

VoiceGuide

User Guide

July, 2016

1. Introduction	1
Welcome To VoiceGuide	1
Which version to use	2
2. Configuration	3
System Requirements	3
Installing v7.x - VoIP (SIP) / HMP	5
Installing v7.x - Telephone lines/trunks	15
Installing v6.x - Telephone lines/trunks	18
Installing v5.x - TAPI devices	20
Installing v5.x - CAPI devices	23
Text To Speech	28
Call Transfers and Conferencing	31
Databases	36
End of Call Detection	40
Distinctive Ring Detection	47
Cisco Call Manager Configuration	50
T1/E1 ISDN Configuration	52
T1/E1 RobbedBit/CAS/R2 Configuration	60
VoIP Line Registration	63
Command Line Options	68
Registering VoiceGuide	69
Unique Identifier	71
3. Script Design	74
Introduction	74
Graphical Design Environment	75
Module Types	77
Paths	78
Result Variables	82
Call Start	89
Call Finish	91
Multilanguage Systems	93
Protected Scripts	95
Sound files	96

Testing Scripts	100
4. Modules	101
Play	101
Record	105
Get Numbers	109
Say Numbers	114
Send Email	116
Database Query	119
Run Program	126
Run VB Script	131
Call Web Service	148
Time Switch	156
Transfer Call	158
Evaluate Expression	166
Fax Send and Receive	178
Send Phone Message	180
Send Pager Message	183
Hangup Call	185
5. Reporting	187
Dashboards	187
REST API	188
Line Status Monitor	190
Automated Report Scheduling	191
6. Voicemail	192
Introduction	192
Voicemail System Manager	195
Voicemail Menus	200
Message Lamps	203
7. Outbound Dialing	205
Loading Numbers to Call	205
Detect Call Answer	216
Outbound VoIP calls	219
Predictive Dialers	221
External Database Source (v7)	222
8. CRM Integration	228

SugarCRM	228
9. Speech Recognition	232
Introduction	232
Install LumenVox	234
10. Logs	237
Script Logs	237
Call Detail Records (CDRs)	239
Temp Files	240
11. COM and WCF Interface	241
Admin_TraceLogAdd	241
Dialer_MakeCall	242
Dialer_OutDialQueAdd	243
Bridge_Connect	246
Bridge_Disconnect	247
Line_Hangup	248
Line_Pickup	249
Play_Start	250
Play_Stop	251
Record_Stop	252
Record_Start	253
Record_2Lines_Start	254
Run_ResultReturn	255
RvGet	256
RvGet_All	258
RvGet_AllXml	259
RvSet	260
RvSet_RvList	261
Script_Gosub	262
Script_Goto	263
Script_Return	264
Serial_Tx	265
Vm_Event	266
Vm_VmbConfig_Get	267
Vm_VmbConfig_Set	268
12. PBX Inband Signaling	269

Inband Signaling	269
Legal Information	273
Copyright & Disclaimer	273

Welcome To VoiceGuide

VoiceGuide was designed to allow fast and easy creation of Inbound and Outbound IVR and Messaging systems which can be easily tailored to individual needs.

VoiceGuide's Graphical Script Designer contains a range of highly functional modules to allow rapid creation and easy administration, and scripted Voicemail and ACD features facilitate a wide range of Inbound/Outbound Self Service and Agent Assisted IVR solutions to be deployed.

Fully working evaluation version of the Voiceguide IVR software can be downloaded, allowing full evaluation of the software before purchasing.

Free Support is provided on the public Support Forum which is monitored by our support staff.

Which version to use

Latest version of VoiceGuide is v7.x

VoiceGuide v7.x can use VoIP, and can control analog and T1/E1 ISDN trunks.

Table below lists some of the common devices used in IVR systems, the types of telephone lines used with these devices, and the appropriate VoiceGuide version to use for each scenario:

Device	Lines Used	VoiceGuide Version
Dialogic cards	Analog, T1/E1 ISDN	VoiceGuide v7 / v6 - select the "Dialogic cards" option during install.
(none used)	VoIP - SIP	VoiceGuide v7 - select the "VoIP/HMP" option during install
Dialogic DNI/HMP cards	T1/E1 ISDN	VoiceGuide v7 - select the "VoIP/HMP" option during install
CAPI cards	BRI ISDN	VoiceGuide v5 (TAPI version)
Voice Modems	Analog	VoiceGuide v5 (TAPI version)
TAPI devices	Various	VoiceGuide v5 (TAPI version)

System Requirements

Minimum system requirements for running VoiceGuide are:

VoiceGuide v7:

Win2012, Win2008, Win10, Win8, Win7, Vista, Win2003_SP2, WinXP_SP3
.NET 4.0
Pentium 1GHz with 1GB RAM, or as per Windows min requirements

VoiceGuide v6:

Win2012, Win2008, Win10, Win8, Win7, Win2003, Win2000, WinXP
Pentium 1GHz with 512MB RAM or as per Windows min requirements

VoiceGuide v5:

Win2003, Win2000, WinXP, Win ME, Win 98, Win 95
Pentium 400MHz with 256MB RAM, or as per Windows min requirements

Hardware Requirements

VoiceGuide can be used with:

Dialogic cards

A range of cards is available including cards for use with traditional 'analog' telephone lines (eg: D/4PCIUF), as well as cards for T1 and E1 based services. (D/240JCT, D/300JCT, etc.)

Please see: [Installing with Dialogic.](#)

VoIP

No hardware is required for VoIP (SIP) deployments. Dialogic HMP drivers are used on VoIP systems.

Please see: [Installing VoIP Systems.](#)

CAPI (ISDN) cards

Any card supporting CAPI can be used as well. These cards will allow VoiceGuide to be used on BRI ISDN lines (2 channel 128Kbs) and PRI (24 channel T1 or 30 channel E1) ISDN lines. There are many manufacturers of CAPI cards, best known are [AVM](#) (Fritz!Card) and [Eicon](#).

Please see: [Installing with CAPI cards.](#)

Voice Modems

Many modems can support "Voice" functionality, but most of them do not support it well, with poor sound quality and unreliable detection of caller's key presses.

Other Telephony Devices

There are a number of other telephony cards and devices which can be used with VoiceGuide - please see: [Recommended Hardware](#)

Installing v7.x - VoIP (SIP) / HMP

VoiceGuide v7.x can use VoIP (SIP) to handle calls over the network interface.

No physical cards are needed.

VoIP deployments can also be made on Virtual Machines and on 'Cloud' services.

Platforms supported: VMware ESXi, Microsoft Azure Cloud.

Installing Dialogic HMP

Dialogic HMP 3.0 drivers should be installed first.

Please always refer to the Dialogic Release Notes and/or Installation Notes to determine what operating system may be used, and what system tests need to be run to confirm HMPs suitability. eg: hpettool.exe to test for HPET compatibility.

It is best if only one network interface is enabled on the system.

If more than one network interface is enabled on the system then Dialogic HMP must use the 'First' network interface for all communications.

On Server class systems if there is an option to set 'C-State' then you need to ensure that C-State is disabled (set to C0).

After installing HMP you will need to ensure that the Dialogic service is started. The Dialogic service can be started using the Dialogic Configuration Manager (DCM).

If Dialogic HMP service has problems starting then you should ensure that there is no other VoIP software installed on the system that would interfere with HMP operation, and that no anti-virus type software is blocking the service start.

If there are still problems then please try fully disabling the Windows' User Access Control and then reinstalling the Dialogic System Release drivers. Windows' User Access Control is disabled in the Windows Registry, by setting the EnableLUA parameter to 0.

To re-install Dialogic System Release you need to first fully uninstall it, selecting 'Do not save configuration', and restart system after uninstall. After installing the Dialogic System Release the system needs to be restarted again.

If using HMP 'thin interface' cards please ensure that the 'Device -> Restore Defaults' is done for all the cards present in system as well as for the 'HMP_Software' device and the 'Configured Devices' root. This is done using Dialogic DCM.

It's recommended to also set the Dialogic service to start automatically. This can be set using DCM 's Settings -> System/Device autostart -> Start System menu, or by using the Windows' Control Panel -> System and Security -> Administrative tools -> Services applet.

HMP comes with a 1 port Evaluation license.

Please contact sales@voiceguide.com regarding purchasing of HMP licenses or obtaining temporary

Testing Dialogic HMP Installation

After installing HMP, start Dialogic's IP Media Server Demo application, and then perform a test call into the system by dialling the IP address of the HMP system from a VoIP phone or softphone on another system.

The IP Media Server Demo application is located here:

`C:\ProgramData\Dialogic\HMP\demos\IPMediaServer\Release\IPMediaServer.exe`

The `C:\ProgramData\` directory is a hidden directory, and needs to be made visible first:

Windows 8, Windows 2012 :

In File Explorer click View on the menu bar at the top. From the displayed icons on the ribbon click Options. From the opened File Options box go to View tab. In the Advanced Settings list select "Show hidden files, folders and drives" option. Click OK to save.

(See also: [here](#) and [here](#))

Windows 7, Windows 2008 :

Click on C:\ drive in Windows Explorer, then click on the "Organize" button in top left corner and select "Folder and Search Options". Click on "View" tab and select "Show hidden files, folders and drives" option. Click OK to save.

On Windows XP and Windows 2003 systems the IP Media Server Demo application is located here:

`C:\Program Files\Dialogic\HMP\demos\IPMediaServer\Release\IPMediaServer.exe`

Softphones

NOTE: The testing softphone MUST NOT be installed on the same system as Dialogic HMP.

It must be installed on some other system.

There can be no other SIP software that uses SIP 5060 port installed on same system as Dialogic HMP.

Only one software can use the SIP 5060 port.

One softphone that you can use to dial the IP address directly is Linphone.

Linphone can be downloaded here: <http://www.linphone.org/>

With Linphone the SIP address to be dialled must be prefixed with "sip:" eg: sip:10.1.1.19 will dial IP address 10.1.1.19 directly.

Also please select Linphone's Options->Preferences menu, and set the following:

Network Settings tab : Media Encryption is set to 'None'

Codecs tab: only enable the PCMU and PCMA codecs, and disable the rest.

Another softphone that you can use to dial the IP address directly is SJPhone.

SJPhone can be downloaded here: <http://www.sjlabs.com/sjp.html>

Once SJPhone is installed the IP address by itself can be just entered as the number to call, and SJPhone will dial that IP address directly.

Trace SIP Messages

WireShark can be used to confirm that SIP packets are arriving at VoiceGuide/HMP system.

To isolate SIP traffic the following should be specified in WireShark's Filter text box:

```
sip
```

To isolate SIP and RTP traffic the following should be specified in WireShark's Filter text box:

```
sip || rtp
```

To save the isolated traces go to WireShark's File->Export Specified Packets menu and save the displayed packets as .pcapng format.

A successful running of the IPMediaServer demo should look like this in the Command Prompt window:

```
C:\Program Files\Dialogic\HMP\demos\IPMediaServer\Release>ipmediaserver
*****
*                                                                 *
*  IP Media Server -  Media services over IP Demo Program.  *
*                                                                 *
*****
DTMFMode inband
TxCoder[0]
        Capability: g711mulaw
        Type: 2
        Direction: 1
        Payload_type: 255
        FramesPerPacket: 20
        VAD: 0.
```

```
RxCoder[0]
```

```
Capability: g711mulaw  
Type: 2  
Direction: 2  
Payload_type: 255  
FramesPerPacket: 20  
VAD: 0.
```

```
[info] CEventManager::Init: Initializing channels...may take a few seconds!
```

```
Number of Fax (& Voice) boards found: 1
```

```
Number of Voice (& Fax boards) found: 1
```

```
Number of IPT boards found: 1
```

```
Number of IPM boards found: 1
```

```
Waiting for key:
```

```
'Q' - to quit
```

```
CIPDevice::processEvent -> receive GCEV_UNBLOCKED on :N_ipmB1C1:P_IP:M_ipmB1C1
```

Installing VoiceGuide

Dialogic's HMP drivers must be installed before proceeding with VoiceGuide installation (see above).

VoiceGuide needs to be installed using the Administrator account. If not logged in as Administrator then right click on install .EXE and select 'Run as administrator'.

When installing VoiceGuide 7 select the "VoIP / SIP / HMP" option to install the configuration files used for VoIP / HMP installations.

Configuring VoiceGuide

System configuration is set using the Config.xml file. Config.xml file is located in VoiceGuide's \conf\ subdirectory.

Config.xml is used to set the script to be used when answering incoming calls, and a number of other settings.

The default Config.xml file opens one VoIP line.

See the sample Config files provided for examples of how to set up the Config.xml file for systems other than the default 1 line VoIP setup.

If not running as Administrator then some permission changes may need to be made to allow editing of Config.xml, VG.INI, etc. files.

In Windows Explorer:

- Right click on the VoiceGuide directory,
- Select Properties,
- Go to the Security tab,
- Make the necessary changes

Alternatively, the text editor application used to edit the Config.xml, VG.INI, etc. files would need to be started using the 'Run as administrator' option. Then Windows will not block the saving of new versions of Config.xml, VG.INI, etc. files.

Perform a test call into the system by dialling the IP address of the HMP system from VoIP phone or softphone on another system.

In order for VoiceGuide to answer calls directed to specific VoIP lines and extensions VoiceGuide must 'register' itself with the relevant VoIP Server. Configuration of the registration process is made through the Config.xml. The relevant VoIP_Registrations and VoIP_Authentications sections in Config.xml must be filled out in order for VoiceGuide to register the VoIP lines/extensions.

After you hear the demonstration script answer the call you are now ready to start creating your own scripts, make outgoing calls etc.

The Script Designer and other VoiceGuide applications can be started by right-clicking on the IVR tray icon and selecting from the context menu.

If not running as Administrator then you will not be able to stop and start the VoiceGuide service using the VoiceGuide Service Monitor applet from the icon tray.

To stop/start the VoiceGuide IVR service you will need to go to Control Panel -> System and Security -> Administrative tools -> Services

Setting Transport Protocol and Port

Default transport protocol is UDP. Most VoIP systems use UDP for SIP communications. If the VoIP systems with which VoiceGuide is to communicate use TCP instead, then VoiceGuide can be configured to use TCP by setting the following entry in VG.INI's [SIP] section:

```
E_SIP_DefaultTransport=ENUM_TCP
```

Port used can be set using the sip_signaling_port entry.

Firewall Configuration

In order to allow incoming VoIP calls to be received by VoiceGuide service the Firewall Inbound Rules will need to be set to allow incoming SIP packets (or, if possible, the Firewall can just be disabled).

Firewall Inbound Rules should be set to allow the vgIvrService.exe program to receive all types of incoming messages (both TCP and UDP).

The specific port ranges used are: SIP: 5060
RTP: 49100 and above
VoiceGuide Reporting and REST API : 7130 - 7140

Wav file format

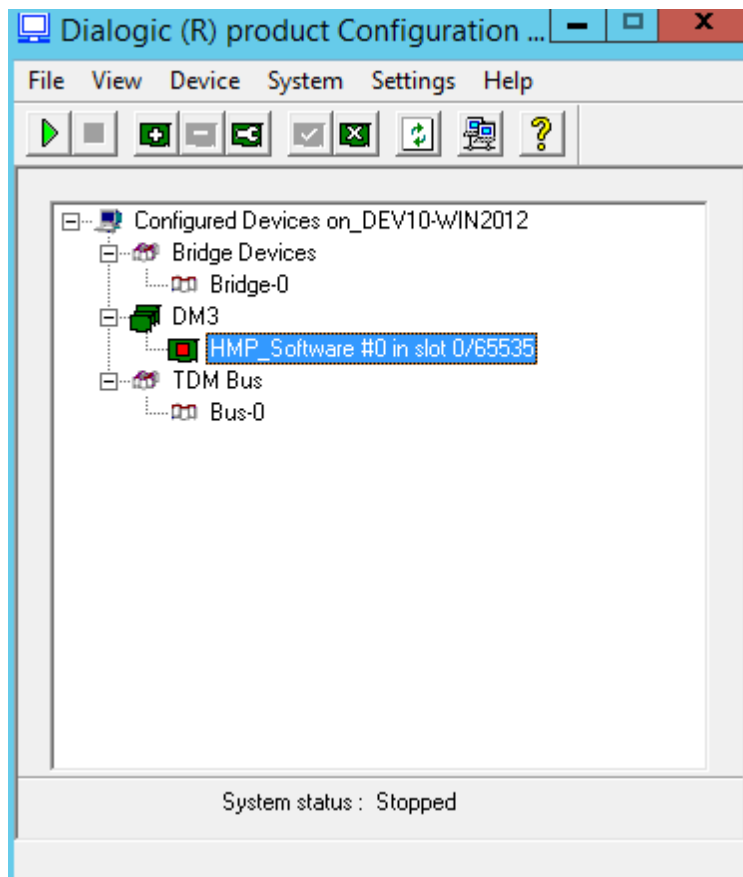
All sound files should be in .WAV uLaw format (8kHz, 8 bit, Mono, uLaw). uLaw is often used for VoIP connections.

The exact same bytes from the .WAV uLaw sound file will be sent in the RTP packets if the connection also uses G.711 uLaw. This allows exact control over quality of sound played over the VoIP connection.

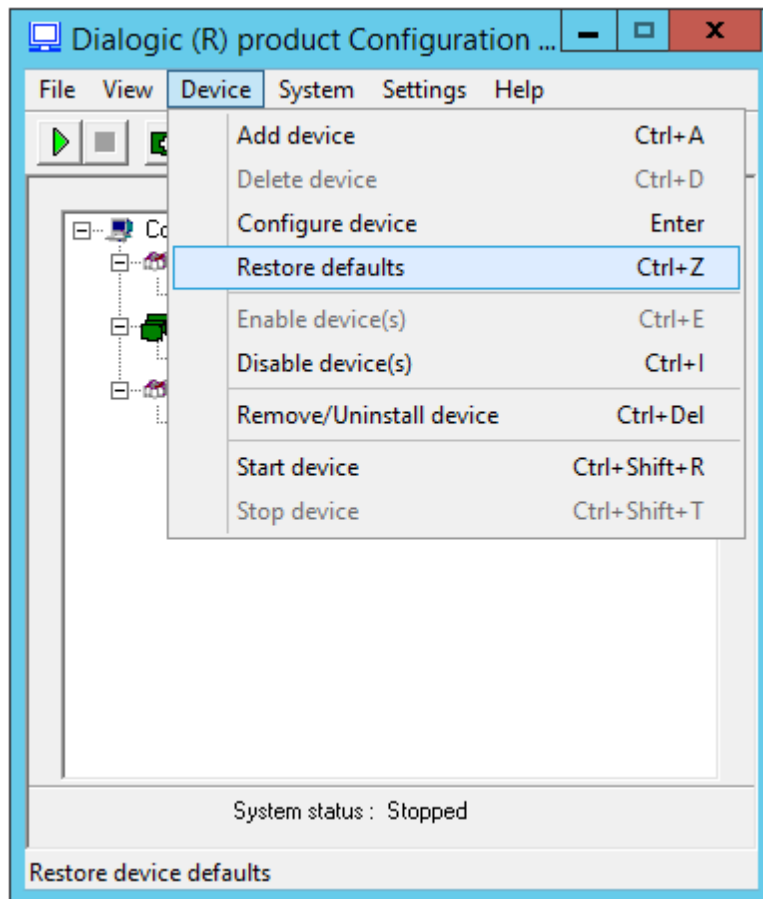
Applying Dialogic HMP License

After selecting the new license in the HMP License Manager the following steps need to be performed in the DCM.

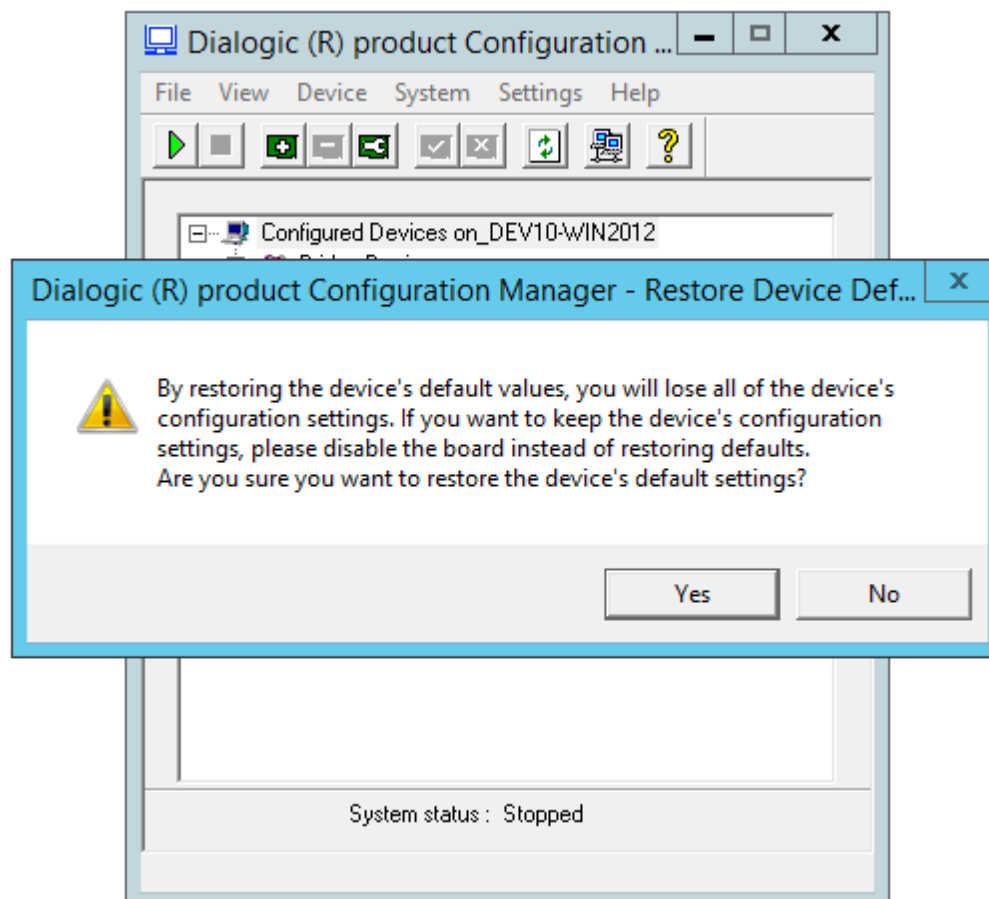
1. Click on the "HMP Software" entry in the DCM to select it.



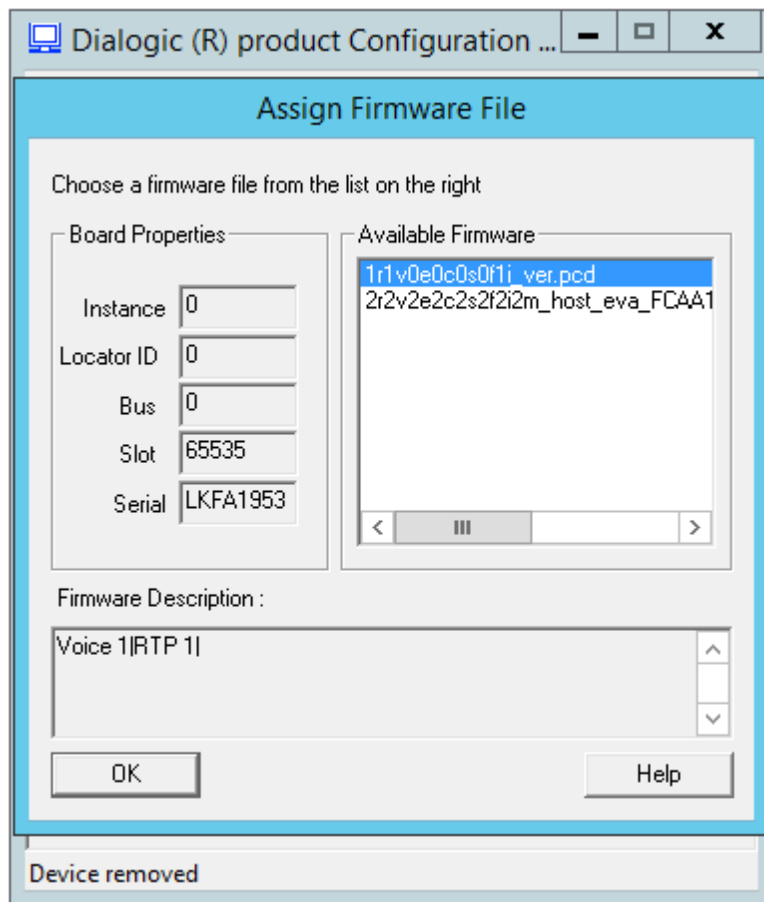
2. Click Device -> Restore Defaults menu.



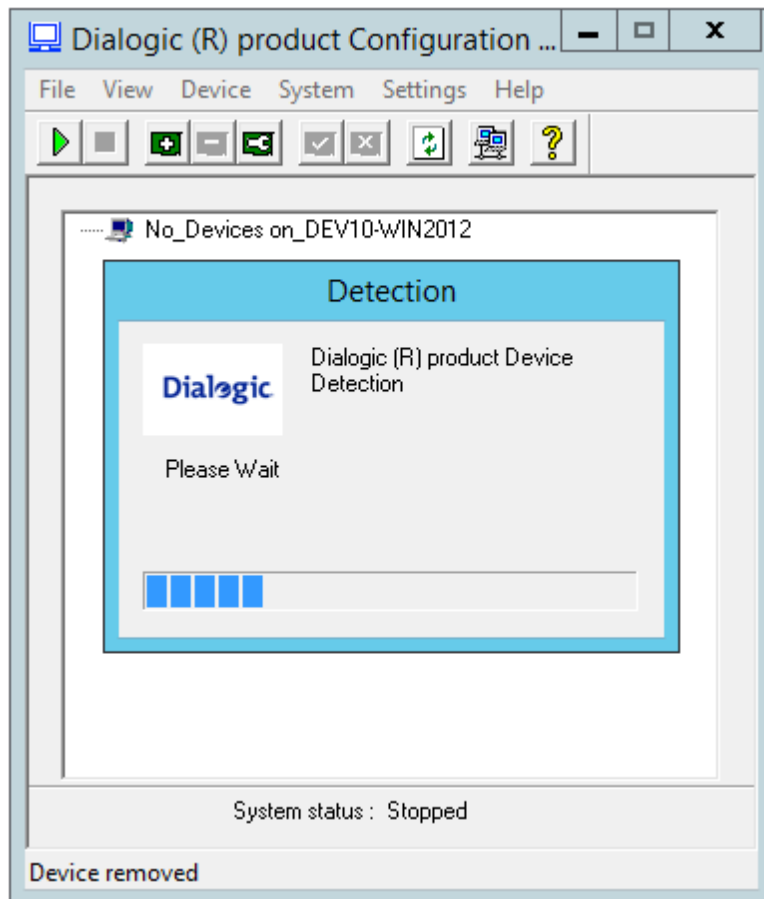
3. Press Yes.



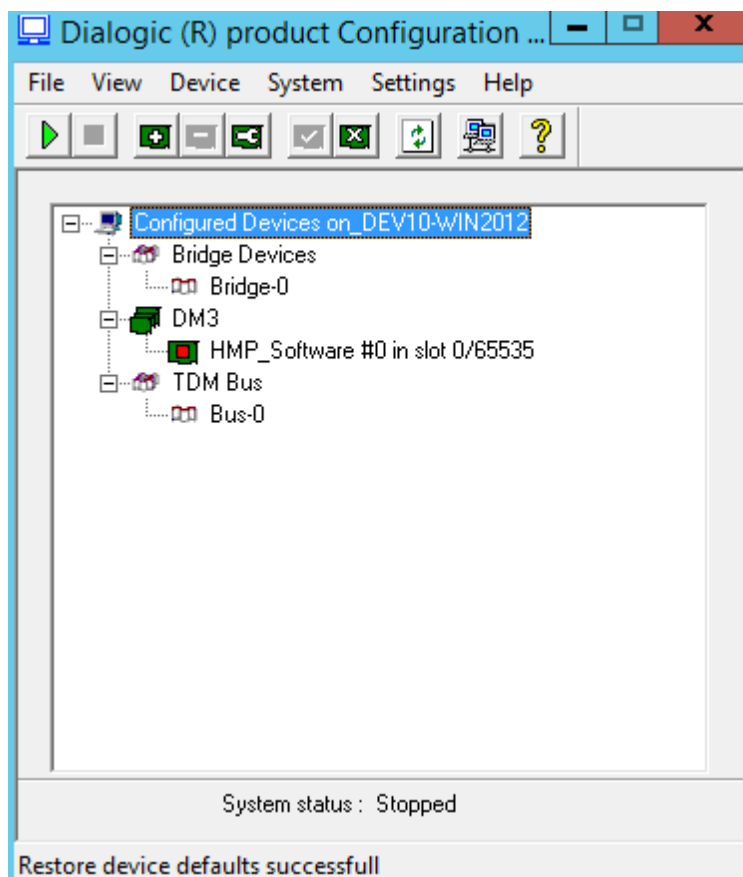
4. The "Assign Firmware File" windows will open. Click on the license which you want to select and then press OK.



5. Another Winow will open for a few seconds.



6. On completion the "Restore device defaults successful" message will be shown. License has now been applied.



Installing v7.x - Telephone lines/trunks

Installing Dialogic Drivers

Dialogic Drivers (System Release) for use with VoiceGuide can be downloaded from the VoiceGuide Downloads page.

Please always refer to the Dialogic's Release Notes and/or Installation Notes to determine what Operating System may be used.

The Dialogic card should be placed in system, and the telephone lines should be attached to the cards before the Dialogic System Release drivers are installed. The Dialogic System Release Drivers will detect the cards present in system upon installation.

Dialogic Drivers install asks user to select options to install. It is best to just select all the options available.

If using a 32-bit version of Windows, the Physical Address Extensions (PAE) need to be disabled. Dialogic install program will display an alert if it detects that PAE is enabled on system.

After installing the Dialogic System Release the machine needs to be restarted.

After restarting the system open the Dialogic Configuration Manager (DCM) and confirm that the Dialogic service can be started.

If there are problems starting the Dialogic service please ensure that the Windows account used has sufficient privileges and the install was started using 'Run as Administrator' option.

If there are still problems then please fully disable the Windows' User Access Control and then reinstall the Dialogic System Release drivers. Windows' User Access Control is disabled in the Windows Registry, by setting the EnableLUA parameter to 0.

To re-install Dialogic System Release you need to first fully uninstall it, selecting 'Do not save configuration', and restart system after uninstall.

After installing the Dialogic System Release the machine needs to be restarted.

The Dialogic service should be set to start automatically. Use DCM 's Settings -> System/Device autostart -> Start System menu, or by using the Windows' Control Panel -> System and Security -> Administrative tools -> Services

Only after the Dialogic System Release drivers are installed on the system and the Dialogic service is started can the VoiceGuide installation proceed.

Testing Dialogic Installation

Dialogic drivers install some sample programs that can be used to test correct card/drivers operation.

The sample programs are located in this directory:

`C:\ProgramData\Dialogic\demos\voice\`

The C:\ProgramData\ directory is a hidden directory, and needs to be made visible first.

To make C:\ProgramData\ directory visible click on C:\ drive in Windows Explorer, then click on the "Organize" button in top left corner and select "Folder and Search Options". Click on "View" tab and select "Show hidden files, folders and drives" option.

On the older Windows XP and Windows 2003 systems the sample programs are located in this directory:

C:\Program Files\Dialogic\demos\voice\

One of the sample programs that can be used is ANSRMT.EXE, it is located in the \demos\voice\ANSRMT\ subdirectory.

Installing VoiceGuide

Dialogic System Release drivers must be installed first.

VoiceGuide needs to be installed using the Administrator account. If not logged in as Administrator then right click on install .EXE and select 'Run as administrator'.

When installing VoiceGuide select the "Telephone Lines (Analog, T1/E1 ISDN)" option to install the configuration files which are used with the Dialogic cards.

Configuring VoiceGuide

System configuration is set using the Config.xml file. Config.xml file is located in VoiceGuide's \conf\ subdirectory.

Config.xml is used to set the script to be used when answering incoming calls, and a number of other settings.

The default Config.xml file opens the first 4 ports of an analog Dialogic card. If an analog Dialogic card is installed then immediately after installing VoiceGuide you will be able to call into the system on any of the 4 ports and you will hear VoiceGuide answer the call and start a demonstration 'Credit Card Payment' script.

If a card other than a 4 port analog card is used then the Config.xml file in VoiceGuide's \conf\ subdirectory will need to be changed to match the card. Sample Config.xml files are provided in that directory as well, showing how Config.xml should be set if a card other than a 4 port analog card is used.

If not running as Administrator then permission changes may need to be made to allow editing of Config.xml, VG.INI, etc. files.

In Windows Explorer:

- Right click on the VoiceGuide directory,
- Select Properties,
- Go to the Security tab,
- Make the necessary changes

Alternatively, the text editor application used to edit the Config.xml, VG.INI, etc. files would need to be started using the 'Run as administrator' option. Then Windows will not block the saving of new versions of Config.xml, VG.INI, etc. files.

After you hear the demonstration script answer the call you are now ready to start creating your own scripts, make outgoing calls etc.

The Script Designer and other VoiceGuide applications can be started by right-clicking on the IVR tray icon and selecting from the context menu.

If not running as Administrator then you will not be able to stop and start the VoiceGuide service using the VoiceGuide Service Monitor applet from the icon tray.

To stop/start the VoiceGuide IVR service you will need to go to Control Panel -> System and Security -> Administrative tools -> Services

Wav file format

When using VoiceGuide v7.x all sound files should be in format: uLaw or ALaw, 8kHz, 8 bit, Mono.

uLaw or ALaw format is selected at install time.

Systems located in North America and Japan should use the uLaw format.

Systems located in the rest of the world use ALaw format.

These are the two formats used by telephone companies to transmit voice over the traditional analog and digital connections, and using this sound format will result in little or no distortion of the sound file when it is transmitted over the phone line.

Installing v6.x - Telephone lines/trunks

NOTE: For new systems it is recommended that VoiceGuide v7 is used.

Installing Dialogic Software

We recommend using Dialogic's System Release 6.0 drivers.

VoiceGuide v6 will still work if Dialogic's System Release 5.1.1 + SR5.1.1SP1 drivers are used, but note that most of Dialogic's newest cards require System Release 6.0 drivers.

Please always refer to the Dialogic's Release Notes and/or Installation Notes to determine what Operating System may be used with these drivers.

The Dialogic card should be placed in system before the Dialogic System Release drivers are installed.

The Dialogic System Release Drivers will detect the cards present in system upon installation.

The Dialogic service must then be started. This can be done using the Dialogic Configuration Manager (Windows' Start -> Dialogic) or using the Windows' Control Panel -> System and Security -> Administrative tools -> Services list.

The Dialogic service should be set to start automatically. Use DCM 's Settings -> System/Device autostart -> Start System menu, or by using the Windows' Control Panel -> Administrative tools -> Services

Only after the Dialogic System Release drivers are installed on the system can the Dialogic service is started can the VoiceGuide installation proceed.

Installing VoiceGuide

Dialogic System Release drivers must be installed and started before installing VoiceGuide v6(see above).

VoiceGuide v6.x comes pre-configured to be used with the first 4 ports of an analog Dialogic card.

If an analog Dialogic card is installed then immediately after installing VoiceGuide you will be able to call into the system and you will hear VoiceGuide answer the call and start a demonstration Credit Card Payment script.

Dialogic lines which VoiceGuide should be using are set in the Config.xml file.
Config.xml file is located in VoiceGuide's \Data\ subdirectory.

If a card other than a 4 port analog card is used then you will need to first change the Config.xml file in VoiceGuide's \data\ subdirectory before starting VoiceGuide. Sample replacement Config.xml files are provided in that directory as well, showing how Config.xml should be setup if a card other

then a 4 port analog card is used.

After you hear the demonstration script answer the call you are now ready to start creating your own scripts. Refer to the Script Design section of the Help file.

Running VoiceGuide

Before running VoiceGuide you need to ensure that the Dialogic service has fully started. Confirming that the service status is 'Started' can be done using the Dialogic Configuration Manager (Windows' Start -> Dialogic) or using the Windows' Control Panel -> System and Security -> Administrative tools -> Services list.

Wav file format

When using VoiceGuide v6.x all sound files should be in format: PCM 8kHz, 8 bit, Mono.

Installing v5.x - TAPI devices

Please note that we do not recommend using voice modems. Voice modems are not really designed to implement professional IVR/Dialer/Voicemail systems.

Many voice modems have one or more of the following problems:

- Poor sound quality/volume.
- Unreliable DTMF tone detection.
- Cannot do call transfers as hookflash length is too long or too short.
- Mistakenly detect a disconnect tone while playing or recording messages and hangup a call halfway through playing/recording of sound file.
- Unable to interrupt the playing of a sound file halfway through.

The sound quality of various modems varies greatly. In general external modems work better than internal modems.

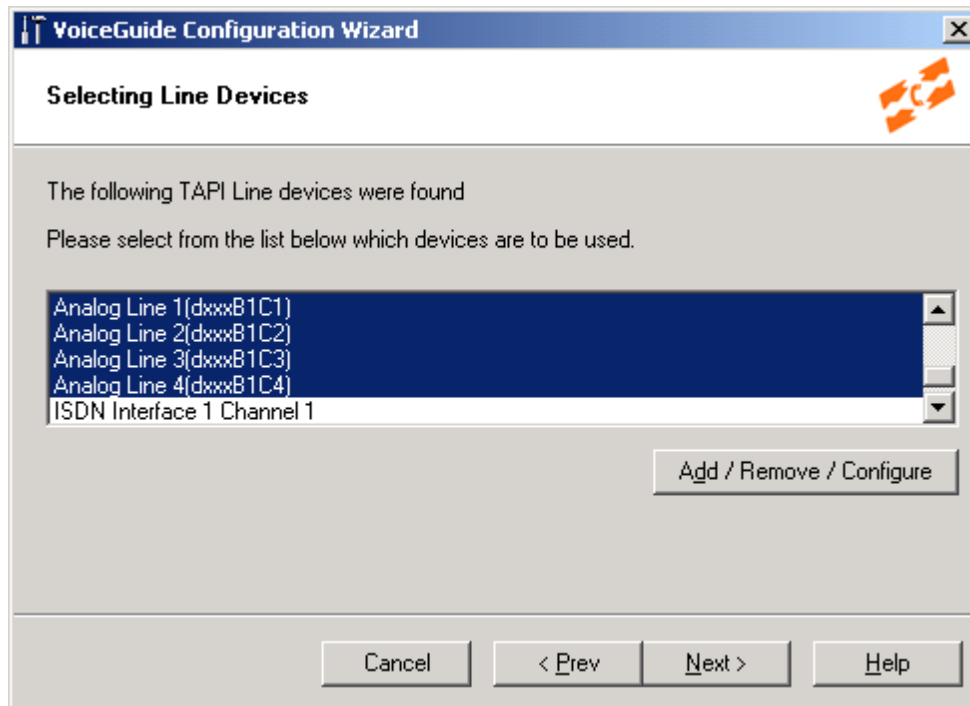
As TAPI devices on the market change frequently we cannot provide a definitive list of current 'best' TAPI devices. We have found that the quality of sound playback and recording can vary between different versions of the same modem, and it will sometimes depend on which version of Windows the modems is installed. When selecting your voice modem we recommend trying a few modems if possible and choosing the best one. The problems faced by many users in finding an adequate sounding voice modem is the reason why many users opt to use telephony cards instead.

Voice modems can be used under Win98/ME, Win2000 and WinXP.

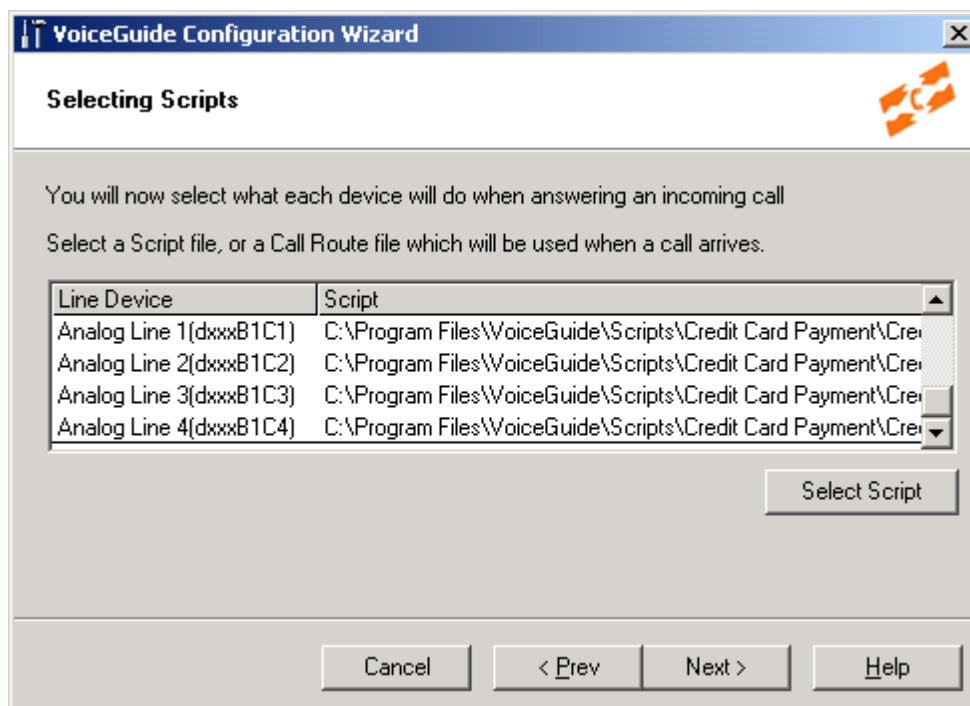
Please ensure that the device's Wave Driver has installed correctly. Without this driver installed VoiceGuide will not regard the device as as a Voice capable device.

Running the Setup Wizard

The Setup Wizard will discover all the TAPI capable telephony devices and will allow you to select which devices you would like to use with VoiceGuide:



Next you need to select the Scripts which will be used by VoiceGuide when an incoming call arrives on the selected devices. To begin with we'd recommend you select the demonstration script in VoiceGuide's **"/Scripts/Credit Card Payment"** directory:



When Setup Wizard configuration has completed you should now be able to start VoiceGuide and call into the system to hear it answer the call and lead you through the selected VoiceGuide script.

You can open the **"/Scripts/Credit Card Payment"** in the Graphical Design Environment to see how the script has been put together.

Sound file format

When using a TAPI/WAVE device all sound files should be in format: PCM 8kHz, 16 bit, Mono.

Installing v5.x - CAPI devices

VoiceGuide v5 can be used to control CAPI capable hardware.

Most CAPI capable cards are BRI ISDN cards which can support 2, 4 or 8 ports. If T1/E1 ISDN systems are considered then VoiceGuide v7 should be used.

Popular CAPI cards are:

[AVM](#) (Fritz!Card, B1, C2, C4, T1, Fritz!GSM, etc)

[Eicon](#) (BRI-2M, 4BRI-8M, etc)

Eicon cards also come with their own TAPI/Wave drivers, which can be used directly by VoiceGuide.

Some CAPI devices do not come with their own TAPI/Wave drivers, and in these cases in order for them to be able to be used by VoiceGuide a 3rd party product is needed: ComISDN, which is essentially a converter between the CAPI interface and TAPI/Wave interface, allowing programs that use the TAPI/Wave interface to work with any CAPI based hardware.

Install order

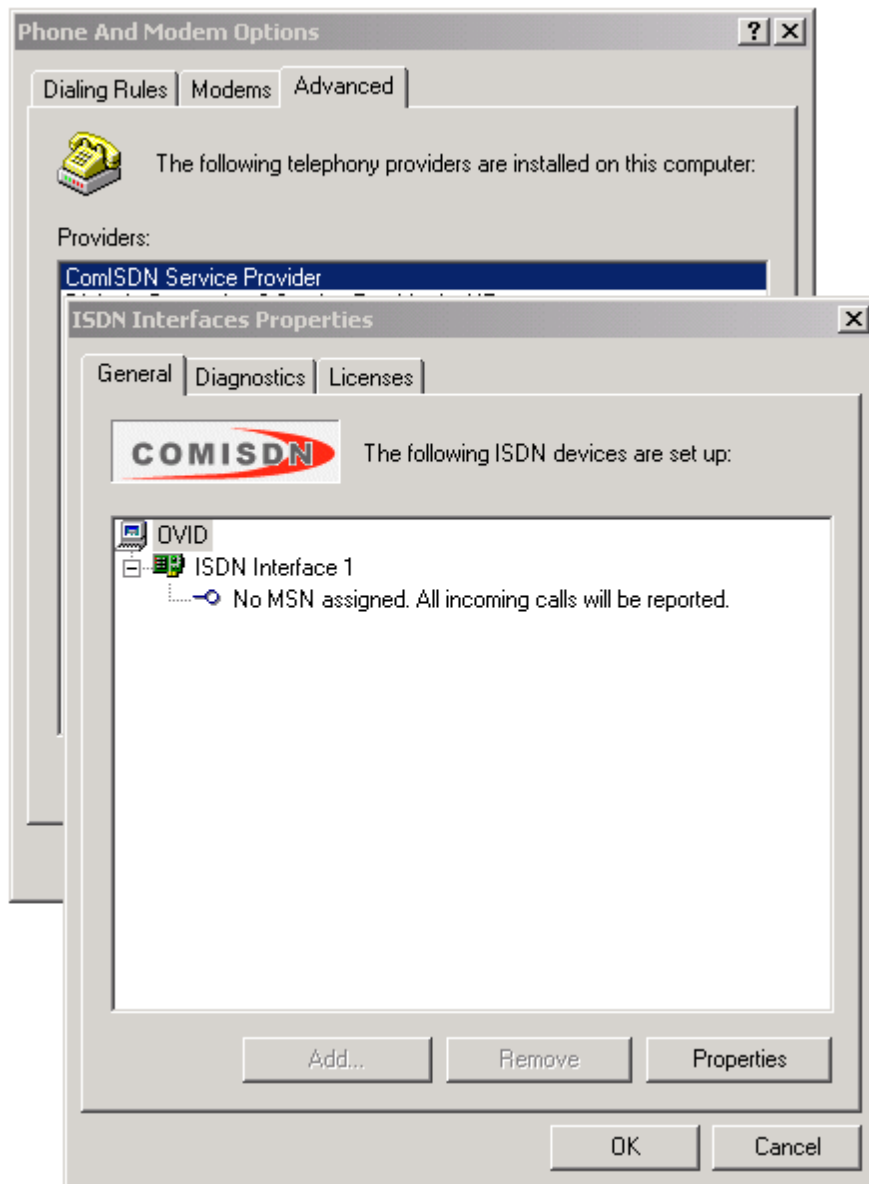
1. Install the CAPI hardware.
2. Download and Install the ComISDN TSP from <https://n-g-media.com/>, and configure it to control the CAPI hardware.
3. Install VoiceGuide (select the "TAPI" install option),

Configuring ComISDN

The ComISDN TSP can be configuration window is accessed using:

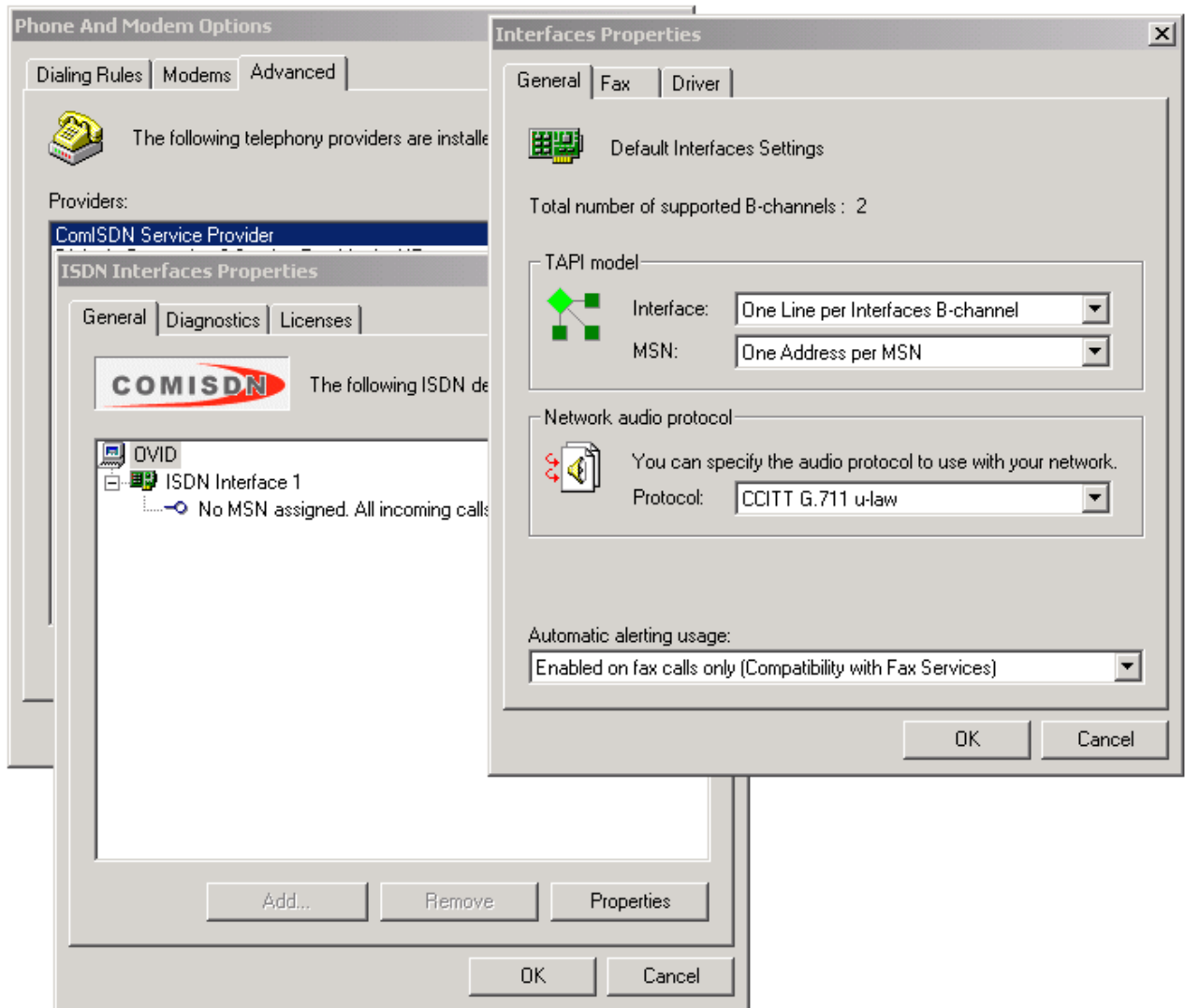
Open Start -> Settings -> Control Panel then click "Phone and Modem Options" or "Telephony", and then under "Advanced" or "Telephony Drivers", select "ComISDN Service Provider" and click "Configure".

The ISDN Interface Properties window will appear:



The main setting that needs to be confirmed as having been correctly set is the "Network audio protocol". If not set correctly the sound played will sound distorted.

To set the Network Audio Protocol select the Computer (root) and then click "Properties".

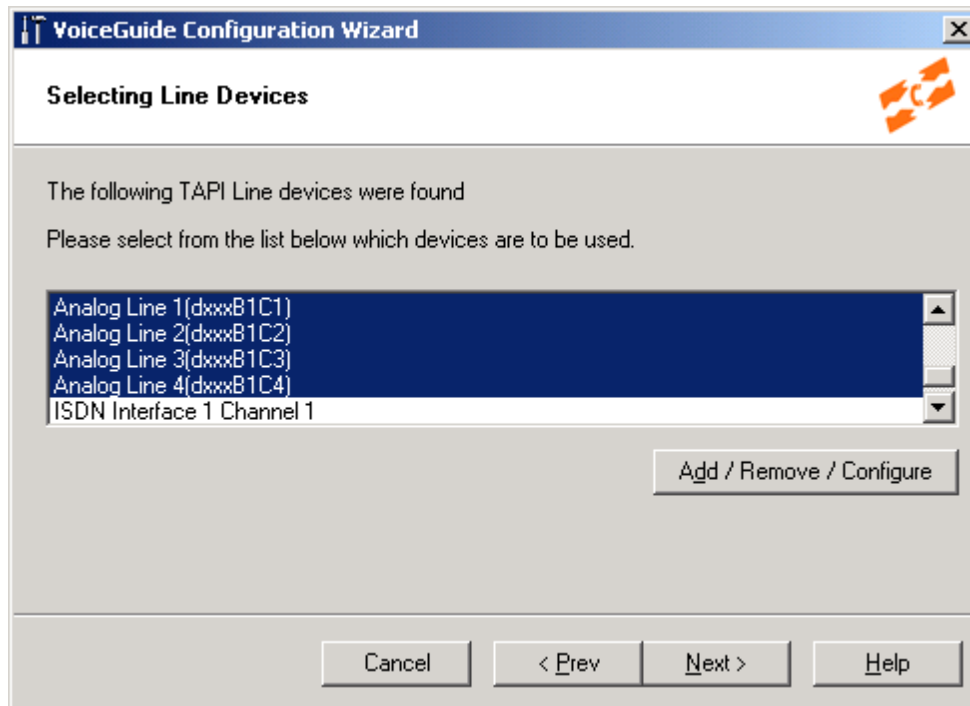


By default, ComISDN will use "u-Law" if it detected that the version of Windows installed is a US, Canadian or Japan one. For all other countries, it will use "A-Law".

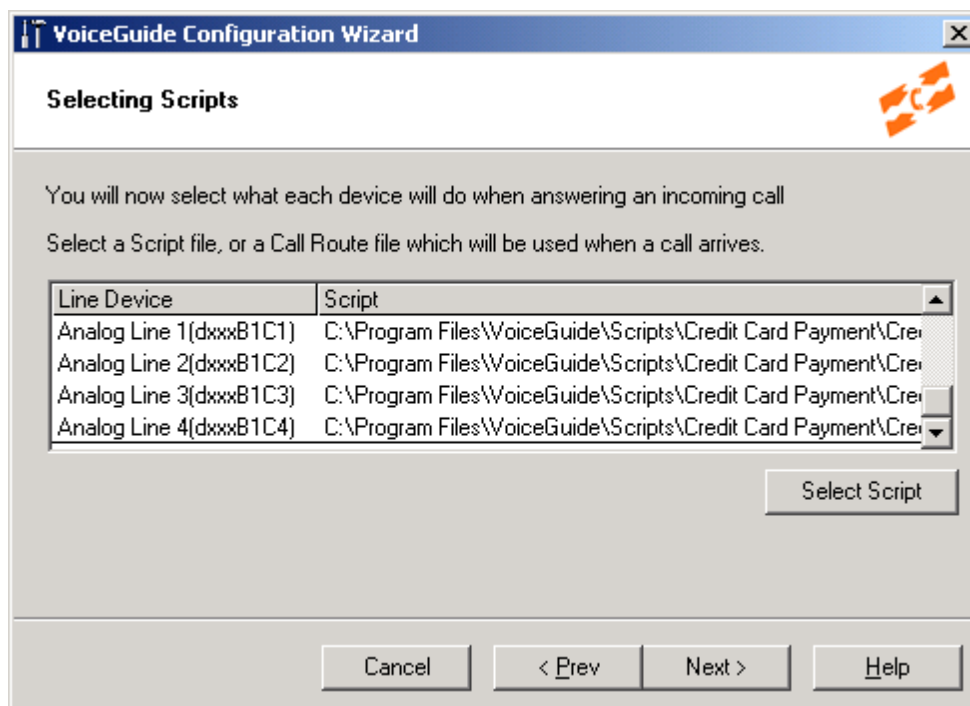
If Windows has not been configured with the correct country setting then it is possible that this setting is incorrect and will need to be set manually. please ensure that it is set to "u-Law" if you are located in US, Canada or Japan, and set it to "A-Law" if you are located in other countries. It may also want to confirm with the telephone company or PBX supplier which protocol should be used - or just try them both and see which one sounds better.

Running the Setup Wizard

The Setup Wizard will discover all the TAPI capable telephony devices and will allow you to select which devices you would like to use with VoiceGuide:



Next you need to select the Scripts which will be used by VoiceGuide when an incoming call arrives on the selected devices. To begin with we'd recommend you select the demonstration script in VoiceGuide's **"/Scripts/Credit Card Payment"** directory:



When Setup Wizard configuration has completed you should now be able to start VoiceGuide and call into the system to hear it answer the call and lead you through the selected VoiceGuide script.

You can open the **"/Scripts/Credit Card Payment"** in the Graphical Design Environment to see how the script has been put together.

Wav file format

All sound files used on a VoiceGuide v5 system should be in format: PCM 8kHz, 16 bit, Mono. Other file formats will not play.

Text To Speech

*Only the Trial and Enterprise versions of VoiceGuide support Text to Speech (TTS).
The Personal and Professional versions of VoiceGuide do not support TTS.*

VoiceGuide can use SAPI compatible TTS engines, and a range of other TTS engines for which MRCPv1, MRCPv2 and native integrations are available.

Please contact sales@voiceguide.com to discuss which TTS engine would best suit your requirements.

VoiceGuide v7 - SAPI TTS

SAPI TTS engine to be used is set in VG.INI, section [SAPI], field TTSEngine. eg:

```
1111111111[SAPI]
TTSEngine=Cepstral Allison-8kHz
```

The list of TTS engine names that are installed on the system can be obtained from the VoiceGuide ktTts trace log file. At the beginning of the trace file there will be a section that looks like this:

```
124244.953 2812 il voice 0 is: [Microsoft Mary], ID=
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSMary]
124244.953 2812 il voice 1 is: [Microsoft Mike], ID=
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSMike]
124244.953 2812 il voice 2 is: [Microsoft Sam], ID=
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSSam]
124244.953 2812 il voice 3 is: [Cepstral Allison-8kHz], ID=
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\Cepstral_Allison-8kHz]
124244.953 2812 il voice 4 is: [Sample TTS Voice], ID=
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\SampleTTSVoice]
```

The name of the engine is listed after the "voice x is:" tag.

The example listing above shows that the following TTS engines are installed on the system:

```
Microsoft Mary
Microsoft Mike
Microsoft Sam
Cepstral Allison-8kHz
Sample TTS Voice
```

so any of the above can be specified as the value of the TTSEngine entry in the VG.INI file.

Format of the WAV file which the SAPI TTS engine should generate can be set in VG.INI, section [SAPI], field SpStreamFormat. eg:

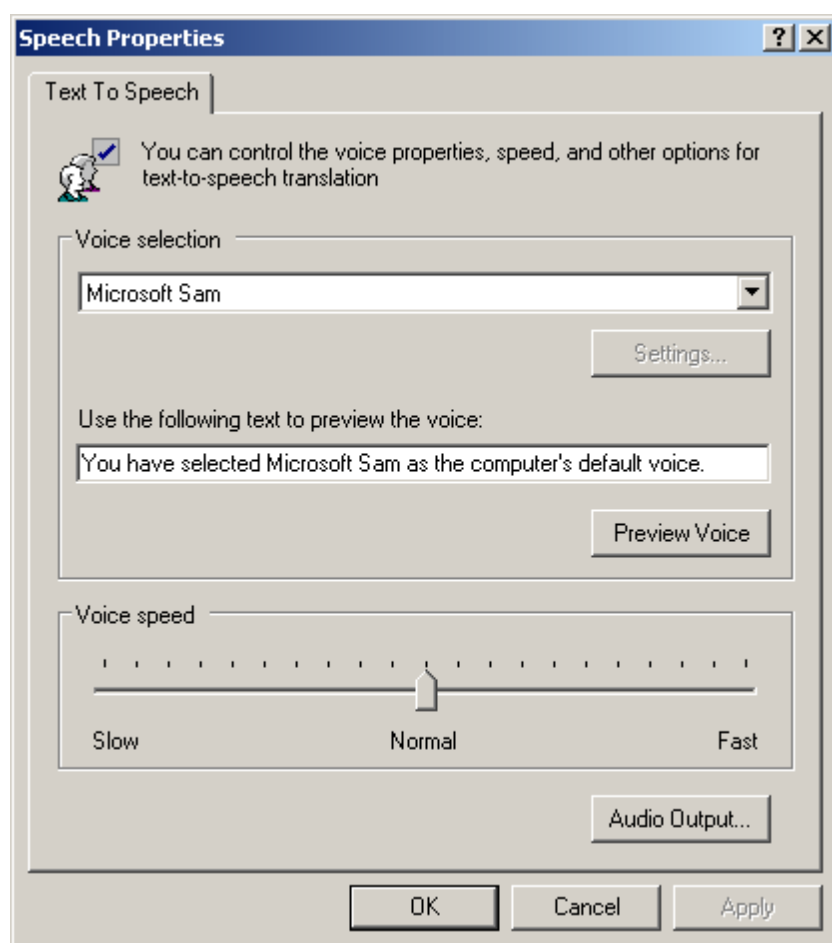
```
[SAPI]  
SpStreamFormat=8khz8
```

The above "8khz8" entry is used to request that SAPI TTS generates 8kHz 8-bit PCM WAV files. Other possible settings are:

```
8khz8  
8khz16  
11khz8  
11khz16  
16khz8  
16khz16  
alaw_8  
ulaw_8
```

VoiceGuide v5 / v6 - SAPI TTS

Default TTS voice can be set using the Speech applet from the Windows Control Panel:



If the Speech applet show above cannot be seen in the Control Panel then it can be found in this

location:

C:\Program Files\Common Files\Microsoft Shared\Speech\sapi.cpl

We have received reports that selecting the default TTS engine using the Control Panel's Speech applet does not work on some systems. In these situations we'd recommend moving to VoiceGuide v7. VoiceGuide v7 sets the TTS engine internally.

Installing and Testing SAPI TTS Engines

Installation of Microsoft's SAPI 5.1 SDK is recommended to allow testing of SAPI TTS engines.

SAPI requires that a sound card is present in the system in order to operate properly.

To test if the SAPI TTS engine installation completed correctly the SAPI SDK's TTSApp program can be used:

Start -> Programs -> Microsoft Speech SDK 5.1 -> Tools -> TTSApp

Then select the TTS engine and press the "Save to .wav" button. If the .WAV file was created and you can play it back using a sound file player then the SAPI TTS engine is properly installed.

Call Transfers and Conferencing

VoIP - SIP

VoiceGuide supports SIP REFER transfers (RFC 3515).

'Tromboned' type transfers are also supported, where VoiceGuide will dial out on another line and connect the two calls together. Using the 2-line-conference type transfer lets VoiceGuide monitor the conversation and react to any selections made during the connection. Afterwards the conference is ended VoiceGuide can take both parties through further separate IVR scripts.

T1, E1, ISDN Trunks

VoiceGuide supports ISDN 'Two B Channel' Transfers (TBCT)

'Tromboned' type transfers are also supported, where VoiceGuide will dial out on another line and connect the two calls together. Using the 2-line-conference type transfer lets VoiceGuide monitor the conversation and react to any selections made during the connection. Afterwards the conference is ended VoiceGuide can take both parties through further separate IVR scripts.

Traditional Analog Lines and 'Robbed Bit' T1 and E1

VoiceGuide can do call transfers on traditional telephone lines in a number of different ways. Most common type of transfer is called a "hookflash transfer". To do this transfer manually from a telephone handset the operator usually just presses the 'flash' button, then dials the telephone number to which the call is to be transferred and then hangs up.

When using hookflash transfers some systems will require you to wait until the destination answers the call before being able to hang up, and some systems will require you to wait only until the destination extension is ringing before hanging up. Some systems will allow you to hangup straight away. Some systems do not support hookflash transfers at all and in such cases you'll need to make a call on another line and have VoiceGuide connect the incoming and outgoing calls together internally to route the call to a new number. These are called 'Tromboned' transfers.

The Help file's section on the [Call Transfer module](#) lists all the different ways in which VoiceGuide can do call transfers.

Before trying to get VoiceGuide to do the transfers it's best to first establish exactly how transfers can be made on your system. Note down step by step what needs to be done and what buttons need to be pressed on your system to make the transfer happen if you just use a telephone handset yourself. Once this is all established you can then go ahead and configure VoiceGuide to do exactly the same things to effect a transfer.

VoiceGuide comes with pre-programmed settings for signals which your PBX or Telephone Network uses to place calls on hold and to forward or conference calls which work for most PBXs, but your PBX / Telephone Network may require different settings. If you do not know what those signals are, you will probably need to ask your PBX supplier or call your Telephone Company.

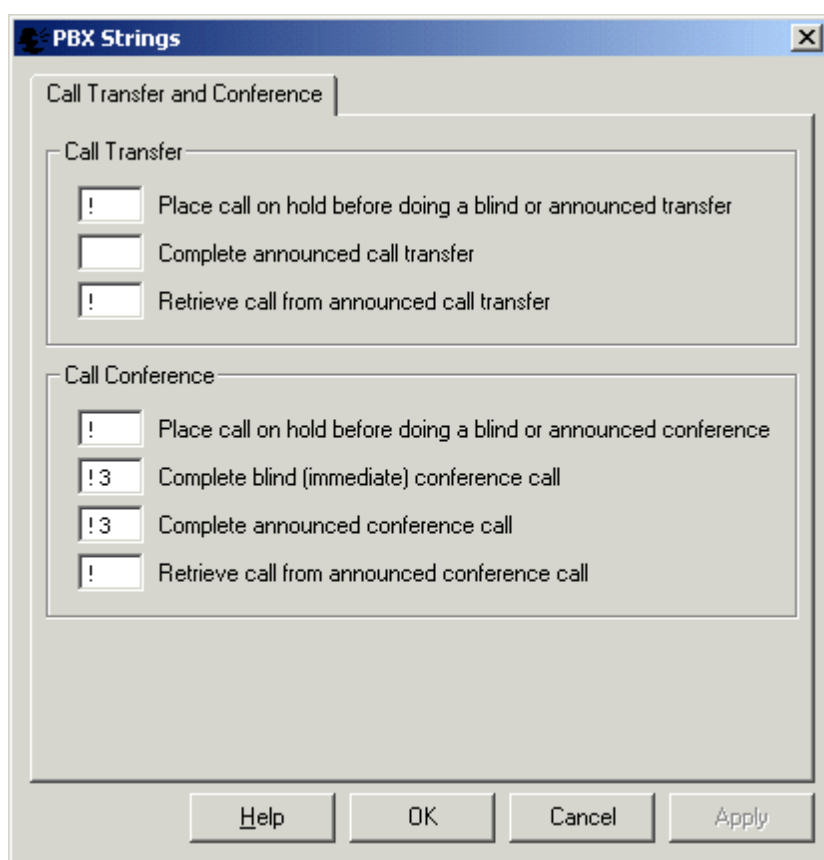
On almost all systems the 'hookflash' (also known as 'flash') signal is used in one way or another. Hookflash is just the action of hanging up the call on-hook for a very short time and then picking up the handset again. The length of the hookflash required differs from system to system - the length used by your Modem or Telephony Card will have to match that required by your PBX / Telephone Network, otherwise the transfers/conferences will not work. If the hookflash is set too short then it will not be noticed by the PBX (the caller will just hear a short click on the line) and if the hookflash is set too long then the PBX will interpret it as a receiver hanging up and will end the call (the caller will then hear just silence or a disconnect tone) and the extension dialed afterwards may be treated by the PBX/Switch as a new call. Note that if the PBX/Switch treats the dialed extension as a new call then the dialed extension number may ring. The time of ringing can be very short (if blind transfer is used) or until the call is answered or times out (if monitored or announced transfer is used).

Most modems cannot vary the length of their hookflash, and if their default length does not match the length required by your PBX / Telephone Network then you will not be able to use that modem to transfer/conference calls.

Telephony Cards allow the hookflash length to be set by the user – ensuring that a Telephony Card can be configured to allow it to successfully perform call transfers and conferencing. This is why it is highly recommended to use a Telephony Card if your application needs to do call transfers / conferencing.

Setting PBX control strings

In VoiceGuide Script Designer click on the **Edit** menu and select **PBX Command Strings**.



The signals sent to your PBX or Telephone Network can be configured here. VoiceGuide will generate the specified signals to command the PBX / Telephone Network to perform call transfers

and conferences.

The "!" character represents a hookflash.

The screen capture above shows a typical configuration which should work for many systems. You should confirm that these settings are OK with your PBX supplier or your Telephone Company, and change them if your PBX supplier or your Telephone Company indicates that they should be changed.

After changing the PBX strings VoiceGuide will need to be restarted to read in the new settings.

If hookflash transfer is not working

If you are encountering problems performing a hookflash transfer please try the following:

Step 1 : Is it possible to do a Hookflash transfer on that line ?

Try attaching a normal analog telephone handset to the line and see if you are able to perform the transfer by just pressing the 'Flash' button and then dialing the destination extension number. If the PBX/Switch does play the dial tone after the hookflash is pressed then it is possible to perform hookflash transfers on that line. After confirming the Switch/PBX react to a hookflash try dialing the transfer destination number and confirm if the dialed number starts ringing (the original caller should be on hold all this time - and usually hearing some on-hold music). Next see if the destination extension will keep ringing if you hang up the phone. If it does keep on ringing after the party performing the transfer hangs up then it looks like you are able to do a "Blind Hookflash Transfer". If the destination extension stops ringing when you hangup the phone then it looks like on this system the destination extension needs to be answered before the extension performing the transfer hangs up - so only monitored or Announced hookflash transfers will work.

If you have verified that it is possible to perform hookflash transfers on the line you are using then you can now move onto confirming if the telephony device used by VoiceGuide is issuing the correct length hookflash.

Step 2 : Is the hookflash length correct ?

Use the VoiceGuide script to answer the original call and use the Call Transfer module to attempt a call transfer. If the hookflash is set too short then it will not be noticed by the PBX (the caller will just hear a short click on the line and will not be put on hold) and if the hookflash is set too long then the PBX will interpret it as a receiver hanging up and will end the call (the caller will then hear just silence or a disconnect tone) and then when the hookflash finishes it will be interpreted by the PBX/Switch as the line going off-hook and treated by the PBX/Switch as a new call (not a transfer), with the extension dialed afterwards just treat as a the number dialed on this new call.

Step 3 : Is the PBX/Switch correctly receiving the transfer destination number ?

It is also possible that there is not enough delay between the hookflash and the dialed extension, in which case the leading dialed digit(s) may not be detected by the PBX/Switch, so the PBX/Switch will have an incomplete number to dial. VoiceGuide leaves enough of a pause between hookflash

and dialed number to not let this happen but if you think this may be occurring then you can add one or more commas before the extension number, like this:

,1234

Each comma usually represents a delay of around 2 seconds. You can also use a Play module to issue the hookflash followed pause(s) and extension number. To do a hookflash transfer to extension 1234 the following may be specified in a Play module: !,1234 or !,,1234 etc.

Setting Hookflash Length

When using VoiceGuide for Dialogic the hookflash length is set in Config.xml file, in the Parameters_DxBd section. The following section of Config.xml should be edited:

```
<Parameters_DxBd>
...
<Parameter>
  <Description>Flash time during dialing (10ms units)</Description>
  <Key>DXBD_FLASHTM</Key>
  <Value>10</Value>
  <Default>10</Default>
</Parameter> ...
</Parameters_DxBd>
```

A setting of 10 means 100ms. The setting is in 10ms units.

Setting Hookflash length on Dialogic cards using the .PRM file

Determine which parameter file is used by the Dialogic Configuration Manager and then change the hookflash length specified in that parameter file.

The release/installation notes which came with the card should inform you what Parameter File you should be specifying in the DCM - if you do not have the release notes then contact the supplier and they should be able to advise you which .PRM file to use.

If you have not specified the parameter file explicitly in the Dialogic DCM's "ParameterFile" setting then you can determine which parameter file is being used based on what "Country" is specified. The filename of the parameter file loaded for a D/4PCI is xx_d4p.prm where "xx" is based on what country is selected.

Eg: If country setting is "Australia/NZ" then the prefix is "an" and the parameter file for a D/4PCI is an_d4p.prm

You will need to change the line:

```
# PARAM 52:(DECIMAL WORD) 50 # Hook Flash/earth recall duration
```

to:

PARAM 52:(DECIMAL WORD) 10 # Hook Flash/earth recall duration

ie: remove the # at the beginning and change 50 to 10. Parameter is in 10ms units, so a setting of 10 will indicate 100ms.

Dialogic service must be restarted to read in the parameter file.

Databases

VoiceGuide can interact with any database or data source.
Any .NET Data Provider, ODBC Source or OLE DB can be used.

.NET and OLE DB Databases and Data Sources

Only a valid Connect String needs to be specified. No other configuration is needed.
A good resource for connection strings is: <https://www.connectionstrings.com/>

Configure ODBC Data Sources

If using the ODBC interface the ODBC driver should be specified in the Connect string, or an ODBC data source must be configured in Windows.

Note that for VoiceGuide v7 and later it is preferable to use the ADO.NET Data Provider, as this method is usually faster than using ODBC approach.

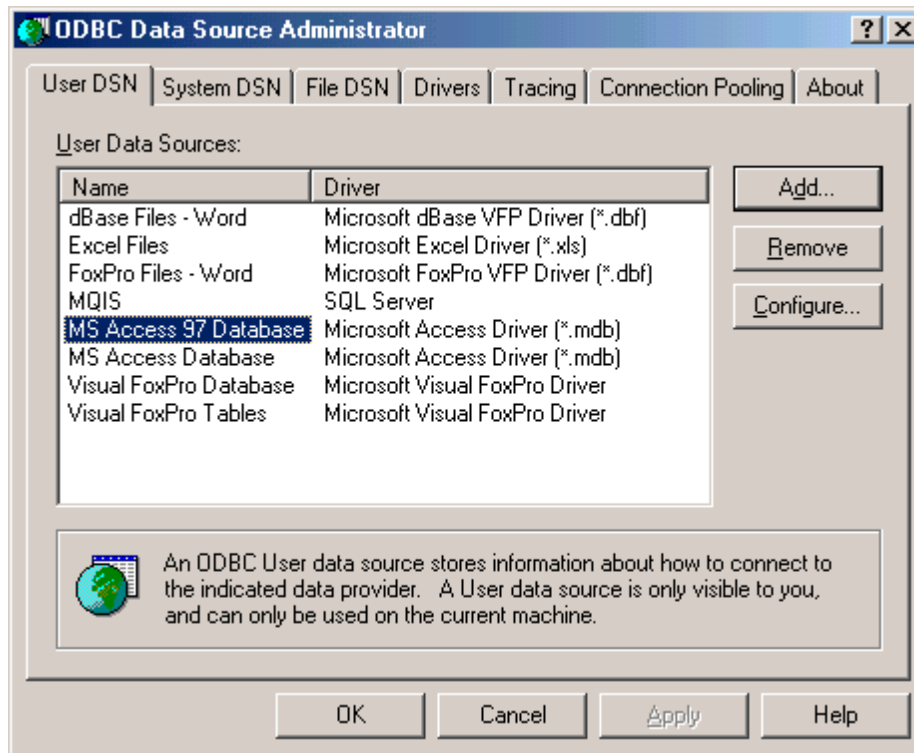
If using ODBC you must ensure that the 32-bit ODBC driver for your database has been installed. VoiceGuide is a 32-bit application and needs to use 32-bit ODBC drivers (not 64-bit ODBC drivers).

On 64-bit systems there are two ODBC Administrator apps. A 32-bit version and 64-bit version. Please read this: <http://support.microsoft.com/kb/942976> for information on these two versions

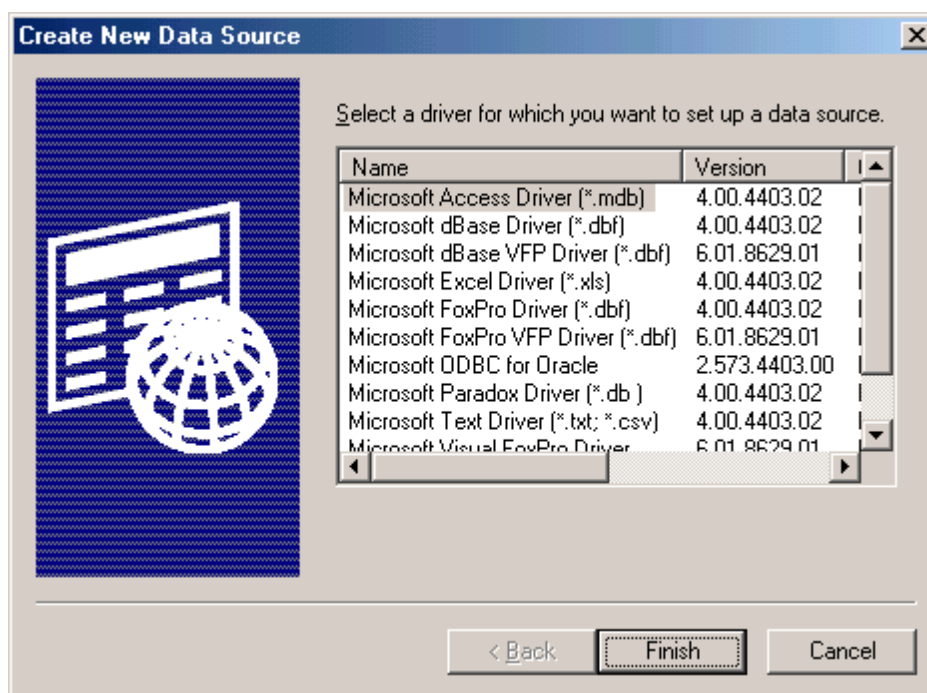
On 32-bit systems to open the ODBC Administrator from the Control Panel:

1. Click **Start**, point to **Settings**, and then click **Control Panel**.
2. Double-click **Administrative Tools**, and then double-click **Data Sources (ODBC)**.

The ODBC Data Source Administrator dialog box appears:

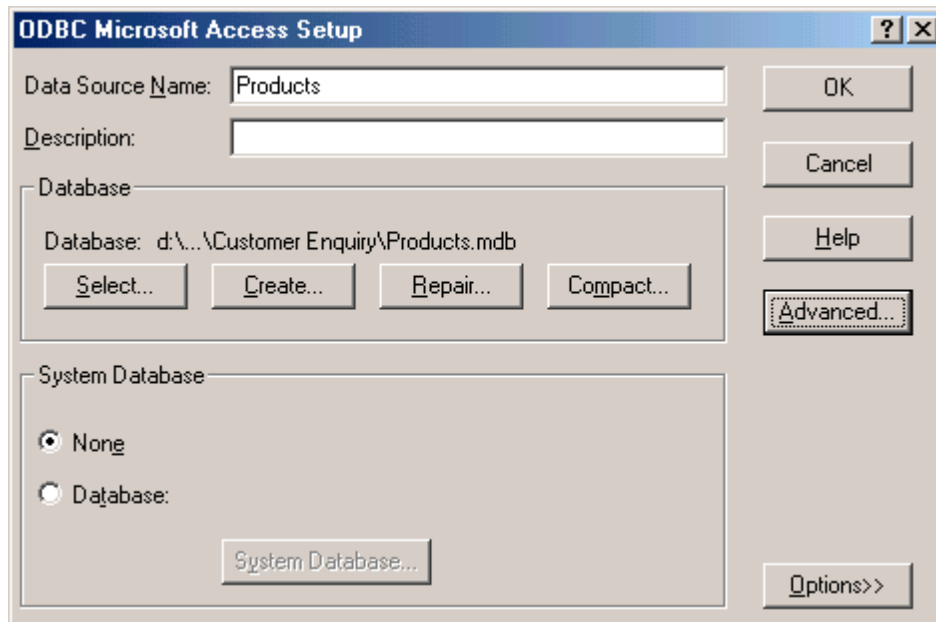


Select the 'System DSN' tab, and then click on the 'Add;' button to display the list of ODBC drivers installed on the system:



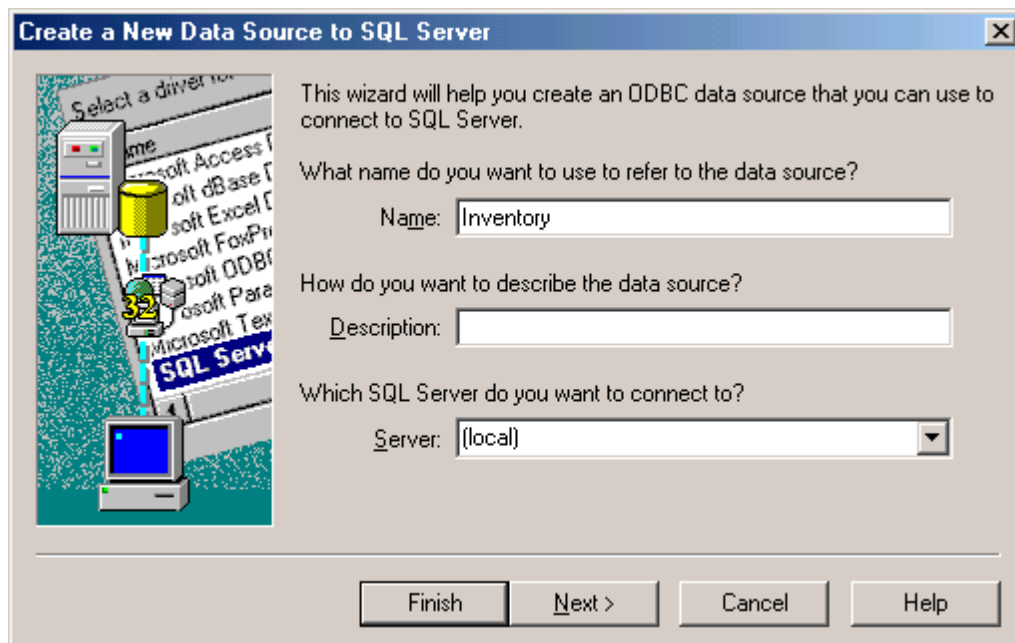
Select the driver which corresponds to the database you want to add, and press Finish. Your Database's ODBC driver should now take you through the rest of the Data Source setup process - the options presented in this stage will be different for each different database source. Please consult your database user's manual for information.

If your database's driver is not listed you should install your databases ODBC drivers. Please consult your database user's manual for information.

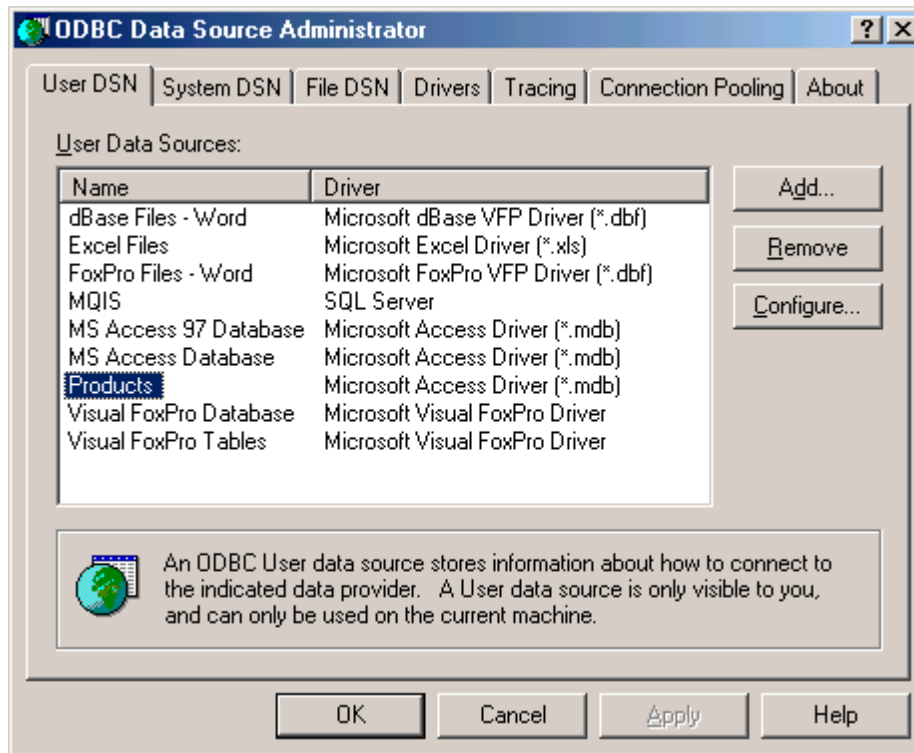


In the example above we selected an Access database Products.mdb and have called our Data Source 'Products'. Further parameters (eg database's password) can be set using the 'Advanced' options button.

Each different database type will have a different setup screen - for example the MS SQL Server ODBC setup screen looks like this:



After setting up our ODBC Data source the new source will appear in the User's list of Data Sources.



This ODBC Data Source is now ready to be used. The ["DB Query" module](#) can be used to access ODBC Data Sources from a VoiceGuide script.

End of Call Detection

On systems using **VoIP - SIP** or **T1/E1 ISDN** end of call is detectable by:

- [End of call message.](#)
- [Timeout awaiting input from caller.](#)

On systems using **Analog Telephony Cards** (eg: Dialogic D/4PCI) end of call is detectable by:

- [Loop Current Drop on the line.](#)
- [Disconnect tone on the line.](#)
- [Special DTMF tone played by PBX/Switch.](#)
- [Timeout awaiting input from caller.](#)

On systems using **Voice Modems** end of call detection is made by:

- [Disconnect tone on the line.](#)
- [Timeout awaiting input from caller.](#)

End of Call Message

VoIP and ISDN systems all use digital signalling to immediately indicate end of call when the other party hangs up the handset. This is the most reliable way of obtaining end of call signal. No special configuration is required in order for the system to detect the ISDN end of call signalling. VoiceGuide will detect the VoIP and ISDN signalling immediately and will end the script and hangup the line as well.

Loop Current Drop

Loop current drop is used by many telephone companies and PBXs to indicate end of call (other party has hung up). Telephony Cards can detect loop current drop and will inform VoiceGuide of it

- VoiceGuide will then immediately hang up as well. Please note that sometimes a loop current drop can be delayed and can arrive several seconds after caller has hung up.

Timeout awaiting input from caller

If there is a timeout awaiting a response from the caller then VoiceGuide will by default hangup

the call.

If however a 'timeout' or a 'fail' path has been set up in that module, then that path will be taken instead.

'Timeout' or 'fail' paths should not be used to advance script from one module that expects a response from caller to another module that again expects some response from caller, as this will just delay the system hanging up the call.

Disconnect Tone

A tone is usually played on the line when the caller has hung up. This tone is known as a 'busy tone' or a 'disconnect tone'. PBXs can be configured to play a tone, or send a DTMF tone (usually tone "D") to indicate end of call.

Detection of disconnect tones: Voice modems come pre-programmed with sets tones that they will detect as 'end-of-call' tones - which cannot be changed. Some voice modems detect end-of-call tones well, but some do not. Telephony cards also come pre-programmed with sets tones that they will detect as 'end-of-call' tones, but can also be be programmed what disconnect tones to listen for. This allows telephony card to have its end-of-call (disconnect) tone detection matched to those played on the line used.

Accuracy of tone detection with telephony cards is much better then with modems. Some modems do not detect tones and some falsely detect disconnect tones while playing/recording sound files or when a DTMF key is pressed by the caller.

Determining the Disconnect tone frequency and cadence :

Dialogic has the PBXpert application which automates the detection and setting of disconnect tones. This is a good tool to use if you have access to 2 lines attached to the Dialogic card.

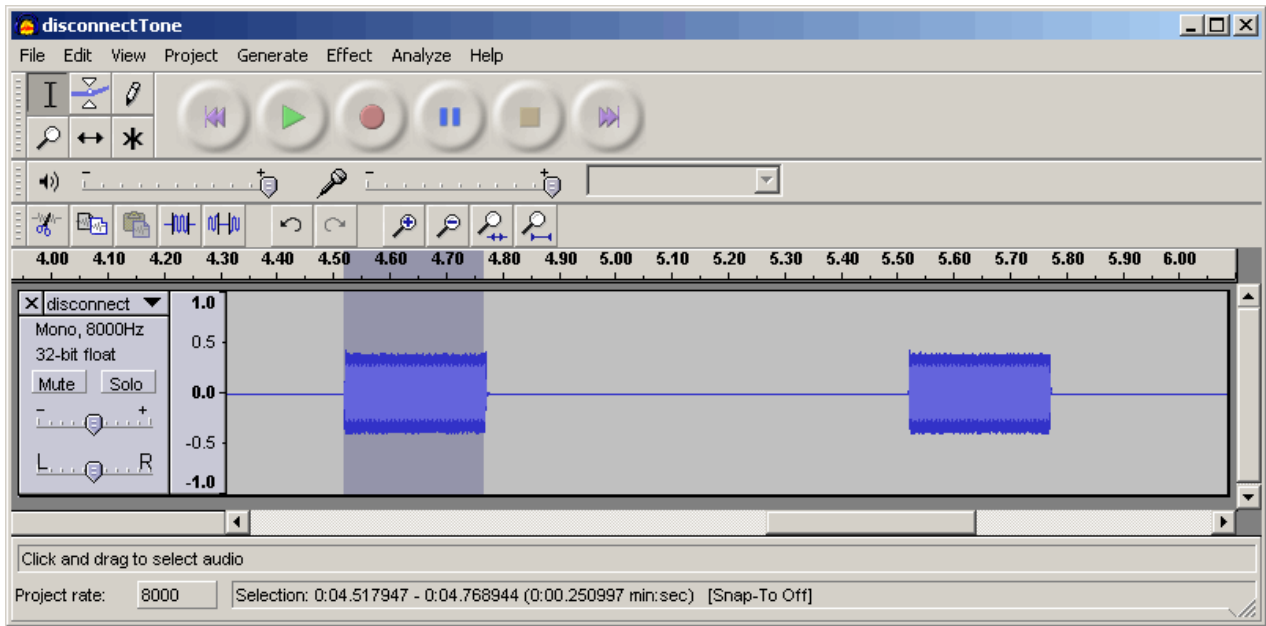
Otherwise please see instructions below for recording and determining frequencies when only a single analog line is available:

To find out the frequencies of the disconnect tone played by your telephone system just record it using VoiceGuide's Record module (just start recording then hang up) and then analyze the frequencies using any of the more advanced sound editors

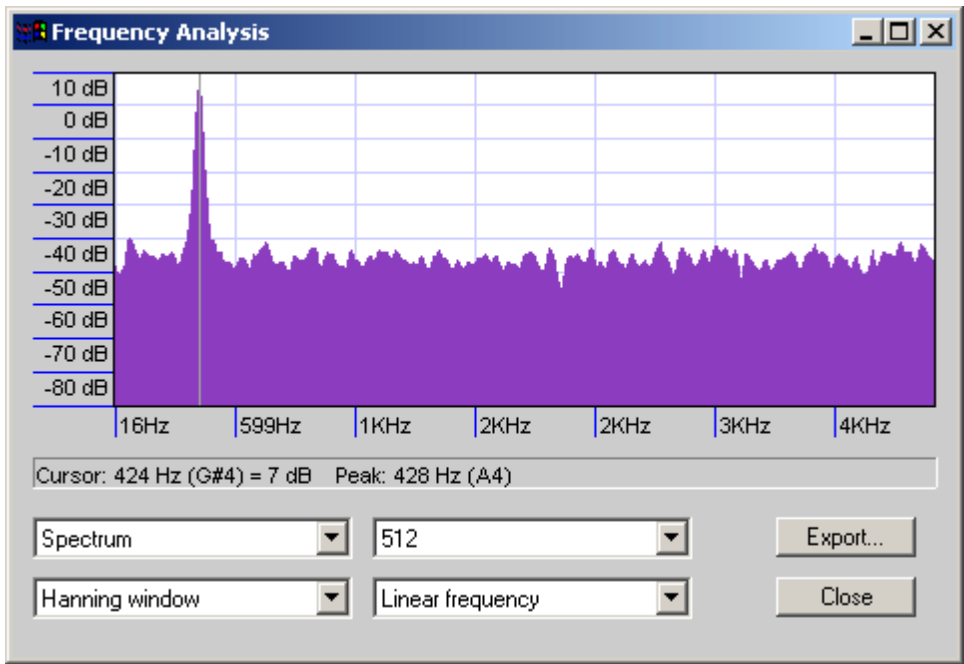
eg: when using Audacity (<http://audacity.sourceforge.net/>) :

1. Open the file in Audacity
2. Highlight the section of the sound file where the tone is present (the "ON" part of the tone)
3. Go to "View" menu and select "Plot Spectrum"
4. Note at what frequency the peak(s) is/are.
5. Highlight the entire section when tone is ON and check the interval, looking at the selection's timing data at bottom of Audacity's window.
6. Highlight the entire section when tone is OFF and check the interval, looking at the selection's timing data at bottom of Audacity's window.

Screenshot below shows Audacity with a disconnect tone with the ON part of the disconnect tone highlighted. The Selection interval timing is displayed at the bottom of the window - in this case the ON part of the tone lasts about 250milliseconds (0.250 seconds).

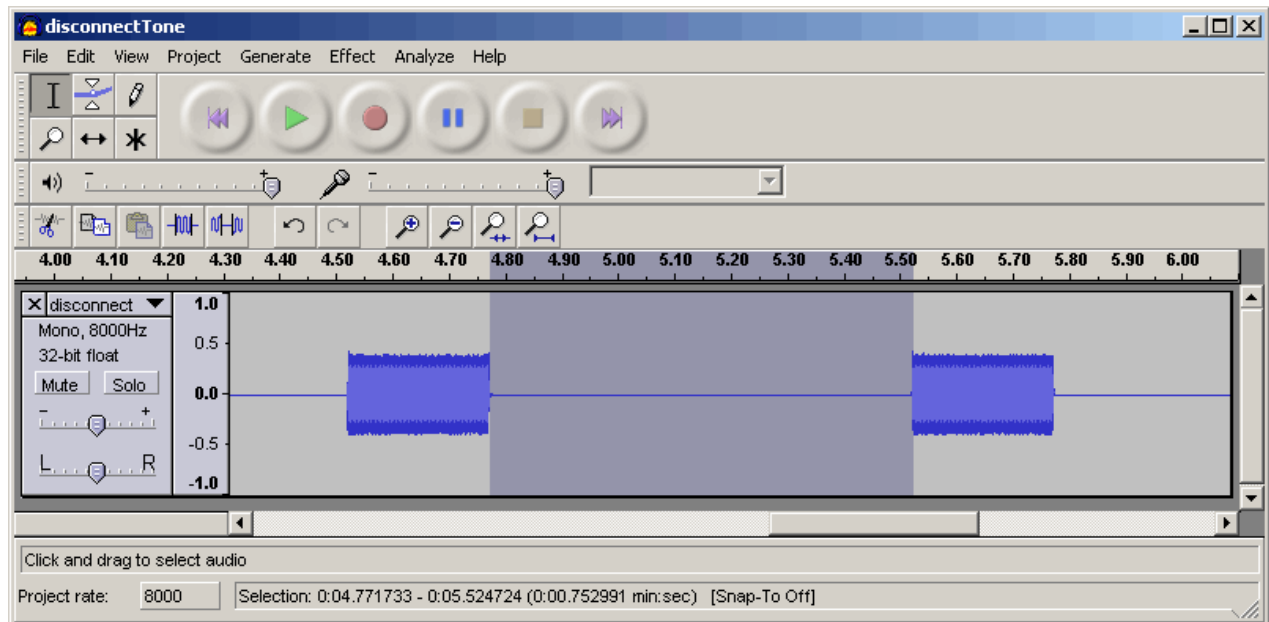


Below screenshot shows the Frequency analysis window which is shown when Audacity's "View"-> "Plot Spectrum" menu is selected.



The peak is at 428Hz.

Screenshot below shows Audacity with a disconnect tone with the OFF part of the disconnect tone highlighted. The Selection interval timing is displayed at the bottom of the window - in this case the OFF part of the tone lasts about 750milliseconds (0.750 seconds).



Based on the above information the disconnect tone in the example above is a single tone of 428Hz which lasts about 250ms and then a period of silence which lasts about 750ms.

The recommended settings for Dialogic cards to detect this tone reliably would be:

Freq1:	428
Freq1 dev:	40
Freq2:	0
Freq2 dev:	0
On time:	25
On time dev:	5
Off time:	75
Off time dev:	5
Repetition count:	3

When specifying the On/Off times (cadence times) for Dialogic the lengths of time are in units of 10milliseconds, so a value of 25 represents 250milliseconds.

For allowed variances (deviations) in frequency and time it's usually appropriate to use about 10% of the value for variance. For 'cadence' usually a value of 5 is OK (a value of 5 is 50milliseconds). If trying to detect a continuous tone which does not have ON/OFF cadence you should set the "Repetition Count" to be 0 and the OFF time to 0 as well.

VoiceGuide 7 and VoiceGuide 6 :

The Disconnect tone must be set in ConfigLine.xml file which can be found in VoiceGuide's \data\ subdirectory.

The Tone definitions which should be edited is the DISCONNECT_USER_1 or DISCONNECT_USER_2 (DISCONNECT_TAPI1 or DISCONNECT_TAPI2 in v6). The TID_DISCONNECT tone definition is only used on outbound calls, and it can also be changed if required.

An example entry in ConfigLine.xml:

```
<Tone Name="DISCONNECT USER 1">
  <Notes>Disconnect Tone</Notes>
```



```

<ID>DISCONNECT_USER_1</ID>
<Freq1>428</Freq1>
<Freq1Dev>50</Freq1Dev>
<Freq2>0</Freq2>
<Freq2Dev>0</Freq2Dev>
<On>25</On>
<OnDev>5</OnDev>
<Off>75</Off>
<OffDev>5</OffDev>
<Count>3</Count>
</Tone>

```

DTMF Disconnect tones :

Detection of DTMF disconnect tones : VoiceGuide is configured by default to react to DTMF key "D" as an indication of disconnect. Selection of which DTMF tone should be used by VoiceGuide as a disconnection tone can be made in the [PBX] section of the VG.INI file.

There have been a few good threads covering this on VoiceGuide's Support Forum:

<http://voiceguide.com/forums/index.php?showtopic=599>

<http://voiceguide.com/forums/index.php?showtopic=768>

In general Dialogic cards are pretty good at correctly detecting disconnection tones and if the tones are defined with low tolerance bands then the probability of false detections is low. If the tolerance bands however are set widely, allowing a wide range of tones and/or frequencies then false detections of disconnect tones can occur just while a person is speaking or a sound file is being played. As disconnect tone detections will result in the call being ended immediately it is imperative that the possibility of false detections is as low as possible.

One of the situations where disconnect tone definition parameters need to be set broadly is when a number of different disconnect tones needs to be detected. To setup the system to detect a variety of tones it would be necessary to record all the tones which you want the system needs to detect, analyze them with Audacity to find out the frequencies and timings and then come up with the one global setting which will be able to detect them all - keeping in mind that the broader the tolerances the higher the chance of false busy detections... In situations like these using T1/E1 ISDN lines would be a better solution.

NB. VoiceGuide allows for a different ConfigLine.xml file to be used for each channel - so each channel can have it's own tone configurations. Use <ConfigLine> tag within

the <Channel> entry to specify the LinesConfig.xml file specific for that channel. eg:

```
<Name>dxxxBlCl</Name>
<Protocol>pdk_na_an_io</Protocol>
<RingsBeforeAnswer>0</RingsBeforeAnswer>
<script>C:\Scripts\MyScript.vgs</Script>
<AllowDialOut>1</AllowDialOut>
<ConfigLine>C:\MyConfigs\ConfigForUsCalls.xml</ConfigLine>
</Channel>
```

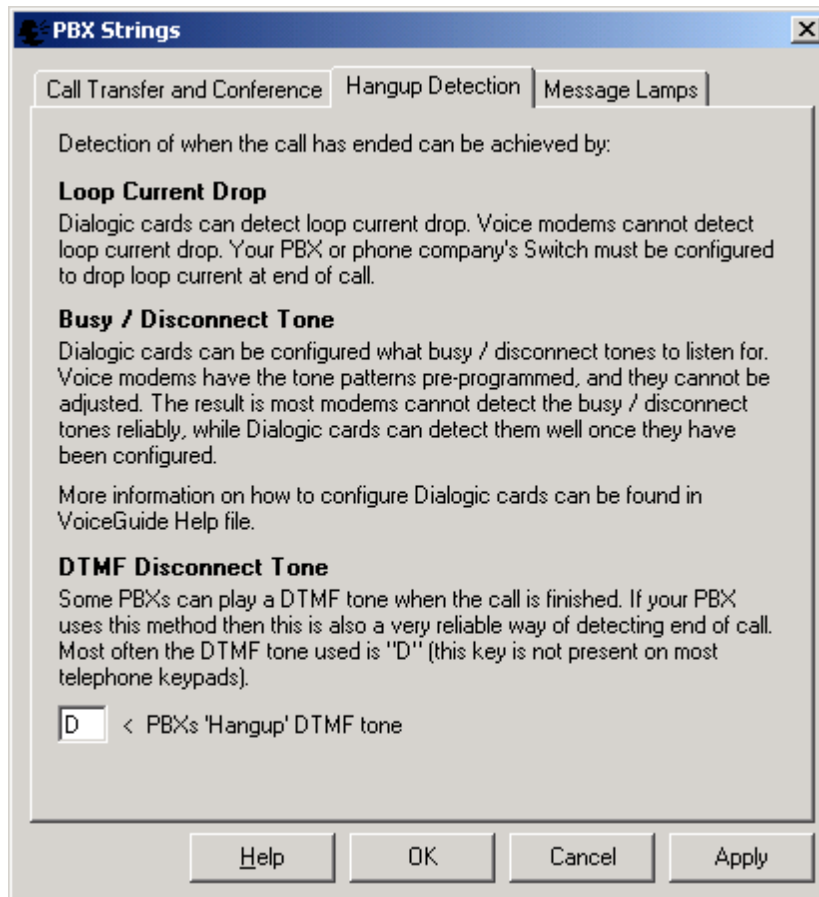
End of Call Detection: Special DTMF tone played by PBX/Switch

Some PBXs and Switches can be set up to play a DTMF tone on the line when the PBX/Switch detects that the other party has hung up. The DTMF tone chosen to indicate that is usually the "D" tone.

Most telephone handsets do not have the DTMF A, B, C and D tones on their keypad, so the callers cannot generate these tones themselves (either intentionally or by mistake). This makes the "DTMF Disconnect Tone" a very useful way of detecting end of call.

If the PBX or Switch is capable of generating DTMF tone at end of call then we highly recommend using this approach as it is a very reliable way of informing VoiceGuide immediately when the call has finished.

The DTMF Disconnect Tone for which VoiceGuide will listen and use as an indication of end of call can be set by selecting "PBX Command Strings" from the "Edit" menu in VoiceGuide Script Designer:



Distinctive Ring Detection

Some traditional analog phone lines (POTS lines) can be set up so that calls to 2 or more telephone numbers will be sent to them, with the recipient being able to tell which number was dialed by the different ring tone played when calls to different numbers arrive. These different ring types are called "Distinctive Ringing"

VoiceGuide can be setup to detect the different type of Ringing and make this information available to the script.

For VoiceGuide to detect the different rings, the cadences of the rings need to be specified in a ConfigLine.xml file. Here is the relevant section of that file with one ring tone defined.

```
<DistinctiveRings>
  <DistinctiveRing>
    <Name>TestPBXDoubleRing</Name>
    <ID>TestPbxStandard</ID>
    <Description></Description>
    <Notes></Notes>
    <Cadence>
      <On>55</On>
      <OnDev>10</OnDev>
      <Off>55</Off>
      <OffDev>10</OffDev>
    </Cadence>
    <Cadence>
      <On>55</On>
      <OnDev>10</OnDev>
      <Off>285</Off>
      <OffDev>10</OffDev>
    </Cadence>
  </DistinctiveRing>
</DistinctiveRings>
```

The time intervals are all specified in 10ms units. The above example shows the definition of a ring that is on for 550ms, the off for 550ms, then on again for 550ms and off for 2850ms, with the pattern repeating afterwards. This definition is loaded by VoiceGuide at startup, and incoming Ring signals (on analog lines) are compared to see if they match any of the definitions.

If a ring matches one of the definitions then the \$RV[DISTINCT_RING] variable is set to hold the Name value of the ring definition entry. eg. if the ring matches the example definition above then VoiceGuide will set \$RV[DISTINCT_RING] to hold a value of: TestPBXDoubleRing also \$RV [DISTINCT_RING_TestPBXDoubleRing] will be set to hold a value of True

Note that VoiceGuide must be set to answer at the beginning of 3rd ring or later in order to be able to hear enough rings to match up their cadence with the example sequence. The 'answer after X rings' setting must always be at least 1 more than the number of Cadence entries in the DistinctiveRing definition.

Confirming the Distinct Ring cadence is loaded properly

Here is the relevant extract from the VoiceGuide log file showing the details of the cadence loaded by Voiceguide at startup.

```
074958.48 0 init load Distinct Ring definitions start [C:\Projects\vg32
\data\ConfigLine.xml]
074958.49 0 init distinct ring definitions found:1
074958.49 0 init ring 1 [TestPBXDoubleRing] total cadences:2
074958.49 0 init distinct ring 1 [TestPBXDoubleRing] cadence 1 [55:10,55:10][110:20]
074958.49 0 init distinct ring 1 [TestPBXDoubleRing] cadence 2 [55:10,285:10][340:20]
074958.49 0 init load Distinct Ring definitions end
```

Relevant log section showing matching incoming ring

Here is the relevant extract from the VoiceGuide log file showing the details of the incoming call, the ring timings and VoiceGuide matching them against loaded cadence templates. At the end of the trace we can see VoiceGuide matched the cadence against the definition of ring tone, and has added \$RV[DISTINCT_RING] to that line's RV set.

Note that with Dialogic cards the cadences are matched using ring rising edge timings. In the example below the rising edge of the second ring arrived 1200ms seconds after the rising edge of first ring, and the 3rd ring arrived 3430ms seconds after the rising edge of the second ring. This fell within the allowed 1st cadence of 900ms-1300ms and the allowed second cadence of 3200ms to 3600ms.

```
112418.28 1 ring 1
112418.28 1 ring time since last ring event (sec): 0.00
112418.28 1 rings=1, min rings before answer=4 (iCallerIdHasArrived=0)
112418.28 1 tw DialogicEvent 134,TDX_CST,0,0,0,DE_RINGS,ET_RON,
112418.28 1 event ScriptEventCode TDX_CST, code=134, state=0
112419.48 1 ring 2
112419.48 1 ring time since last ring event (sec): 1.20
112419.48 1 ring match against pattern 1, cadences=1
112419.48 1 ring template 1, cadence 1 = [110|20] edge time=1203.3
112419.48 1 ring match distinct ring pattern 1 cycle 1 ok
112419.48 1 rings=2, min rings before answer=4 (iCallerIdHasArrived=0)
112419.48 1 tw DialogicEvent 134,TDX_CST,0,0,0,DE_RINGS,ET_RON,
112419.48 1 event ScriptEventCode TDX_CST, code=134, state=0
112422.92 1 ring 3
112422.92 1 ring time since last ring event (sec): 3.43
112422.92 1 ring match against pattern 1, cadences=2
112422.92 1 ring template 1, cadence 1 = [110|20] edge time=120.3
112422.92 1 ring match distinct ring pattern 1 cycle 1 ok
112422.92 1 ring template 1, cadence 2 = [340|20] edge time=343.4
112422.92 1 ring match distinct ring pattern 1 cycle 2 ok
```

```
112422.92 1 ring full match ring pattern 1 (2 cadences)
112422.92 1 rv add [DISTINCT_RING]{TestPBXDoubleRing}
```

The ring timing marks : "ring time since last ring event" allow you to examine any new ring types' cadences and set the Distinctive Ring definitions accordingly.

Note that with Dialogic cards the cadences are matched using ring rising edge timings, so it is not really essential to correctly set the ON and OFF values, as long as the sum of the ON and OFF values within the cadence matches the time to the next ring rising edge. The <OnDev> and <OffDev> values are also summed to give the overall allowed time deviation.

Cisco Call Manager Configuration

The simplest way to have Cisco send calls VoiceGuide IVR system is to set up a trunk within Cisco that routes calls directly to the IVR system.

The "Trunk type" should be set to "SIP Trunk"

The "Device Protocol" should be set to "SIP".

eg:

Then in the Trunk Configuration screen you will need to:

Select a Device Pool and set Media Resource Group List (Media Resource Group List needs to exist)

And in the SIP Information section:

In the "Destination Address" field enter the IP address of your VoiceGuide IVR server,

In the "Destination Port" field enter "5060"

In the Preferred Codec field select "711ulaw"

In the "SIP Trunk Security Profile" select "Non Secure SIP Trunk Profile"

In the "SIP Profile" select "Standard SIP Profile"

Also, it is necessary to:

Select "UDP" as the "Outgoing Transport Type" (see System > Security Profile > SIP Trunk Security Profile menu):

Cisco Unified CM Administration
For Cisco Unified Communications Solutions

Navigation: Cisco Unified CM Administration Go
administrator About Logout

System Call Routing Media Resources Voice Mail Device Application User Management Bulk Administration Help

SIP Trunk Security Profile Configuration Related Links: Back To Find/List Go

Save

Status
Status: Ready

SIP Trunk Security Profile Information

Name*
Description

Device Security Mode Non Secure

Incoming Transport Type* TCP+UDP

Outgoing Transport Type TCP

☐ Enable Digest Authentication

Nonce Validity Time (mins)* 600

X.509 Subject Name

Incoming Port* 5060

☐ Enable Application Level Authorization

☐ Accept Presence Subscription

☐ Accept Out-of-Dialog REFER

☐ Accept Unsolicited Notification

☐ Accept Replaces Header

Save

* indicates required item.

Change This From TCP to UDP

WireShark can be used to verify that sip packets are arriving at VoiceGuide IVR server.

To filter for SIP packets specify:

sip

in the WireShark's filter text box.

T1/E1 ISDN Configuration

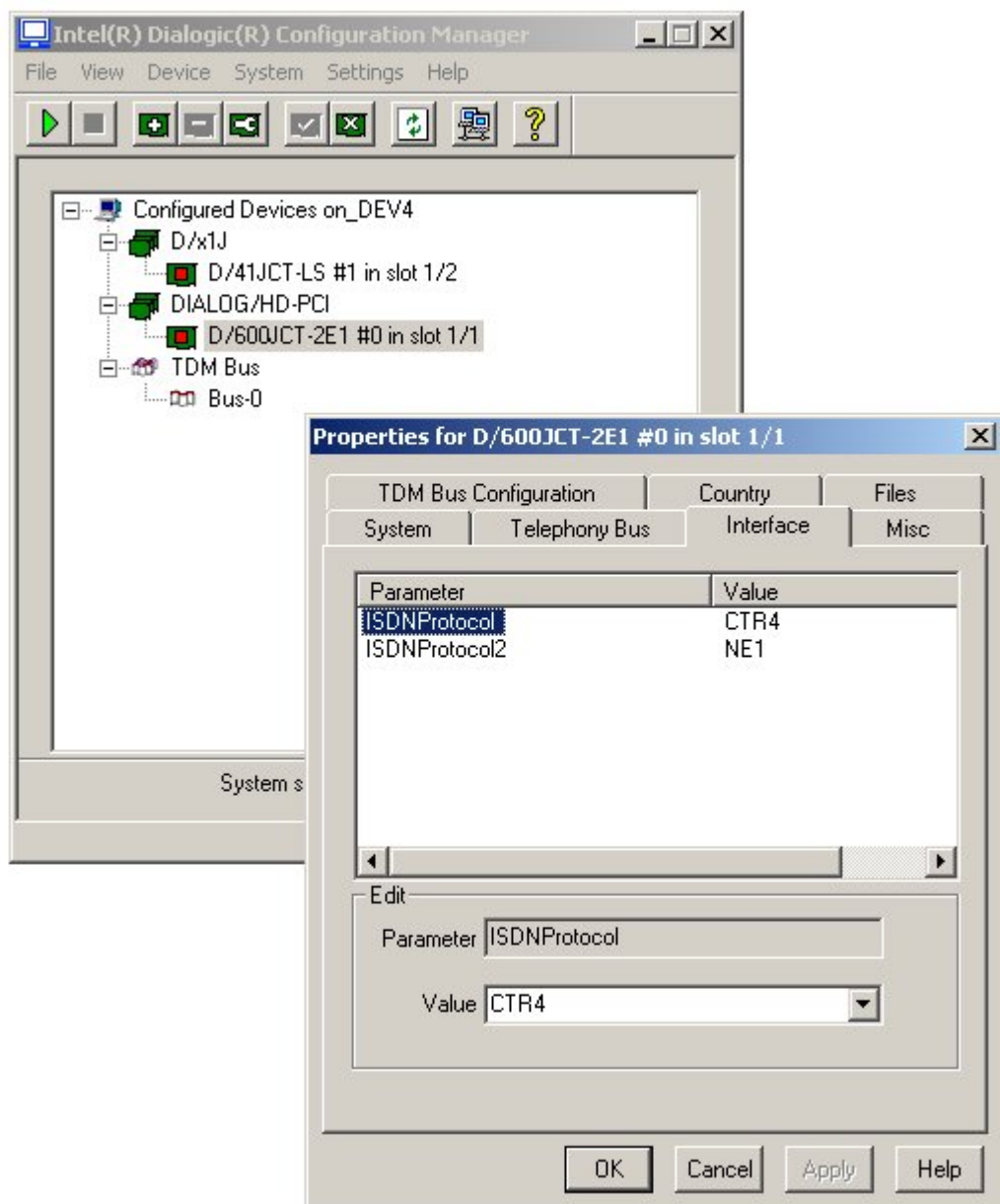
Installing and Starting Dialogic T1/E1 cards

Please refer to installation instructions supplied with your card, and contact the card supplier if you encounter any issues starting the Dialogic service.

It is recommended that a working T1/E1 trunk is attached to the card's RJ45 port before starting the Dialogic service. Otherwise the Dialogic service may not start. A 'loopback' connector may often be used instead of the working T1/E1 trunk.

Selecting ISDN Protocol

The appropriate ISDN protocol in the Dialogic Configuration Manager (DCM) system needs to be selected. After selecting the appropriate ISDN protocol, and starting the Dialogic service the system should now be able to receive calls. Screenshot below shows where in the DCM the ISDN protocol is set for a JCT card:



The telephone company or PBX/Switch administrators will advise which ISDN protocols are available on the trunks they supply.

It is recommend that calls are presented on the trunk (or set of trunks) in a 'round robin'/'circular' order. This would result in the incoming calls being evenly distributed across all available channels.

Usually the Dialogic cards are expected to be set up in 'User' mode.

Below table shows what protocol selection should be made for Dialogic JCT series cards:

T1 ISDN Protocol	Mode: User or Network	ISDN Protocol in JCT
QSIG-T1	User	QTU
QSIG-T1	Network	QTN
NI2	User	NI2
NI2	Network	NT1
4ESS	User	4ESS
4ESS	Network	NT1
5ESS	User	5ESS

5ESS	Network	NT1
DMS	User	DMS
DMS	Network	NT1

E1 ISDN Protocol	Mode: User or Network	ISDN Protocol in JCT
NET5 (Euro-ISDN)	User	CTR4
NET5 (Euro-ISDN)	Network	NE1
QSIG-E1	User	QTE
QSIG-E1	Network	QNT

Testing ISDN Line

The Dialogic ISDIAG.EXE application can be used to test the T1/E1 lines and obtain ISDN traces.

When using the "DMV" family cards. eg: DMV480, DMV600, etc. the Dialogic ISDNTRACE.EXE application can be used obtain ISDN traces.

Any ISDN traces captured can be converted into readable format using Dialogic's ISDTRACE.EXE utility.

ISDIAG

Dialogic provides the `isdiag.exe` command line application which will allow you to test if the ISDN connection is functioning and can capture the ISDN layer trace as well. `Isdiag.exe` will only work on 'Springware' Dialogic cards, ie. 'JCT' series cards. `Isdiag.exe` is located in directory `C:\Program Files\Dialogic\bin`

To test an E1 connection the `isdiag.exe` would usually be started like this at the DOS Command Prompt:

```
isdiag 1 1 e s v
```

To test a T1 connection the `isdiag.exe` would usually be started like this at the DOS command prompt:

```
isdiag 1 1 t s v
```

The examples above would allow user to receive and make calls on that first channel of first T1/E1 interface only. You will need to confirm with the Telco that they will send a call on that channel after the ISDN D channel is restarted.

ISDN's D channel is restarted when `isdiag` is restarted, or when Dialogic Service is restarted.

Screenshot below shows menu displayed when `isdiag` is started:

```
Administrator: Command Prompt - isdiag 1.1.0.0

MAIN MENU
1. set call parameters
2. request calling party number<ANI>
3. send maintenance request
4. display information
5. drop call
6. make call
7. play/record/dial
8. set info
9. send message
10. start/stop/browse trace
11. restart and waitcall
12. shell to DOS
13. change current channel
14. Hold/Retrieve Call
15. dpnss supp service
ESC exit
Please enter choice: _

ISDIAG Version Package 3 - Beta 1.00
LOS OOF RAI AIS CRC R/E DCH
0 0 0 0 0 0 0

F1. help menu
```

Isdiag displays current state of ISDN trunk at top right. All the error flags should be "0".

eg. if the number under DCH is "1" this indicates a D channel error (D channel not present).

VoiceGuide should not be started when using isdiag.exe, as isdiag.exe itself opens the channel on the ISDN trunk.

If you are unable to receive/make calls and play/record sound files with isdiag then you will need to contact your Dialogic supplier to establish why the hardware they provided does not work on your T1/E1 trunks. It is unlikely that VoiceGuide will work if isdiag cannot receive and make calls.

isdiag.exe can be used to capture traces of the ISDN messages as well. The log files produced by isdiag.exe need to be converted into human-readable form by using the isdtrace.exe application.

ISDNTRACE

Dialogic provides the ISDNTRACE.EXE application which can capture a trace of the ISDN messages. Isdntrace will only work on DM3/DMV type Dialogic cards. Isdntrace is located in Dialogic's \bin\ subdirectory.

The ISDN traces are used to confirm what ISDN parameters are sent by the Telco on that ISDN line, and to confirm what parameters are being sent out by VoiceGuide when an outbound call is made. The outgoing parameters should usually match what is sent by the Telco's switch.

To use isdntrace to obtain the trace of the ISDN layer messages please follow the following steps.

- 1. Stop VoiceGuide.
- 2. Restart Dialogic Service.
- 3. Start isdntrace using this command .

```
isdntrace -b0 -dl -f c:\myisdnlog
```

4. Start VoiceGuide.
5. Make a call into VoiceGuide. Hangup after VoiceGuide answers.
6. Make a call out of VoiceGuide, loading the call using the "Outbound Call Loader".
7. After calls is ended press "q" in the `isdntrace` Command Prompt window to exit `isdntrace`.
8. Trace file would be created in C:\, name of file begins with "myisdnlog".

Configuring VoiceGuide

The channels which VoiceGuide is to use are specified in Config.xml file (see VoiceGuide's \conf\ subdirectory).

Sample Config.xml files are provided, and cover the most common deployment scenarios.

In the Config.xml file the Protocol just needs to be set to `ISDN`

It is recommended that all channels are opened: 23 on a T1 ISDN line, and 30 on an E1 ISDN line.

Some PBXs and Switches will not properly handle the calls unless all channels on the line are opened.

A sample ConfigLine.xml file for use with ISDN systems is also provided. Analog system related settings have been removed from that file, and ISDN related settings have been added in.

ISDN Outgoing Calls

Setting the ISDN protocol to the correct value is usually sufficient for outgoing calls to function as well. Some telephone companies however have specific non-standard requirements for outgoing calls.

This section outlines the configuration options available within VoiceGuide to allow the user to explicitly specify the individual settings on outgoing calls.

Most of these requirements relate to the structure and the information contained within the "Setup" message sent to the switch when a new call is made. VoiceGuide offers extensive control over the information sent within the Setup message, allowing easy configuration of the system to match requirements of the Telco switch or PBX.

The Setup message parameters are configurable using the ConfigLine.xml file, in section `<ConfigLine><isdn><msgSetup>`.

Below is an example of what this section looks like, with the default values set.

```

<isdn>
<msgSetup>
  <defsrc>NONE</defsrc>
  <defcc>
    <cc_bc_xfer_cap>BEAR_CAP_SPEECH</cc_bc_xfer_cap>
    <cc_bc_xfer_mode>ISDN_ITM_CIRCUIT</cc_bc_xfer_mode>
    <cc_bc_xfer_rate>BEAR_RATE_64KBPS</cc_bc_xfer_rate>
    <cc_usrinfo_layer1_protocol>ISDN_UI11_G711ULAW</cc_usrinfo_layer1_protocol>
    <cc_usr_rate>ISDN_NOTUSED</cc_usr_rate>
    <cc_destination_number_type>NAT_NUMBER</cc_destination_number_type>
    <cc_destination_number_plan>UNKNOWN_NUMB_PLAN</cc_destination_number_plan>
    <cc_destination_sub_number_type>OSI_SUB_ADDR</cc_destination_sub_number_type>
    <cc_origination_number_type>NAT_NUMBER</cc_origination_number_type>
    <cc_origination_number_plan>UNKNOWN_NUMB_PLAN</cc_origination_number_plan>
    <cc_origination_phone_number></cc_origination_phone_number>
    <cc_origination_sub_number_type>OSI_SUB_ADDR</cc_origination_sub_number_type>
    <cc_origination_sub_phone_number></cc_origination_sub_phone_number>
    <cc_facility_feature_service>ISDN_NOTUSED</cc_facility_feature_service>
    <cc_facility_coding_value>ISDN_NOTUSED</cc_facility_coding_value>
  </defcc>
  <defgc>
    <gc_destination_address></gc_destination_address>
    <gc_destination_address_type>NAT</gc_destination_address_type>
    <gc_destination_address_plan>UNKNOWN</gc_destination_address_plan>
    <gc_destination_sub_address></gc_destination_sub_address>
    <gc_destination_sub_address_type>UNKNOWN</gc_destination_sub_address_type>
    <gc_destination_sub_address_plan>UNKNOWN</gc_destination_sub_address_plan>
    <gc_origination_address></gc_origination_address>
    <gc_origination_address_type></gc_origination_address_type>
    <gc_origination_address_plan></gc_origination_address_plan>
    <gc_origination_sub_address></gc_origination_sub_address>
    <gc_origination_sub_address_type></gc_origination_sub_address_type>
    <gc_origination_sub_address_plan></gc_origination_sub_address_plan>

    <gc_chan_info_medium_id>1</gc_chan_info_medium_id>
    <gc_chan_info_medium_sel>MEDIUM_PREF</gc_chan_info_medium_sel>

    <gc_call_info_category>SUB_NOPRIOR</gc_call_info_category>
    <gc_call_info_address_info>ENBLOC</gc_call_info_address_info>
    <gc_ext_data></gc_ext_data>
  </defgc>
</msgSetup>
</isdn>

```

The <defsrc> setting selects whether the "CC" or "GC" approach (or neither) will be used to configure Setup message. Both approaches set almost the same information, but some Dialogic cards require the use of CC approach and some require the GC approach, which is why the option to use one or the other is offered. Setting the <defsrc> filed to NONE should always be tried first. This will result in the default settings for the selected ISDN protocol being used - the protocol set in Dialogic's Configuration Manager (DCM).

The DM3/DMV series cards can only use the "GC" setting parameters.

Settings used for individual fields within the <ConfigLine><isdn><msgSetup> section are detailed below :

Valid Settings

Field	Values
defsrc	CC, GC, NONE
cc_bc_xfer_cap	BEAR_CAP_SPEECH, BEAR_CAP_UNREST_DIG, BEAR_REST_DIG
cc_bc_xfer_mode	ISDN_ITM_CIRCUIT
cc_bc_xfer_rate	BEAR_RATE_64KBPS, BEAR_RATE_128KBPS, BEAR_RATE_384KBPS, BEAR_RATE_1536KBPS, BEAR_RATE_1920KBPS, PACKET_TRANSFER_MODE
cc_usrinfo_layer1_protocol	SDN_UIL1_CCITTV.110, ISDN_UIL1_G711ULAW, ISDN_UIL1_G711ALAW, ISDN_UIL1_G711ADPCM, ISDN_UIL1_G722G725, ISDN_UIL1_H261, ISDN_UIL1_NONCCITT, ISDN_UIL1_CCITTV120, ISDN_UIL1_CCITTX31
cc_usr_rate	ISDN_UR_EINI460, ISDN_UR_56000, ISDN_UR_64000, ISDN_UR_134, ISDN_UR_12000
cc_destination_number_type	EN_BLOC_NUMBER, INTL_NUMBER, NAT_NUMBER, LOC_NUMBER, OVERLAP_NUMBER
cc_destination_number_plan	UNKNOWN_NUMB_PLAN, ISDN_NUMB_PLAN, TELEPHONY_NUMB_PLAN, PRIVATE_NUMB_PLAN
cc_destination_sub_number_type	OSI_SUB_ADDR, USER_SPECIFIED_SUB_ADDR, IA_5_FORMAT
cc_origination_number_type	EN_BLOC_NUMBER, INTL_NUMBER, NAT_NUMBER, LOC_NUMBER, OVERLAP_NUMBER
cc_origination_number_plan	UNKNOWN_NUMB_PLAN, ISDN_NUMB_PLAN, TELEPHONY_NUMB_PLAN, PRIVATE_NUMB_PLAN
cc_origination_phone_number	Default CallerID phone number for outgoing calls made using this channel can be specified here.
cc_origination_sub_number_type	OSI_SUB_ADDR, USER_SPECIFIED_SUB_ADDR, IA_5_FORMAT
cc_origination_sub_number	Default CallerID phone sub-number for outgoing calls made using this channel can be specified here
cc_facility_feature_service	ISDN_NOTUSED, ISDN_FEATURE, ISDN_SERVICE
cc_facility_coding_value	ISDN_CPN_PREF, ISDN_SDN, ISDN_BN_PREF, ISDN_ACCUNET, ISDN_LONG_DIS, ISDN_INT_800, ISDN_CA_TSC, ISDN_ATT_MULTIQ
gc_destination	TRANSPARENT, NAT, INTL, LOC, IP, URL, DOMAIN, EMAIL

<code>_address_type</code>	
<code>gc_destination</code> <code>_address_plan</code>	UNKNOWN, ISDN, TELEPHONY, PRIVATE
<code>gc_destination</code> <code>_sub_address_t</code> <code>ype</code>	UNKNOWN, OSI, USER, IA5
<code>gc_destination</code> <code>_sub_address_p</code> <code>lan</code>	UNKNOWN
<code>gc_origination</code> <code>_address</code>	Default CallerID phone number for outgoing calls made using this channel can be specified here.
<code>gc_origination</code> <code>_address_type</code>	TRANSPARENT, NAT, INTL, LOC, IP, URL, DOMAIN, EMAIL
<code>gc_origination</code> <code>_address_plan</code>	UNKNOWN, ISDN, TELEPHONY, PRIVATE
<code>gc_origination</code> <code>_sub_address</code>	Default CallerID phone sub-number for outgoing calls made using this channel can be specified here.
<code>gc_origination</code> <code>_sub_address_t</code> <code>ype</code>	UNKNOWN, OSI, USER, IA5
<code>gc_origination</code> <code>_sub_address_p</code> <code>lan</code>	UNKNOWN
<code>gc_chan_info_m</code> <code>edium_id</code>	Specifies the timeslot to be connected.
<code>gc_chan_info_m</code> <code>edium_sel</code>	MEDIUM_PREF, MEDIUM_EXCL
<code>gc_call_info_c</code> <code>ategory</code>	SUB_NOPRIOR, SUB_PRIOR, MAINT_EQUIP, COIN_BOX, OPERATOR, DATA, CPTP, SPECIAL, MOBILE, VPN
<code>gc_call_info_a</code> <code>ddress_info</code>	ENBLOC, OVERLAP

VoiceGuide ISDN Tracing

VoiceGuide can also initiate a trace the ISDN D-channel messages.

A file titled `ktTelControl_TraceSelect.txt` needs to be placed in "C:\\" with the contents of the file indicating which Dialogic's board D-channel is to be traced.

Example contents should be:

```
isdn_trace_dtiB1
```

which indicates which interface board should have it's D-channel traced.

VoiceGuide will create the traces in "C:\\", with the trace files named in this format:

```
isdn_trace_dtiB1_0325_203334.log
```

Traces can be converted into readable format using Dialogic's ISDTRACE.EXE utility.

T1/E1 RobbedBit/CAS/R2 Configuration

Installing and Starting the Dialogic T1/E1 cards

Please refer to installation instructions supplied with your card, and contact the card supplier if you encounter any issues starting the Dialogic service.

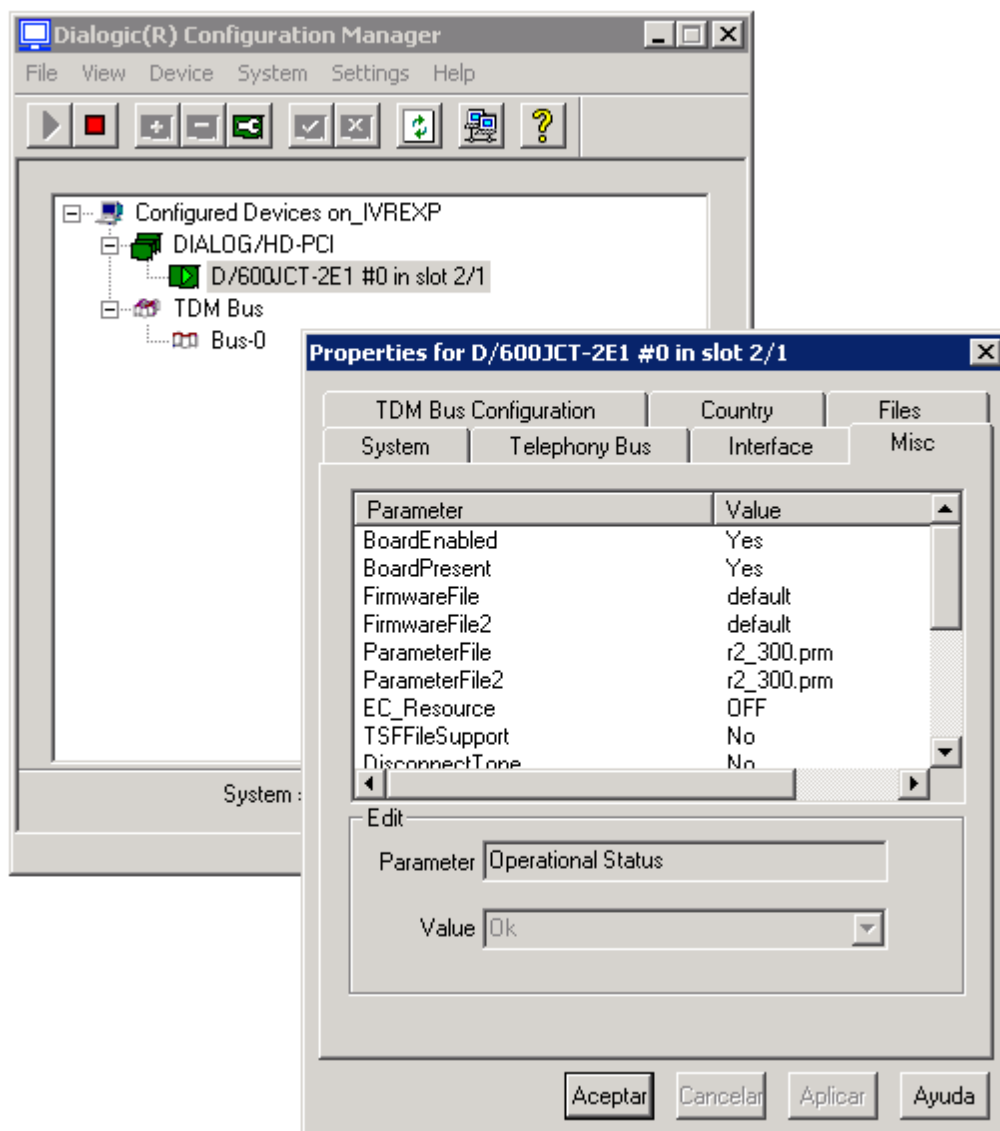
It is recommended that a working T1/E1 trunk is attached to the card's RJ45 port before starting the Dialogic service. Otherwise the Dialogic service may not start. A 'loopback' connector may often be used instead of the working T1/E1 trunk.

JCT series Dialogic cards

To configure the Dialogic "JCT" series T1/E1 cards to work with a Robbed Bit or CAS or R2 type line the following steps must be taken:

1. The Parameter File must be specified in the Dialogic Configuration Manager.
2. Config.xml file must have the correct protocol specified in the <Protocol> field.
3. Optionally Dialogic's `spandti.prm` and `voice.prm` configuration files may need to be edited

Screenshot below shows where in the DCM the Parameter File is set:



The example screenshot above shows the `r2_300.prm` file being used on the E1 line.

A common setting for the T1 Robbed Bit lines is `us_240.prm`

More information on the various configuration options which can be used to configure the "JCT" series cards can be found in this Dialogic help file: [springware_config_win_v1.pdf](#)

DMV series Dialogic cards

To configure the Dialogic "DMV" series T1/E1 cards to work with a Robbed Bit or CAS or R2 type line the protocol family is chosen in the "Trunk Configuration" tab in the Dialogic's Configuration Manager.

The specific protocol (eg: `pd_k_us_mf_io`) is the specified in the VoiceGuide's Config.xml file, in the <Protocol> field.

More information on the various configuration options which can be used to configure the "DMV" series cards can be found in this Dialogic help file: [dm3_pci_config_win_v1.pdf](#)

Information on the various diagnostics programs which can be used to debug the DMV cards' operation can be found in this Dialogic help file: dm3_diagnostics_win_v7.pdf

Configuring VoiceGuide

The channels which VoiceGuide is to use are specified in Config.xml file (see VoiceGuide's \conf\ subdirectory).

Sample Config.xml files are provided, and cover the most common deployment scenarios.

In the Config.xml file the Protocol files needs also needs to be set to the Dialogic protocol file which has been designed to work with the type of protocol used on the line. Common setting is `pdk_us_mf_io` for T1 Robbed Bit lines. The protocol .cdp files can be edited to change the Dialogic card behavior. Refer to Dialogic documentation for information on .cdp file structure and their fields.

VoIP Line Registration

VoiceGuide can register itself with the SIP provider of choice, resulting in VoiceGuide receiving calls directed to registered telephone numbers and being able to place outgoing calls using the registered accounts.

The VoIP registrations are specified in the Config.xml file, in section <VoIP_Lines>

<VoIP_Lines> contains two sections: <VoIP_Registrations> and <VoIP_Authentications>.

<VoIP_Registrations> can contain multiple <VoIP_Registration> sections, and

<VoIP_Authentications> can contain multiple <VoIP_Authentication> sections.

Section <VoIP_Registration>:

```
<VoIP_Registration>
<Protocol>SIP</Protocol>
<RegServer>ServerAddress</RegServer>
<RegClient>RegisteredClient</RegClient>
<LocalAlias>LocalAlias</LocalAlias>
</VoIP_Registration>
```

<Protocol>	Leave as SIP
<RegServer>	IP address of the registration server or the domain name of the registration server. If domain name is specified then HMP will resolve the domain name to IP address before issuing the registration request.
<RegClient>	Client name is usually specified as: <i>AuthUsername@Realm</i> or <i>AuthUsername@RegServer</i>
<LocalAlias>	Value of RegClient is sent in the From: and the To: fields of SIP Register messages.
	Any string is OK here. Value of LocalAlias is used in the Contact: field of SIP Register messages.

<VoIP_Authentication> holds information about the SIP digest authentication. It contains:

```
<VoIP_Authentication>
<Realm>Domain</Realm>
<Identity>AccountName</Identity>
<AuthUsername>AuthUser</AuthUsername>
<AuthPassword>AuthPassword</AuthPassword>
</VoIP_Authentication>
```

<Realm>	The 'realm' for which this authentication applies. It is recommended that this field be left blank, unless you are registering with multiple SIP servers. If registering with multiple servers then the "realm" used by the SIP server should be specified here. WireShark can be used to view 401/407/etc response contents to see realm setting in those responses.
<Identity>	Account for which this authentication applies. It is recommended that this field be left blank, unless you are registering multiple accounts/trunks and require a different authentication to be used for each account/trunk. If specified then this authentication entry will only be used if the Identity matches the To: field contents in the 401 or 407 response from the registration

	server. This field usually is in this format: sip:1010@10.1.1.11
<AuthUsername>	Username used for authentication.
<AuthPassword>	Password used for authentication.

The Dialogic HMP service must also be restarted after any changes to <VoIP_Registration> or <VoIP_Authentication> entries. This is necessary to clear the old Registration/Authentication entries that have been previously loaded into HMP.

WireShark can be used to confirm what SIP packets are exchanged between the SIP server and the VoiceGuide/HMP system. WireShark traces are usually necessary in determining causes of any registration failures.

SIP registration and authentication examples can be found in the Config.xml file. Information used for SIP registration is very similar for all SIP switches/providers.

Below are some examples as well:

CallCentric (www.callcentric.com)

```
<VoIP_Lines>

<VoIP_Registrations>
<VoIP_Registration>
  <Protocol>SIP</Protocol>
  <RegServer>callcentric.com</RegServer>
  <RegClient>17771111111@callcentric.com</RegClient>
  <LocalAlias>17771111111@10.1.1.9</LocalAlias>
</VoIP_Registration>
</VoIP_Registrations>

<VoIP_Authentications>
<VoIP_Authentication>
  <Realm></Realm>
  <Identity></Identity>
  <AuthUsername>17771111111</AuthUsername>
  <AuthPassword>Password</AuthPassword>
</VoIP_Authentication>
</VoIP_Authentications>

</VoIP_Lines>
```

Asterisk

The registration config below demonstrates how VoiceGuide would register to accept calls to a particular Asterisk extension (ext 3000).

Asterisk was installed on another server. Asterisk server's IP address was: 10.1.1.11 VoiceGuide is installed on IP address 10.1.1.9

```
<VoIP_Lines>

<VoIP_Registrations>
<VoIP_Registration>
<Protocol>SIP</Protocol>
<RegServer>10.1.1.11</RegServer>
<RegClient>1010@10.1.1.11</RegClient>
<LocalAlias>sip:1010@10.1.1.9:5060</LocalAlias>
</VoIP_Registration>
</VoIP_Registrations>

<VoIP_Authentications>
<VoIP_Authentication>
<Realm></Realm>
<Identity></Identity>
<AuthUsername>3000</AuthUsername>
<AuthPassword>1234</AuthPassword>
</VoIP_Authentication>
</VoIP_Authentications>

</VoIP_Lines>
```

FreeSWITCH

The registration config below demonstrates how VoiceGuide would register to accept calls to a particular FreeSWITCH extension (ext 1010).

FreeSWITCH was installed on another server. FreeSWITCH server's IP address was: 10.1.1.11

Note that this is all that is required to allow multiple calls to extension 1010 to be all sent to VoiceGuide at the same time. The number of actual calls handled will only be limited by the number of VoiceGuide lines, so for example a 20 line VoiceGuide system still requires only one extension to be registered with the VoIP switch.

FreeSWITCH will send all calls to extension 1010 to VoiceGuide, regardless of how many ext 1010 calls VoiceGuide is currently handling.

You can of course register multiple extensions if you want VoiceGuide to run different services depending on which extension was called.

```
<VoIP_Lines>

<VoIP_Registrations>
<VoIP_Registration>
<Protocol>SIP</Protocol>
<RegServer>10.1.1.11</RegServer>
```

```

<RegClient>1010@10.1.1.11</RegClient>
<LocalAlias>1010@10.1.1.9</LocalAlias>
</VoIP_Registration>
</VoIP_Registrations>

<VoIP_Authentications>
<VoIP_Authentication>
<Realm></Realm>
<Identity></Identity>
<AuthUsername>1010</AuthUsername>
<AuthPassword>1234</AuthPassword>
</VoIP_Authentication>
</VoIP_Authentications>

</VoIP_Lines>

```

Skype Connect

```

<VoIP_Lines>

<VoIP_Registrations>
<VoIP_Registration>
  <Protocol>SIP</Protocol>
  <RegServer>sip.skype.com</RegServer>
  <RegClient>99051000000000@sip.skype.com</RegClient>
  <LocalAlias>99051000000000@sip.skype.com</LocalAlias>
</VoIP_Registration>
</VoIP_Registrations>

<VoIP_Authentications>
<VoIP_Authentication>
  <Realm></Realm>
  <Identity></Identity>
  <AuthUsername>99051000000000@</AuthUsername>
  <AuthPassword>Password</AuthPassword>
</VoIP_Authentication>
</VoIP_Authentications>

</VoIP_Lines>

```

BroadSoft

Many SIP providers use BroadSoft's platform.

```

<VoIP_Lines>

```

```
<VoIP_Registrations>
<VoIP_Registration>
  <Protocol>SIP</Protocol>
  <RegServer>sip.NSW.iinet.net.au</RegServer>
  <RegClient>0299998888@sip.NSW.iinet.net.au</RegClient>
  <LocalAlias>0299998888@10.1.1.9</LocalAlias>
</VoIP_Registration>
</VoIP_Registrations>

<VoIP_Authentications>
<VoIP_Authentication>
  <Realm></Realm>
  <Identity></Identity>
  <AuthUsername>0299998888</AuthUsername>
  <AuthPassword>Password</AuthPassword>
</VoIP_Authentication>
</VoIP_Authentications>

</VoIP_Lines>
```


Command Line Options

VoiceGuide v7.x runs as a Windows Service only. This section only applies to v5.x and v6.x of VoiceGuide.

-icon : Will minimize the application to the system tray

-hide : Will hide the main application window

-min : Will minimize the window to the taskbar

Script Designer

The filename of the script to be edited can be passed as a parameter on the command line

Registering VoiceGuide

VoiceGuide can be registered using online order form at www.VoiceGuide.com

You may also register by filling out the form below and forwarding it to sales@voiceguide.com. After your registration has been processed you will receive by e-mail or Fax your own personalized License Key which will register your copy of VoiceGuide.

Name: _____

Company: _____

Address1: _____

Address2: _____

City/state: _____

Country: _____

Telephone: _____ Fax: _____

E-Mail: _____

Required: VoiceGuide [Unique Identifier](#) : _____

License Type: _____

License Price (as listed on www.VoiceGuide.com) : _____

Payment can be made by Visa, Mastercard, PayPal or Bank Transfer.

Cardholders Name: _____

Credit Card Number: _____

Credit Card Expiry Date: _____

Cardholders Signature: _____

Comments: _____

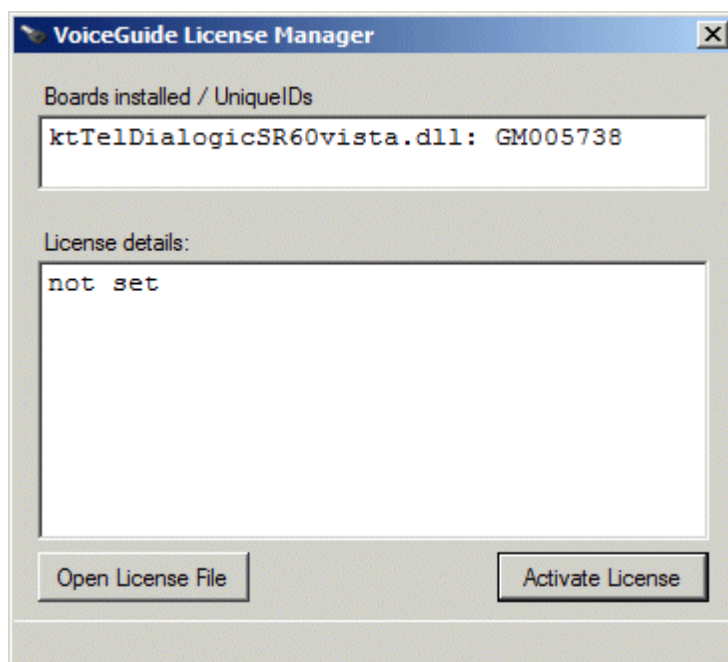
Unique Identifier

The Unique ID is based on either the Dialogic card serial number, or the Network adapters MAC address.

VoiceGuide v7

To find out the 'Unique ID' for the system go to Start -> Program Files -> VoiceGuide menu and start the License Manager, which will display the Unique ID.

The Dialogic service must be started before starting the License Manager in order for the Unique ID to show.



If Dialogic card is used the VoiceGuide v7 license will be linked to the Dialogic card, and the license will work when that Dialogic card is present in the system.

If moving software to new system the licensed Dialogic card needs to be moved to new system as well.

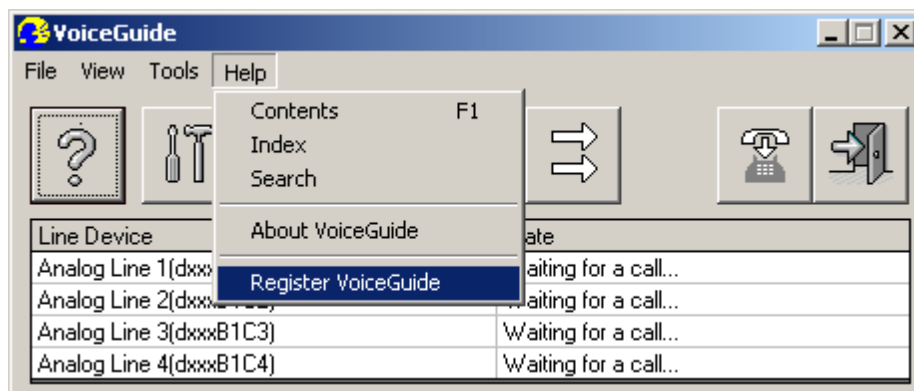
For VoIP deployments that do not use a Dialogic card the VoiceGuide v7 license will be linked to the network adapter, and the license will work when that network adapter is used for VoIP communication.

For VoIP deployments it is recommended that a PCIe based network card is used for VoIP communication, and any motherboard based network interfaces are disabled.

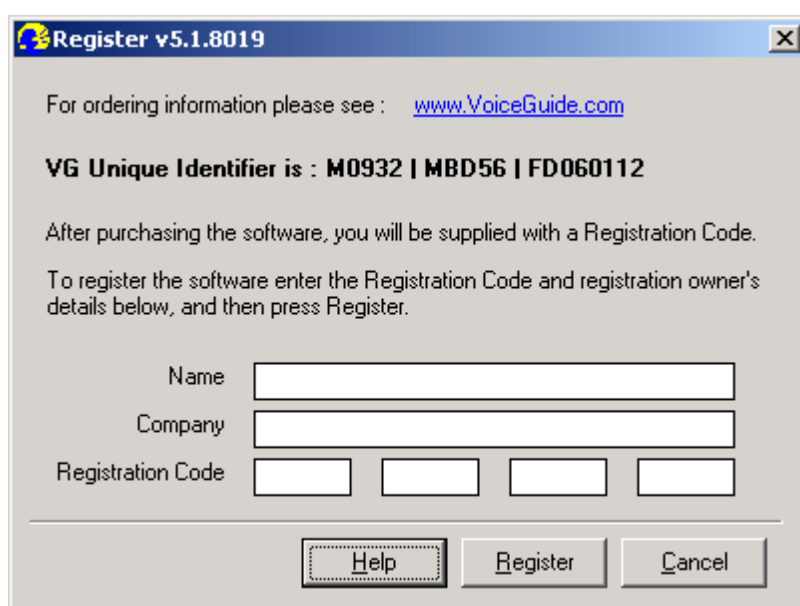
If moving software to new system the licensed network card needs to be moved to new system as well.

VoiceGuide v6 and v5

To find out the Unique ID for the system go to the Help menu and select 'Register VoiceGuide' option:



The registration screen will then appear with the various Unique IDs showing:



5 digit Unique IDs beginning with a letter 'M' are linked to the Network Card.
5 digit Unique IDs beginning with a letter 'H' are linked to the Hard Disk.
8 digit Unique IDs are linked to the Dialogic card.

In the example screenshot above, three Unique IDs are shown on the registration screen: M0932, MBD56 and FD060112. The first two Unique IDs are based on network cards in the system and the third one is based on the Dialogic card in the system.

For systems which use a Dialogic card it is recommended to link the license to the Dialogic card. This will allow the VoiceGuide license to be used in the new system that Dialogic card is moved to.

For systems which do not use a Dialogic card and where the ability to easily move the license from one machine to another is required it is recommended that a PCIe based network adapter is used (or a 'USB to Ethernet' network adapter). Moving the adapter to new system will allow license to work on the new system.

For older systems where license was issued linked to an on-board network adapter which cannot be moved to a new system a new registration code will be issued. Please contact sales@voiceguide.com

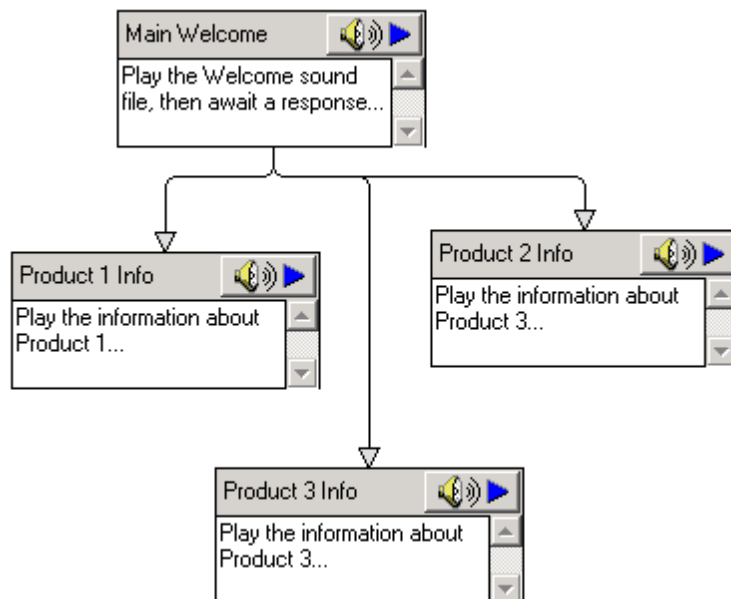
NB: There are no letters 'O' in any of the Unique IDs - if it looks like a zero then it is a zero.

Introduction

A Script specifies what the system is to do when an incoming call arrives, or an outgoing call is answered.

VoiceGuide provides a Graphical Script Designer to simplify script design and allow easy modification of existing scripts.

For example, to play a sound file, and then play a different file depending on which key the customer has pressed the script would look like this:



The modules above all play different sound files. There are other modules which perform other actions. eg: [ask for caller's input](#), [record sound files](#), [interact with databases](#), [interact with web services](#), [run other programs](#), [act as voicemail boxes](#), and [more](#).

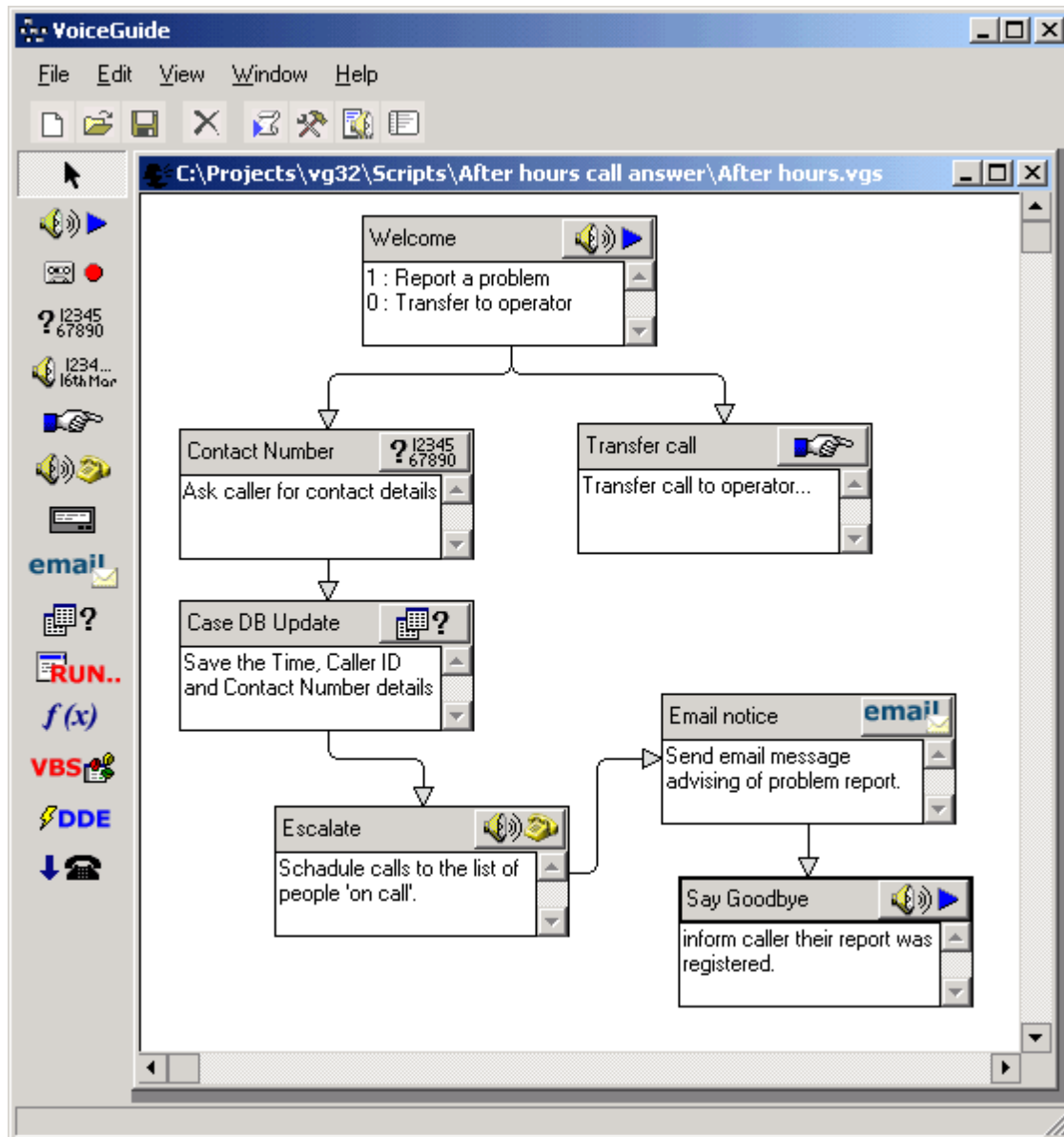
To create a system which you need is as simple as drag-and-dropping the modules which do the actions that you need on the screen and then specifying what events and actions result in Script taking caller from one module to another.

Graphical Design Environment

VoiceGuide's Graphical Script Designer allows creation of the IVR callflow by drag-and-dropping the function modules onto the screen and specifying how the caller will be taken through the script by editing the properties of each module.

NOTE: The script designer will not work under 'Aero' themes used by Windows7/Win2008/Vista. The workaround is to use 'Windows Basic' or 'Windows Classic' themes, or select 'Disable desktop composition' from vgScriptDesigner.exe's "Right Click->Properties->Compatibility" tab.


An example below shows a simple script being edited.



Adding new modules

The toolbar on the side of the window is used to select what modules to add to the script. To add a module select the module type to add, and then click on the script's workspace. To stop adding modules select the 'arrow' button.

Deleting modules

Select the module to delete and press the  button.

Creating paths between modules















Select the start module and press the 'Properties' button located in the top right corner of the module. This will bring up the module's Properties pages. The path options are set in the text box on the *Paths* tab of the Properties window. For more instructions on creating paths please [click here](#).

Running scripts

If the edited script is currently used by VoiceGuide then just saving the script will result in VoiceGuide using the new version of the script for any new calls arriving on the lines on which that script is active.

Module Types

VoiceGuide scripts can be put together using a variety of function modules:

	Play Sound File
	Record Sound File
	Get Number
	Say Number
	Transfer Call
	Make Call
	Web Service
	Send Email
	Database Query
	Run Program
	Time Switch
	Evaluate Expression
	Run VB Script
	Hangup Call

VoiceGuide Voicemail system can be accessed from anywhere in the VoiceGuide scripts. Callers can be directed to various parts of the Voicemail system, depending on whether they want to leave messages, listen to the messages in their mailbox, or just browse through the Voicemail boxes on the system.



Record a message for a particular Voicemail box



Access the messages in a particular Voicemail box



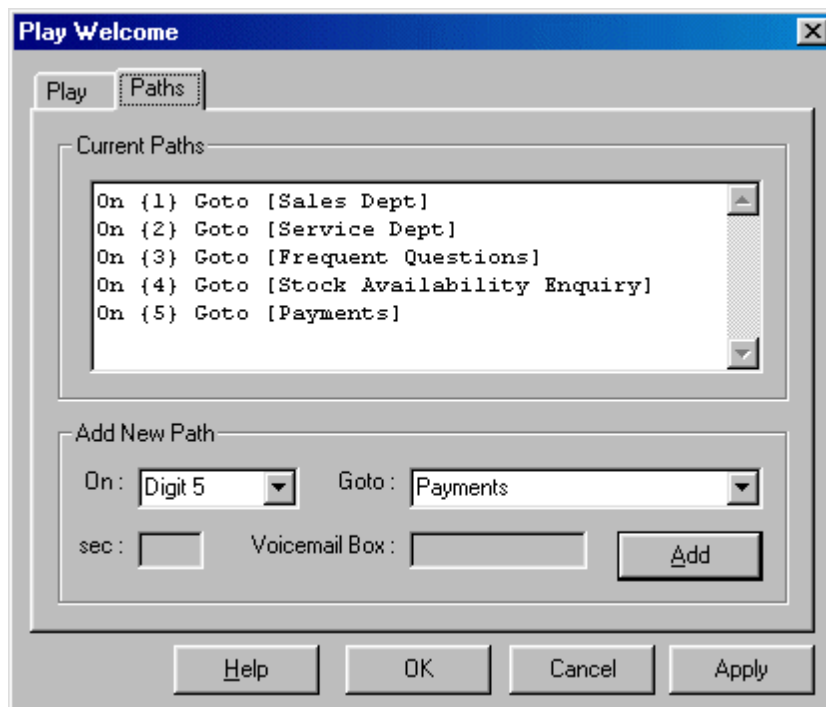
Browse through the Voicemail system, leaving messages in various Voicemail boxes

Paths

Paths is the name given to the transitions between the Script Modules.

Creating new paths

To create a new path, press the Properties button on the module from which the path will start, and go to the **Paths** tab.



Paths are specified using the format:

```
On {Trigger} Goto [Module Title]
```

The text in the Paths window can be edited directly, or you can use the Add New Path help frame.

Selection Paths

Taken when caller has made a matching selection. This can be a keypress or a series of keypresses, a keyword or phrase recognised by the Speech Recognition system, matching data retrieved from database or web service, or data returned by external program or VBScript/JavaScript etc.

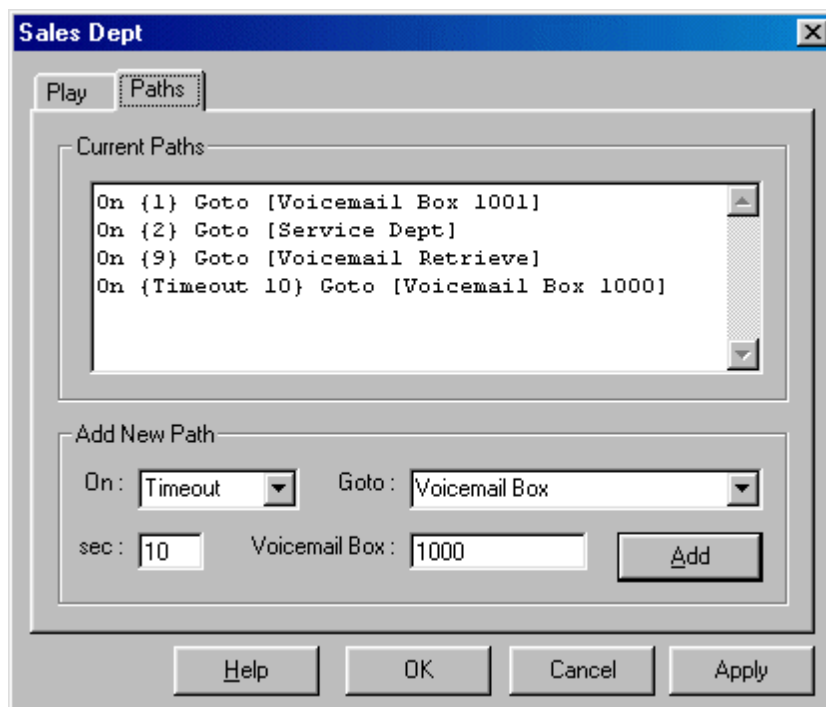
Timeout Paths

A Timeout path will be taken if a caller has not made a selection within the specified number of seconds. A Timeout path with a wait time of zero seconds will be taken immediately after the last

sound file in the module has completed playing.

Paths to Voicemail System

Paths can direct the caller to a Voicemail Box, Voicemail Box Menu or Voicemail Retrieve Menu, eg:



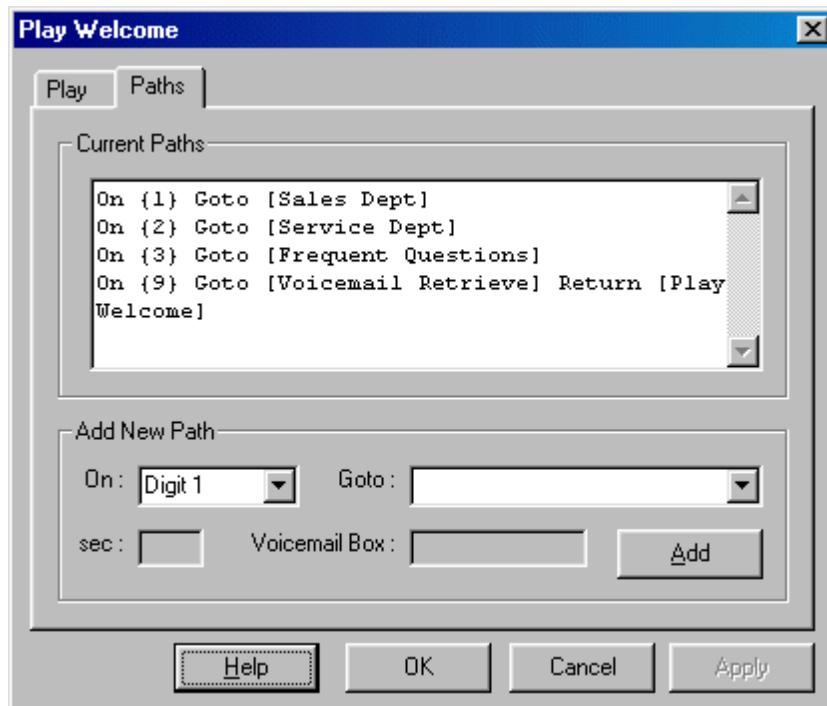
When specifying the Voicemail Box as the destination, the Voicemail Box number must be specified as well. make sure that the Voicemail Box specified exists and has been configured using the [Voicemail System Manager](#).

Returning from the Voicemail system

If you would like the caller to go to a certain module after returning from the Voicemail system then you can specify a return module. The Path definition needs to be:

```
On {Trigger} Goto [Voicemail System Module] Return [Module Title]
```

Eg: if after Retrieving Voicemail the caller should be sent to the main menu again, the path will look like this:



If the titles of the modules are too long to fit on one line, the path description will wrap around to the following lines. This is OK.

Branching to other scripts and calling subscripts.

The Enterprise and Evaluation versions of VoiceGuide can call subscripts.

To branch (goto) to a script you can specify a path like this:

```
on {event} goto [script filename|module name]
```

eg:

```
on {1} goto [c:\scripts\myscript.vgs|PlayWelcome]
```

To run a subscript (gosub) to a script you can specify a path:

```
on {event} gosub [script filename|module name]
```

eg:

```
on {1} gosub [c:\scripts\myscript.vgs|PlayWelcome]
```

The script name or the module name can be left blank. If the Script Filename is left blank then the current script is assumed, and if the Module Name is left blank then the default starting module in the script is used.

```
eg: on {1} gosub [|PlayWelcome]
```

```
on {1} gosub [c:\scripts\myscript.vgs|]
```

If the [*module name*] notation is used the path to the destination module will not be drawn. This notation can be used to visually de-clutter highly linked scripts.

If "|" is omitted from the destination specification then the destination name is assumed to be just a module in the current script. So for example this path will not work:

```
on {1} gosub [c:\scripts\myscript.vgs]
```

To specify what module should be the next module once the subscript returns the "return" path option should be used. If the "return" path option is not used when the subscript returns then VoiceGuide will start running the default start module in the returned to script.

To specify a return module:

```
on {event} gosub [script filename|module name] return [return script filename|return module name]
```

eg:

```
on {1} gosub [c:\scripts\admin.vgs|GetUserPin] return [c:\TakeOrder.vgs|MainMenu]
```

The Return To script can be a different script. The return script filename can be omitted if you would like the subscript to return to the same script.

To return from a subscript use a path like this in the subscript:

```
on {event} return
```

eg: on {4} return

The various goto/gosub/return paths are briefly demonstrated in the sample scripts in VoiceGuide's \scripts\more sample scripts\paths directory.

Subscripts using COM/WCF interface.

Please see the VoiceGuide COM/WCF Reference section in this Help file for information on how the goto/gosub/return commands can be issued to VoiceGuide from external applications.

Result Variables

Result Variables store information on activity in each module during the call. These Result Variables (\$RVs) can then be used later in the VoiceGuide script, allowing the VoiceGuide script to act based on the information in the previously set \$RVs.




Result Variables can also be accessed and created/set by external systems, using VoiceGuide's COM/WCF/REST/etc interfaces. This is one of the mechanisms by which external systems can directly monitor and control the progress of the VoiceGuide script in real-time.

Please refer to each module's reference for more information on \$RVs created by that module. VoiceGuide vgEngine trace files also contain details of all created/set \$RVs.

Often Used Result Variables :

\$RV[*module title*]

Used to access the 'main' information saved by a particular module. eg:

	Module Type	Result Variable will contain
	Play	The key pressed by the caller
	Get Number Sequence	The number entered by the caller
	Record	The last recorded filename.

All modules create more than one Result Variable. Please see documentation for each module type and the vgEngine trace file to see what \$RVs are created by each module.

The module Result Variables are created for the duration of the call, and all modules running on that line during the call will be able to access them. Even if the script jumps to another script then the next script will still be able to access the Result Variables created by any previous script during current call.

\$RV_CIDNAME

Caller's Name (if available). Can be used in [Evaluate Expression](#) modules to switch to different areas on the script based on who is the caller.

On traditional analog telephone lines the Caller ID information is often sent between the first and the second Ring. Hence it is usually necessary to set analog systems to answer after the second Ring if this Result Variable is to carry any information. On ISDN and VoIP systems Caller ID is provided at beginning of the call.

\$RV_CIDNUMBER

Caller's Telephone Number (if available). Can be used in [Evaluate Expression](#) modules to switch to different areas on the script based on who is the caller.

On analog telephone systems the Caller ID information is often sent between the first and the second Ring. Hence it is usually necessary to set analog systems to answer after the second Ring if this Result Variable is to carry any information. On ISDN and VoIP systems Caller ID is provided at beginning of the call.

\$RV_DNIS

Telephone number called by the caller. This information is usually only provided on ISDN lines, and is usually used when multiple numbers are terminated the the T1/E1 ISDN line. Using \$RV_DNIS the system can then determine which script it should be starting for the call.

\$RV_LINEID

Identification number of the line device which is handling this call. Can be used in Run Program module to help the called program generate the appropriate Results filename, or in the COM interface to return results to the correct line. The ID number refers to an internal line ID number - not the position in which that line's status is displayed in VoiceGuide's status window.

\$RV_PORTNUMBER

Line Device's position is VoiceGuide's line status listing. Numbering starts from the top, with first line/port having a \$RV_PORTNUMBER of 1, the seconds line/port having a \$RV_PORTNUMBER of 2, etc.

\$RV_STARTTIME

The Date and Time the call was started. The 'Medium' Date format and the 'Long' Time format is used. These formats can be set in the Windows' Control Panel - Regional Settings applet.

\$RV_CALLLENGTH

Number of seconds since the call started. This variable can be used to limit the length of time callers spend using the system.

\$RV_LINESINUSE

Number of lines on system currently busy taking calls.

\$RV_LIC_LINES

Number of lines system is licensed for.

\$RV_LIC_TYPE

Professional, Professional+Dialer, Enterprise or Enterprise+Dialer

\$RV_SOFT_VERSION

Version number of software running.

\$RV_RINGCOUNT

Number of rings an incoming call has rung so far. This variable can be used to determine at which point to answer the call if the script was started with the "Start the script before answering the call" option.

\$RV_DIALEDNUMBER

If the call is an outbound call this variable stores the telephone number which was dialed.

\$RV_LastKeyPress

Last Key that was pressed, or "timeout" if a timeout event fired.

\$RV_PreviousModule

Title of the previous module, ie: from which module the script arrived at the current module.

\$RV_EventListXml

Significant events which occurred during the call along with the times when those events occurred. This information is usually used in in system usability analysis.

\$RV_CALLSTATE

Current state of call.

\$RV_STATUSDISPLAYED

What is currently displayed in the Line Status Monitor.

\$RV[RUNAFTERHANGUP]

The filename of the script ran when the caller hangs up. This filename can be set in the Evaluate Expression module. Setting of \$RV[RUNAFTERHANGUP] is only valid during the current script. This "OnHangup Script" setting is wiped whenever a goto/gosub is made to another script, and the new script's "OnHanup Script" setting is used.

\$RV[*module title*_RowCount]

\$RV[*module title*_ColumnIndex_RowIndex]

Please see the [Database Query module](#) for more information on the two Result Variables above.

\$RV_PathScript

The path to the location where the script is located. Does not include the "\" at the end.

\$RV_PathVoiceGuide

The path to the location where the VoiceGuide application is located. Does not include the "\" at the end.

\$RV[VoicemailMessage]

\$RV[VoicemailMessageXXXX]

The last recorded voicemail message during this call.

\$RV[VoicemailMessage] returns the filename of the last recorded voicemail message regardless of in which voicemail box it was recorded.

\$RV[VoicemailMessageXXXX] returns the filename of the last recorded voicemail message in a particular voicemail box - which is specified by replacing XXXX with the number of the voicemail box. The message must be recorded during the current call for them to be accessible using these Result Variables.

\$RV[VmbId]

ID of the last voicemail box that was accessed (either logged into or message left for).

Outbound dialing related Result Variables

\$RV[OutDial_RetriesLeft]

Number of retries left for an outgoing call. If this is the last call attempt then this RV will have a value of 0.

\$RV[OutDial_Result]

When the outgoing call has been answered or the number of retries has been used up this RV stores the type of outcome. Possible values are: Contacted_Human, Contacted_AM, Contacted_Pager, Contacted_Fax, Uncontactable_OnDontDialList, SIT_Reorder, SIT_NoCircuit, SIT_CustIrReg, SIT_Unknown, SIT_Unavailable, Uncontactable_NoAnswer

\$RV[AmWelcMsg_RecLen100ms]

Length of Answering Machine message. in 100ms units. eg: a value of 40 would indicate answering machine message was 4 seconds long.

Conference & Call Transfers related Result Variables

\$RV[Conf_DevName_X]

The device name of party X in a conference call. eg: \$RV[Conf_DevName_2] would be the device name of device carrying the 2nd leg of a call in the "Dial and

Conference" call.

\$RV[Conf_LineID_X]

The Line number of party X in a conference call. eg: \$RV[Conf_LineId_2] would be the Line ID number of device carrying the 2nd leg of a call in the "Dial and Conference" call.

Script Branching (Goto/Gosub/Return) related Result Variables

These RVs are set when a Goto or Gosub branch is made to another script. The names are fairly self explanatory:

```
$RV[ScriptEnd_Time]  
$RV[ScriptEnd_Goto_Script]  
$RV[ScriptEnd_Goto_Module]  
$RV[ScriptStart_Time]  
$RV[ScriptStart_CalledFrom_Script]  
$RV[ScriptStart_CalledFrom_Module]
```

Current Date/Time Result Variables

\$RV_WEEKDAY

Takes on a value between 1 and 7, depending on what day it is. 1 is a Monday and 7 is a Sunday. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on the day of the week.

\$RV_DD

Takes on a 2 digit value between "01" and "31", depending on what day of the month it is. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on what day of the month it is.

\$RV_MM

Takes on a 2 digit value between "01" and "12", depending on what month it is. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on the current month.

\$RV_YY

Takes on a 2 digit value of current year (eg: "03"). Can be used in Evaluate

Expression modules to allow you to switch to different areas on the script depending on the current year.

\$RV_HH

Takes on a 2 digit value between "00" and "23", indicating the hour of the current time. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on the time of day.

\$RV_NN

Takes on a 2 digit value between "00" and "59", indicating the minute of the current time. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on the time of day.

\$RV_SS

Takes on a value between "00" and "59", indicating the second of the current time. Can be used in Evaluate Expression modules to allow you to switch to different areas on the script depending on the time of day.

\$RV_MS

Takes on a value between "000" and "999", depending on the millisecond fraction of the current time.

Other Time Related RVs

The following \$RVs are also available. They are pretty self-explanatory:

`$RV_MONTHNAME, $RV_HOUR, $RV_MINUTE, $RV_SECOND, $RV_TimeStamp_Long,`
`$RV_TimeStamp_Short, $RV_DateStamp_Long, $RV_DateStamp_Short $RV_MONTH, $RV_YEAR,`
`$RV_DATE`

eg. use this expression to create a date and time 'timestamp' :

`RV_YYRV_MMRV_DDRV_HHRV_NNRV_SS.$RV_MS`

Where can Result Variables be used

\$RVs can be used anywhere in the VoiceGuide script. eg:

- Specifying the 'trigger' and the destination modules in Paths leaving a module,
- Filenames of sound files which are to be played or recorded,
- What digits/dates/amounts are to be spoken to the caller in the [Say Number](#) module,
- Looking up information in a database,
- As parameters when calling other programs, web services, etc.

- Evaluate Expression modules,
- Send Email module: in destination Email address, Title, Message Body and Attachment Filename.

Global RVs

Global RV can be preset in VG.INI, in section [Scripts], using an expression that lists multiple RVs on one line, like this:

```
GlobalRV=[MyValue1]{23}[SomeOption]{yes}[SomeEmail]{admin@xyz.com}
```

These RVs will then be loaded and set when the VoiceGudie service starts and can be used in scripts. eg. If using example above then \$RV[SomeOption] can be used in the scripts.

Notes

If a Result Variable is used in the script that has not yet been defined, it will be replaced with nothing (an empty string), not a digit zero.

Eg: If you use expression: C:\SoundFiles\Info\$RV[InfoNumber].wav and the script has not yet defined \$RV[InfoNumber] then the resulting string after the Result Variables are replaced will be: C:\SoundFiles\Info.wav

If \$RV are used in Evaluate Expressions module's comparison expressions it is usually best to use quotes around the RVs which may be empty.

Also, quotes should be used when using RVs in a VBScript in situations where replacement with an empty string could potentially lead to syntax errors unless quotes have been used.

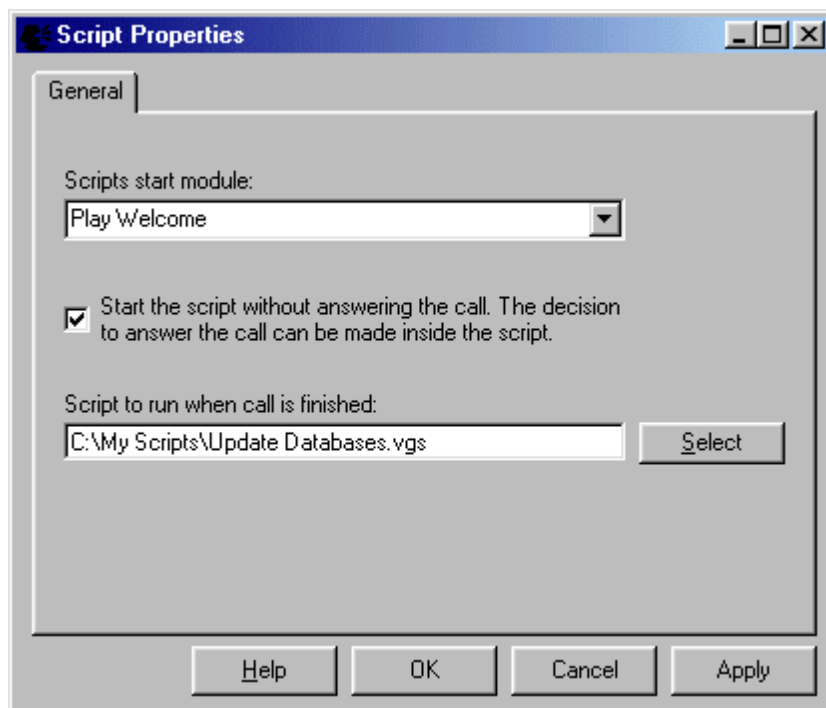
Call Start

It is sometimes desirable to carry out some processing before the call is answered. Quite often the purpose of this processing is to determine whether the call should be answered or not.

VoiceGuide script can be configured to start running without answering the telephone line. This allows VoiceGuide to conduct any database or web service lookups etc. based on provided CallerID in order to determine how the call should be handled.







To select this option use the "Script Properties" menu entry from the "Edit" menu in the Graphical Design Environment, then check the "Start the script without answering call" box.

For a given script to answer all incoming calls leave the "Start the script without answering call" box unchecked (this is the default setting for each new script).



Modules allowed before the call is answered

Any modules that do not need to play a sound file can be used. eg:

	Database Query
	Web Service Call
	Time Switch
	Evaluate Expression
	Run Program
	Send Email



Make Call



Run VB Script



Hangup the Call

Arriving at the "Hangup Call" module results in the script finishing and the call never being answered.

Answering the call

Calling any module which plays or records a sound file or transfers the call will result in a call being answered.

Reaching the following modules in the script will effectively answer the call:



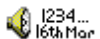
Play Sound File



Record Sound File



Get Number



Say Number



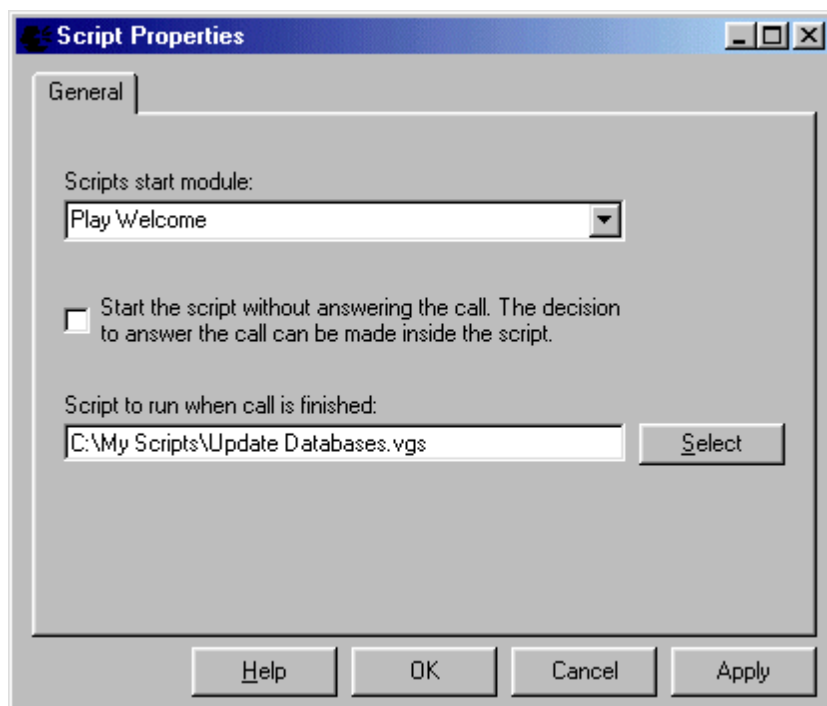
Transfer Call

Call Finish

It is sometimes desirable to perform some functions at the time when the call finishes, regardless of at which point in the script the call ended. Usually such a script will perform 'clean-up' type operations like database updates, call other programs, send out emails or schedule outbound calls.

VoiceGuide can be instructed to run a separate script when the call has finished using the "Script Properties" option from the "Edit" menu in the Graphical Design Environment.







Below shows the script "C:\My Scripts\Update Databases.vgs" is to be called when the currently edited script is finished:



Modules allowed in the Cleanup/OnHangup script

Only certain types of modules can be used in a Cleanup script. As the call has already ended, modules which play or record sound files or expect user input cannot be used.

The following modules can be used:

	Database Query
	Web Service Call
	Time Switch
	Evaluate Expression
	Run Program
	Send Email



Make Call



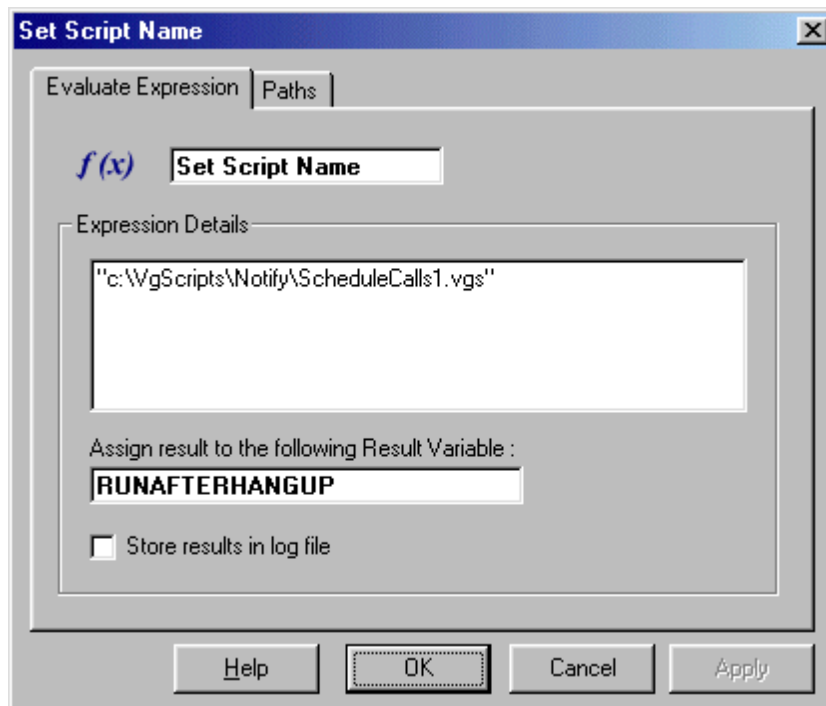
Run VB Script



Hangup

Changing the Cleanup/OnHangup script at runtime

It is sometimes desirable to change the Cleanup script depending on user selections or other factors. To change the script we need to assign the new filename to the `$RV[RUNAFTERHANGUP]` [Result Variable](#). The [Evaluate Expression](#) module can be used to do this:



Result Variables

All Result Variables from the script which was handling the call are available to the 'Cleanup' script. This allows all information entered by caller or retrieved/created by the original script to be used in the call cleanup script.

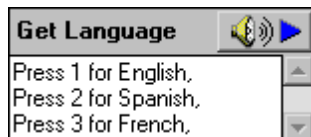
Multilanguage Systems

When implementing multi-language systems:

- Script prompts need to be recorded in multiple languages.
- If using any of VoiceGuide's pre-recorded system prompts in your script, then a new set of system prompts needs to be recorded in the new language as well.

Script Sound Files

Usually towards the beginning of the script a following module would be used:



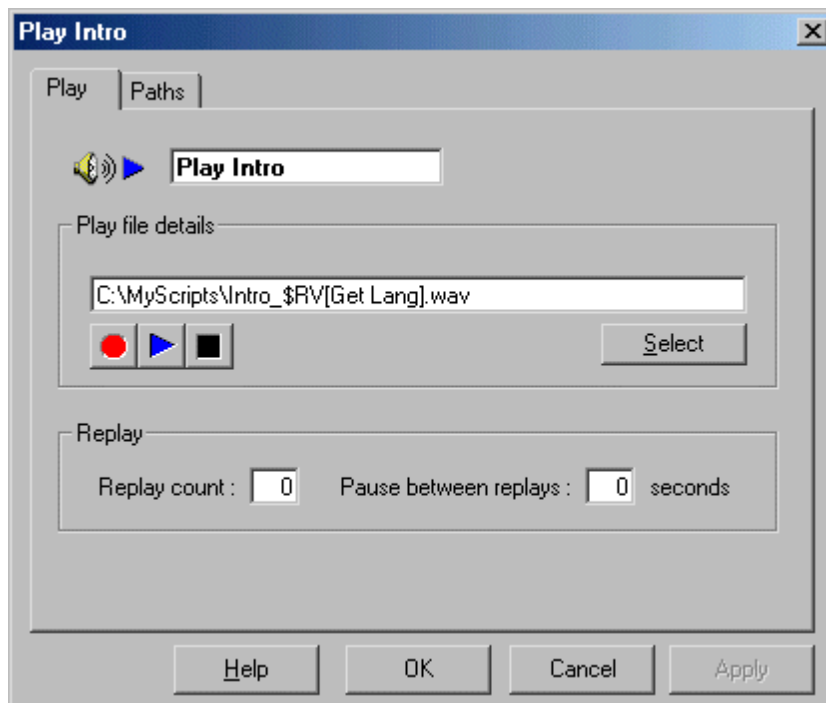
The user selection would then be available by using the `$RV[Get Language]` [Result Variable](#).

You can use this result variable to select the right language version of the file you will want to play in the rest of the script.

eg:

`C:\MyScripts\Intro_$RV[Get Language].wav`

Would be translated by VoiceGuide to indicate either `Intro_1.wav`, `Intro_2.wav` or `Intro_3.wav`, where the three files are recorded in English, Spanish and French respectively.

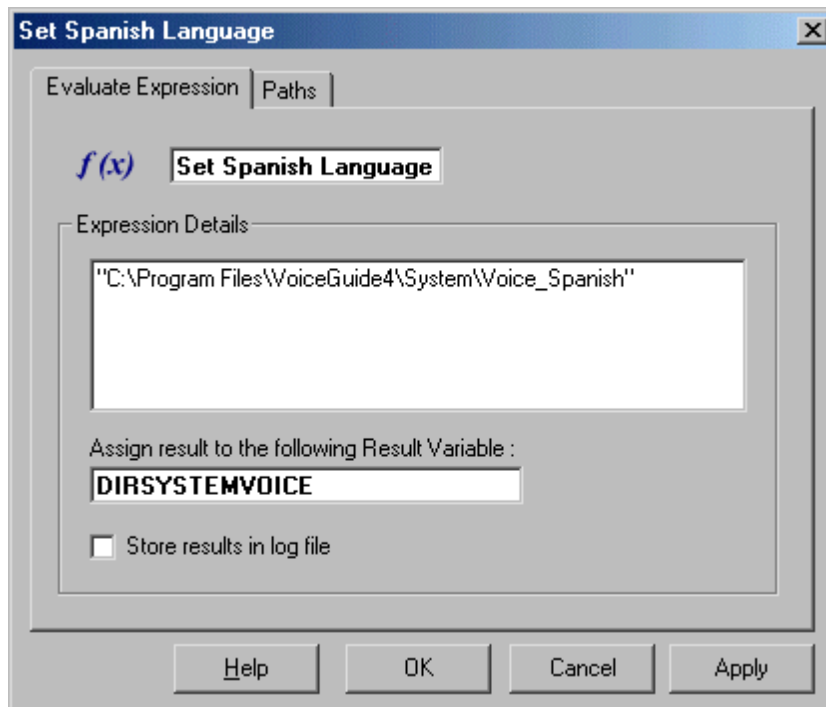


This approach allows you to use a single script for a number of languages, saving you the need to duplicate the scripts.

VoiceGuide System Sound Files

VoiceGuide's default system sound files are stored in the VoiceGuide's "\\system\\voice\\" subdirectory.

To select a different set of system files to be used for the current call, the path pointing to the new system directory current for this call needs to be written to the `$RV[DIRSYSTEMVOICE]` Result Variable.



The path has to be written in double quotes. Note that only `DIRSYSTEMVOICE` is specified in the "Assign to Result Variable" text box, not `$RV[DIRSYSTEMVOICE]`.

Protected Scripts

Saving VoiceGuide Script as 'Protected Script' encrypts the script file.

Protected Scripts can only be viewed/edited if the password is known.

Protected Scripts should be used if the VoiceGuide Script contains sensitive information, like database or web service access passwords etc.

When running Protected Scripts VoiceGuide will decrypt the script in memory, preventing the sensitive information from being disclosed.

Protected Scripts have the .VGP suffix.

Sound files

The format of sound files which are to be used depends on which version of VoiceGuide is used.

- Versions 7.x** .WAV : U-Law 8kHz 8-bit Mono or A-Law 8kHz 8-bit Mono.
Can also use .WAV PCM: 8kHz or 11kHz, 8-bit or 16-bit, Mono.
Preferred format is selected at install time.
- Versions 6.x** .WAV : PCM 8kHz, 8-bit, Mono.
Can also use .VOX files: 8kHz, 4bit, ADPCM.
- Versions 5.x** .WAV : PCM 8kHz, 16-bit, Mono.
When using VoiceGuide versions 5.x with Dialogic TAPI/WAVE drivers then the sound files must be recorded in the format used by the Dialogic Wave driver, which is WAV PCM 11kHz, 8bit, Mono.

VoiceGuide comes with a number of pre-recorded system files which are installed in directory \system\voice\

The format of the sound files installed may depend on what options were selected during install.

An alternative system directory can be specified in the script by assigning the directory path to the DIRSYSTEMVOICE Result Variable.

Please see the [Multilanguage](#) section for more information on this.

The sound files say "Pound Key" to indicate the "#" key. Alternate copies of the sound files have been supplied if you would like VoiceGuide to use "Hash Key" to indicate the "#" key. To use these files, just copy the alternate files over the system files used by VoiceGuide. The alternate files have "_HashKey" appended to their filename.

Converting Sound Files

Converting of large number of sound files can be done using the free SOX Sound eXchange toolkit from <http://sox.sourceforge.net/> or Adobe Audition (<http://www.adobe.com/products/audition/>). Other tools are also available as well.

System Sound Files

Files in \system\voice\ contain the following recordings:

0-23, 30, 40, 50, 60, 70, 80, 90	These files contain the digit which corresponds to the name of the file.
AcceptAutoCall	This is an automated telephone call, please press any number on the telephone keypad to accept this call
am	am as in 8:30 am
and	...and...
beep1	A short beep played before recording sound files.
billion	billion
cents	cents
dollars	dollars
ErrorPlayingFile	There has been an error playing this file, please check the file format

GetNbrsConfirmIntro

GetNbrsConfirmMenu

hundred

MaxRecTime

million

minus

month01 ... month12

place01 ...place31

pm

point

press

pressplz

RecMsgMenu

SoundFileNotFound

thousand

trillion

transferto

TsfrCallFrom

TsfrAskAccept

You have entered the following number:

Please press '1' if that is correct, or press '2' if you would like to re-enter the number

hundred

I'm sorry, the maximum record time has been reached

million

minus

January ... December

first ... thirty first

pm as in 5:30 pm

point - as in the decimal point

press

please press

Please press '1' to replay the message, '2' to delete the message and record again, or '3' to save this message and go back to the voicemail box menu

The sound file specified could not be found

thousand

trillion

to be transferred to

This is a transferred call

Please press 1 to accept the call, or press any other number to reject it

ACD

acd_EstWaitTimeIsLessThen

acd_EstWaitTimeIs

acd_Minute

acd_Minutes

acd_YouAre

acd_InTheQueue

acd_LongWaitTimes

acd_IfYouDoNotWantToWait

acd_NotLosePlaceAndCallBack

acd_NotLosePlace

acd_PleaseSayYourName

acd_WeHaveRecorded

acd_Press1orRecAgain

acd_WeHaveYourNumber

acd_EnterPhoneNumber

acd_YouHaveEntered

acd_Press1orEnterAgain

acd_SystemWillCallBack

acd_WeWillCallBackOn

acd_InApprox

acd_ThisIsACallFor

acd_When

acd_WhenAtTelephonePress1

acd_YouWillNowBeConnected

Your estimated wait time is less then

Your estimated wait time is

minute.

minutes.

You are ...

...in the queue.

We are experincing longer then normal wait times.

If you do not want to wait, you can request a callback.

You will not lose your place in the queue and the system will call you back once you have reached the start of the queue.

You will not lose your place in the queue.

Please say your full name after the tone and press any key when you have finished.

The name we have recorded is:

Please press 1 if that is correct or press any other key to record again.

Please press 1 if you would like us to call you back on the phone number from which you are calling from, or press any other key to enter the contact phone number.

Please enter the phone number on which you would like us to call you back.

You have entered:

Please press 1 to confirm, or press any other key to re-enter the number.

The system will call you back once you have reached the start of the queue.

We will call you back on ...

... in approximately ...

This is a callback telephone call for...

When...

... is on the line please press 1 to continue.

Thank you, you will now be connected.

acd_ThankYou

Thank You.

Automated Attendant

aa_connectto

To be connected to

aa_getextorspell

If you know the extension number of the person to which you would like to be transferred please enter it now, otherwise please press 1 to spell the name of the person to whom you want to be transferred using your telephone keypad , or press 0 to be transferred to the operator.

aa_nomatchext

This extensions does not exist

aa_nomatchext2

Extension could not be found

aa_nomatchnames

No matching names can be found

aa_press0operator

or press 0 to speak with the operator

aa_press0reception

or press 0 to speak with reception

aa_spellname

Please enter the first 4 characters of the persons name, spelling the surname using the keys on your telephone keypad.

aa_spellname2

Please enter the first 4 characters of the persons surname, spelling the surname using the keys on your telephone keypad.

aa_transferringto

Transferring you now to...

aa_xfernoanswer

The call was not answered, press 1 to leave a message in the voicemail box, or press 2 to try another extension. You can press 0 to speak with the operator.

aa_xferextbusy

The extension is busy, press 1 to leave a message in the voicemail box, or press 2 to try another extension. You can press 0 to speak with the operator.

Voicemail

VmbAccessGetVmbId

Please enter the voicemail box number for which you want to leave the message and then press the pound key

VmbAccessGetVmbIdRetrieve

Please enter your voicemail box number and then press the pound key

VmbAccessGetVmbPin

Please enter your 4 digit PIN number and then press the pound key

VmbAccessInvalidVmbId

This voicemail box does not exist

VmbAccessInvalidVmbPin

I'm sorry, the PIN number you have entered was incorrect

VmbAccessListenMsgMenu

Please press '1' to replay the message, '2' to save the current message and listen to the next message, '3' to delete the current message and listen to the next message, or press '0' to return to the voicemail box menu

VmbAccessMenu

Press '1' to listen to new messages, '2' to listen to saved messages, press '3' to change your greeting message, or press '4' to change your mailbox PIN number.

VmbAccessNewMessagesAnd

... new messages and ...

VmbAccessNoMessages

No Messages

VmbAccessNoNewMessages

There are no new messages

VmbAccessNoSavedMessages

There are no saved messages

VmbAccessNoWelcMsg

You currently do not have a personalized welcoming message

VmbAccessSavedMessages

... saved messages

VmbAccessWelcMsgMenu

Press '1' to play the current greeting message, '2' to record a new welcoming message, '3' to save the current message and exit and '4' to delete the current message and exit

VmbAccessWelcMsgRecPrompt	Record your new welcoming message after the tone, when you have finished recording press '1'
VmbAccessYouHave	You have ...
VmbAdminName_Menu	Press '1' to play the current name, '2' to record a new name, '3' to save the current name and exit or '4' to delete the current name and exit
VmbAdminName_NoName	You currently do not have a name recorded for this mailbox
VmbForwardAskAccept	This is the VoiceGuide voicemail system. You have a new voicemail message. Press '1' to hear your new message.
VmbForwardPhonePrompt1	The current forwarding phone number is: Enter the phone number to which your voicemail messages will be forwarded, and press the pound key when you have finished. You can press the pound key to stop forwarding your voicemail messages or press the star key to keep your current forwarding phone number.
VmbForwardPhonePrompt2	You currently do not have a forwarding phone number set up.
VmbForwardPhonePrompt3	You have entered the following phone number:
VmbForwardPhoneConfirm1	Please press 1 to confirm, or press 2 to re-enter the phone number.
VmbForwardPhoneConfirm2	The forwarding phone number has been updated.
VmbForwardPhoneChangedOK	This mailbox does not exist
VmbDoesNotExist	Your PIN number has been changed
VmbPinChangedOK	Please enter your new PIN number
VmbPinEnterNew	Please re-enter the PIN number
VmbPinEnterNew2	The two entered PIN numbers do not match
VmbPinNotMatching	Please record your message after the tone
VmbWelcomeMsgDefault	Please press 2 to delete the recorded message and start recording again, 3 to save the recorded message and start entering the destination voicemail groups or 4 to delete the recorded message and return to the voicemail box main menu.
VmSendMainMenu	Please enter the next destination followed by the pound key, or just press the pound key if you have finished entering all destinations.
VmSendGetNextDest	Messages have been sent.
VmSendSuccess	Please record the message after the tone. When you have finished recording please press the Pound key.
VmSendRecPrompt	Please enter the next next recipient followed by the pound key, or just press the pound key if you have finished entering all recipients
VmSendGetNextDest	Please press 1 to play the recorded message, 2 to delete the message and record again, 3 to save the message and start entering recipients, or press 0 to delete the recorded message and return to the voicemailbox menu.
VmSendMainMenu	Please record the message you would like to send after the tone. when you have finished recording please press the pound key.
VmSendRecPrompt	Messages have been sent
VmSendSuccess	

Testing Scripts

The best way to test the IVR system is to dial into it just like the caller would.

To test the script while developoing them a good approach is to setup VoiceGuide in VoIP mode and dial into it by just dialing the IP address of the machine on which VoiceGuide is installed using a VoIP phone or a soft phone from another PC.

or

Use a small PBX appliance and plug the analog lines from the small PBX into the Dialogic card.

Automated Testing

VoiceGuide can be used to create scripts which can test IVR systems by dialing into them and running a test script which simulates callers responses.

Please contact sales@voiceguide.com for more information on designing such automated IVR testing solutions.

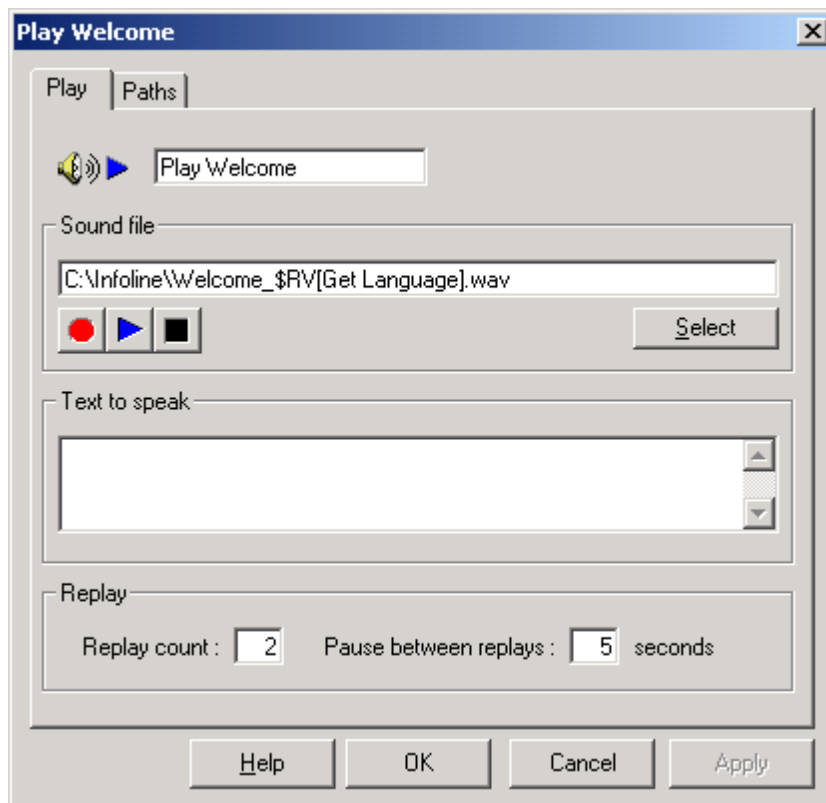
Play

The Play module can:

- Play a sound file,
- Speak the typed text (Text to Speech),
- Speak the specified text file (Text to Speech),
- Play DTMF tones
- Send Hookflash signal

While playing the module will listen for a response from the caller. You can select the number of times the message will be re-played, and the pause between the replays.

[Result Variables](#) can be used when specifying the sound file to be played. In the example below the file played will be different depending on what selection the caller made in the 'Get Language' module.



Multiple sound files can be selected to be played by separating the successive files by commas. eg:

```
prompts\1.wav, prompts\2.wav
```

would result in two files from the script's prompts subdirectory being played (1.wav and 2.wav) one after another.

If the sound files are specified without a full path then VoiceGuide will search for the sound files in the script's directory, or in a script's \voice\ subdirectory (if it exists).

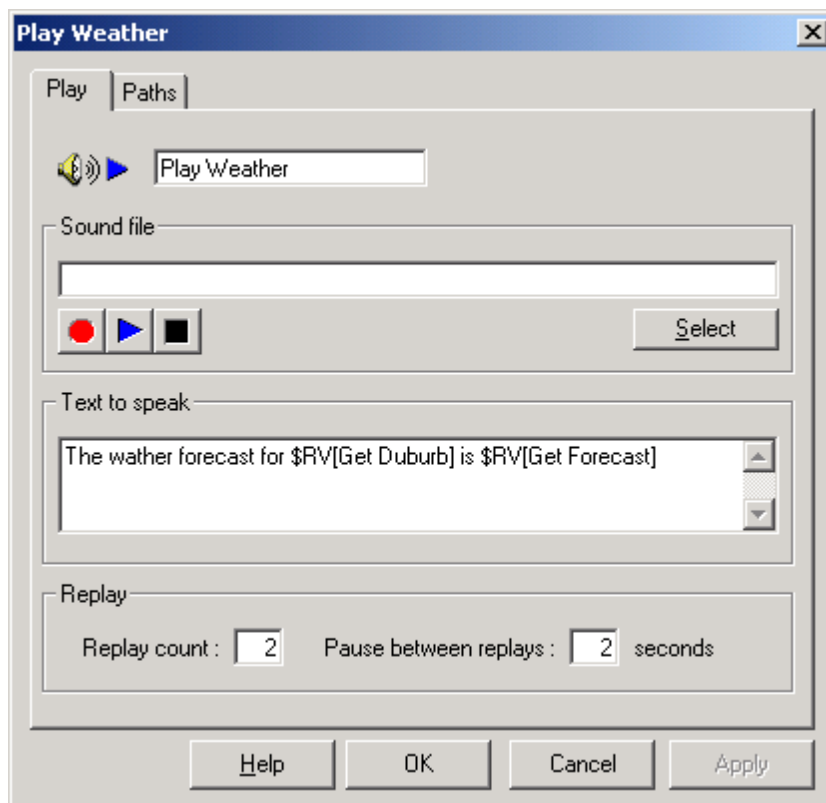
If one or more of the sound files to be played cannot be found then VoiceGuide will see if a path `SoundFileNotFound` is defined in the module.

If the path `SoundFileNotFound` is defined then that path will be taken.

If the path `SoundFileNotFound` is not defined then VoiceGuide will play the system sound file `SoundFileNotFound.wav` in place of the sound files that cannot be found.

Text to Speech

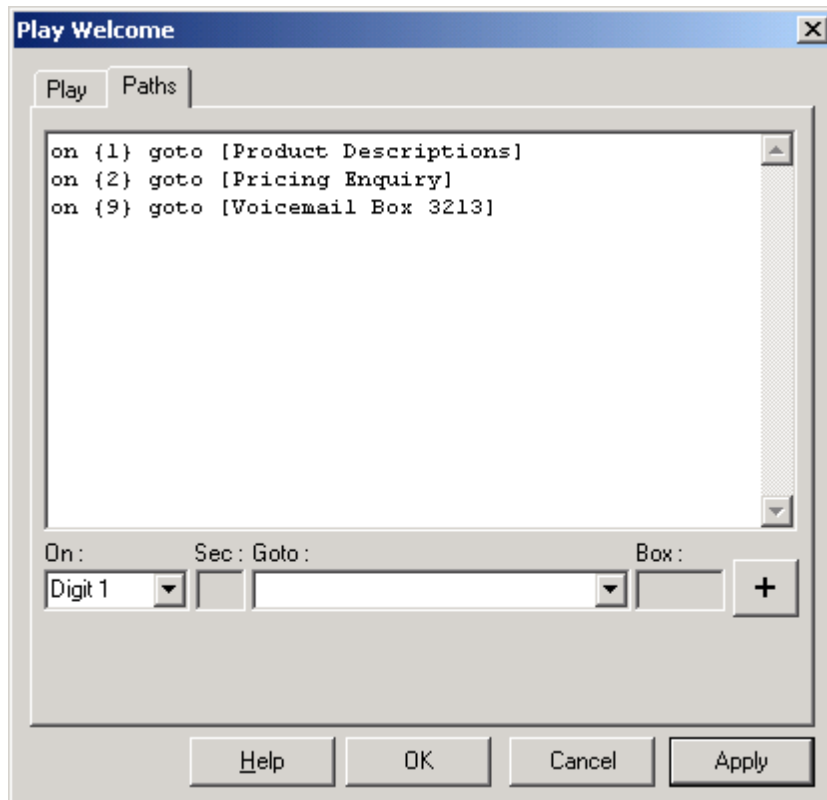
The text entered in the "Text to speak" box will be spoken. XML tags can be used when specifying the text to be spoken. For full details on the XML tags which can be used please refer to Microsoft's SAPI 5 specification, or to your TTS engine's reference.



If a text file can be specified in the "Sound File" text box then VoiceGuide will read in the text file and will speak (TTS) the contents of that file.

Paths

Paths specify which module the caller will go to if a matching response is made by the caller. eg:



A Timeout path is taken if no key has been pressed within the specified number of seconds after the last replay of sound file(s) has completed. If no Timeout path is specified and no selection is made within 10 seconds of the last message replay finishing, VoiceGuide will hang up the call.

While sound file is playing, keypresses can be used to skip forward/back or to adjust playback speed and volume. Special PLAY ACTION type paths need to be specified to achieve this:

```
On {Trigger} Goto [PLAY ACTION - Pause]
On {Trigger} Goto [PLAY ACTION - Resume]
On {Trigger} Goto [PLAY ACTION - Jump Forward]
On {Trigger} Goto [PLAY ACTION - Jump Back]
On {Trigger} Goto [PLAY ACTION - Jump To Start]
On {Trigger} Goto [PLAY ACTION - Jump To End]
On {Trigger} Goto [PLAY ACTION - Speed Increase]
On {Trigger} Goto [PLAY ACTION - Speed Decrease]
On {Trigger} Goto [PLAY ACTION - Speed Reset]
On {Trigger} Goto [PLAY ACTION - Volume Up]
On {Trigger} Goto [PLAY ACTION - Volume Down]
```

eg:

```
On {4} Goto [PLAY ACTION - Jump Back]
```

etc.

The skip forward/back size can be adjusted by setting a value of following Result Variables in the Evaluate Expression module:

```
$RV[playaction_skipbackbytes]
$RV[playaction_skipforwardbytes]
```

Playing DTMF tones and Hookflash

The Play module can also play DTMF tones, and send Hookflash signals. To specify the DTMF tones to send, type in the DTMF tone sequence in the "Sound File" text box. An exclamation mark "!" indicates a hookflash, and a comma "," indicates a pause, whilst characters "0-9" and "A-D" are used to indicate DTMF tones

When using Voice Modems you can only specify either a hookflash or DTMF tones in a Play module. This means that in order to have the modem send a hookflash followed by DTMF tones you will need to use two Play modules one after another. The first Play module would just contain the hookflash, and the next module will contain the DTMF string. You should then use a Timeout path to link the two modules together (eg: On {Timeout 1} Goto [Dial Number]). In this situation using the Timeout path allows you to specify exactly how long after the hookflash will the DTMF tones be dialed. Pause characters cannot be used at all with most Voice modems.

When using Dialogic cards any combination of DTMFs, hookflashes and pauses can be specified on the one line and the Dialogic card will play them correctly.

Automated DNIS based file selection

VoiceGuide allows per-DNIS sound file specification.

If DNIS is supplied on the call then VoiceGuide will see if a "DNIS-tagged" derived filename exists. The "DNIS-tagged" derived filename is the original filename with a "_DNIS" inserted just before the ".wav" in the filename.

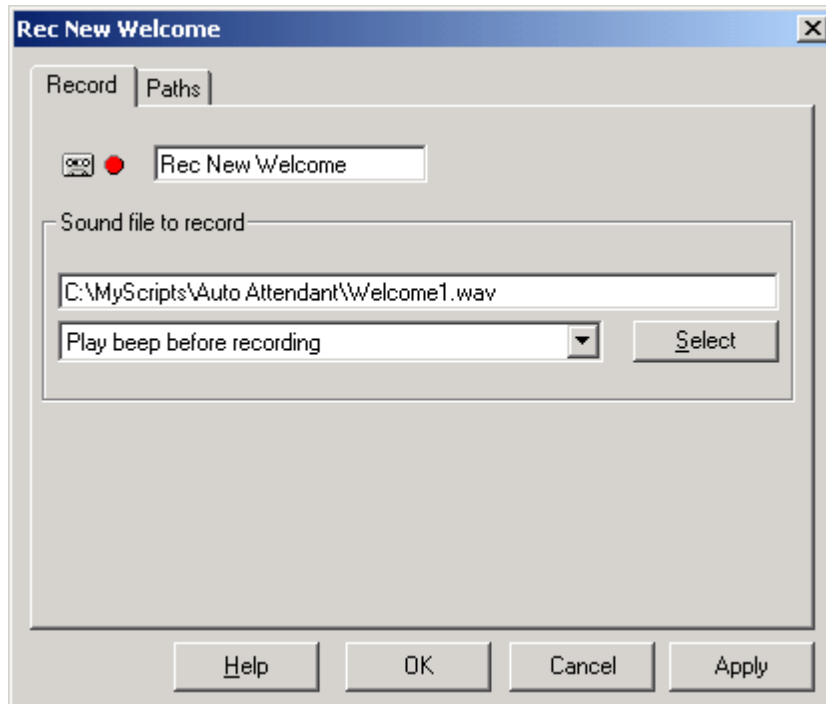
eg. if the Play module is configured to play a file `welcome.wav` and the DNIS on the call is 1234 then VoiceGuide will first search for file `welcome_1234.wav` If `welcome_1234.wav` does not exist then VoiceGuide will play the sound file `welcome.wav`

NB: DNIS is an expression indicating the number dialed by the caller, and is only really available on ISDN and VoIP systems. On analog systems a close replacement of this functionality can be achieved by using \$RVs within the filename to be played, and setting RV value previous based on Inband Signaling or user selection etc.

Record

The Record module will record a sound file.

The Record module can also be used to do Call Recording. It can be used to record the conversation between two callers connected together using "Dial and Conference" (tromboned transfer) or 3-way-call. For more information see [here](#).



Recording will finish when:

- a user settable timeout is reached, or
- a key that has a path defined for it is pressed, or
- silence is detected, or
- if the caller hangs up.

In the 'Record file details' text box you may:

Specify the full filename of the destination file:

Recording will be copied to the specified destination. If a file with this name exists already then it will be overwritten.

Specify just the directory in which the recordings are to be placed

A filename will be generated and the recording will be placed in the specified directory. The directory name must finish with a "\"

Leavethe entry blank

A filename will be generated and the recording will be placed in the script's directory

If the destination filename is not specified the following file name will be generated by VoiceGuide:

`MMDDHHNNSS_LineId_CallerID.wav`

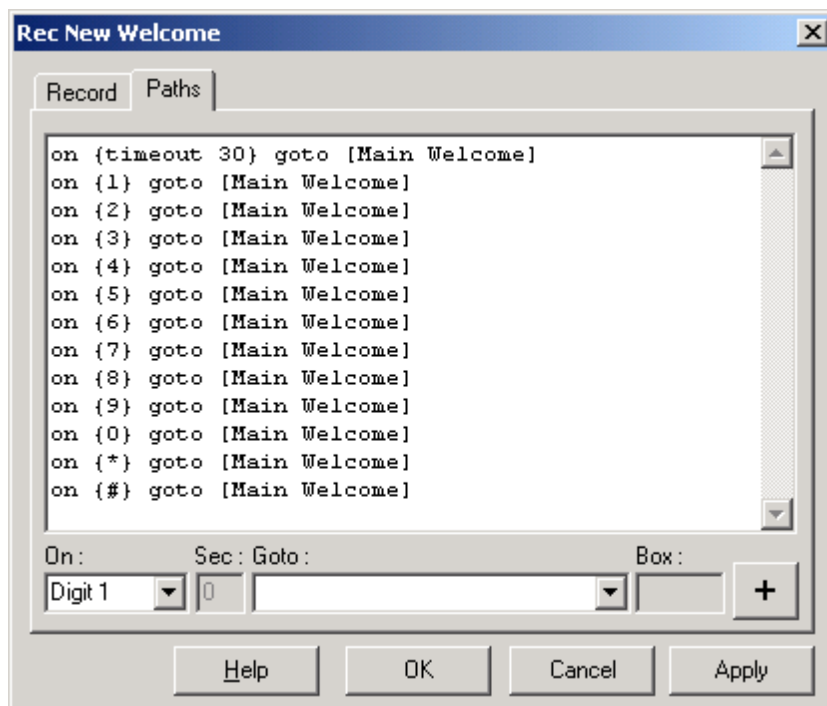
where:

MM	2 digit month
DD	2 digit day
HH	2 digit hour
NN	2 digit minute
SS	2 digit second
LineId	on which line the call arrived
CallerID	Caller ID of the caller who left the message

The record module is usually used to allow remote recording of a file which is used by other parts of a script or other scripts. For messages left by callers a Voicemail box is usually used.

The Timeout path is used to specify the maximum length of recording.

The [paths](#) screen below shows that the recording can go on for 30 seconds, after which the caller will be sent to the Main Welcome module. If the caller presses any of the keys listed, the recording will terminate and the caller will be sent to the Main Welcome module.



Silence Detection

The recording will also stop if silence is detected. The following type of path can be used to tell VoiceGuide which module to go to next if a silence is detected;

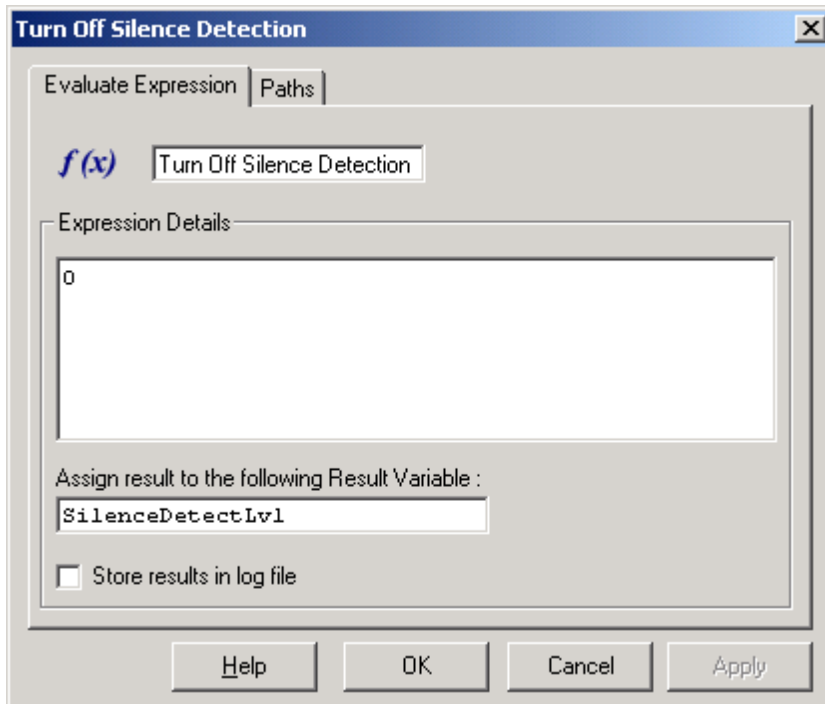
```
On {silence} Goto [module title]
```

Minimum silence length and volume parameters can be set by editing the VG.INI

configuration file, in section [PlayRecordConfig]. From VG.INI file:

```
;SilenceDetectLength: 40=4 seconds  
SilenceDetectLength=40  
SilenceDetectLevel=10
```

The silence detection settings can be set from within the script as well, by assigning values to Result Variables SilenceDetectLvl and SilenceDetectLen using the Evaluate Expression module. eg. To disable silence detection from within the script use:



Result Variables

`$RV[ModuleTitle]` will store the filename of the recorded sound file. Includes the full directory path.

`$RV[ModuleTitle_EndRecCause]` will store the reason why recording was stopped.

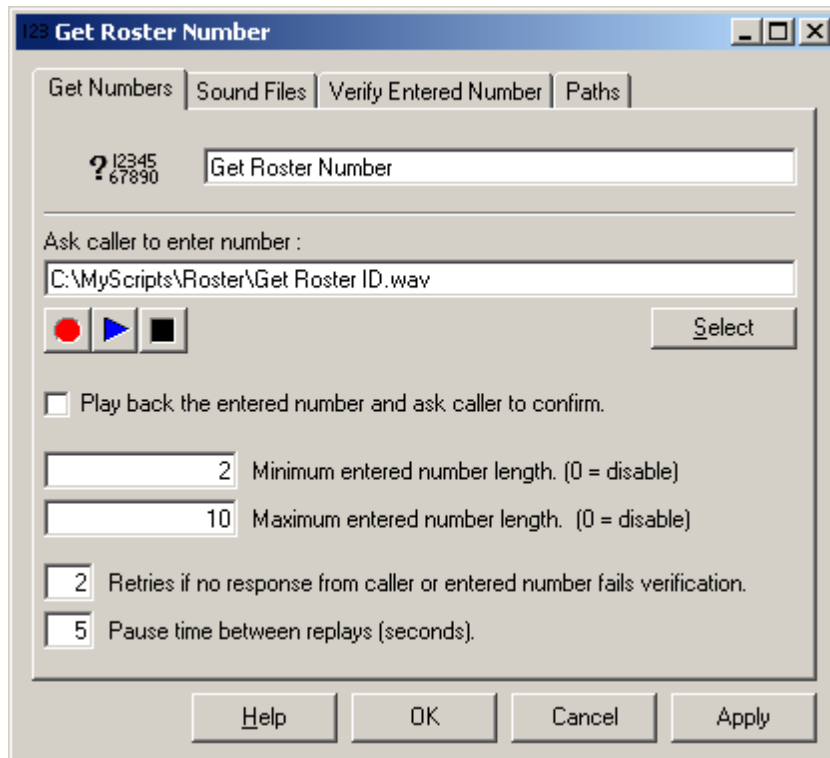
`$RV[ModuleTitle_RecLen100ms]` will store the recording length in 100ms units (eg. a value of 75 equals seven and a half seconds).

Call Recording - Recording Conversations

The [Record_2Lines_Start](#) function can be used to record both sides of the conversation. Record_2Lines_Start supports recording of both sides of a bridged conference call. Record_2Lines_Start also supports recording of both sides of the Caller<->IVR connection - recording what IVR plays to the caller and the callers responses.

Get Numbers

The Get Number module plays a sound file, and gathers a number sequence response from the caller. Caller can indicate that they have finished entering the number by pressing the "#" key, or module can complete when number reaches a pre-set length or times out awaiting further input. Module can optionally run a verification check on the entered number. Module can also optionally play back the number sequence for customer verification.



Sound files

Multiple sound files can be selected to be played by separating the successive files by commas. eg:

`prompts\1.wav,prompts\2.wav`

would result in two files from the prompts subdirectory being played (1.wav and 2.wav) one after another. If you do not want a sound file to be played at all you can also specify "none" in the sound file text field.

If Text to Speech (TTS) is enabled then a text file can be specified in the "Sound File" text box and VoiceGuide will read in the text file and will speak (TTS) the contents of that file.

Play back entered number and ask caller to confirm

If this option is selected and the caller has entered something then VoiceGuide will play back the entered number and ask the caller to confirm it. The caller will need to press '1' to confirm the entry, any other keypress will result in VoiceGuide asking for the caller to enter the number again.

Minimum entered number length

The minimum length of there entered number which this module expects to receive from user.

Maximum entered number length

The maximum length of the entered number which this module should wait for. Once this many digits have been entered the module will immediately perform the next action - be it taking the best matching path or running any selected confirmation and verification options.

Verify Entered Number Tab

Quite often it is desirable to determine if the entered number satisfies some criteria, and if it does not then the option should be given to the caller to enter the number again. A VBScript can be specified here to check if the number passes any required checks/tests.

For example to check that the entered number is between 13 and 16 characters long and begins with either a 3 or a 4 the following VBScript would be used:

```
set vg = CreateObject("vgServices.CommandLink")
iLen = Len("$RV_ENTEREDNUMBER")
sFirstChar = Left("$RV_ENTEREDNUMBER",1)
if iLen<13 or iLen>16 or (sFirstChar <> "3" and sFirstChar <> "4") then
    sResult = "verify_failed"
else
    sResult = "verify_passed"
end if
vg.Run_ResultReturn $RV_LINEID, sResult
set vg = Nothing
```

For more information on VBScripts and how they are used from within VoiceGuide please refer to this Help file's entry on Run VBScript module.

As a quick-start guide lets just say here that this line should be at the beginning of every VBScript used to verify the number:

```
set vg = CreateObject("vgServices.CommandLink")
```

and these lines should be at the end:

```
vg.Run_ResultReturn $RV_LINEID, sResult
set vg = Nothing
```

the value of sResult should be set within the script to be either "verify_passed" or "verify_failed".

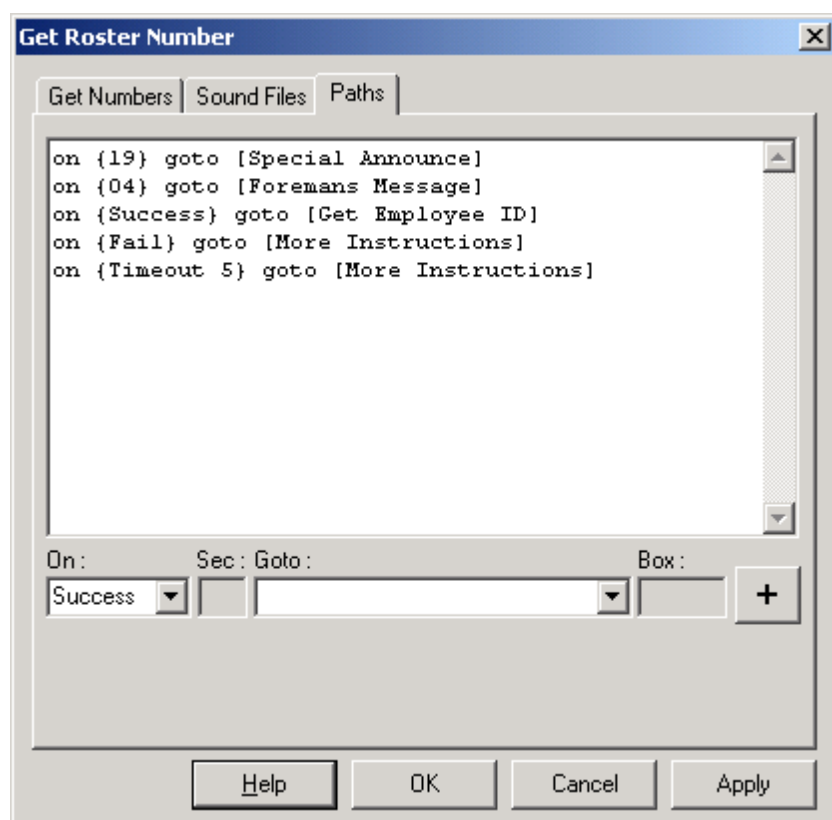
This is the method by which VoiceGuide is informed whether the verification of the entered number passed or failed.

If "verify_failed" is returned back to VoiceGuide then VoiceGuide will either play the sound file(s) specified in the "file to play if entered number is wrong length or verification failed" text box, or if that text box is not set it will just play the main prompt asking the caller to enter the number again. If the verification script starts playing another file before returning "verify_failed" then VoiceGuide will just wait for caller to start entering the number again, without starting to play any other sound files. This allows customized advice sound files to be played depending on the cause of failed verification.

\$RV_ENTEREDNUMBER contains the number just entered by the caller. In the verification script use \$RV_ENTEREDNUMBER instead of the \$RV[ModuleName] style \$RV. When the verification script is ran the \$RV[ModuleName] style result variable has not yet been set.

Paths

Here is an example of valid paths:



Success Path

If any of the paths match the entered number exactly then that path is taken, otherwise the 'Success' path will be taken.

If the 'Success' path is not defined and none of the other paths match the entered number exactly then the caller will be asked to enter the number again.

Fail Path

Taken if the caller has not entered any numbers, ie. caller just presses the "#" key or VoiceGuide times out awaiting first number entry.

If the 'Fail' path is not defined then the system will hang up the call.

Timeout Paths

The timeout between key presses is set by default to 6 seconds. Default timeout values used by this module can be changed in VG.INI file - in the [moduleGetNbrs] section.

If a 'Timeout' path is specified then that path overrides the default inter-digit key press timeout, and that path will be taken whenever a timeout occurs awaiting a key press.

'Exact Match' Paths

If any of the paths match the entered number exactly then that path is taken immediately. It will be taken immediately even if the path does not satisfy the minimum and maximum number length limits. The "Confirm entered number" option will also be ignored, and no verification script will be ran.

Note: Manual editing of the path trigger entry loaded by script designer in the {} brackets will be necessary to specify the 'exact number' used. ie. the `exact_number` placeholder loaded by the script designer needs to be replaced with the actual number combination to trigger on.

'*' Paths

If an "on {*} " path is defined then pressing the * key results in VoiceGuide assigning a "*" to the Result Variable for the module and going down the "on {*} " path. This feature can be used to easily implement a "press * to correct" option - just point the "on {*} " path back to the same module and then when caller presses the * key they will be able to just start entering the number again from start.

If a "on {*} " path is not specified then the * key will be included in the string of numbers captured by the module.

Result Variables

Main RVs created by the 'Get Numbers' module:

`$RV[moduleTitle]`

Stores the characters entered by caller.

`$RV[moduleTitle_PathTaken]`

Stores which path was taken when exiting the module. This could be "success", "fail", "Timeout", "*" or the entered number if the exact path match was made.

Notes

1. When entering numbers it is usually best to ask callers to press a # key or * key to indicate they have finished entering the number. Using such terminating characters allows you to act immediately after the # or * is pressed, without the need to rely on timing-out awaiting next keystroke to determine when caller has finished entering number.

Say Numbers

The Say Numbers module can speak the supplied number as:

Digits	"12345" would be spoken as "one two three four five". Up to forty digits will be spoken.
Amount Dollars	"12345" would be spoken as "twelve thousand three hundred forty five dollars". Amounts up to twelve digits long will be spoken.
Amount Cents	"12345" would be spoken as "one hundred twenty three dollars forty five cents". Amounts up to fourteen digits long will be spoken.
Amount Cents, Decimal Point	"12" would be spoken as "twelve dollars zero cents". "12.3" would be spoken as "twelve dollars thirty cents". "12.34" would be spoken as "twelve dollars thirty four cents". Amounts up to fourteen digits long will be spoken. This setting is usually used for speaking amounts retrieved from databases.
Number	"12345" would be spoken as "twelve thousand three hundred forty five". Numbers up to twelve digits long will be spoken.
DateMMDD	"0123" would be spoken as "twenty third January". Only the first four digits of the supplied string are looked at.
DateDDMM	"2301" would be spoken as "twenty third January". Only the first four digits of the supplied string are looked at.
DateMMDDHHNN	"01231456" would be spoken as "twenty third January two fifty six pm". Only the first eight digits of the supplied string are looked at.
DateDDMMHHNN	"23011456" would be spoken as "twenty third January two fifty six pm". Only the first eight digits of the supplied string are looked at.
TimeHHNN	"1745" would be spoken as "five forty five pm", "0730" would be spoken as "seven thirty am"
TimeHHNN 24	"1745" would be spoken as "seventeen forty five"

[Result Variables](#) can be used when specifying what number is to be spoken. Sound files to be played before and after the spoken number can also be selected. Result Variables can also be used in those filenames.

If a filename is specified then VoiceGuide will read in the file contents and read the data specified in the first line of the file.

If **Digits**, **Amount** or **Number** options are selected, and the number to be spoken starts with a "-", then the word "minus" will be spoken first before speaking the rest of the number.

If **Date** or **Time** options are selected, and the supplied number is the wrong length or does not make sense in the context of the selected option (eg: the supplied number is 8933 and the DateMMDD option is selected) then nothing will be played and the Fail path will be taken. If the Fail path is not defined then the Success path will be taken. If neither the Fail or Success paths are defined then VoiceGuide will hang up the call.

The Date/Time checking feature allows the Say Number module to be used for verifying validity of any caller-entered times and dates. The Time Switch module can also be used for checking the validity of date/time entered by caller.

Sound files played before & after the number

The sound files used in the module have to be of the same format as VoiceGuide current 'system' sound files (usually found in VoiceGuide's \system\voice\ subdirectory). This would be either "ALaw 8kHz 8bit Mono", or "ULaw 8kHz 8bit Mono".

Multiple sound files can be selected to be played by separating the successive files by commas. eg:

```
prompts\1.wav, prompts\2.wav
```

would result in two files from the prompts subdirectory being played (1.wav and 2.wav) one after another.

Changing the way Say Number module speaks numbers

Users can edit the way in which the numbers are spoken, or even add their own Say Number functions.

In VoiceGuide v5 and v6 the way the numbers are spoken can be changed by editing the file: `lib_num2wav.vbs` located in VoiceGuide's \system\vbs\ subdirectory. This is useful if it is required to say numbers/amounts in a language other than English. Please read the `lib_num2wav.vbs` file for more information.

In VoiceGuide v7 a DLL file is used to generate the list of WAV files needed to speak the number. The DLL is called `vgLib_SayNumbers.dll` and is located in VoiceGuide's \system\dll\ subdirectory. The source code and full VS2005 C# project required to build the DLL can be found in this VoiceGuide subdirectory: \system\dll\Source\vgLib_SayNumbers. After building the new DLL just place it in VoiceGuide's \system\dll\ subdirectory and restart VoiceGuide in order for VoiceGuide to start using the new DLL.

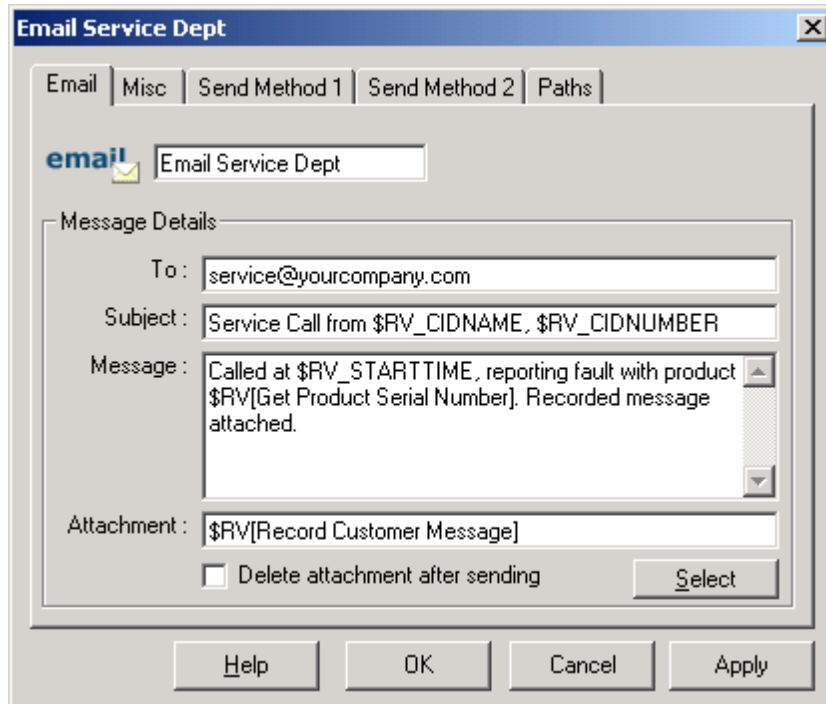
In VoiceGuide v7 you can also select to have the system use the `lib_num2wav.vbs` to generate the spoken number by changing the VG.INI file setting. In VG.INI, section `[moduleSayNumbers]`, change the entry `WavListGenerator` from `DLL` to `VBScript`.

Send Email

This module will send an Email message.

You can specify the Destination, Title, Message Body and an Attachment File. [Result Variables](#) can be used.

Example:



If the message needs to be sent to multiple email addresses the individual addresses must be separated by semicolons ";" or commas ",".

Other information may be set in the Misc tab, including return email address, CC, BCC.

The two Send Method tabs specify which SMTP servers (primary and backup) the email should be sent through.

Please note that many SMTP gateways require that you specify a valid return email address before they will accept and send the email.

VoiceGuide can send emails by connecting to an SMTP server, or a server that supports SSL secure SMTP connections started using STARTTLS (TLS Security). SMTPS servers are not supported.

Most STARTTLS connections are made using port 587. If a different port is to be used for a STARTTLS connection then the server name should be prefixed with `ssl:` to indicate that STARTTLS is to be used.

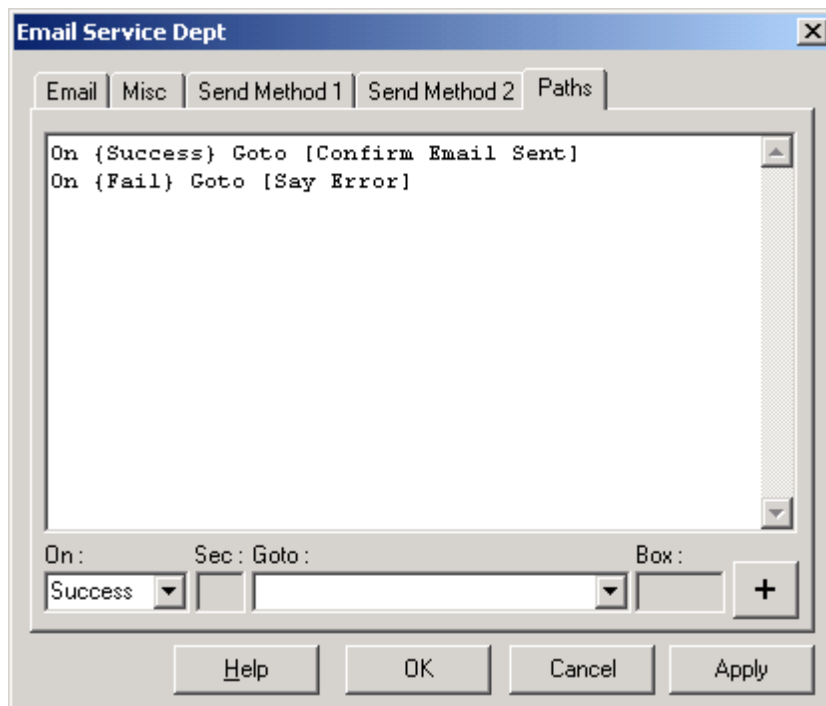
eg:

`ssl:smtp.gmail.com`

Gmail supports STARTTLS connections for outgoing emails, so if you encounter problems with your local SMTP server not supporting STARTTLS then a free Gmail account can be used to send out the

emails.

If the email was queued to be sent then the Success path will be taken. If there were problems with queuing the email then the Fail path will be taken. Note that a 'Success' result does not indicate that email was successfully sent out or delivered. A 'Success' result just indicates that the email was queued for sending.



What the above paths do:

On {Success} Goto [Confirm Email Sent]
On {Fail} Goto [Say Error]

Go to module called 'Confirm Email Sent'
Go to module called 'Say Error'

Result Variables

The following Result Variables are created by this module:

`$RV[module title]`

Will store the destination email address.

`$RV[module title_Subject]`

Will store the email message subject.

`$RV[module title_Message]`

Will store the email message body text.

`$RV[module title_Attachment]`

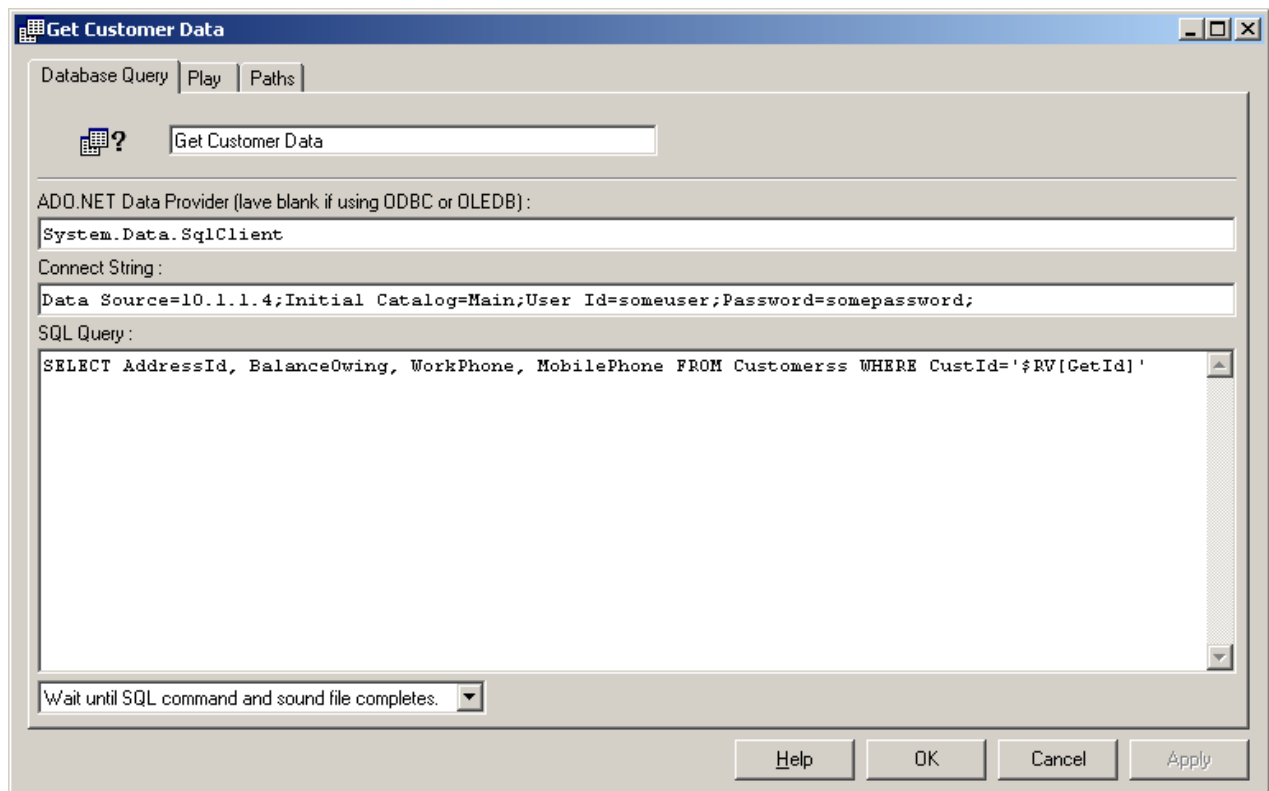
Will store the filename of the attached file sent.

`$RV[module title_SendResult]`

Will store whether sending of the email was successful or not. Stores 'OK' if the message was sent, or stores an error code if the message could not be sent. An error code of 3 means that Microsoft Outlook/Exchange/Messaging was not running at the time.

Database Query

The Database module can access databases and data files. Any information retrieved is made available for use by subsequent modules.



Following technologies can be used to interact with databases:

- .NET Data Providers (ADO.NET)
- ODBC
- OLE DB

All databases support at least one of the above interfaces.

Module Parameters

.NET Data Provider	Used to specify the .NET Data Provider to use. Can be left blank if using ODBC or OLE DB.
Connect String	Connect String to use.
SQL Query	The SQL statement to run, or the range of cells to retrieve from Excel.

Result Variables can be used in all of the above fields.

Below are some examples illustrating the different Connect String which can be used to access various databases. More examples of connect strings can be found at: <http://www.connectionstrings.com>

.NET : MS SQL Server

Data Provider: `System.Data.SqlClient`

Connect String: `Data Source=myServerAddress;Initial Catalog=myDataBase;User Id=myUsername;Password=myPassword;`

.NET : MySQL

Data Provider: `MySql.Data.MySqlClient`

Connect String: `Server=myServerAddress;Database=myDataBase;Uid=myUsername;Pwd=myPassword;`

ODBC : MS SQL Server

Connect String: `Driver={SQL Server Native Client 10.0};Server=myServerAddress;`

`Database=myDataBase;Uid=myUsername;Pwd=myPassword;`

ODBC : MySQL

Connect String: `Driver={MySQL ODBC 5.1 Driver};Server=myServerAddress;Database=myDataBase;`

`User=myUsername;Password=myPassword;Option=3;`

ODBC : Oracle

Connect String: `Driver={Oracle in OraHome92};Server=myServerAddress;Dbq=myDataBase;`

`Uid=myUsername;Pwd=myPassword;`

ODBC : DSN Data Sources

Connect String: `DSN=myDataSourceName;`

NOTE: on 64 bit systems the 32-bit "ODBC Data Source Administrator" must be used to create the data source. The 32 bit "ODBC Data Source Administrator" can be found at `C:\Windows\SysWOW64\odbcad32.exe`

OLEDB : MS Access

Connect String: `Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\mydatabase.accdb;`

OLEDB : MS Excel

Connect String: `Provider=Microsoft.ACE.OLEDB.12.0;Data Source=c:`

`\myFolder\myExcel2007file.xlsx; Extended Properties="Excel 12.0 Xml;HDR=NO;IMEX=1";`

Query Results

The results of the database query are accessible using Result Variables. The following Result Variables can be used to access data retrieved by a Db Query module:

`$RV[ModuleTitle_ColumnIndex_RowIndex]`

If more than one item was requested in the query, the Column Index can be used to access the individual items in each retrieved row.

eg: module LookupContactDetails uses a query: `SELECT cust_telephone_mobile, cust_email FROM Customer WHERE CustId='$RV[Get CustId]'`

After running above query `$RV[LookupContactDetails_1_1]` can be used to access the contents of the `cust_telephone_mobile` column, and `$RV[LookupContactDetails_2_1]` can be used to access the contents of the `cust_email` column for customer 1.

`$RV[LookupContactDetails_1_2]` can be used to access the contents of the `cust_telephone_mobile` column for customer 2, etc.

The following Result Variable can be used to access the number of database entries which were returned:

`$RV[`

Play Tab

If "Wait Until SQL Command Completes" option is selected, the sound file specified in the Play tab will be played while the database query is performed.

Paths:

Three paths can be selected from this module:

True	Some data was retrieved.
False	No data was retrieved.
Timeout	Query retrieval took too long and a specified Timeout path is taken.

SQL Commands

For a complete reference SQL see the Access Help files, or other Database reference books. The few examples below are included to demonstrate simple applications.

Retrieving Data

As an example, let's have an application where after entering the postal code, the caller would like to know how many clients live in this postal area code area, and then be told their telephone number, and amounts which they owe.

Suppose the caller has entered the postal area code in a Get Numbers module titled 'Ask for ZIP'. We can now use a Result Variable to use the entered value in a query.

The SQL below will retrieve selected details of all clients living in this area code:

```
SELECT AddressId, BalanceOwing, WorkPhone, MobilePhone FROM Addresses WHERE  
PostalCode='$RV[Ask for ZIP]'
```

The count of how many clients were found to live in this postal code is accessible via \$RV[Get ClientDetails_RowCount]. The first clients details would be accessible using \$RV[GetClient Details_1_1] through to \$RV[Get Client Details_4_1]

Say Number modules can be used to play the numeric information, like the balance and telephone numbers. TTS can be used to play the text information.

Inserting Data

The following SQL statement shows how to insert data into Payments table. In this example the Payments table has the following fields (amongst others): CustomerID, PayAmount, CardNumber, ExpDate. The values to be inserted into the database have been entered by the user in previous modules and Result Variables are used to in the expression. The result variables will be replaced by VoiceGuide by the actual entered data before the SQL statement is executed.

```
INSERT INTO Payments (CustomerID, PayAmount, CardNumber, ExpDate) VALUES ('$RV  
[GetCustId]', '$RV[GetPayAmount]', '$RV[GetCardNumber]', '$RV[GetExpiryDate]')
```

Updating Data

The following SQL statement shows how to modify data in the Payments table. The Payments table has the following fields (amongst others) CustomerID, PaymentStatus. The result variables will be replaced by VoiceGuide by the actual entered data before the SQL statement is executed.

```
UPDATE Payments SET PaymentStatus='Paid' WHERE CustomerID='$RV[GetCustId]'
```

In the examples above we have used single quotes as part of the WHERE and VALUES sections of the SQL statement. Single quotes should be used when referring to string or text fields in the database. When referring to number fields the single quotes should not be used: eg:

```
UPDATE Products SET UnitPrice=$RV[AddToPrice] WHERE ProductID=$RV[GetProductId]
```

Running Stored Procedures

Stored Procedures can be ran using the SQL EXEC statement:

```
EXEC UpdateStatusToPaid '$RV[GetCustId]'
```

More advanced Stored Procedure calls can be performed using VBScript - for which the 'Run VBScript' module can be used.

Working with BLOBs

VoiceGuide allows BLOB handling. BLOBs are usually used to store sound or graphics files or other large data files.

To specify a BLOB data element at insert the following needs to be to be specified as part of the SQL statement:

```
<BLOB><file>filename</file></BLOB>
```

eg:

```
INSERT INTO VmMsgs (Vmb, MsgKey, WavBlob) VALUES (1111, '$RV[MsgID]', <BLOB><file>$RV[Rec]</file></BLOB>)
```

To specify a BLOB data element for retrieval the following needs to be to be specified as part of the SQL statement:

```
<BLOB><dbcolumn>ColumnName</dbcolumn><file>filename</file></BLOB>
```

eg:

```
SELECT Vmb, <BLOB><dbcolumn>WavBlob</dbcolumn><file>c:\retrievedBLOB.wav</file></BLOB>  
FROM VmMsgs WHERE MsgKey='$RV[MsgID]';
```

BLOB reading/writing is only supported if .NET Data Providers are used to connect to

the database.

A sample script which writes and reads .WAV sound files as BLOB to/from database is provided in VoiceGuide's "scripts\Database BLOB WriteRead" subdirectory.

Books on SQL

A good introductory book on SQL is:

"SQL Queries for Mere Mortals" by Michael J. Hernandez, John Viescas
ISBN 0201433362

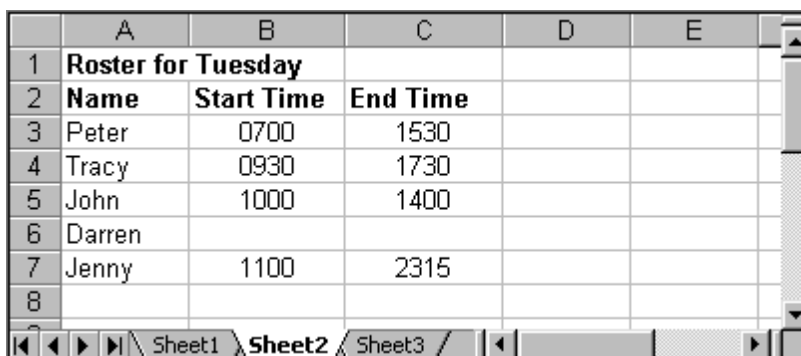
Microsoft Excel Example

The DB Query module can also retrieve data directly from an Excel file.

Excel range query specification follows this format :

```
SELECT * FROM [SheetName$StartCell:EndCell]
```

For an application which allows people to query a daily roster the Excel spreadsheet may look something like this:



	A	B	C	D	E
1	Roster for Tuesday				
2	Name	Start Time	End Time		
3	Peter	0700	1530		
4	Tracy	0930	1730		
5	John	1000	1400		
6	Darren				
7	Jenny	1100	2315		
8					

The caller would enter their Roster Number - which corresponds to the row in the Excel spreadsheet, and then we could retrieve the information in the 'start time' and the 'end time' columns using the following query:

```
SELECT * FROM [Sheet2$B$RV[Get Roster Number]:C$RV[Get Roster Number]]
```

The returned data would be saved in Result Variables. If the database query module's title was GetTimes then the Start Time would be accessed using \$RV[GetTimes_1_1] and the End Time would be accessed using \$RV[GetTimes_2_1]

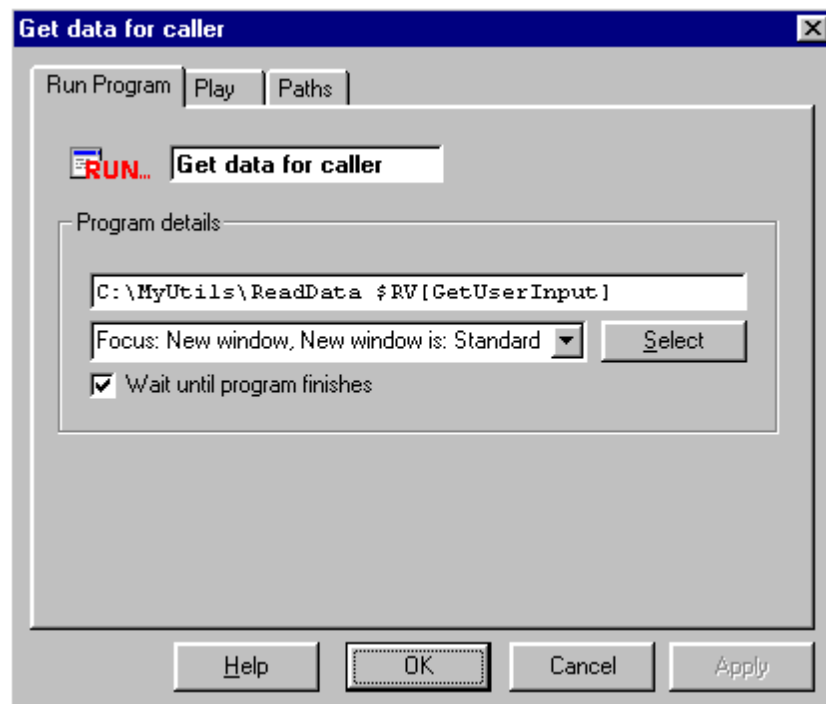
If we had each day of the week on a different page, we could also ask the customer to enter the day that they want to enquire about, and then our Query would be:

```
SELECT * FROM [Sheet$RV[GetDay]$B$RV[Get Roster Number]:C$RV[Get Roster Number]]
```

Please note that to retrieve only the value of a single cell (say B5 on Sheet2) the cell address still needs to be specified as a range, so the cell location expression should be: "Sheet2\$B5:B5" (Using "Sheet2\$B5" will result in an error returned by Excel)

Run Program

This module will run the specified program. [Result Variables](#) can be used when specifying the program's name and the parameters passed to the program.



Run Program Tab

The Program name and parameters passed to the program are specified here.

Play Tab

If "Wait Until Program Finishes" option is selected, the sound file specified in the Play tab will be played while the started program is executing. The playing of the sound file will be stopped as soon as the the program finishes.

Paths Tab

Wait until program finishes

Don't Wait

If the result file contains some data, then the "Success" path is taken. If the result file is empty then the "Fail" path is taken. If the program could not be started then the "Fail" path is taken.

If the program was started successfully then the "Success" path is taken. If the program could not be started then the "Fail" path is taken.

After reading in the result file, VoiceGuide will rename the file by adding a ".last" to the filename.

Timeout Paths

When used in Run Program module, a Timeout Path can be used to limit the maximum amount of time the module will wait for the called program to finish (when "Wait until program finishes" option is selected). When the timeout is reached the module will close the program it has called, and go the module specified in the Timeout Path.

Returning data from ran program back to VoiceGuide

There are two ways in which the ran program can return data back to VoiceGuide:

- Calling VoiceGuide's COM/WCF functions. eg: [Run_ResultReturn](#)

or:

- Creating a 'Result File' which is then read in by VoiceGuide.

Result Files

If "Wait Until Program Finishes" option is selected, then VoiceGuide will wait until the program completes and afterwards VoiceGuide will look for the result file which can be optionally created by the called program. The file can be created in the script's directory, or in the directory where the program was ran from, or in VoiceGuide's \data\ subdirectory.

The file must be named `VGRUNRESULT_LineNumber.TXT`

The *LineNumber* is the ID of the line which is executing the Run Program module, and it can be passed to your program using the `$RV_LINEID` Result Variable.

The result file can be used to return results back to VoiceGuide. This file should contain the list of the Result Variables that you want VoiceGuide to read in, along with their values.

The syntax for the `VGRUNRESULT_LineNumber.TXT` file is:

```
[Result Variable Name]{Result Variable Value}
```

Multiple Result Variables can be specified. For Example, the following contents:

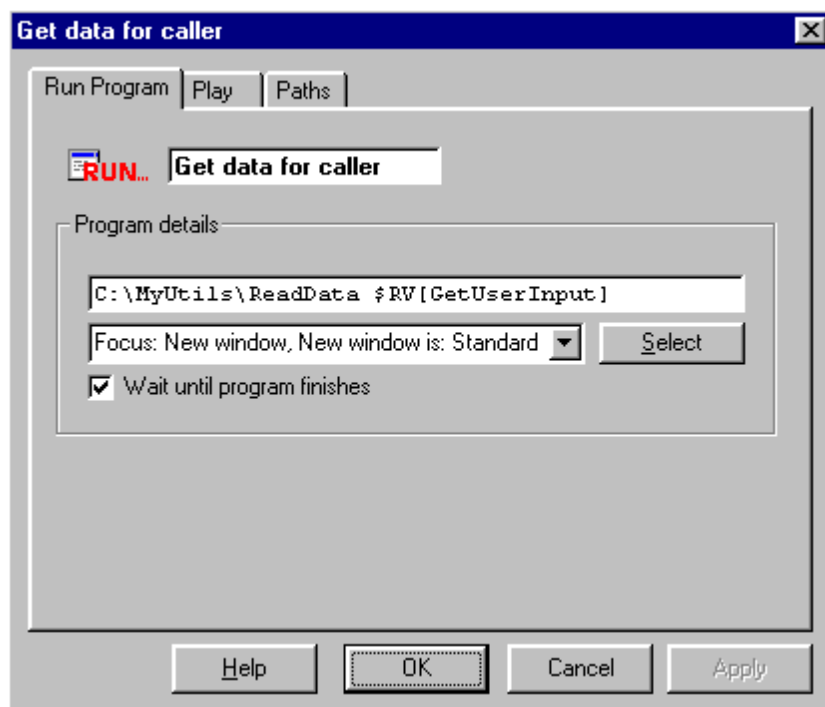
```
[RvName1]{Value1}[RvName2]{Value2}[RvName3]{Value3}[RvName4]{Value4}
```

Would return 4 Result Variables to VoiceGuide, whose values would be whatever is contained in the curly brackets. All Result Variables must be listed on a single line.

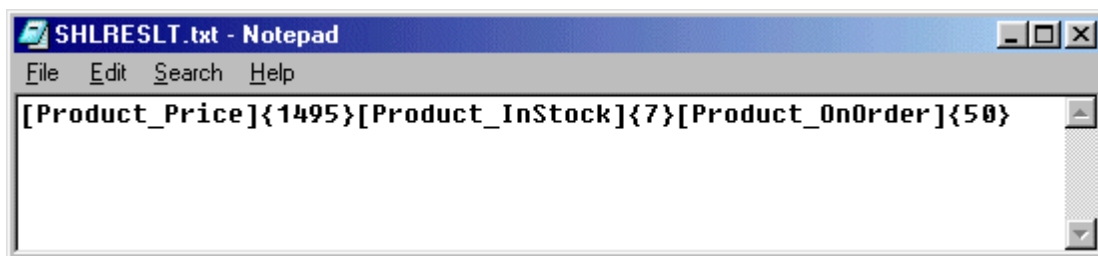
Example 1:

A custom written utility **ReadData.exe** to retrieve information about a product has been written and the system has a Get Number Sequence module called **Get Data Type** which is used to ask caller to select what data is to be retrieved. The ReadData.exe is then

called with the information entered in Get Data Type passed as a parameter.



Result File is then be used to return the retrieved data. ReadData.exe writes the following data into the file:



The above will create 3 new Result Variables in VoiceGuide, which can be accessed using:

```
$RV[Product_Price]  
$RV[Product_InStock]  
$RV[Product_OnOrder]
```

whose values are:

```
1495  
7  
50
```

respectively. These Result Variables can then be used in Say Numbers modules to speak out this information to the caller.

Example 2:

The Run Program module can be used to write out data that the caller has entered into

a file. The command below will write out the 3 pieces of data gathered in Get Number Sequence modules out into the file output.txt:

```
command.com /c echo $RV[GetUserId], $RV[GetProductID], $RV[GetOrderAmount] >> c:\output.txt
```

Use ">" instead of ">>" if you would like to overwrite the output.txt file instead of appending to it.

Always specify the full path of the file to which the output is echoed - if you do not specify the full path sometimes the output file will not be in the scripts' directory.

Note: If writing to same log file from subsequent modules in same script or from multiple lines then it's preferable to use the [Run VB Script](#) module to write log files, as the DOS echo command does not handle concurrent writing to the same file well.

Example 3:

Sound Recorder application specifying that it should play the specified sound file and then close afterwards using this command line:

```
sndrec32 /play /close c:\test.wav
```

The Media Player application can also be used for this purpose:

```
mplayer /play /close c:\test.wav
```

If you are finding that the program does not run as VoiceGuide cannot find it then try specifying the full path to where your program is on your system. For example, the Media Player program on some systems is actually in:

```
C:\Program Files\Windows Media Player\mplayer2.exe
```

which would make the actual command line required to be used in the Run Module:

```
C:\Program Files\Windows Media Player\mplayer2.exe /play /close c:\test.wav
```

Example 4:

Send a message to another computer, alerting a particular user of an event. eg. when the caller on the VoiceGuide system is about to be transferred to their extension.

```
net send someusername "VG call from $RV_CIDNUMBER"
```

Example 5:

Running batch files (.bat) : To run a batch file from a Run Program module it is best to create a shortcut (a link) to the batch file and run the shortcut itself. You may

need to use the actual name of the file as shown when doing a DIR listing in the DOS Command Prompt window.

Example 6:

Concatenation of sound files can be done using the "shntool" utility.

Eg: to concatenate wav files 1.wav, 2.wav and 3.wav into one file 1_2_3.wav you would run the following command from a "Run Program" module:

```
command /c shntool join -stdout 1.wav 2.wav 3.wav > 1_2_3.wav
```

shntool can be obtained from:

<http://www.etree.org/shnutils/shntool/>

<http://shnutils.freeshell.org/shntool/>

Run VB Script

This module will run a VBScript or ECMAScript / JScript.

[Result Variables](#) can be used throughout the script.

After starting the script VoiceGuide can:

- > Continue, without waiting for script to complete.
- > Wait until script completes.
- > Wait until script and sound file completes.

Continue, without waiting for script to complete option

VoiceGuide will immediately continue down the "Success" path once script was started successfully. If script could not be started successfully then the "Fail" path is taken. If the path that VoiceGuide should be following is not defined then VoiceGuide will hang up the call.

Wait until script completes option

VoiceGuide will wait until the script finishes running, or a response from the executing script is received.

Script can send responses back to VoiceGuide while it is executing by calling one VoiceGuide's ActiveX/COM functions. The functions which are considered to return a result back to VoiceGuide are: Run_ResultReturn(), Script_Gosub(), Script_Goto(), Script_Return(). For more information on VoiceGuide's ActiveX/COM interface please see the COM Interface section of Help file.

Once VoiceGuide detects that a script has completed and no COM response was received beforehand then VoiceGuide will see if a "Result File" has been created by the script. If one has been created then VoiceGuide will read in it's contents and then determine what to do next based on the contents of the file.

The syntax of the Result File is the same as that used by the Run Program module. Please refer to the Run Program module Help file's section for more information.

A "Success" or Result Variable list must be returned to VoiceGuide (either through a Run_ResultReturn COM function or a Result File) in order for it to go down the Success path. If no result is returned then the "Fail" path is taken after the VBScript completes. Calls to Script_Gosub(), Script_Goto() and Script_Return() functions result in immediate running of the next module.

Any sound files still playing when VBScript completes will be stopped.

Wait until script and sound file completes option

VoiceGuide will wait until the script finishes running (or a response from the executing script is received) and until the sound file playing completes. Sound file to play can be specified either using the "sound file to play" text box or can be started from within the script itself using the Play_Start

COM function.

Other functionality is similar to the "Wait until script completes" option above.

Limiting maximum execution time

To limit the length of time the script is allowed to run for, a Timeout Path should be defined. If the script does not finish before the timeout occurs the script will be terminated and the timeout path will be taken. The timeout value is in seconds and should not be set to 0 - a value of 0 will result in the script being aborted immediately after it is started, without giving it any chance to run. Minimum timeout value used should be 2 seconds.

Play Tab

If "Wait Until script Finishes" option is selected, the sound file specified in the Play tab will be played while the started program is executing.

Writing VBScript

A pretty good book on VBScript is: ["VBScript in a Nutshell" by Ron Petrusha, Matt Childs, Paul Lomax.](#)

VBScript editors which you can use to develop your scripts before moving them over the the Run VBScript module can be found at: www.vbsedit.com.

Following are some examples of VBScripts which can be ran in the Run VBScript module. Extensive use of VBScripting is also made by the VoiceGuide's Voicemail system. See VoiceGuide's \system\vm\ directory.

Using ECMAScript / JScript instead of VBScript

To have the module execute JScript the first line of the script must begin with this:

```
//ECMAScript OR
```

```
//JScript
```

This indicates to VoiceGuide that the contents of this module are JScript and not the default VBScript.

Using Result Variables in VBScripts

Before running the VBScript VoiceGuide will first see if there are any Result Variables specified within the script, and if there are then VoiceGuide will first replace them with their current values and run

the resulting script then.

If the Result Variable is not defined then VoiceGuide will replace it with an empty string. For this reason it is often desirable to have the Result Variables used with quotes specified around them.

Eg: If Caller ID information did not arrive then VoiceGuide will replace \$RV_CIDNUMBER with an empty string before letting the VBScript run.

So, if Caller ID information did not arrive this code:

```
If $RV_CIDNUMBER = "" Then
    CallerID = "Unknown"
Else
    CallerID = $RV_CIDNUMBER
End if
```

would be changed to

```
If = "" Then
    CallerID = "Unknown"
Else
    CallerID =
End if
```

Which you can see is not valid VBScript and hence an error will occur.

But if you write the code using quotes around the RVs, like this:

```
If "$RV_CIDNUMBER" = ""
    Then CallerID = "Unknown"
Else
    CallerID = "$RV_CIDNUMBER"
End if
```

and Caller ID information did not arrive then the resulting VBScript which VoiceGuide actually runs would be:

```
If "" = "" Then
    CallerID = "Unknown"
Else
    CallerID = ""
End if
```

Which is a valid VBScript and when running it no errors will be generated and the script will work as expected.

Result Variables should not be directly used in places where replacement with a blank would cause a syntax error. Instead a variable which gets set to the RV value beforehand should be used in those situations. That way there is no syntax error and the check for whether RV holds any data can be made earlier in the script and the appropriate action taken if RV does not hold a value.

Notes:

After the Result Variables have been replaced in the VBScript, the new VBScript can be optionally saved in VoiceGuide's **\temp** sub-directory, named as **vbs_LineId_ThisCallScriptIndex.vbs** Viewing this file will allow you to confirm that Result Variable replacement was done correctly. Saving to .vbs file can be set in VG.INI, in [moduleRunScript] section.

VBScript .vbs files can be executed by using the Windows' wscript.exe utility.

On 32-bit systems the .vbs files can be ran by just double clicking on them.

On 64-bit systems please open a command prompt (may need to be an 'Administrator Command Prompt') then change directory to the VoiceGuide \temp\ sub-directory and run this command line:

```
C:\Windows\SysWow64\WScript.exe vbs_x_y.vbs
```

Where vbs_x_y.vbs is the filename of the .vbs file which you wish to run.

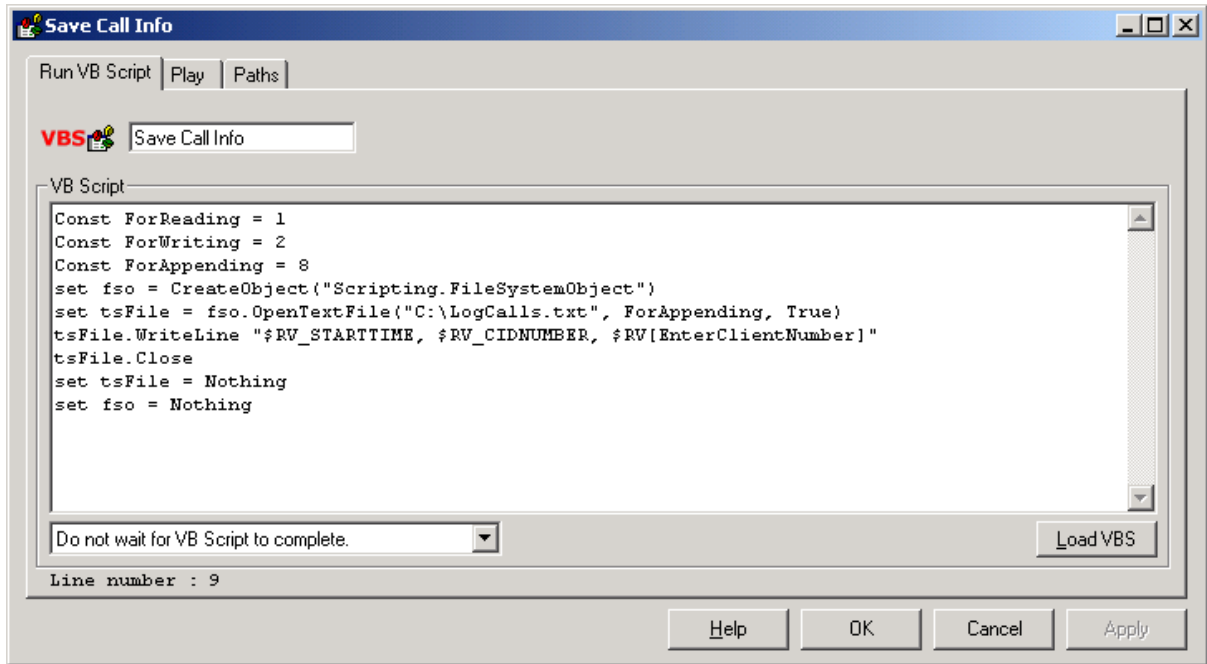
VoiceGuide is 32 bit software so any VBScripts that use the VoiceGuide COM interface need to be ran using the 32 bit version of VBScript interpreter.

On 64-bit systems if you change the Registry Key Computer\HKLM\SOFTWARE\Classes\VBSFile\Shell\Open\Command to use the 32-bit C:\Windows\SysWow64\WScript.exe then the .vbs files can be ran by just being double-clicked.

Example : Save information to a file

The VBScript below will append a line of text to the file C:\LogCall.txt - the line of text will contain information about start of the call, caller's telephone number and information entered by caller in module "EnterClientNumber". [Result Variables](#) are used in this script to allow information from VoiceGuide to be visible to the VBScript.

```
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8
set fso = CreateObject("Scripting.FileSystemObject")
set tsFile = fso.OpenTextFile("C:\LogCalls.txt", ForAppending, True)
tsFile.WriteLine "$RV_STARTTIME, $RV_CIDNUMBER, $RV[EnterClientNumber]"
tsFile.Close
set tsFile = Nothing
set fso = Nothing
```



Example : Read information from a file

The VBScript below will read the contents of file C:\CurrentPrices.txt and will assign them to VoiceGuide Result Variable \$RV[ReadInPrice].

```
set fso = CreateObject("Scripting.FileSystemObject")
set fileUnitPrice = fso.OpenTextFile("C:\CurrentPrice.txt")
sEntireFile = fileUnitPrice.ReadAll
set fileUnitPrice = Nothing 'always deallocate after use...
set fso = Nothing
set vg = CreateObject("vgServices.CommandLink")
vg.RvSet $RV_LINEID, "ReadInPrice", sEntireFile
vg.Run_ResultReturn $RV_LINEID, "success"
set vg = Nothing 'always deallocate after use...
```

Example : Read information from Excel

The VBScript below retrieves information from an Excel spreadsheet.

```
Dim xlApp, xlBook, xlSht
Dim filename, value1, value2, value3, value4

filename = "c:\Warehouse.xls"
```

```

Set xlApp = CreateObject("Excel.Application")
set xlBook = xlApp.WorkBooks.Open(filename)
set xlSht = xlApp.activesheet

value1 = xlSht.Cells(2, 1)
value2 = xlSht.Cells(2, 2)

'the MsgBox line below would be commented out in a real application
'this is just here to show how it works...
msgbox "Values are: " & value1 & ", " & value2

xlBook.Close False
xlApp.Quit

'always deallocate after use...
set xlSht = Nothing
Set xlBook = Nothing
Set xlApp = Nothing

```

Example: Save information to Excel

The VBScript below saves information to an Excel spreadsheet.

```

Dim xlApp, xlBook, xlSht
Dim filename, value1, value2, value3, value4

on error resume next

filename = "c:\warehouse.xls"

Set xlApp = CreateObject("Excel.Application")
set xlBook = xlApp.WorkBooks.Open(filename)
set xlSht = xlApp.activesheet

xlApp.DisplayAlerts = False

'write data into the spreadsheet
xlSht.Cells(2, 2) = "New Data"

xlBook.Save
xlBook.Close SaveChanges=True
xlApp.Close

```

```
xlApp.Quit
```

```
'always deallocate after use...  
set xlSht = Nothing  
Set xlBook = Nothing  
Set xlApp = Nothing
```

Example : Creating Result File

The VBScript below demonstrates how the \$RV_LINEID Result Variable is used to generate a Result file from which the data is read back into VoiceGuide. Please note that using the COM function Run_ResultReturn() is a preferable way of returning information to VoiceGuide (it's faster) - but a result file can be used if there is no other way.

```
Dim iIndexDow, iIndexNasdaq, iIndexSP500  
  
'Do some work here to retrieve the data and initialize  
'the iIndexDow, iIndexNasdaq and iIndexSP500 variables  
  
strResultVariables= "[IndexDow]{" & iIndexDow & "}" & _  
"[IndexNasdaq]{" & iIndexNasdaq & "}" & _  
"[IndexSP500]{" & iIndexSP500 & "}"  
  
iRet = WriteResultFile(strResultVariables)  
  
Function WriteResultFile(strResult)  
    Const ForReading=1, ForWriting=2, ForAppending=8  
    filename = "VGRUNRESULT_$RV_LINEID.TXT"  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    Set ts = fso.OpenTextFile(filename, ForWriting, True)  
    ts.WriteLine(strResult)  
    ts.Close  
    WriteResultFile=0  
  
    'always deallocate after use...  
    set ts = Nothing  
    set fso = Nothing  
end function
```

It is recommended that the full path to the result file be specified, otherwise the Windows' current 'default' path will be used by the file subsystem - and that does not always point to

the same path as the script's.

Example : MS Access Database read/write

Retrieving data from an MS Access database using an SQL query, and then updating the same record with new values:

```
'ADO related const values from ADOVBS.INC file, usually found in: C:\Program
Files\Common Files\System\ado
Const adOpenForwardOnly = 0
Const adOpenKeyset = 1
Const adOpenDynamic = 2
Const adOpenStatic = 3

'other related const values
const vbGeneralDate = 0
const vbLongDate = 1
const vbShortDate = 2
const vbLongTime = 3
const vbShortTime = 4

set vg = CreateObject("vgServices.CommandLink")
set cn = CreateObject("ADODB.Connection")
set rs = CreateObject("ADODB.Recordset")
cn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\data\Ads.mdb"

if cn.State = 1 then
    vg.Admin_TraceLogAdd $RV_LINEID, 5, "vbs    connection to Ads.mdb made OK"
    set rs.ActiveConnection = cn
    rs.Open "SELECT TOP 1 AdID, Filename, PlayCount FROM AdList WHERE PlayCount <
PlayCountMax ORDER BY LastPlayTime", cn, adOpenStatic

    if rs.RecordCount > 0 then
        vg.Admin_TraceLogAdd $RV_LINEID, 5, "vbs    records found OK"
        iAdID = rs.Fields("AdID").Value
        sFilename = rs.Fields("Filename").Value
        iPlayCount = rs.Fields("PlayCount").Value

        'Do some other processing here. eg: vg.Play_Start $RV_LINEID, sFilename

        'now update the database
        iPlayCount = iPlayCount + 1
        strDateTime = "#" & FormatDateTime(Now, vbShortDate) & " " & FormatDateTime
(Now, vbLongTime) & "##"
        cn.Execute "UPDATE AdList SET PlayCount=" & iPlayCount & ", LastPlayTime="
```

```

& strDateTime & " WHERE AdID=" & iAdID
else
    'no records could be retrieved, we can specify to play something instead
here
    vg.Admin_TraceLogAdd $RV_LINEID, 5, "vbs    no records found"
end if

else
    vg.Admin_TraceLogAdd $RV_LINEID, 5, "vbs    connection to Ads.mdb could not be
made"
end if

cn.Close
Set rs = Nothing
Set cn = Nothing
Set vg = Nothing

```

Example : MS SQL Server Database read/write

A script like this can be used :

```

set vg = CreateObject("vgServices.CommandLink")
set cn = CreateObject("ADODB.Connection")
set rs = CreateObject("ADODB.Recordset")

cn.Open "Provider=SQLOLEDB;Server=DBSERVER1;UID=user;PWD=user;Database=Ads"
'MSSQL authentication

if cn.State <> 1 then
    vg.Admin_TraceLogAdd iLineId, 5, "login LeadingAd connection to database
could not be made"
    vg.Run_ResultReturn iLineId, "fail"
else

    set rs.ActiveConnection = cn
    sSQL = "SELECT TOP 1 AdID, Filename, PlayCount, LastPlayTime FROM AdList
WHERE    PlayCount < PlayCountMax AND Active <> 0 ORDER BY LastPlayTime"
    rs.Open sSQL, cn, 3

    if rs.RecordCount <= 0 then
        rs.Close
        vg.Admin_TraceLogAdd iLineId, 5, "no records retrieved"
    else
        iAdID = rs.Fields("AdID").Value
    end if
end if

```



```

sFilename = rs.Fields("Filename").Value
iPlayCount = rs.Fields("PlayCount").Value
dateLastPlayTime = rs.Fields("LastPlayTime").Value
rs.Close

sSQL = "UPDATE AdList SET PlayCount=" & iPlayCount & ", LastPlayTime=" &
strDateTimeNow & " WHERE AdID=" & iAdID
vg.Admin_TraceLogAdd iLineId, 5, "sql=[" & sSQL & "]"
cn.Execute sSQL

sSQL = "INSERT INTO PlayLog (AdID, CallID, PlayDateTime, PlayedFrom)
VALUES (" & iAdID & ", 0, " & strDateTimeNow & ", 'AnswerCall')"
vg.Admin_TraceLogAdd iLineId, 5, " sql=[" & sSQL & "]"
cn.Execute sSQL
end if

cn.Close
end if
set rs = Nothing
set cn = Nothing
set vg = Nothing

```

Example : Stored Procedure defined in the database (no parameters).

Let's assume that you have defined in your database a stored procedure names `RetrieveUserNames` that returns a recordset but takes no parameters. (note this example assumes that the database connection `cn` and `VG.CommandLink` are defined outside this function) The following code will retrieve the records.

```

set cmd = CreateObject("ADODB.Command")
set rs = CreateObject("ADODB.Recordset")
With cmd
    .ActiveConnection = cn
    .CommandType = adCmdStoredProc
    .CommandText = "RetrieveUserNames"
    Set rs = .Execute
    'you can now access the individual fields in the recordset rs
End With
Set cmd = Nothing
Set rs = Nothing

```

Example : Stored Procedures defined in the database (with parameters).

Let's assume that you have defined in your database a parameterized stored procedure query named `UpdateUser` as follows:

```
UPDATE USERS
SET USERS.FIRSTNAME = [inFirstName], USERS.MIDDLEINITIAL = [inMiddleInitial],
USERS.LASTNAME = [inLastName], USERS.DEPARTMENT = [inDepartment],
USERS.EMAIL = [inEmail], USERS.TELEPHONE = [inTelephone],
USERS.EXTENSION = [inExtension]
WHERE (((USERS.USERNAME)=[inUserName]));
```

You should probably use a function like this as part of our VBScript to run this stored procedure (note this example assumes that the database connection and `VG.CommandLink` are defined outside this function):

```
Public Function UpdateUser(ByRef cn As Connection, ByVal Firstname As String,
ByVal MiddleInitial As String, _
                        ByVal LastName As String, ByVal Department As
String, ByVal Email As String, _
                        ByVal Telephone As String, ByVal Extension As
String, ByVal UserName As String ) As Boolean

    Const adParamInput = 1
    Const adParamOutput = 2
    Const adInteger = 3
    Const adCmdStoredProc = 4
    Const adExecuteNoRecords = 128
    Const adVarChar = 200

    Set cmd = CreateObject("ADODB.Command")
    Dim lngRecordsAffected As Long
    With adoCmd
        .ActiveConnection = cn
        .CommandType = adCmdStoredProc
        .CommandText = "UpdateUser"
        .Parameters.Append .CreateParameter("inFirstName", adVarChar,
adParamInput, 50, Firstname)
        .Parameters.Append .CreateParameter("inMiddleInitial", adVarChar,
adParamInput, 2, IIf(Len(MiddleInitial) > 0, MiddleInitial, Null))
        .Parameters.Append .CreateParameter("inLastName", adVarChar,
adParamInput, 50, LastName)
        .Parameters.Append .CreateParameter("inDepartment", adVarChar,
adParamInput, 50, IIf(Len(Department) > 0, Department, Null))
        .Parameters.Append .CreateParameter("inEmail", adVarChar, adParamInput,
50, IIf(Len(Email) > 0, Email, Null))
```

```

        .Parameters.Append .CreateParameter("inTelephone", adVarChar,
adParamInput, 50, IIf(Len(Telephone) > 0, Telephone, Null))
        .Parameters.Append .CreateParameter("inExtension", adVarChar,
adParamInput, 10, IIf(Len(Extension) > 0, Extension, Null))
        .Parameters.Append .CreateParameter("inUserName", adVarChar,
adParamInput, 50, UserName)
        .Execute lngRecordsAffected, , adExecuteNoRecords
        UpdateUser = CBool(lngRecordsAffected)
    End With

    Set cmd = Nothing

End Function

```

The only thing you need to do is to match the order and data type of the parameters that are 'appended' to the command object with those of the MS Access query.

Example : Retrieving data from an ASMX / SOAP Web Service

NOTE: The VoiceGuide Web Service module is the recommended way of querying web services.

The VBScript below retrieves the WSDL from the ASMX / SOAP Web Service provided by cdyne.com, and then calls the Web Service function GetCityWeatherByZIP.

In this example the ZIP code 10004 (New York, NY) was used.

Returned data is then formatted as a Result Variable list and returned to VoiceGuide.

Microsoft's SOAP Toolkit must be installed on system to use the MSSOAP.SoapClient30 COM object.

```

Set soapClient = CreateObject("MSSOAP.SoapClient30")
soapClient.MSSoapInit "http://wsf.cdyne.com/WeatherWs/Weather.asmx?WSDL"
Set node_list = soapClient.GetCityWeatherByZIP(10004)

For Each node In node_list
    sRv = sRv & "[" & node.nodeName & "]"{" & node.Text & "}"
Next

set soapClient = Nothing

```

```
set node_list = Nothing
```

```
'MsgBox sRv
```

```
set vg = CreateObject("vgServices.CommandLink")
```

```
vg.Run_ResultReturn $RV_LINEID, sRv
```

```
set vg = Nothing
```

The Result Variables list returned would be something like this:

```
[Success]{true}[ResponseText]{City Found}[State]{NY}[City]{New York}  
[WeatherStationCity]{White Plains}[WeatherID]{4}[Description]{Sunny}  
[Temperature]{46}[RelativeHumidity]{17}[Wind]{NW16G26}[Pressure]{30.20F}  
[Visibility]{}[WindChill]{}[Remarks]{}]
```

Example : Retrieving data from a HTTP / REST Web Service using HTTP GET

NOTE: The [VoiceGuide Web Service module](#) is the recommended way of querying web services.

Request to a Web Service can be made directly to the service/function URL.

eg. a GET call to cdyne.com GetCityWeatherByZIP function can be made directly to this HTTP address:

<http://wsf.cdyne.com/WeatherWs/Weather.asmx/GetCityWeatherByZIP?ZIP=10004> and since the GetCityWeatherByZIP function returns data in XML format we can use the XML functions to to extract the required data.

```
Set Server = CreateObject("MSXML2.ServerXMLHttp")
```

```
Server.open "GET", "http://wsf.cdyne.com/WeatherWs/Weather.asmx/
```

```
GetCityWeatherByZIP?ZIP=10004", False
```

```
Server.setRequestHeader "Content-Type", "text/xml"
```

```
Server.send
```

```
set objXMLDoc = Server.responseXML
```

```
'MsgBox Server.Status & " " & Server.StatusText & vbCrLf & objXMLDoc.xml
```

```
Set objChildNodes = objXMLDoc.documentElement.childNodes
```

```
For Each node In objChildNodes
```

```
    sRv = sRv & "[" & node.nodeName & "]"{" & node.Text & "}"
```

```
Next
```

```
'MsgBox sRv

set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, sRv
set vg = Nothing
```

Example : Uploading data to HTTP / REST Web Service using HTTP POST

NOTE: The [VoiceGuide Web Service module](#) is the recommended way of querying web services.

Request to a Web Service can be made directly to the service/function URL.

eg. a POST call to cdyne.com GetCityWeatherByZIP function can be made directly to this HTTP address:

```
http://wsf.cdyne.com/WeatherWs/Weather.asmx/GetCityWeatherByZIP
```

and since the GetCityWeatherByZIP function returns data in XML format we can use the XML functions to to extract the required data.

```
Set Server = CreateObject("MSXML2.ServerXMLHttp")
Server.open "POST", "http://wsf.cdyne.com/WeatherWs/Weather.asmx/
GetCityWeatherByZIP", False
Server.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"
Server.send("ZIP=10004")

set objXMLDoc = Server.responseXML
'MsgBox Server.Status & " " & Server.StatusText & vbCrLf & objXMLDoc.xml

Set objChildNodes = objXMLDoc.documentElement.childNodes
For Each node In objChildNodes
sRv = sRv & "[" & node.nodeName & "]"{" & node.Text & "}"
Next
'MsgBox sRv

set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, sRv
set vg = Nothing
```

eg. a new ticket in the zendesk.com system can be created using a HTTP POST call to URI:

`https://yourname.zendesk.com/api/v1/tickets.xml`

with ticket data provided as data attached to the POST:

```
sPostData = "<ticket><subject>Ticket from VoiceGuide</subject><description>Please call back $RV_CIDNUMBER</description></ticket>"
Set Server = CreateObject("MSXML2.ServerXMLHttp")
Server.open "POST", "https://yourname.zendesk.com/api/v1/tickets.xml",
False, "youruser", "yourpassword"
Server.setRequestHeader "Content-Type", "application/xml"
Server.send sPostData

set objXMLDoc = Server.responseXML
MsgBox Server.Status & " " & Server.StatusText & vbCrLf & objXMLDoc.xml

set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, "[Server.Status]{" & Server.Status & "}
[Server.StatusText]{" & Server.StatusText & "]"
set vg = Nothing
```

Example : Screen Scraping from Web Page

If the required information is not available through a web service but is shown on the web page itself, a screen-scraping approach can be used.

The VBScript below retrieves stock market levels from www.finance.yahoo.com and returns this data to VoiceGuide. (Please see the Web Site Scraping demo script).

```
Set IE = CreateObject("InternetExplorer.Application")
With IE
    .RegisterAsDropTarget = False
```

```

.Visible = False
.Silent = True
.Navigate("finance.yahoo.com")
While .Busy
    'sleep
Wend
With .Document.Body
    readWwwHtml = .InnerHTML
    readWwwText = .InnerText
End With
End With
IE.Quit
Set IE = Nothing

iIndexDow = GetIntegerAfterLabel("Dow")
iIndexNasdaq = GetIntegerAfterLabel("Nasdaq")
iIndexSP500 = GetIntegerAfterLabel("S&P 500")

strResultVariables= "[MarketDow]{" & iIndexDow & "}" & _
"[MarketNasdaq]{" & iIndexNasdaq & "}" & _
"[MarketSP500]{" & iIndexSP500 & "}"

'make sure the returned data does not contain any commas
strResultVariables = replace(strResultVariables, ",", "")

set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, strResultVariables
Set vg = Nothing

function GetIntegerAfterLabel(strLabel)
    'it is assumed that the integer terminates with a decimal point
    iLblPos1= Instr(readWwwText, strLabel)
    iValuePos1 = iLblPos1 + len(strLabel)
    iValuePos2 = Instr(iValuePos1, readWwwText, ".")
    GetIntegerAfterLabel = mid(readWwwText, iValuePos1, iValuePos2-iValuePos1)
end function

```

Other Resources:

<http://en.wikipedia.org/wiki/VBScript>

<http://www.youtube.com/watch?v=oRM5osg7gGs>

<http://www.visualbasicscript.com>

Call Web Service

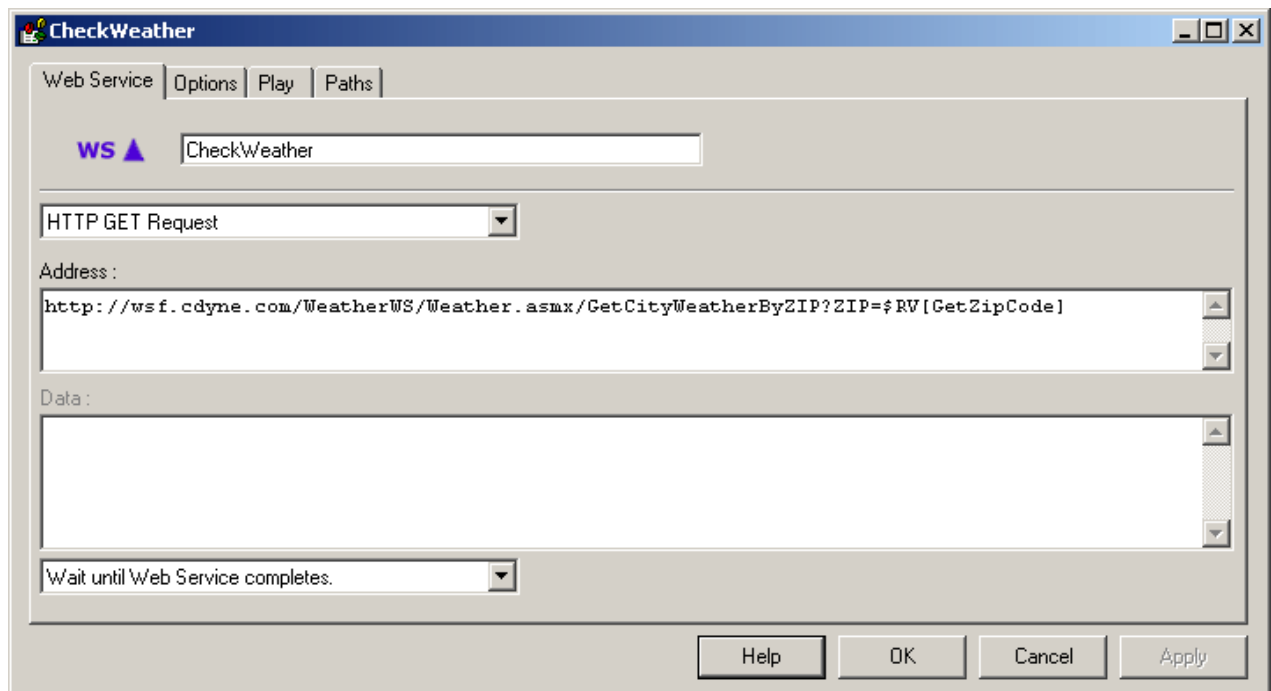
This module lets VoiceGuide communicate with Web Services.

For http Web Services the following HTTP request types can be selected:

- DELETE
- GET
- POST
- PUT
- UPDATE

Module supports setting of:

- URL Address
- Authentication
- Post Data
- Headers



After issuing the request to the Web Service, VoiceGuide can:

- > Continue, without waiting for web service query to complete.
- > Wait until web service query completes.
- > Wait until web service query and sound file completes.

Continue, without waiting for web service to complete : VoiceGuide will immediately continue down the "Success" path once the web service query was started successfully. If web service query could not be started successfully then the "Fail" path is taken. If the path that VoiceGuide should be following is not defined then VoiceGuide will hang up the call.

Wait until web service query completes : VoiceGuide will wait until the web service query completes.

Any sound files still playing when the web service query completes will be stopped.

Wait until web service and sound file completes : VoiceGuide will wait until the web service completes and until the sound file playing completes.

Limiting maximum execution time

To limit the length of time the system waits for the web service to return, a Timeout Path should be defined. If the web service does not return before the timeout occurs the web service query will be terminated and the timeout path will be taken. The timeout value is in seconds and should be set to a value of 1 or higher.

Play Tab

If "Wait Until Web Service Completes" option is selected, the sound file specified in the Play tab will be played while the web service query is performed.

Using Result Variables

Before running the web service VoiceGuide will first see if there are any Result Variables specified within the web service's HTTP address (URI), Post Data, Headers etc. and if present then VoiceGuide will first replace them with their current values.

If the Result Variable is not defined then VoiceGuide will replace it with an empty string.

Returned Result Variables

Data returned by Web Service is saved in Result Variables available for use throughout remainder of the call. The Headers of the returned response are also saved as Result Variables.

Paths

The following are valid path options:

- Success/Fail
- HTTP Response Codes
- Timeout

eg: The following HTTP Response Code paths are all valid:

```
on {200} goto [Say all went OK]
on {OK} goto [Say all went OK]
on {406} goto [Say trans not accepted]
on {Not Acceptable} goto [Say trans not accepted]
```

Script Examples

Sample script excerpts are provided in VoiceGuide's \Scripts\Web Service\ subdirectory. Script provided there demonstrate the use of the various HTTP request methods from within the VoiceGuide Web Service module.

Example : GET Request : Get Weather by ZIP code

Cdyne Corp. provides a web service that can be used to retrieve weather information at a particular ZIP code. The Web service can be called using the a HTTP GET request to the following URI:

```
http://wsf.cdyne.com/WeatherWS/Weather.asmx/GetCityWeatherByZIP?ZIP=[zip code]
```

eg:

```
http://wsf.cdyne.com/WeatherWS/Weather.asmx/GetCityWeatherByZIP?ZIP=10005
```

The response to this HTTP GET request is:

```
<?xml version="1.0" encoding="utf-8" ?>
<WeatherReturn xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema" xmlns="http://ws.cdyne.com/WeatherWS/">
<Success>true</Success>
<ResponseText>City Found</ResponseText>
<State>NY</State>
<City>New York</City>
<WeatherStationCity>White Plains</WeatherStationCity>
<WeatherID>4</WeatherID>
<Description>Sunny</Description>
<Temperature>46</Temperature>
<RelativeHumidity>17</RelativeHumidity>
<Wind>NW16G26</Wind>
<Pressure>30.20F</Pressure>
<Visibility />
<WindChill />
<Remarks />
</WeatherReturn>
```

The following Result Variables would be created by VoiceGuide for this response:

```
$RV[Success] - "true"
$RV[ResponseText] - "City Found"
$RV[State] - "NY"
$RV[City] - "New York"
$RV[WeatherStationCity] - "White Plains"
$RV[WeatherID] - "4"
$RV[Description] - "Sunny"
$RV[Temperature] - "46"
```

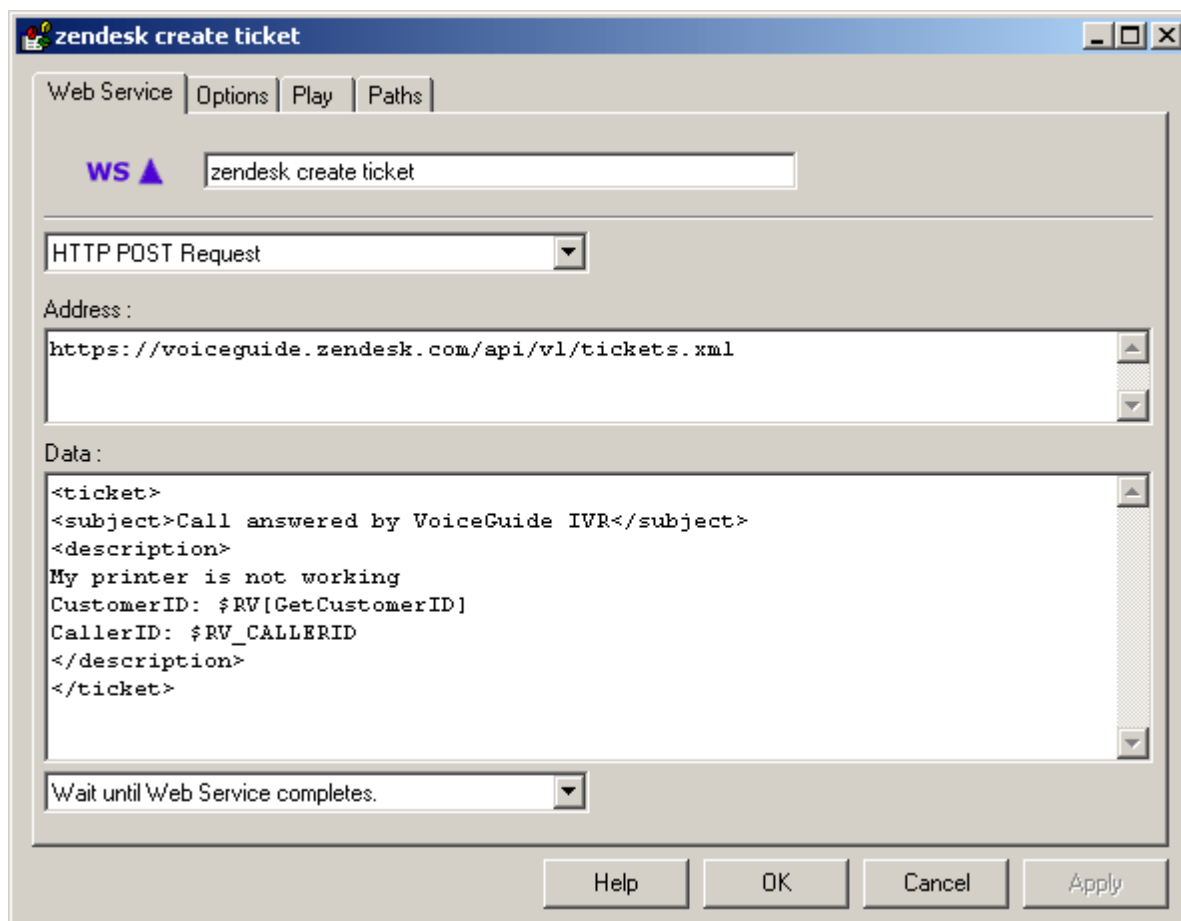
```
$RV[RelativeHumidity] - "17"  
$RV[Wind] - "NW16G26"  
$RV[Pressure] - "30.20F"  
$RV[Visibility] - ""  
$RV[WindChill] - ""  
$RV[Remarks] - ""
```

The above Result Variables can then be used for the remainder of the call.

Example : POST Request : Add Ticket to Zendesk

Zendesk (<http://www.zendesk.com/>) provides a HTTP REST API.

Following example shows how VoiceGuide's Web Service module can be used to add a new ticket to your Zendesk account:



NB. this POST request will require authentication, and hence would require that your own Zendesk account is set up first. The URL would need to be modified accordingly.

Example : POST Request to ASMX Web Service

Issuing a SOAP 1.2 POST request to an ASMX Web Service:

GetPaymentOptions

Web Service | Options | Play | Paths

WS ▲ GetPaymentOptions

HTTP POST Request

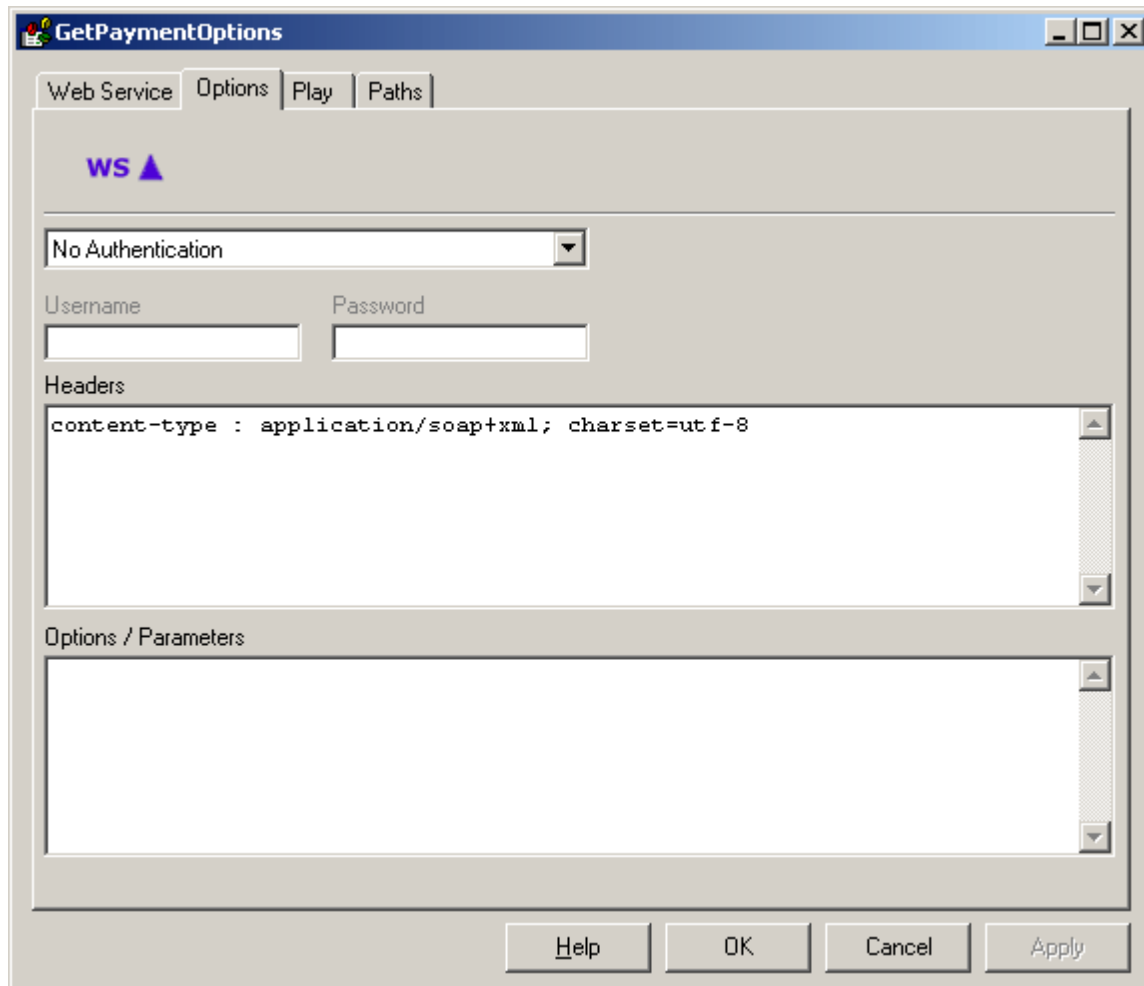
Address :
http://www.mywebsite.com/webservices/payments.asmx

Data :

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <LookUpAvailableOptions xmlns="http://tempuri.org/">
      <iCustomerID>$RV[CustID]</iCustomerID>
    </LookUpAvailableOptions>
  </soap12:Body>
</soap12:Envelope>
```

Wait until Web Service completes.

Help OK Cancel Apply



Then SOAP XML structures to use is displayed when the Web Services' ASXM URL is viewed in a web browser.

Most ASMX Web Services support SOAP 1.2 format. Issuing a SOAP 1.1 format POST request is only slightly different:

GetPaymentOptions SOAP 1.1

Web Service | Options | Play | Paths

WS ▲ GetPaymentOptions SOAP 1.1

HTTP POST Request ▼

Address :

http://www.mywebsite.com/webservices/payments.asmx

Data :

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <LookUpAvailableOptions xmlns="http://tempuri.org/">
      <iCustomerID>$RV[CustID]</iCustomerID>
    </LookUpAvailableOptions>
  </soap:Body>
</soap:Envelope>
```

Wait until Web Service completes. ▼

Help OK Cancel Apply

GetPaymentOptions SOAP 1.1

Web Service | Options | Play | Paths

WS ▲

No Authentication ▼

Username Password

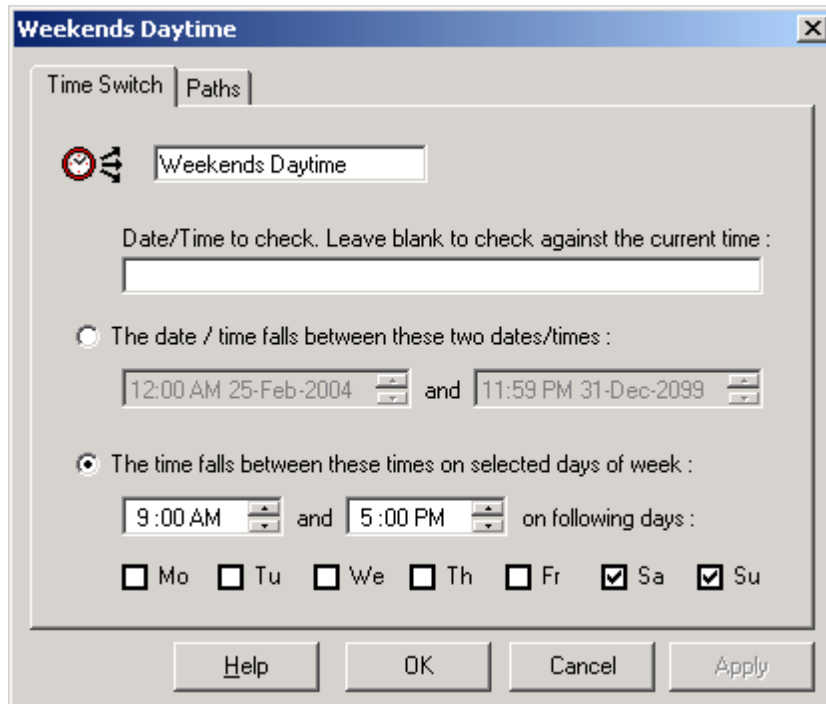
Headers

content-type : text/xml; charset=UTF-8

Options / Parameters

Time Switch

This module is used to direct the caller to a different part of the script based on Date or Time of Day.



The screenshot shows a dialog box titled "Weekends Daytime" with two tabs: "Time Switch" and "Paths". The "Time Switch" tab is active. It features a clock icon and a text field containing "Weekends Daytime". Below this is a text field labeled "Date/Time to check. Leave blank to check against the current time :". There are two radio button options. The first option, "The date / time falls between these two dates/times :", is unselected. It has two date/time pickers showing "12:00 AM 25-Feb-2004" and "11:59 PM 31-Dec-2099" separated by "and". The second option, "The time falls between these times on selected days of week :", is selected. It has two time pickers showing "9:00 AM" and "5:00 PM" separated by "and", followed by the text "on following days :". Below this are checkboxes for the days of the week: Mo, Tu, We, Th, Fr, Sa, and Su. The checkboxes for Sa and Su are checked. At the bottom are buttons for "Help", "OK", "Cancel", and "Apply".

The screenshot above shows a time range to covering 9am till 5pm on Saturday and Sunday. The Date/Time to Check field is left blank so the current time when the script is executed will be used. When specifying the 'times on selected days' the start time must be earlier in the day then the stop time.

Paths Chosen

The path "True" is taken if the current time falls within the specified time range, otherwise the "False" path is taken.

Date/Time to Check

Most of the time this field will be left blank as usually the intention is to check what is the current time and if it falls within the specified time range.

If the module is used to check whether a date/time entered by caller or retrieved from the database falls within a certain range then the date to be checked should be specified in this text field.

The format in which the date is to be specified will vary according to your computers settings for preferred date/time format. If specifying date and time then usually one of the following formats will be OK (hours are specified in a 24 hour format) :

MM/DD, YYYY HH:NN

DD/MM, YYYY HH:NN

MM/DD/YYYY HH:NN

DD/MM/YYYY HH:NN

HH:NN

If the date is not specified then current date is used.

How can the caller enter date/time? Usually by just entering the require information in a series of Get Numbers modules and then having all the data entered specified in the Date/Time to Check, eg:

`$RV[GetMonth]/$RV[GetDay], $RV[GetYearYYYY] $RV[GetHour24]:00`

If date is retrieved from database you should ensure that the correct format is supplied.

If the date/time format supplied is invalid then the "False" path is taken.

Note

Evaluate Expression module can also be used to set up time-switching.

Transfer Call

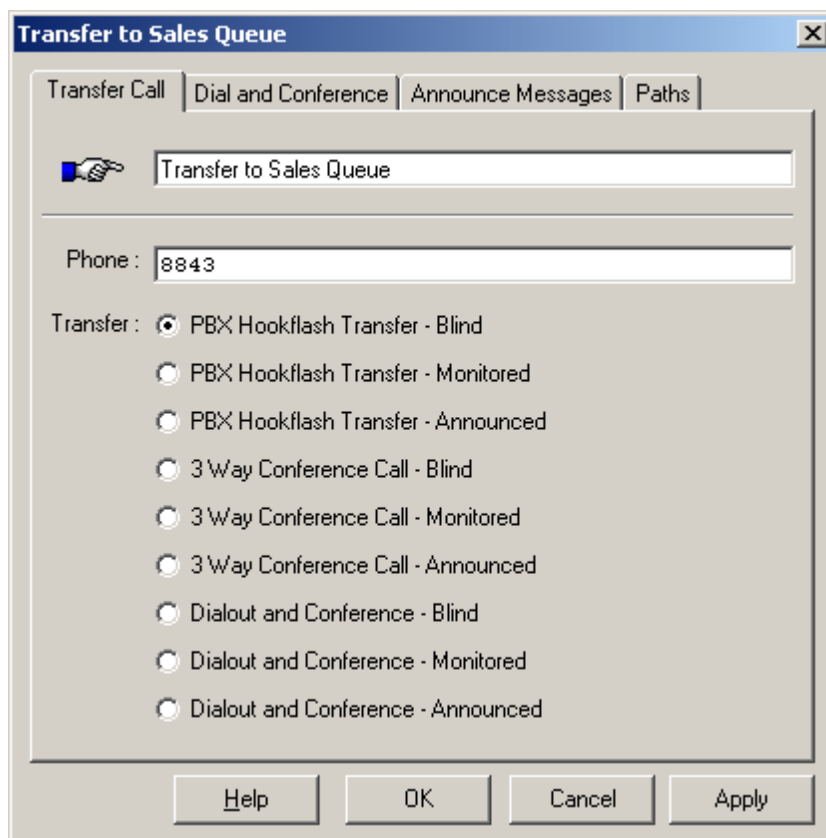
The Transfer Call module will transfer the call to another extension or telephone number.

All types of call transfers are supported:

- 'Hookflash' transfers
- 'Dial and Conference' / 'Trombone' / 'Bridge' transfers
- TBCT transfers (used on ISDN trunks)
- SIP REFER transfers (used on VoIP - SIP)

The 'Dial and Conference' ('Trombone' / 'Bridge') transfer uses two lines for the entire length of the call. In all other transfers types the line becomes free to take another call as soon as the transfer is made.

All PBXs support hookflash transfers, and some phone companies support hookflash transfers as well. You will need to check with your phone company to see if it supports hookflash transfers on it's lines. If your system is using telephone lines which do not support hookflash transfers and you do not have a PBX then you will need to use the "Dial and Conference" (Trombone / Bridge) transfer method.



When specifying the number to dial you can use commas ',' to indicate pauses in the dialing sequence, ie: to allow a pause between the hookflash and the telephone number being dialed add a couple of commas before the phone number. (eg: the text box in the screenshot above would read ", , 8843

Note that this can only be done on analog lines. On ISDN and VoIP connections commas cannot be used.

[Result Variables](#) can be used when specifying the telephone number or extension.

Hookflash Transfer - Blind

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Hangs up immediately

The caller will now be connected to another extension. VoiceGuide no longer monitors the call.

Some PBXs/Switches require that the destination extension starts ringing before VoiceGuide can hang up. When using such systems it may be necessary to add a comma or two at the end of the destination number to make VoiceGuide pause for a while after dialing the number and before hanging up the call.

Some PBXs/Switches require that the destination extension is answered before VoiceGuide can hangup. In such situations the "Announced" or "Monitored" transfer option will need to be used.

Hookflash Transfer - Monitored

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Listens to see if a call is answered by a human or an answering machine. If a human or answering machine answer is detected then VoiceGuide hangs up, which completes the transfer. If no answer is detected or the extension is busy then VoiceGuide will dial the 'retrieve call from announced transfer' signal (usually just a hookflash) and go down the Fail path.

Hookflash Transfer - Announced

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Plays a message announcing the Caller's ID, and awaits the selection from the called person whether to accept the call or not. "1" is used to accept the call, any other digit is used to decline the call. If "VoiceGuide for Dialogic" is used then the announce message will only start when VoiceGuide detects that the outgoing call has been answered.
4. If the call is accepted VoiceGuide will dial the 'complete announced call transfer' signal (usually nothing is needed) and hang up and allow the two new parties to continue conversation. If the Success path is defined then that path will be taken, but script should be designed in such a way that VoiceGuide hangs up soon, so that the two parties can speak to one another. If the call is declined, the called party does not answer, or the extension is busy then VoiceGuide will dial the 'retrieve call from announced transfer' signal (usually just a hookflash) and then the "Fail" path is taken. If other more appropriate path is defined then that path is taken instead.

SIP REFER Transfer (RFC 3515)

VoiceGuide v7 can perform SIP transfers. All that is required to perform the RFC3515 REFER transfer is to select "Blind Hookflash Transfer" option and type in the IP address of the transfer destination.

3 Way Conference - Blind

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Dials the signal as specified in 'Complete blind conference call' setting (usually "!3")
4. Waits until the call is finished.

VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- either party hangs up.
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

3 Way Conference - Monitored

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Listens to see if a call is answered by a human or an answering machine. If a human or answering machine answer is detected that VoiceGuide hangs up. If no answer is detected then VoiceGuide will dial the 'retrieve call from announced transfer' signal (usually just a hookflash).
4. If a human or answering machine answer is detected VoiceGuide will dial the 'complete announced conference' signal (usually "!3"). Then if the Success path is defined then that path will be taken, otherwise VoiceGuide will wait until end of call. If no answer is detected or the extension is busy then VoiceGuide will dial the 'retrieve call from announced conference' signal and then the Fail path is taken (If other more appropriate path is defined then that path is taken instead).

If call is accepted VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- either party hangs up.
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

3 Way Conference - Announced

1. Dials the signal specified (usually just a hookflash),
2. Dials the telephone number,
3. Plays a message announcing the Caller's ID, and awaits the selection from the called person whether to accept the call or not. "1" is used to accept the call, any other digit is used to decline the call. If "VoiceGuide for Dialogic" is used then the announce message will only start when VoiceGuide detects that the outgoing call has been answered.
4. If the call is accepted VoiceGuide will dial the 'complete announced conference' signal (usually

"!3"). Then if the Success path is defined then that path will be taken, otherwise VoiceGuide will wait until end of call. If the call is declined, the called party does not answer, or the extension is busy then VoiceGuide will dial the "retrieve call from announced conference" signal and then the Fail path is taken.

If call is accepted VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- either party hangs up.
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

Dial and Conference - General Notes:

The D4PCIU* family cards do not support "Dial and Conference". You need a JCT or DMV series card (eg. D/41JCT, D/120JCT, DMV160, D/240JCT, DMV600B etc.)

The ports which can be used to place the call can also be specified. The allowed posts need to be specified in comma delimited format. eg: 2,3,4

If ports are not specified then any free port will be used.

VG Dialer add-on is required to use the "Dial and Conference" option.

End of call detection on analog lines is done by listening for disconnect tone. Some phone companies hold off with playing the disconnect tone until some time after hangup, and sometimes the time before the disconnect tone is played differs depending on whether the person who hung up was the one who made the call or received the call.

To be able to hangup calls immediately the phone company must either play the disconnect tone immediately after either party hangs up, or you should use T1/E1 ISDN lines. If the application needs to detect the precise time of when either party hung up then T1/E1 ISDN lines should be used.

Dial and Conference - Blind

1. Dials the second number on another line.
2. Connects the two calls together.

VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- busy
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

Dial and Conference - Monitored

1. Dials the number on another line.
2. Listens to see if the outgoing call is answered by a human or an answering machine. If a human or answering machine answer is detected then VoiceGuide connects the two calls

together. If an announce message is specified the announce message is played to the call recipient before the call is connected.

If call is connected VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- busy
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

Dial and Conference - Announced

1. Dials the second number on another line.
2. Waits until the outgoing call is answered. When call is answered VoiceGuide plays a message announcing the Caller's ID, and awaits the selection from the called person whether to accept the call or not. "1" is used to accept the call, any other digit is used to decline the call.

If call is accepted VoiceGuide will take the Success path if it is defined, otherwise VoiceGuide will wait and hang up when:

- busy
- maximum conference time limit is reached (settable in VG.INI file)
- digit "*" is pressed during conversation

Two B-Channel Transfer (TBCT)

VoiceGuide v7 can perform TBCT transfers on ISDN lines which support this feature. To perform a TBCT transfer a 'Dial and Conference' transfer type needs to be selected, and then in the 'Dial and Conference Options' text box you need to include either:

```
<tbct_on>2ND_LEG_ALERTING</tbct_on>
```

or

```
<tbct_on>2ND_LEG_CONNECTED</tbct_on>
```

The `<tbct_on>2ND_LEG_ALERTING</tbct_on>` option will result in TBCT command being issued to the switch at the time when the 2nd leg of the call is in Alerting phase (2nd number begins to ring), and the `<tbct_on>2ND_LEG_CONNECTED</tbct_on>` option will result in TBCT command being issued to the switch at the time when the 2nd leg of the call has connected (2nd call was answered).

ACD Queue Transfer

ACD queues are supported in VoiceGuide v7 only.

Call can be placed in ACD queue, and from the queue the call will then be transferred to an agent when the appropriate agent becomes available.

To place call in an ACD queue the following expression is used: `acd:QueueName`

eg:

acd:sales

VoiceGuide will search for agents that are qualified to receive calls from specified queue, and the call is then connected (using a 'tromboned' connection) to the agent.

To receive ACD calls, agents need to use the 'VoiceGuide Agent' application on their PC.

'Announced' transfers timeout

The default length for how long VoiceGuide will await a confirmation to receive a call is 30 seconds. If you would like to change this, specify a timeout path in the 'Announced' modules to indicate the number of seconds you would like VoiceGuide to wait.

3 Way Conference Calls

3-Way Call Service must be enabled on the telephone line by the telephone company or PBX administrator before you can use this feature.

Some telephone systems do not allow 3 Way Blind Conference Calling, and will only allow a 3 way conference call connection to be made when the called person has answered the phone. In those situations you must use the 3 Way Announced Conference Call option.

Many North American telephone networks allow immediate blind conferencing.

Dial and Conference Options

Following user settable fields are available in Transfer module when "Dial and Conference" transfers are used:

- On-hold sound file to play while connection is made
- Line selection list
- When to connect the two callers
- Options

The Options field can be used to specify the outgoing CallerID used for the call by using the XML syntax. eg: specifying `<CallerId>5551234</CallerId>` would result in CallerId to be set to 5551234 on outgoing calls. Note that the CallerID settings in Config.xml can be used to override the settings made here as the Config.xml CallerId settings would take precedence.

Paths

The following paths can be taken from the module:

SUCCESS	the call was transferred successfully
FAIL	the call was not transferred, or transfer was declined
TIMEOUT	there was a timeout awaiting for transfer to be accepted (announced transfers only)

When using a Dialogic card the following paths can also be specified in the module:

BUSY	number was busy
FAX	call was answered by a fax
NOANSWER	there was no answer
NODIALTONE	there was no dialtone
NORINGBACK	call was not answered and no ringback was heard
OPERATOR	SIT tone was heard indicating number is most likely disconnected
CONNECT	transfer completed successfully
AM	transfer completed successfully with Answering Machine answering the call.
VOICE	transfer completed successfully with a live Human answering the call.
CADENCE	transfer completed successfully based on cadence analysis.
LOOPCURRENT	transfer completed successfully based on loop current detection.

Result Variables

For Tromboned transfers the following \$RVs are created on both lines:

```
$RV[Conf_DevName_1]
$RV[Conf_LineId_1]
$RV[Conf_LineNbr_1]
$RV[ModuleTitle_DevName_1]
$RV[ModuleTitle_LineId_1]
$RV[ModuleTitle_LineNbr_1]

$RV[Conf_DevName_2]
$RV[Conf_LineId_2]
$RV[Conf_LineNbr_2]
$RV[ModuleTitle_DevName_2]
$RV[ModuleTitle_LineId_2]
$RV[ModuleTitle_LineNbr_2]
```

These \$RVs can be further used to manage the connected calls in the future. vgEngine trace file contains more information on other \$RVs created during the call transfer process.

Troubleshooting

If the call transfers are not working for you please go through this checklist:

- 1.Can you do the call transfer manually on that line using a normal telephone handset?
- 2.Is the hookflash the right length? (ie. is the original caller placed on hold?)
- 3.Do settings in VG's "PBX Command Strings" config screen match the keys you need to press when doing it manually?
- 4.If transfers are still not working please post a question on our online Support Forum.

- a) Full detailed step-by-step description of how you just did the call transfer manually on this system,
- b) Your VoiceGuide script
- c) VG.INI file, and
- d) Copy of the trace log capturing the entire call.

Note:

If your system requires other specialized PBX signaling keep in mind that Hookflash and DTMF signals can also be generated using the "Play" module.

"!" is used to generate a hookflash and digits are used to generate DTMF tones.

Timeout paths between successive Play modules can be used if specific delays between successive PBX signals are required.

Evaluate Expression

Used to direct the script to a different module based on the value of Result Variable, Boolean expression or an Arithmetic expression specified. This module is used when the call is to go to different parts of script depending on:

- Caller ID
- Time and Date of the call
- Information entered by caller
- Information retrieved using DB Query / Web Service / Run Program / Run Script modules
- Any VBScript expression ([see here for complete list of functions](#))

The result of the evaluated expression will be assigned to the `$RV[module title]` Result Variable.

Optionally the result can also be assigned to a user specified Result Variable. This is useful when implementing counters or other more advanced call flow management.

The name of the user specified Result Variable can be anything. The user specified variable can be accessed in other parts of the script using `$RV[user specified variable]` format.

Selecting "Make Global RV" will make the created variable a "Global Result Variable". Global RVs need only be used if many different lines want to be able to read/set a central globally available variable which is retained by VoiceGuide as long as the software is running and does not get reset when any new calls arrive on any lines. (This option is used only in rare circumstances).

Selecting the "Store results in log file" option will ensure the results will be stored along with other call details. If the option is not selected the Result Variables will still be accessible within other parts of the script, but they will not be saved in the call log.

Paths Chosen:

If a [Result Variable](#) or an Arithmetic expression is specified:

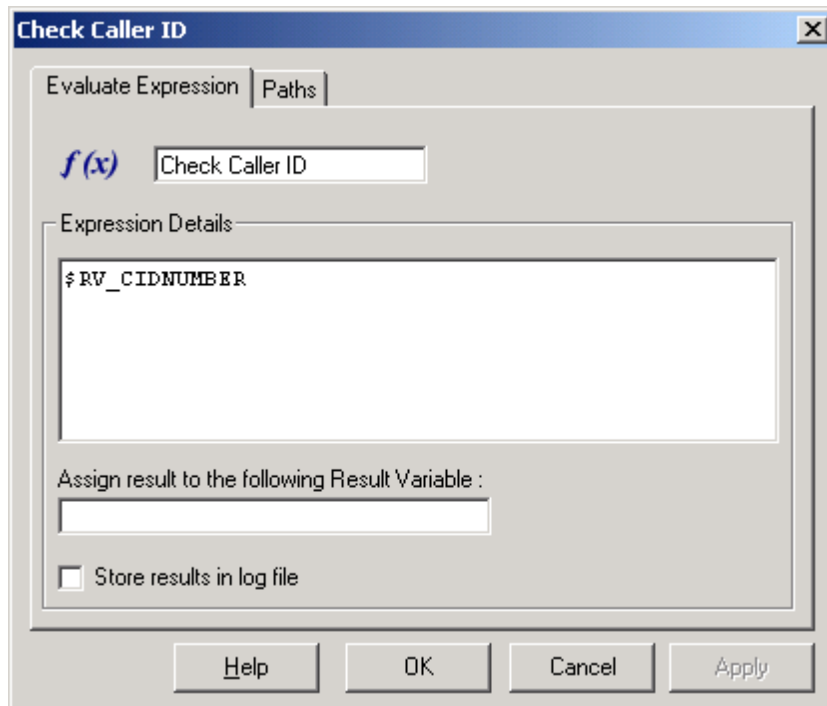
- if any of the [paths](#) match the result exactly then that path will be taken.
- If none of the Paths match the value exactly, but the result is non-empty then the "True" path will be taken.
- If the result is empty, then the False path will be taken.

If a Boolean expression is specified:

- A True or a False path will be taken depending on what was the result of the expression.

Example 1:

The Evaluate Expression can be used to switch to different parts of the script based on Caller ID. To switch based on the caller's phone number use `$RV_CIDNUMBER` as the Result Variable to be evaluated :



And when specifying the paths, use the telephone number in the { } section.

eg: If \$RV_CIDNUMBER was used, and you want to handle calls from number 5625551234 in a special way, the path to send all the calls from this number to a particular module would be:

```
On {5625551234} Goto [List Administrator Options]
```

you could also use:

```
on {} goto [NoCallerId]
```

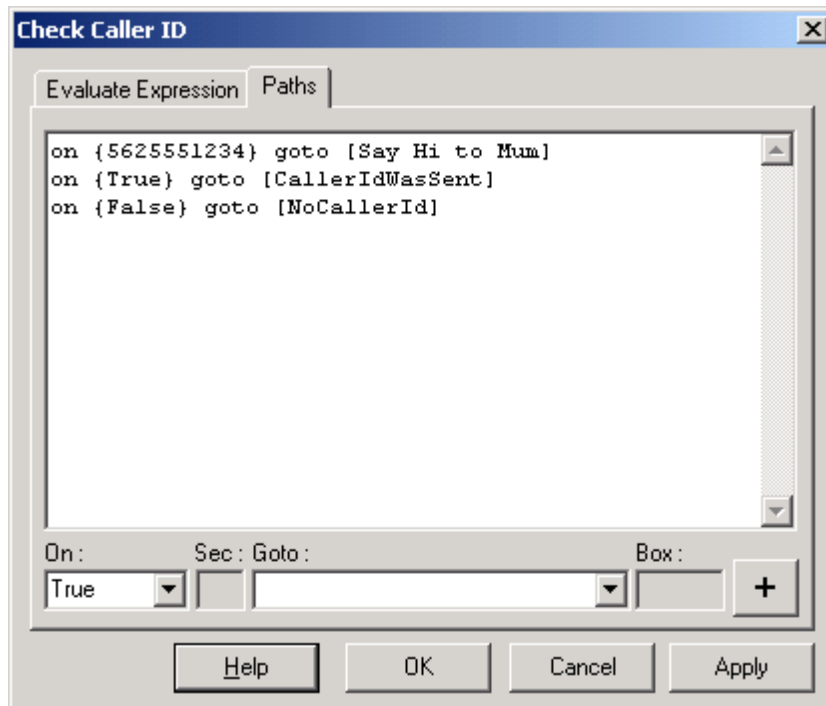
or

```
on {False} goto [NoCallerId]
```

both of which will match if CallerID number is blank.

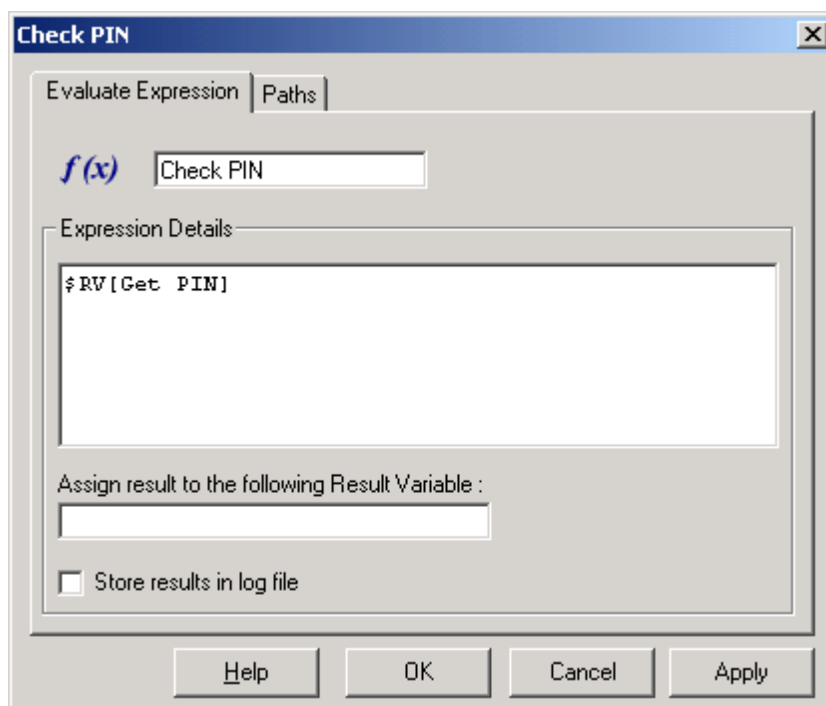
```
on {True} goto [CallerIdWasSent]
```

will be taken if CallerID was provided but no specified path explicitly matched the number:

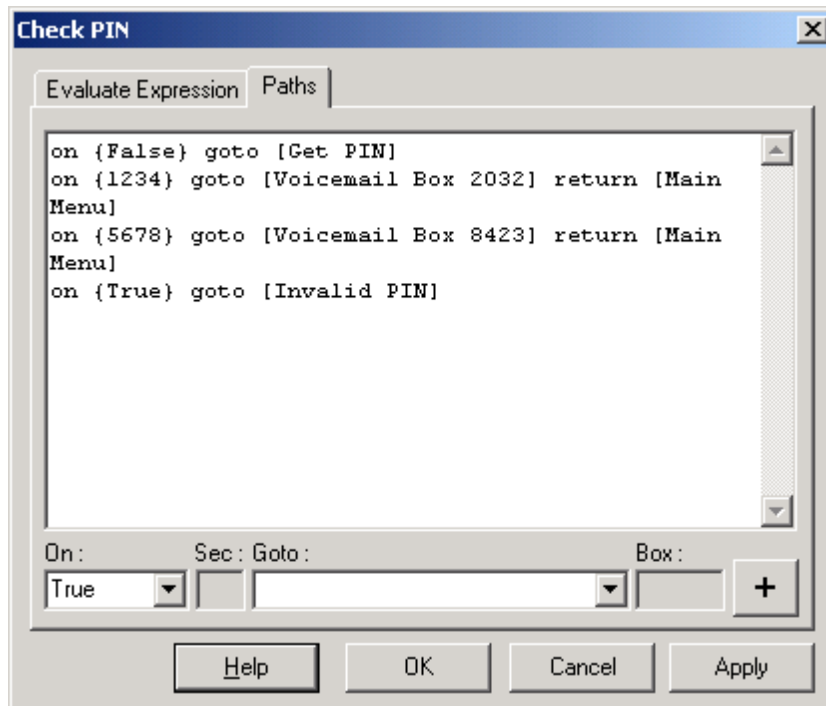


Example 2:

The Evaluate Expression module is used to check the PIN entered in previous Get Number module titled "Get PIN".



And the paths in this case are defined as:



What do the paths do:

On {False} Goto [Get PIN]

If for some reason no PIN was entered, and \$RV[Get PIN] holds no numbers, then goto the Get PIN module

On {1234} Goto [Voicemail Box 2032] Return [Main Menu]

If the entered PIN was 1234 then go to the Voicemail box number 2032, and then when the caller has finished leaving the message return the caller to the Main Menu module

On {5678} Goto [Voicemail Box 8423] Return [Main Menu]

If the entered PIN was 5678 then go to the Voicemail box number 8423, and then when the caller has finished leaving the message return the caller to the Main Menu module

On {True} Goto [Invalid PIN]

A PIN was entered and \$RV[Get PIN] does hold some entered number, but that number does not match exactly any of the paths specified. Go to the Invalid PIN module.

It is a good idea to always specify the TRUE and FALSE paths, since VoiceGuide will hang up the call if it cannot find a valid path to go to the next module.

Example 3:

The Evaluate Expression module can also be used to perform calculations.

Eg: To calculate whether the number of widgets ordered is more then the available amount you can use the following expression:

```
($RV[LokupStockAvail_1_1]- $RV[GetOrderAmount]) >= 0
```

A True or a False path will be taken depending on what was the result of the expression.

Example 4:

The Evaluate Expression module can also be used to compare strings. An example of valid

expression to evaluate is:

```
"$RV[DB_Retrieve_1_1]"="Available"
```

A True or a False path will be taken depending on what was the result of the above expression

Example 5:

If you need to evaluate a Result Variable that holds a string, and select different paths depending on the string's value then the string will need to be specified using quotes or "CStr()" function:

```
"$RV[DB_Retrieve_1_1]"
```

or:

```
Cstr($RV[DB_Retrieve_1_1])
```

you will need to use and use the different possible values of this variable when specifying the On ('Paths' where the script

Example 6:

The "+" or the "&" operator can also be used to concatenate strings together.

eg: The following is a valid expression to use:

```
["1_" & Cstr($RV_DAY) & "_2"]
```

The result is going to be(on a Saturday):

```
1_6_2
```

Example 7:

If comparing dates the # character should be used around the date expressions.

eg: The following expression determines if current date is between 7th July 2017 and 31 December 2018:

```
(#7/7/2017# <= #$RV_MONTH/$RV_DATE/$RV_YEAR#) and (#$RV_MONTH/$RV_DATE/$RV_YEAR# <= #31/12/2018#)
```

Example 8:

To add the value of the number entered in module "GetValue1" to the value of the number entered in module "GetValue2" the following expression would be used:

```
$RV[GetValue1] + $RV[GetValue2]
```

The result will be assigned to the result variable name for the module, or can be assigned to a Result Variable specified in the "Assign result to following Result Variable" field.

Example 9:

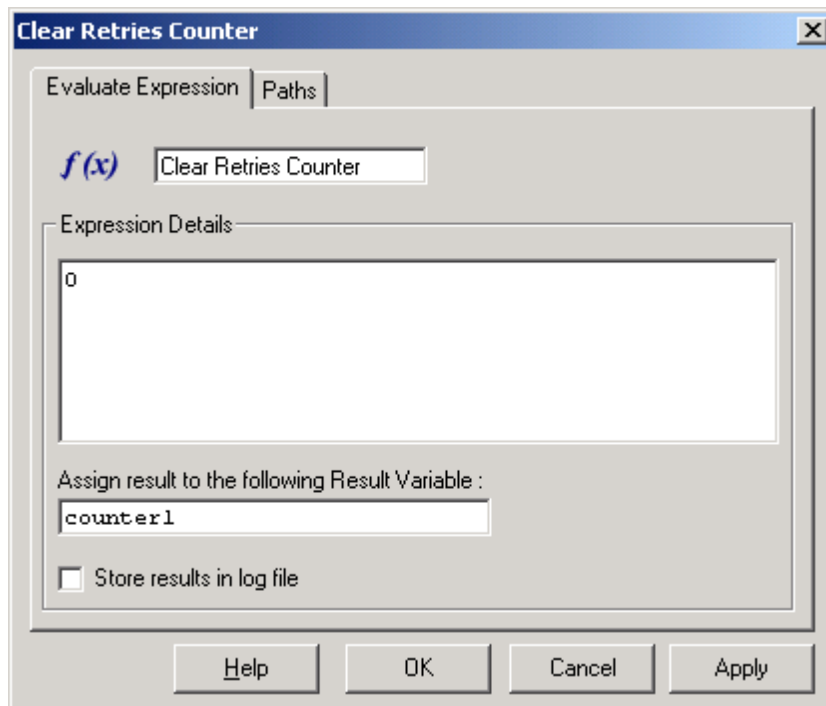
Evaluate expression can be used as a sophisticated Time-Switch as well. eg. To check if the current day is either New Year's Day or Christmas Day the following expression can be used:

```
(( "$RV_DD" = "01" ) and ( "$RV_MM" = "01" )) or ( ( "$RV_DD" = "25" ) and ( "$RV_MM" = "12" ))
```

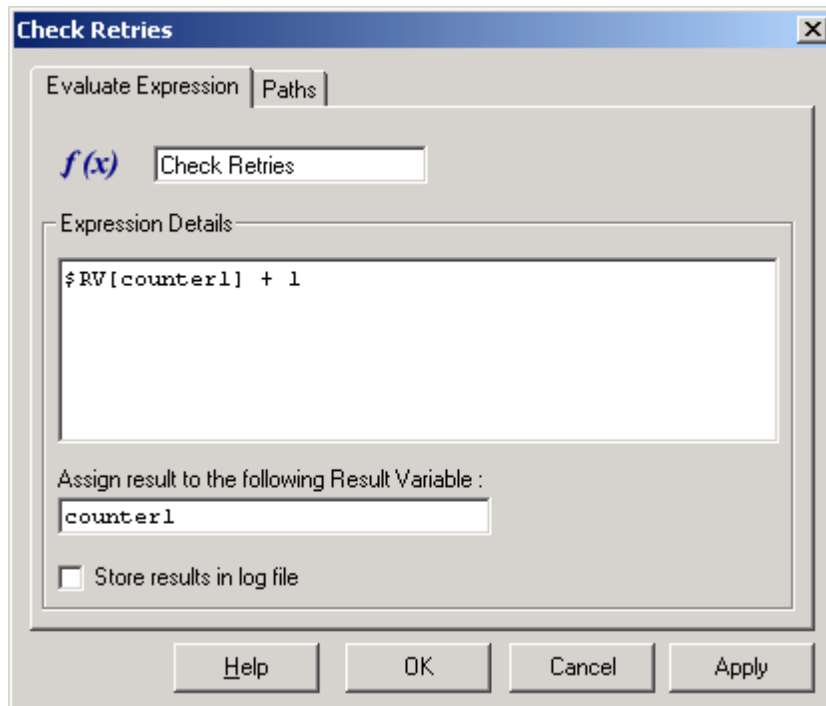
Example 10:

Evaluate Expression module can be used to implement a counter to check the number of times the caller goes through certain parts of the call flow. This example demonstrates how a counter can be implemented.

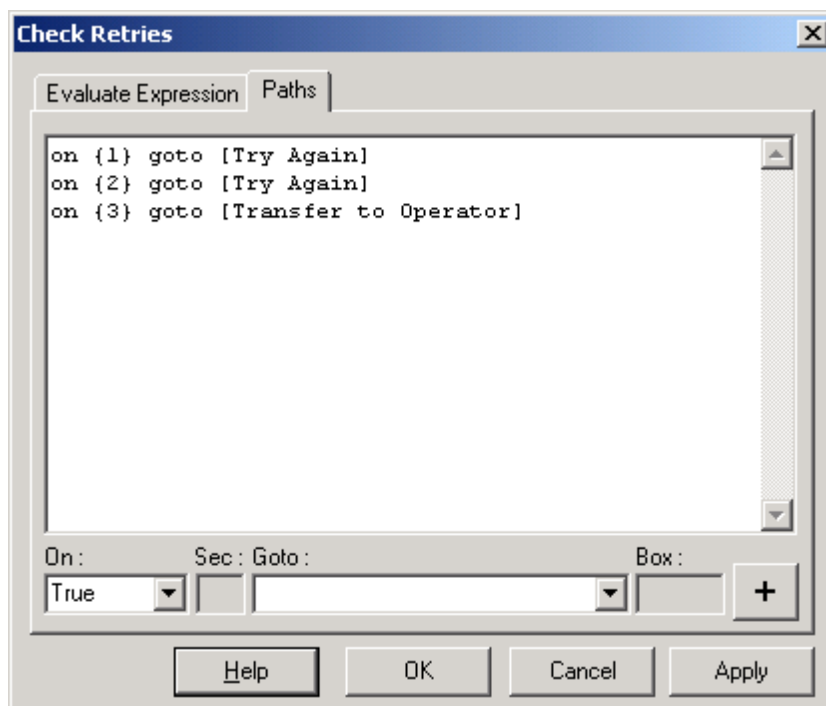
A user defined Result Variable is created and its value is set to zero. Note that as the Result Variable was assigned a value of 0, the 'False' path will be taken from this module.



The counter is incremented by retrieving the current value of the user defined Result Variable, adding 1 to this current value, and then saving the result back into the user defined Result Variable.



The value saved by this Evaluate Expression module is then used in the Paths section to determine the next module the script should branch to.



List of functions which can be used:

Any VBScript type expression can be evaluated using the Evaluate Expression module. The full list of functions/operators/constants/keywords etc which can be used can be found at:

http://www.w3schools.com/asp/asp_ref_vbscript_functions.asp

Functions which can be used in the Evaluate Expression module:

**Date/Time
Functions**

CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week
Year	Returns a number that represents the year

**Conversion
Functions**

Asc	Converts the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype

	Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDBl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Hex	Returns the hexadecimal value of a specified number
Oct	Returns the octal value of a specified number

Format Functions

FormatCurrency	Returns an expression formatted as a currency value
FormatDateTime	Returns an expression formatted as a date or time
FormatNumber	Returns an expression formatted as a number
FormatPercent	Returns an expression formatted as a percentage

Math Functions

[Top](#)

Abs	Returns the absolute value of a specified number
Atn	Returns the arctangent of a specified number
Cos	Returns the cosine of a specified number (angle)
Exp	Returns e raised to a power
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Fix	Returns the integer part of a specified number
Log	Returns the natural logarithm of a specified number
Oct	Returns the octal value of a specified number
Rnd	Returns a random number less than 1 but greater or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sin	Returns the sine of a specified number (angle)
Sqr	Returns the square root of a specified number

[Tan](#)

Returns the tangent of a specified number (angle)

Array Functions

[Top](#)

[Array](#)

Returns a variant containing an array

[Filter](#)

Returns a zero-based array that contains a subset of a string array based on a filter criteria

[IsArray](#)

Returns a Boolean value that indicates whether a specified variable is an array

[Join](#)

Returns a string that consists of a number of substrings in an array

[LBound](#)

Returns the smallest subscript for the indicated dimension of an array

[Split](#)

Returns a zero-based, one-dimensional array that contains a specified number of substrings

[UBound](#)

Returns the largest subscript for the indicated dimension of an array

String Functions

[InStr](#)

Returns the position of the first occurrence of one string within another. The search begins at the first character of the string

[InStrRev](#)

Returns the position of the first occurrence of one string within another. The search begins at the last character of the string

[LCase](#)

Converts a specified string to lowercase

[Left](#)

Returns a specified number of characters from the left side of a string

[Len](#)

Returns the number of characters in a string

[LTrim](#)

Removes spaces on the left side of a string

[RTrim](#)

Removes spaces on the right side of a string

[Trim](#)

Removes spaces on both the left and the right side of a string

[Mid](#)

Returns a specified number of characters from a string

[Replace](#)

Replaces a specified part of a string with another string a specified number of times

[Right](#)

Returns a specified number of characters from the right side of a string

[Space](#)

Returns a string that consists of a specified number of spaces

[StrComp](#)

Compares two strings and returns a value that represents the result of the comparison

[String](#)

Returns a string that contains a repeating character of a specified length

[StrReverse](#)

Reverses a string

[UCase](#)

Converts a specified string to uppercase

Other Functions

CreateObject	Creates an object of a specified type
Eval	Evaluates an expression and returns the result
IsEmpty	Returns a Boolean value that indicates whether a specified variable has been initialized or not
IsNull	Returns a Boolean value that indicates whether a specified expression contains no valid data (Null)
IsNumeric	Returns a Boolean value that indicates whether a specified expression can be evaluated as a number
IsObject	Returns a Boolean value that indicates whether the specified expression is an automation object
RGB	Returns a number that represents an RGB color value
Round	Rounds a number
ScriptEngine	Returns the scripting language in use
ScriptEngineBuildVersion	Returns the build version number of the scripting engine in use
ScriptEngineMajorVersion	Returns the major version number of the scripting engine in use
ScriptEngineMinorVersion	Returns the minor version number of the scripting engine in use
TypeName	Returns the subtype of a specified variable
VarType	Returns a value that indicates the subtype of a specified variable

VBScript Keywords

Empty	<p>Used to indicate an uninitialized variable value. A variable value is uninitialized when it is first created and no value is assigned to it, or when a variable value is explicitly set to empty.</p> <p>Example:</p> <pre>Dim x 'the variable x is uninitialized! x="ff" 'the variable x is NOT uninitialized anymore x=Empty 'the variable x is uninitialized!</pre> <p>Note: This is not the same as Null!!</p>
IsEmpty	<p>Used to test if a variable is uninitialized.</p> <p>Example: If (IsEmpty(x)) 'is x uninitialized?</p>
Nothing	<p>Used to indicate an uninitialized object value, or to disassociate an object variable from an object to release system resources.</p> <p>Example: Set myObject=Nothing</p>
Is Nothing	<p>Used to test if a value is an initialized object.</p> <p>Example: If (myObject Is Nothing) 'is it unset?</p>

Note: If you compare a value to Nothing, you will not get the right result! Example: If (myObject = Nothing) 'always false!

Null Used to indicate that a variable contains no valid data.

One way to think of Null is that someone has explicitly set the value to "invalid", unlike Empty where the value is "not set".

Note: This is not the same as Empty or Nothing!!

Example: x=Null 'x contains no valid data

IsNull Used to test if a value contains invalid data.

Example: if (IsNull(x)) 'is x invalid?

True Used to indicate a Boolean condition that is correct (True has a value of -1)

False Used to indicate a Boolean condition that is not correct (False has a value of 0)

Operators which can be used in Evaluate Expression module:

Addition (+)	Sum two numbers.
And	Perform a logical conjunction on two expressions.
Assignment (=)	Assign a value to a variable or property.
Concatenation (&)	Force string concatenation of two expressions.
Division (/)	Divide two numbers and return a floating point result.
Eqv	Perform a logical equivalence on two expressions.
Exponentiation (^)	Raise a number to the power of an exponent.
Imp	Perform a logical implication on two expressions.
Integer Division (\)	Divide two numbers and return an integer result.
Is	Compare to object reference values.
Mod	Divide two numbers and return the remainder.
Multiplication (*)	Multiply two numbers.
Negation (-)	Indicate the negative value of a numeric expression.
Not	Perform logical negation of an expression.
Or	Perform logical disjunction on two expressions.
Xor	Perform a logical exclusion on two expressions.
Subtraction (-)	Find the difference between two numbers.

Fax Send and Receive

VoiceGuide v7 can send and receive faxes. All current Dialogic cards support fax send and receive functionality.

Sending Fax

A Play module can be used to send a fax. Just specify a PDF or TIFF or JPEG file as the file to be "played" by a Play module. The Play module will begin fax transmission immediately if a .tif/.tiff/.pdf/.jpg source file is specified.

Once fax sending is complete the success path will be taken and the script can continue. Please note that the remote end will usually hangup after receiving the fax, so any post fax sending updates etc. should be performed in the "After Hangup" script.

Its simplest if TIFF files are used for the outgoing fax. The TIFF file must be in "Fax Group 3" format and have a horizontal resolution of 1728 pixels across the page. Vertical pixel count will depends on length of fax page, but a standard page is 2200 pixels long. The DPI resolution of the TIFF file should be 200. Some sample TIFF files are provided in the directories of the sample VoiceGuide scripts that perform fax sending.

Any PDF file can be used as input as well. If faxing PDF files you must aggregate Ghostscript with VoiceGuide by placing Ghostscript's gswin32c.exe and gsdll32.dll file into VoiceGuide's \gs\ subdirectory. Ghostscript can be downloaded from <http://www.ghostscript.com>

JPEG (.jpg) files are used if "Color Fax" needs to be sent. Please contact support@voiceguide.clom if you need to send "Color Fax".

Receiving Fax

A Record module can be used to receive a fax. To make the Record module receive a fax just specify the filename to end in .pdf suffix or end in .tif suffix. The Record module will begin fax reception immediately if a .pdf target file or a .tif target file is specified.

The received fax is automatically saved both in in the PDF and in the TIFF file formats.

After receiving the fax VoiceGuide will create a PDF version of the received fax, and will also perform OCR on the fax and scan the fax for any barcodes. The OCR text and text from decoded barcodes is then stored as Result Variables. During OCR VoiceGuide will also try to identify the "To:" and "From:" fields and make the text that follows them available through Result Variables as well. VoiceGuide will also save the fax header line as a separate Result Variable.

The following Result Variables are created when a fax is received:

<code>\$RV[ModuleTitle_fax]</code>	Filename of the sent or received fax.
<code>\$RV[ModuleTitle_fax_pdf]</code>	Filename of the PDF file create after TIF->PDF conversion.
<code>\$RV[ModuleTitle_fax_tif]</code>	Filename of the TIF file used as a source for PDF conversion.
<code>\$RV[ModuleTitle_ocr]</code>	Full OCR of the received fax.
<code>\$RV[ModuleTitle_ocr_header]</code>	Fax Header line which usually contains the Date/Time,

<code>\$RV[ModuleTitle_ocr_to_1]</code>	CSID (Call Subscriber ID), fax number, and page number.
<code>\$RV[ModuleTitle_ocr_to_2]</code>	First word after the "To:"
<code>\$RV[ModuleTitle_ocr_from_1]</code>	First two words after the "To:"
<code>\$RV[ModuleTitle_ocr_from_2]</code>	First word after the "From:"
	First two words after the "From:"
<code>\$RV[ModuleTitle_barcode_1]</code>	Contents of the first barcode encountered on the fax page.
<code>\$RV[ModuleTitle_barcode_2]</code>	Contents of the second barcode encountered on the fax page.
<code>\$RV[ModuleTitle_barcode_count]</code>	Number of barcodes found on the fax.
<code>\$RV[ModuleTitle_barcode_all]</code>	Contents of all the barcodes encountered on the fax page, delimited by " ".

To have VoiceGuide save the PDF filed in 'searchable PDF' format please contact sales@voiceguide.com

To install the barcode recognition add-on please contact sales@voiceguide.com

Fax Routing

The OCR and barcode decoding allow advanced automated fax routing to be done, automatically forwarding the received document by:

- Email
- Printing to selected printer
- Ftp
- Re-faxing to another number (optionally with additional pages added to original fax)
- any other processing called from command line or VBScript

Note that as the call is usually ended immediately after the fax is received, the fax routing should be done from the "After Hangup" script. The "After Hangup" script can look at the OCR \$RVs as well as the \$RVs containing the CallerID or the DNIS (number dialed by caller) and decide how to forward the fax, and then perform the email or printing or loading of a new outbound call etc. This allows fax routing to be done based on any combination of:

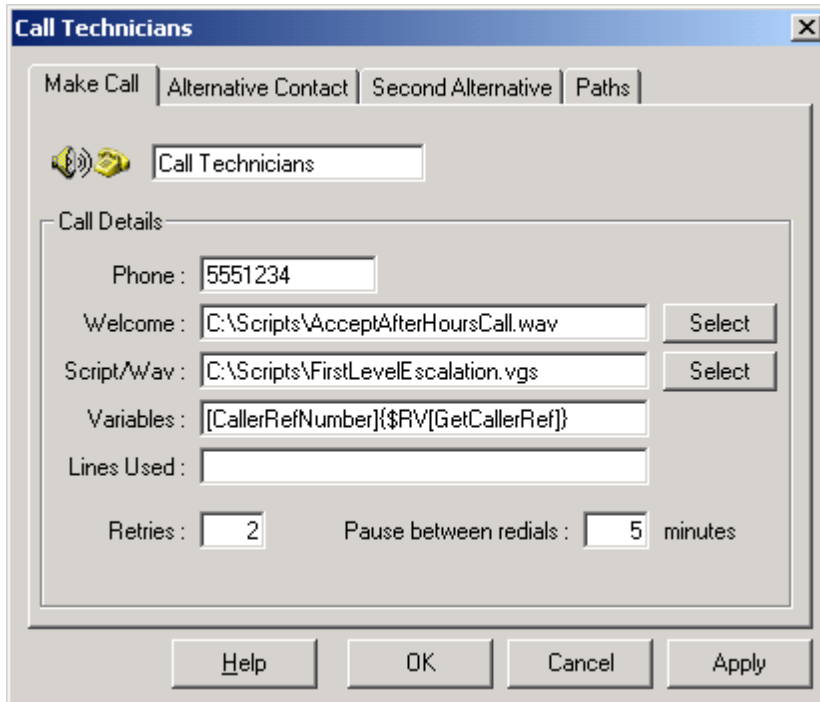
- CallerID
- DNIS
- CSID (Call Subscriber ID) of the machine sending the fax
- To: or From: fields in the received fax
- Any other text in the received fax
- Barcodes in the received fax
- Time/Day etc

Send Phone Message

This module will make an outgoing call on another line to the phone number specified, and run the specified script or sound file when the call is answered.<

Note: More advanced call scheduling features can be accessed by loading the calls using the 'Out Dial' XML file method, WCF/COM/REST interfaces, or inserting call data directly into the VoiceGuide database. For more information on these methods of queuing outgoing calls please [see here](#).

The [Dialer](#) option must be enabled to allow VoiceGuide to perform any outbound dialing.



You can specify the telephone number to call, the welcome message to play when the call has been made, the VoiceGuide script to run once the call has been accepted, and custom Result Variables which can be used by the called script.

[Result Variables](#) can be used when specifying any of the entries in this module. <>

Phone

The phone number to dial.

Welcome

Optional. Prompt played to caller caller to accept the call. The welcome message asks the called person whether they want to accept the call or not. The called person should press '1' on their telephone keypad to indicate that they want to accept the call. Hence any welcome message should advise to 'Press 1' to accept the call'.

If the welcome message is not specified then the default VoiceGuide welcome message will be played. If welcome message is set to '**none**' then no welcome message will be played and VoiceGuide will go directly to running the script.

Script

Which script will be ran once the call is accepted, or the sound file to be played.

Variables

Result Variables which can be accessed by the called script. Use these to pass any information (eg. callers contact details) to the script which will be used when the outbound call is made. The format of this field is `[RvName]{RvValue}`. Multiple Result variables can be specified by listing the name-value pairs as needed.

In the properties page screen capture above we see that the caller's problem reference number is being passed to the `FirstLevelEscalation.vgs` script. The reference number would have been generated previously in module `CallerRef`, and `$RV[CallerRef]` will be replaced with the actual reference number when the call is queued. The reference number can then be accessed using `$RV[CallerRefNumber]` in the `FirstLevelEscalation.vgs` script.

Lines Used

If the outbound call may only be made on particular phone lines then these lines should be listed here. Lines should be specified as a comma delimited list of "LineIDs" (eg: 6,7,8) or of the Dialogic line identifiers (eg: `dxxxB1C2,dxxxB1C3`). If any of the lines are allowed to to be used then this setting should be left blank.

Retries

The number times the number will be redialed before abandoning attempts to contact the number. For example, if the number of retries is set to 2, with 5 minutes between redials then the phone number will be called 3 times in total (1 call + 2 redials).

Escalation Dialing

In case VoiceGuide is unable to contact anybody at the first phone number supplied, alternative phone numbers and scripts can be specified. This is useful if:

- A person you want to contact can be under different numbers.
- Problem escalation situations, where if one person is unavailable then further people will be called until one of the people on the list answers the call.

Alternative Contact and Second Alternative Contact tabs are used to specify alternative contact numbers. Once the number of retries for one number is exhausted, the next alternative number will be dialed.

Notes:

For instructions on how to set your own escalation dialing without using the Make Call module, see the VoiceGuide Dialer section of this help file.

Paths

The 'Success' path is taken if the call was scheduled. If for any reason the call could not be

scheduled, the 'Fail' path will be taken

Scheduling Calls

The Send Phone Message module schedules the call to be made immediately. If you need to schedule a call to be made at a later time VoiceGuide's COM function Dialer_OutDialQueAdd needs to be used. This function can be called from within a Run VB Script module.

The future date/time needs to be specified in format "YMMDDHHNN" in Dialer_OutDialQueAdd's *lActivateTime* parameter.

eg. To specify a date/time exactly 3 days in future you can use the following VB Script snippet:

```
dDate = DateAdd("d", 3, Now)
sYear = Right(Year(dDate), 1)
sCallTime = sYear * 100000000 + Month(dDate) * 1000000 + Day(dDate) * 10000 + Hour(dDate)
* 100 + Minute(dDate)
```

and then you use sCallTime when calling Dialer_OutDialQueAdd

You can hard code the hour and minute settings if you want the call to be made at a cetein time. eg to call at 10:30am the sCallTime formula would become:

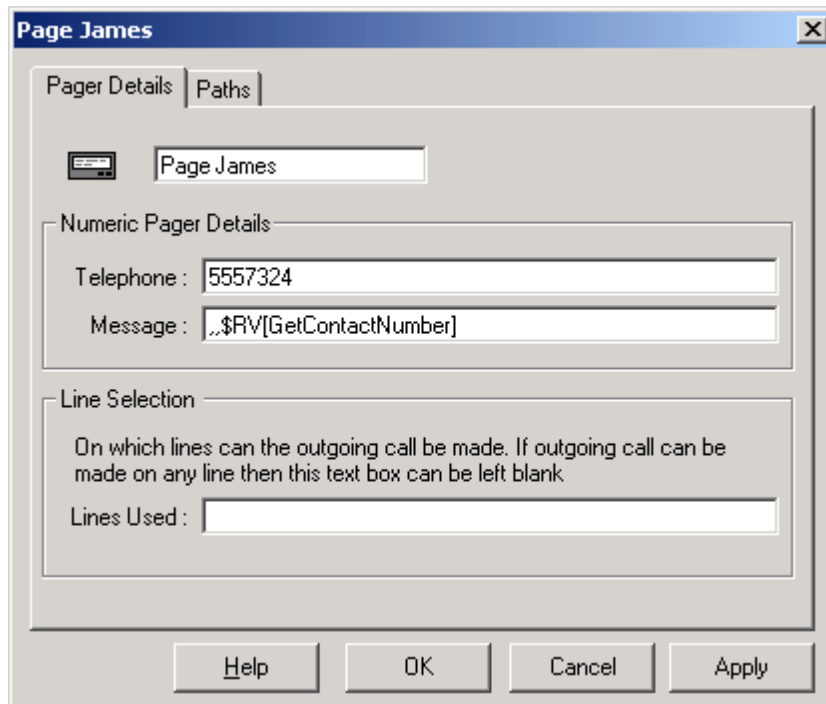
```
sCallTime = sYear * 100000000 + Month(dDate) * 1000000 + Day(dDate) * 10000 + 1030
```

Send Pager Message

This module will queue a pager message to be sent.

[VG Dialer](#) option must be enabled to allow VoiceGuide to perform any outbound dialing and pager message delivery.

The pager message will be sent as soon as VoiceGuide has a line available to make a call on. When sending the message VoiceGuide will dial the Telephone number specified, and then will start dialing the digits specified in the Message text box.



The screenshot shows a Windows-style dialog box titled "Page James". It has two tabs: "Pager Details" (selected) and "Paths". Under "Pager Details", there is a label "Page James" next to a small icon. Below that is a "Numeric Pager Details" section containing a "Telephone" field with the value "5557324" and a "Message" field with the value "..\$RV[GetContactNumber]". Below the message field is a "Line Selection" section with a text box labeled "Lines Used". At the bottom of the dialog are four buttons: "Help", "OK", "Cancel", and "Apply".

VoiceGuide will pause for a couple of seconds after dialing the telephone number and before dialing the message. If this pause is not long enough, extra wait time can be specified by tying commas at the beginning of the message. The example in the captured screen above has two commas.

The length of a pause generated by each comma differs from system to system and its best to determine by trial and error the number of commas to use in order to have reliable pager message delivery from your VoiceGuide system to your pager provider.

Also in the example above a [Result Variable](#) was used in the message. This shows how data entered by the caller in one part of the script can be sent as part of a pager message.

Lines Used

If the outbound call may only be made on particular phone lines then these lines should be listed here. Lines should be specified as a comma delimited list of "LineIDs" (eg: 6,7,8) or of the Dialogic line identifiers (eg: dxxxxB1C2,dxxxxB1C3). If any of the lines are allowed to to be used then this setting should be left blank.

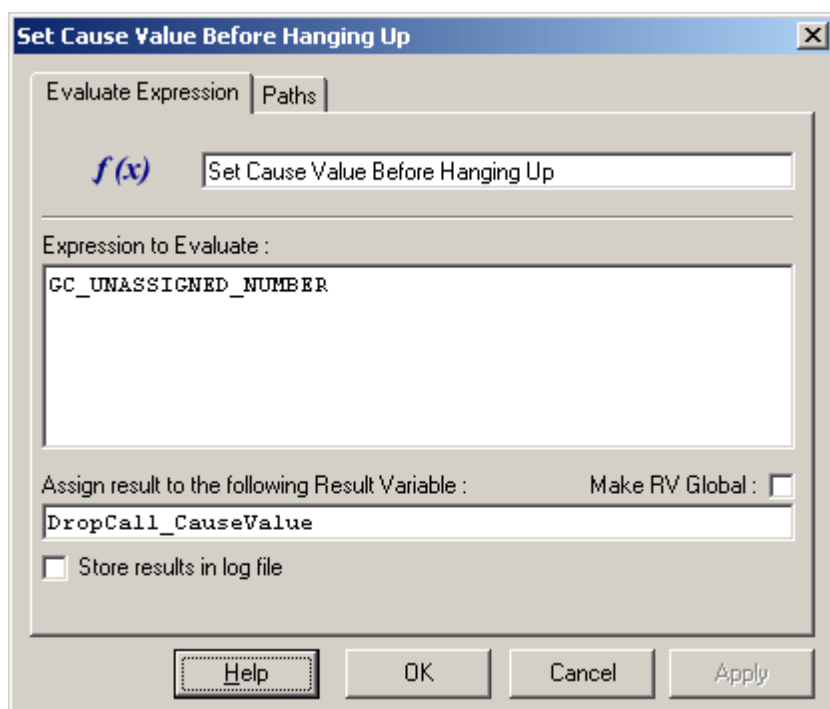
Paths

The 'Success' path is taken if the call to the pager was scheduled. If for any reason the call could not be scheduled, the 'Fail' path will be taken.

Hangup Call

Hangs up the current call.

On ISDN systems the 'cause value' which is sent at the time of call disconnection can be specified by setting the `$RV[DropCall_CauseValue]` to hold the cause value tag. You can use the Evaluate Expression module to assign a cause value to `$RV[DropCall_CauseValue]`



Valid cause values are:

Cause Value Tag

ACCESS_INFO_DISCARDED
BAD_INFO_ELEM
BEAR_CAP_NOT_AVAIL
CAP_NOT_IMPLEMENTED
CHAN_DOES_NOT_EXIST
CHAN_NOT_IMPLEMENTED
FACILITY_NOT_IMPLEMENT
FACILITY_NOT_SUBSCRIBED
FACILITY_REJECTED
GC_CALL_REJECTED
GC_CHANNEL_UNACCEPTABLE
GC_DEST_OUT_OF_ORDER
GC_NETWORK_CONGESTION
GC_NORMAL_CLEARING
GC_REQ_CHANNEL_NOT_AVAIL
GC_UNASSIGNED_NUMBER
GC_USER_BUSY
INCOMING_CALL_BARRED
INCOMPATIBLE_DEST
INTERWORKING_UNSPEC

Cause Value Description

Access information discarded
Information element nonexistent or not implemented
Bearer channel capability not available
Bearer channel capability not implemented
Channel does not exist
Channel type not implemented
Requested facility not implemented
Facility not subscribed
Facility rejected
Call was rejected
Channel is not acceptable
Destination is out of order
Call dropped due to traffic volume
Call dropped under normal conditions
Requested channel is not available
Requested number is unknown
End user is busy
Incoming call barred
Incompatible destination
Interworking unspecified

INVALID_CALL_REF	Invalid call reference
INVALID_ELEM_CONTENTS	Invalid information element
INVALID_MSG_UNSPEC	Invalid message, unspecified
INVALID_NUMBER_FORMAT	Invalid number format
MANDATORY_IE_LEN_ERR	Message received with mandatory information element of incorrect length
MANDATORY_IE_MISSING	Mandatory information element missing
NETWORK_OUT_OF_ORDER	Network out of order
NO_CIRCUIT_AVAILABLE	No circuit available
NO_ROUTE	No route. Network has no route to the specified transient network or to the destination
NO_USER_RESPONDING	No user responding
NONEXISTENT_MSG	Message type nonexistent or not implemented
NUMBER_CHANGED	Number changed
OUTGOING_CALL_BARRED	Outgoing call barred
PRE_EMPTED	Call preempted
PROTOCOL_ERROR	Protocol error, unspecified
RESP_TO_STAT_ENQ	Response to status inquiry
SERVICE_NOT_AVAIL	Service not available
TEMPORARY_FAILURE	Temporary failure
TIMER_EXPIRY	Recovery on timer expired
UNSPECIFIED_CAUSE	Unspecified cause
WRONG_MESSAGE	Message type invalid in call state or not implemented
WRONG_MSG_FOR_STATE	Message type not compatible with call state

The full definition of all the various cause codes can be found in the ITU-T's Q.850 recommendation:
<http://www.itu.int/rec/T-REC-Q.850/en>

ITU-T's Q.931 (ISDN) recommendation itself can be found at: <http://www.itu.int/rec/t-rec-q.931-199805-i/en>

Dashboards

Preconfigured VoiceGuide IVR and ACD dashboards can be accessed at these locations:

`http://localhost:7140/dash/ivr.html`

`http://localhost:7140/status/port-grid.html`

`http://localhost:7140/dash/acd-agent-list.html`

`http://localhost:7140/dash/acd-agent-grid.html`

`http://localhost:7140/dash/acd-agent-list-2.html`

"localhost" would need to be replaced with VoiceGuide system's IP address if dashboard is viewed remotely.

The dashboards are user configurable and retrieve system data from VoiceGuide through VoiceGuide's [REST API](#).

Creation of new custom Dashboards is also supported. New dashboards can be added to the system by creating a new dashboard definition file and placing it in VoiceGuide's `\system\web\dash\` subdirectory. The newly created dashboard can then be accessed using:

`http://localhost:7140/dash/my-new-dashboard.html`

REST API

VoiceGuide's REST API can be used to retrieve current and historical data for IVR and ACD systems. Generation of PDF reports is also supported through the REST API.

VoiceGuide's REST API can be accessed at this location:

```
http://localhost:7131/v1/
```

"localhost" is replaced with system's IP address if REST API is accessed remotely.

REST API for IVR data:

```
http://localhost:7131/v1/ivr/calls/[direction]/[range]/[bin]/[port]/[data_or_labels]
```

[direction] : in | out | all

[range] : today | yesterday | YYYYMMDD | YYMMDD | MMDD | last24hr | last1hr | last30min | 24hr | 1hr | 30min

[bin_size] : day | 24hr | 1hr | 30min

[port] : all | X

[data_or_labels] : count | labels | detail | pdf

Examples of use:

```
http://localhost:7131/v1/ivr/calls
```

```
http://localhost:7131/v1/ivr/calls/group/live
```

```
http://localhost:7131/v1/ivr/calls/group/historical
```

```
http://localhost:7131/v1/ivr/calls/in/24hr/1hr/all/count
```

```
http://localhost:7131/v1/ivr/calls/in/24hr/1hr/all/labels
```

```
http://localhost:7131/v1/ivr/calls/in/24hr/1hr/all/pdf
```

"localhost" would need to be replaced with Voiceguide system's IP address if dashboard is viewed remotely.

REST API for ACD data:

```
http://localhost:7131/v1/acd/agent/[agent_id]/[range]
```

```
http://localhost:7131/v1/acd/queue/[queue_name]/[range]
```

[agent_id] : all | list | X

[range] : today | yesterday | YYYYMMDD | YYMMDD | MMDD

[queue_name] : all | list | X

Examples of use:

<http://localhost:7131/v1/acd/agent/list>

<http://localhost:7131/v1/acd/queue/list>

Line Status Monitor

VoiceGuide Line Status Monitor is a Windows application that shows real-time status of each port.

Automated Report Scheduling

VoiceGuide allows for automated scheduling of creation of PDF reports and forwarding of them to specified email list.

Automated reporting is configured in Config.xml file, located in VoiceGuide's /conf/ subdirectory.

Introduction

VoiceGuide has a built in a fully-featured voicemail system, and can support an unlimited number of voicemail boxes.

Voicemail Features:

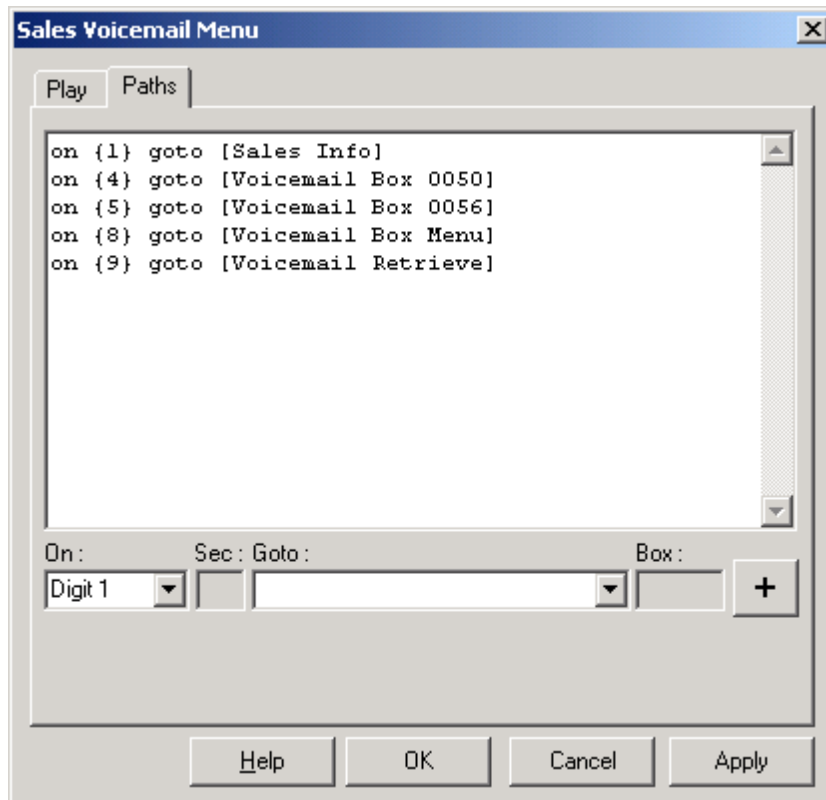
- Auto-attendant system allowing spelling of user's names and diverting to voicemail if extension is not answered.
- Each mailbox can have a personalized greeting message, which can be changed remotely.
- Each mailbox can have a personalized name/description, which can be changed remotely.,
- Voicemail messages can be retrieved remotely.
- Messages can be forwarded to other voicemail boxes and/or broadcast groups automatically, or as specified by listener of message.
- Messages can be be forwarded to an email address automatically as an attached Wave file.
- Messages can be be forwarded to another phone number automatically. The forwarding telephone number can be changed remotely by the voicemail box owner.
- Messages can be uploaded by FTP to selected FTP server.
- Voicemail box owners can change their PIN number remotely.
- Broadcast Groups, allowing the sending of broadcast messages to selected groups of people.
- Message Lamp for the extension associated with the mailbox is turned ON when new messages arrive and OFF when messages are listened to.
- All Voicemail system details stored in XML format, allowing easy integration with other systems.

Voicemail boxes have to first be created before they can be used. The [Voicemail Manager](#) tool is used to create and manage the voicemail boxes.

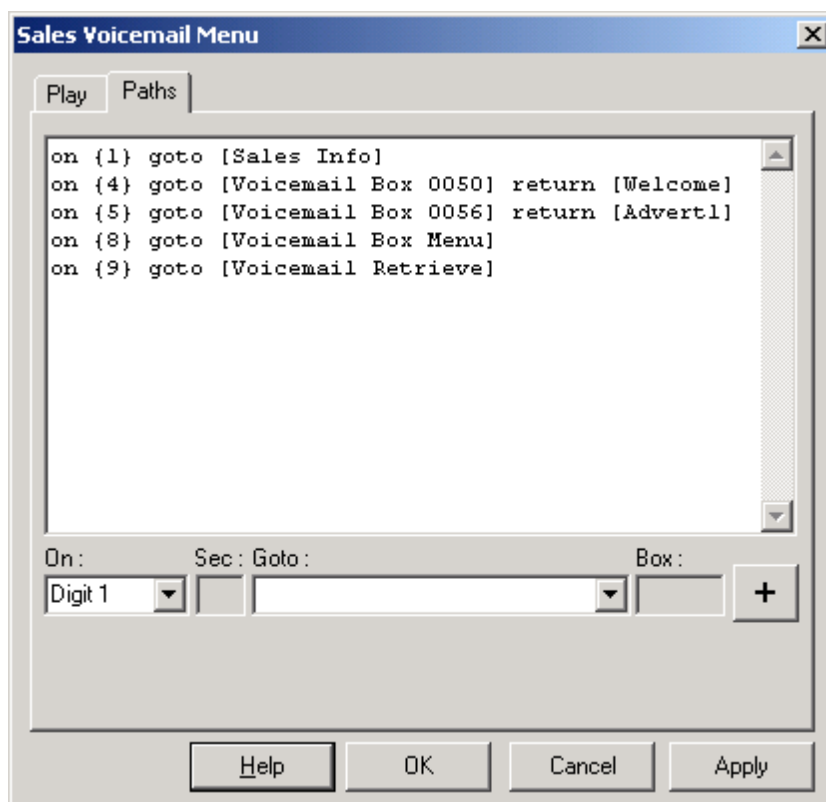
The voicemail boxes and menus can be accessed from any module. Paths can be specified to go to:

- 1.A particular voicemail box,
- 2.Menu which will ask the caller to enter the Voicemail box for which they want to leave a message,
- 3.Retrieve Menu where the caller will be asked their voicemail box number and voicemail PIN before being able to retrieve messages.
- 4.Auto Attendant which allows callers to spell the name of the voicemail box owner, and will then transfer the call to related extension (if defined), and if extension does not answer will divert call to related voicemail box.

An example below shows how the paths can be defined:



You can indicate to which module in the script the caller is to return to after leaving the voicemail system by appending **Return [module title]** after the path specification. The Return option will need to be typed in by hand. eg:



Once the caller has finished recording and/or retrieving messages and presses **'0'** to exit from the voicemail system, the script will then go to the module specified in the Return option. See [here](#) for more information on the **Return** option.

Modifying the Voicemail System

The Voicemail system is written as a set of VoiceGuide scripts located in VoiceGuide's \system\vm\ directory. These scripts can be edited giving the user full control and freedom over any modifications and extensions which may be required of the voicemail system.

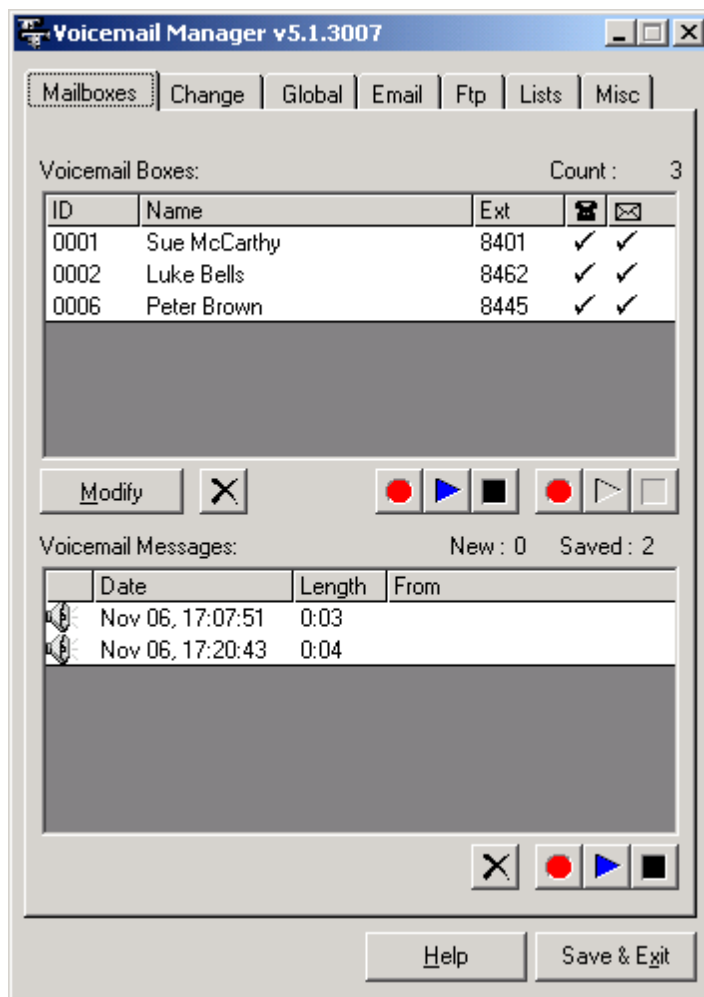
Only the Enterprise (and Evaluation) versions of VoiceGuide use the modifiable scripted version of the voicemail - ie: the .vgs scripts in the \system\vm\ directory.

The Standard and Professional versions of VoiceGuide use the non-modifiable version of the voicemail system.

Voicemail System Manager

Using the Voicemail Manager you can browse through the voicemail boxes, and change any of their settings:

- create/remove voicemail boxes,
- play/delete/record messages left in individual voicemail boxes,
- play/delete/record the greeting messages for individual voicemail boxes.
- edit the forwarding email address for new voicemail messages
- edit the forwarding phone number for new voicemail messages
- edit the pager telephone number
- edit the message distribution lists.



To add a new voicemail box select the 'Add/Modify' tab, and fill out the mailbox details and forwarding information.

Field	Description
Number	Voicemail box numbers (VMB IDs) can be up to 6 digits long, allowing up to 1,000,000 voicemail boxes on the system. In most applications 4 digit digit voicemail ID's are used, allowing up to 10,000 mailboxes.
PIN	Voicemail box access PIN number length is set to 4 digits. If no PIN is specified the caller will be asked to specify the PIN the first time they log into the voicemail box.
Name	Name of the voicemail box owner. This is the name which can be spelled by caller when using the Auto Attendant's 'spell name' feature.
Ext	Extension number associated with this voicemail box. This is the extension number to which caller will be transferred when using the Auto Attendant's 'spell name' feature.
Comments	Administrator's Comments.
Groups	Voicemail groups to which this mailbox belongs to. Voicemail groups are used when sending broadcast messages, and when browsing voicemail boxes' welcoming messages.
Forward tel	Any new messages will be forwarded to the phone number specified. <i>VG Dialer option must be enabled to allow VoiceGuide to dial out and deliver any new voicemail messages.</i>
Email	If a forwarding Email address is specified, all recorded messages for this mailbox will be sent by Email to the address specified. If the message needs to be sent to multiple email addresses the individual addresses must be separated by semicolons "
Copy To	What other Voicemail boxes any messages left in this voicemail box should be copied to. The list can include individual voicemail boxes or Distribution Lists.
Pager tel	The telephone number dialed to deliver a numeric pager message.

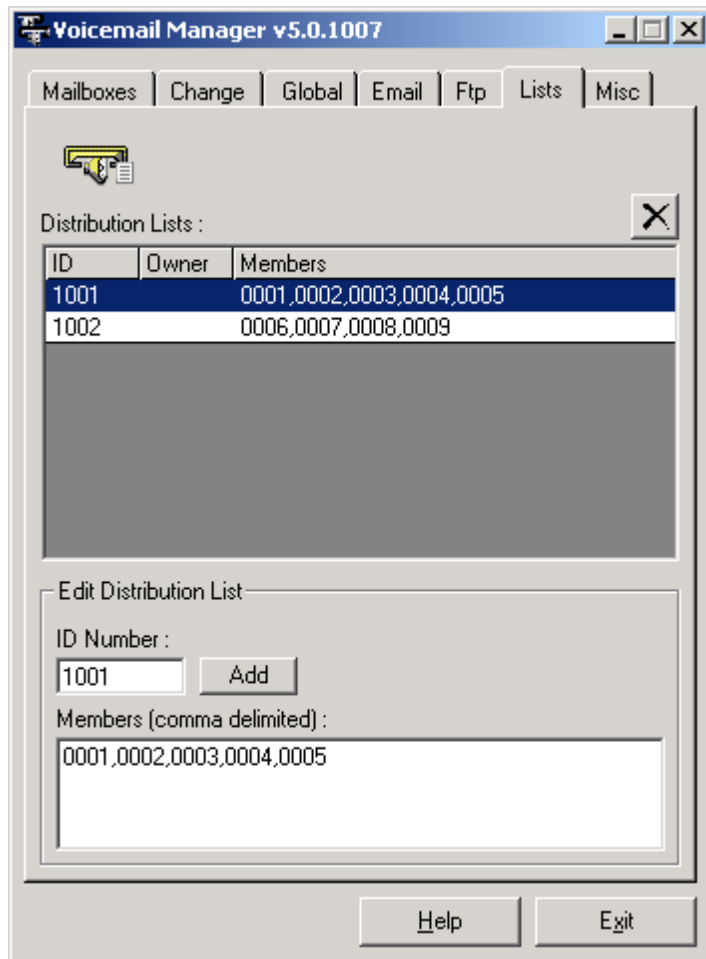
VG Dialer option must be enabled to allow VoiceGuide to dial out and deliver any

new voicemail messages.

Pager msg	<p>The message sent to the pager. VoiceGuide will pause for a couple of seconds after dialing the telephone number and before dialing the message. If this pause is not long enough, extra wait time can be specified by tying commas at the beginning of the message. The example in the captured screen above has three commas.</p> <p>The length of a pause generated by each comma differs from system to system and its best to determine by trial and error the number of commas to use in order to have reliable pager message delivery from your VoiceGuide system to your pager provider.</p> <p>Also in the example above a Result Variable was used in the message. This shows how the caller's ID or any other data entered by the caller in another part of the script can be sent as part of a pager message.</p>
Email	<p>If a forwarding Email address is specified, all recorded messages for this mailbox will be sent by Email to the address specified. If the message needs to be sent to multiple email addresses the individual addresses must be separated by semicolons "</p>
Run	<p>What program to run after a message has been left for this voicemail box. This can be used to start 3rd party command line utilities to forward alerts to alphanumeric pagers, or to mobile phones via SMS. Another possible use is for post-processing of recorded message or forwarding of the recorded message through other means. (The filename of the just recorded message can be accessed using the \$RV[VoicemailMesasgeXXXX] Result Variable - it returns the filename of the last recorded voicemail message in a particular voicemail box - which is specified by replacing XXXX with the number of the voicemail box.)</p>

Distribution Lists

Messages recorded in the voicemail system can be sent to individual voicemail boxes or Distribution Lists. Distribution Lists are by default global and available for use to anyone. This screenshot shows how the Distribution Lists are defined:



Voicemail configuration file

The voicemail information is stored in file `VmBoxList.xml` in the Data subdirectory.

The Voicemailbox data is stored in an XML format, allowing easy viewing and manipulation of this data by users own tools.

Any changes made to the Voicemail configuration file by the Voicemail Manager or by any other program will also be loaded by VoiceGuide immediately after the new version of the file is saved. This allows for outside programs to add and delete voicemail boxes at will.

Email Forwarding

The Voicemail Manager allows you to define the standard template for the Subject and Body of the emails sent with forwarded recorded messages.

The Email's Subject and Body can also be customized on a per voicemail box basis by creating these test files in VG's `\system\vm\` subdirectory:

```
\system\vm\EmailForwardTemplate_XXXX_subject.txt
```

`\system\vm\EmailForwardTemplate_XXXX_body.txt`

Where XXXX is to be replaced by the voicemail box number.

If the Voicemail-box specific files subject or body can be found then the following files will be used if they exist:

`\system\vm\EmailForwardTemplate_subject.txt`

`\system\vm\EmailForwardTemplate_body.txt`

RV's can be used within these template files as well.

The voicemail scripts can also be modified to append information to the message body - please examine the voicemail scripts to see how this is done.

Note

Voicemail Manager reads the contents of the `VmBoxList.xml` file on start, and saves all the information on exit - overwriting the previous `VmBoxList.xml`. This means that any changes made to `VmBoxList.xml` by voicemail users managing their accounts (changing PINs or Forwarding numbers) by any other programs while the Voicemail Manager is open will be lost.

Hence it is recommended that Voicemail Manager only be used only at times when no callers are performing administration functions on their Voicemail account.

The voicemail introduction messages are stored by default in VG's `\data\VmWelc\` subdirectory.

The messages left in voicemail boxes are stored by default in VG's `\data\VmSave\` subdirectory.

Voicemail Menus

This is a summary of what keys can be used in the various voicemail menus. A .PDF file (VoicemailMenus.pdf) showing the menus in a graphical format is also supplied in VoiceGuide's directory.

Greeting Message Menu

While listening to the greeting message the following keys can be used:

1	Stop playing the greeting message, and start recording a message
4	Group browse to the previous voicemail box and play it's greeting message
5	Replay the greeting message for this voicemail box
6	Group browse to the next voicemail box and play it's greeting message
8	Go to Voicemail Login Menu
0	Exit the voicemail system
#	Stop recording and play the menu which speaks the options listed above

Leave Message Menu

This is a menu used when recording a message to be left as voicemail, or editing that recorded message. To access this menu the keys **1** or **#** must be pressed after the message has been recorded.

1	Stop recording and play the recorded message
2	Delete the recorded message and start recording again
3	Save the recorded message and play the greeting message for this voicemail box again
4	Delete the recorded message and group browse to the previous voicemail box and play it's greeting message
5	Delete the recorded message and play the greeting message for this voicemail box again
6	Delete the recorded message and group browse to the next voicemail box and play it's greeting message
8	Save the recorded message and go to Voicemail Login Menu
0	Save the recorded message and exit the voicemail system
#	Stop recording and play the menu which speaks the options listed above

Voicemail Login

When logging into the voicemail box, a prompt will be played asking for the voicemail ID, and then for the PIN number to be entered. If the two have been entered correctly then the Voicemail Box Main Menu will be played.

Voicemail Box Main Menu

After entering the voicemail box ID and PIN, the following keys can be used:

1	Listen to new voicemail messages
----------	----------------------------------

2	Listen to saved voicemail messages
3	Administer the greeting message
4	Change voicemail box PIN
5	Change the forwarding telephone number
6	Record a message and send it to other voicemail boxes
7	Administer voicemail box name
8	Exit and go to Login Menu
0	Exit the voicemail system
#	Play the message counts

Retrieve Messages Menu

When listening to new or saved messages, the following keys can be used:

1	Replay the current message
2	Save the current message and play the next message
3	Delete the current message and play the next message
4	Play the previous message.
5	Replay the current message
6	Play the next message.
7	Play the date and time when the message was recorded, and the Caller ID of who left the message.
8	Exit and go to Login Menu
9	Forward current message to other voicemail boxes
0	Return to Voicemail Access Menu

Administer Greeting Message Menu

When recording voicemail box's greeting message the following keys can be used:

1	Replay the current greeting message
2	Delete the current greeting message and record a new greeting message
3	Save the current greeting message and return to Voicemail Access Menu
4	Delete the current greeting message and return to Voicemail Access Menu

Administer Mailbox Name/Description Menu

When recording voicemail box's name/description the following keys can be used:

1	Replay the current greeting name
2	Delete the current name and record a new name
3	Save the current name and return to Voicemail Access Menu
4	Delete the current name and return to Voicemail Access Menu

Record Message Menu

When recording a message to be sent to other voicemail boxes or distribution lists.

1	Stop recording and play the recorded message
2	Delete the recorded message and start recording again
3	Save the recorded message and start entering the recipients of the voicemail message. These can be individual voicemail boxes or distribution lists. Please

4

0

#

see the "Select Message Recipients Menu" on how the recipients of the message are specified.
Delete the recorded message and return to the Voicemail box Administration Menu
Save the recorded message and exit the voicemail system
Stop recording and play the menu which speaks the options listed above

Select Message Recipients Menu

This menu allows the caller to select further recipients of the message just recorded. Once in this menu the caller must enter the voicemail group numbers or the voicemail box numbers.

After entering each group number or voicemail box number the caller must enter the "#" key. After the last destination is entered the caller must press the "#" key twice, and they will be returned to the Voicemail box Administration Menu.

Changing the keys used

The Voicemail system is written as a set of VoiceGuide scripts located in VoiceGuide's \system\vm\ directory. These scripts can be edited giving the user full control to make any modifications and extensions which may be required. This includes changing of the keys used to access various voicemail system commands and features.

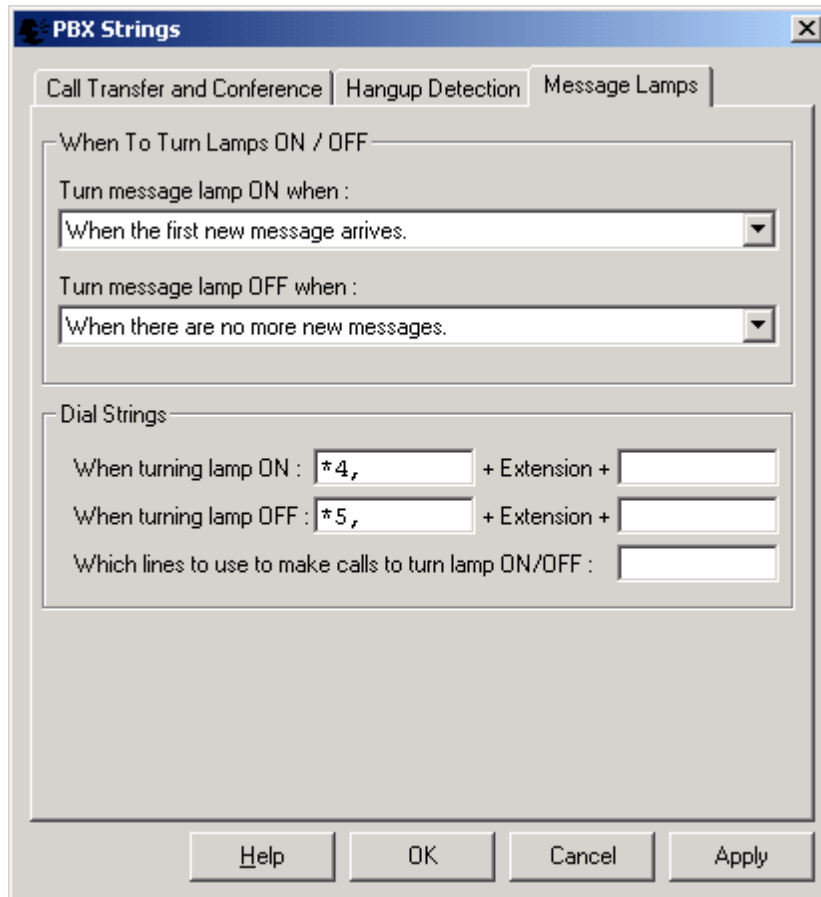
Only the Enterprise (and Evaluation) versions of VoiceGuide use the modifiable scripted version of the voicemail - ie: the .vgs scripts in the \system\vm\ directory.

The Standard and Professional versions of VoiceGuide use the non-modifiable version of the voicemail system.

Message Lamps

VoiceGuide can instruct the PBX to turn on the telephone extension's 'message lamp' when a new voicemail message has arrived, and to turn off the 'message lamp' once the message has been heard.

In VoiceGuide **Script Editor** got to the **Edit** menu and select go **PBX Command Strings** option, then click on the **Message Lamps** tab. The following should be displayed:



The screenshot shows a dialog box titled "PBX Strings" with three tabs: "Call Transfer and Conference", "Hangup Detection", and "Message Lamps". The "Message Lamps" tab is selected. The dialog is divided into two main sections. The top section, "When To Turn Lamps ON / OFF", contains two dropdown menus. The first is labeled "Turn message lamp ON when :" and has the option "When the first new message arrives." selected. The second is labeled "Turn message lamp OFF when :" and has the option "When there are no more new messages." selected. The bottom section, "Dial Strings", contains three text input fields. The first is labeled "When turning lamp ON :" and contains the text "*4," followed by a small text box and "+ Extension +". The second is labeled "When turning lamp OFF :" and contains the text "*5," followed by a small text box and "+ Extension +". The third is labeled "Which lines to use to make calls to turn lamp ON/OFF :" and is an empty text box. At the bottom of the dialog are four buttons: "Help", "OK", "Cancel", and "Apply".

The top section is used to select when the signals to the PBX to turn the message lamps on/off should be sent.

The "Dial Strings" section is used to define what are the DTMF signals which VoiceGuide needs to send to the PBX to turn the message lamps on/off.

In the example shown in the above screenshot VoiceGuide has been configured to send DTMF tones *4,*Ext* (where *Ext* is the Extension number) when the first new voicemail message arrives and *5,*Ext* when the voicemail box owner has saved or deleted all of the 'New' voicemail messages.

The sending of the DTMF tones is done by VoiceGuide picking up one of the lines and dialing the DTMF tones specified. Eg. if in the above example the extension number was 1234 then to turn the lamp ON VoiceGuide would pick up a line and dial *4,1234 (with the comma indicating a short pause) . It is possible to limit on what lines VG is allowed to send these lamp on/off messages. The lines on which the sending of message lamp messages is allowed can be specified by just typing in the Line IDs separated by commas. If the line selection text box is left blank then any of the currently

available lines will be used.

The Message Lamp feature is only enabled in the 'Evaluation' and 'Enterprise+Dialer' versions of VoiceGuide.

Loading Numbers to Call

The Dialer add-on enables VoiceGuide to dial out and play a message or run a VoiceGuide script once the call has been answered. Outbound calls can be scheduled using any of these methods:

- Dial List Loader program.
- [Send Phone Message](#) module in a VoiceGuide script.
- Using VoiceGuide WCF/COM interface function [Dialer_OutDialQueAdd](#).
- Saving list of numbers to be called in an 'Out Dial' file, which VoiceGuide reads in automatically.
- Adding dial entries directly to the OutDialQue database.

When using Dialogic cards the system can detect whether a real person or an answering machine has answered the call. Different messages or scripts can be specified in either case. When an answering machine answers the call the message or script will start after the welcoming message stops playing.

When using a voice modem to dial out a prompt message needs to be specified which will be played in a loop until the person who answers the call presses key on their telephone keypad. This is because a voice modem cannot detect when a call is answered. Only after the called person presses the key on their telephone keypad will the message or script be played. If an answering machine answers calls placed using voice modems it will end up recording the prompt message being played in a loop.

Outbound Call Loader

Allows easy scheduling of calls.

Outbound Call Loader

Phone Numbers

Phone Numbers (one per line):

```

2023575900|David S Ferriero
2023575900|Debra Steidel Wall
2023575100|John O Hamilton
3018370939|Ismael Martinez
3018371750|Gary M Stern
3018373026|Gary M Stern (A)
3018372928|Chris M Runkel
2023575263|Kathleen M Williams
3018373018|James E Springs (A)
3018371966|John M Simms
3018372941|Matthew T Elliott
3018373018|James E Springs
2023577464|Donna M Garland
2023577467|Katherine B Balanza
2023575300|Chris Isleib
3018373534|Paula Jonak

```

Phone Numbers File:

Answer Timeout (sec) Campaign Name:

Retries Number Scheduler

Retries Delay (min)

Call Priority (1=highest)

Added to CallQueue : 176/176. Not added (in DNC list) : 0

Outgoing Call parameters set using the Dial List Loader:

Phone numbers

The telephone numbers to be dialed can be typed directly into this text box. The numbers should be entered one per line.

A range of numbers can be specified by placing an "X" in the number location which covers the desired range. eg: 5627XXX results in a thousand telephone numbers: 5627000 to 5627999 to be called. Typing 56274XX would result in a hundred numbers being called: 5627400 to 5627499.

Prefix

The number to add to before each of the numbers provided in the 'Phone numbers' text box or in the 'list file'. This entry is used if a certain number has to be dialed to reach an outgoing line or if using an override code when dialing.

Phone Numbers File

Allows to select the text file from which to import the list of numbers to call. The file should contain each number on its own new line.

A range of numbers can be specified by placing an "X" in the number location which covers the desired range. eg: by typing 5627XXX we indicate that a thousand telephone numbers: 5627000 to 5627999 should be called. Typing 56274XX would result in a hundred numbers being called: 5627400 to 5627499.

Live Person Answer

What VoiceGuide script will be started or what sound file will be played when the call is answered by a real person. Specifying the starting module can be made using notation: *Script Filename|Module Title*

Answering Machine Answer

When the call is answered by an answering machine VoiceGuide will play this message/script. Answering machines can only be detected if a Dialogic card is used.

If an answering machine is detected VoiceGuide will wait until the answering machine's welcome message finishes before starting the script/sound file specified here.

RETRY : call will be attempted again if an answering machine is encountered (up to the maximum number of redials allowed).

IGNORE The script/file specified when the live person answers will be used.

DISABLE Detection of when the call has been answered is disabled.

Answering Machine detection relies on the Dialogic card listening to the first word spoken by the caller, and determining based on what it heard whether that word sounded like it was spoken by a live person answering the call, or whether that word sounded like it was spoken by an answering machine. This means that VoiceGuide needs to wait until the Dialogic card hears the word being spoken and then advises VoiceGuide of its decision. This incurs a delay of about a second after the first word is spoken before VoiceGuide is advised whether to play the "real person" or the "answering machine" script.

If the call is made on T1/E1 ISDN lines (and usually VoIP) VoiceGuide is able to detect the exact time at which the call has been answered. If the "Answering Machine" message option is not set then VoiceGuide will commence playing of the "Live person answer" script immediately when the handset is picked up.

Fax to Send

If a fax is specified then VoiceGuide will send the specified fax on this call. This can be a .PDF or a .TIFF file. Note that when this field is specified VoiceGuide will start playing the Fax Calling Tone immediately after the number is dialed and only a fax send will be attempted during the call.

Do Not Call List (DNC List)

File containing a list of telephone numbers which will never be called by VoiceGuide, even if they were inadvertently specified in the "Phone numbers to be called" file or text box. The file must contain each number on a separate line.

Result Variables

Result Variables can be supplied to be used by the scripts which are ran once outgoing call is answered.

The format of this field is `[RvName]{RvValue}`. Multiple Result Variables can be specified by listing as many name-value pairs as needed. eg: `[rv1]{val1}[rv2]{val2}[rv3]{val3}`

Call Options

The Options field can be used for other special settings. Eg. to specify the outgoing CallerID used for the call. Including this setting: `<CallerId>5551234</CallerId>` would result in CallerID to be set to 5551234 on outgoing calls (lines used must support the setting of outgoing CallerID). For information on configuring ISDN lines to send out the CallerID Name as well please see: <http://voiceguide.com/forums/index.php?showtopic=6079>

Result Files

In addition to CDR Logs and Script Logs, outcome of all outgoing calls is saved in VoiceGuide's `\data\` subdirectory.

`OutDial_Contacted_Human.txt`

List of calls answered by a real person.

`OutDial_Contacted_AM.txt`

List of calls answered by an answering machine.

`OutDial_Contacted_Fax.txt`

List of calls answered by a fax machine person.

`OutDial_Uncontactable_NoAnswer.txt`

List of calls which were not answered and rung out awaiting answer.

`OutDial_Uncontactable_Busy.txt`

List of calls which were not answered as the busy tone was heard.

`OutDial_Uncontactable_OnDontDialList.txt`

List of calls which were not made as the telephone number was on the supplied "Do Not Call list".

OutDial_Uncontactable_NoDialer.txt

List of calls which were not dialed as the Dialer option is not enabled.

OutDial_SIT_Reorder.txt

List of calls for which the Special Information Tone (Reorder) was played by the phone company.

OutDial_SIT_NoCircuit.txt

List of calls for which the Special Information Tone (No Circuit) was played by the phone company.

OutDial_SIT_CustIrReg.txt

List of calls for which the Special Information Tone (CustIrReg) was played by the phone company.

OutDial_SIT_Unknown.txt

List of calls for which an unknown Special Information Tone was played by the phone company.

Out Dial file

An 'OutDial' file can also be used to add new telephone numbers to be dialed by VoiceGuide. When the file is created, VoiceGuide will read in the file and delete it after reading it's contents.

'OutDial' file should be created in VoiceGuide's \data\ subdirectory.

The filename must begin with "OutDial" and have the ".xml" suffix.

Eg: OutDial_080521175323a.xml

Multiple 'OutDial' files can be created at the same time, and will be read in by VoiceGuide within about a second of them being created.

When supplying outgoing call data to VoiceGuide using input files it is advisable to first create the file under a different name, and after the file writing/copying is completed then rename the file to the OutDial*.xml format. This would ensure that VoiceGuide would not attempt to open the file while it's contents are still being written to by another process. If creating multiple

Out Dial Input file

The syntax of the OutDial file as used by VoiceGuide v7 is:

```
<OutDialEntry>
  <PhoneNumber>sPhoneNumber</PhoneNumber>
  <PhoneNumberPrefix>sPhoneNumberPrefix</PhoneNumberPrefix>
  <ActivateTime>sCallTime</ActivateTime>
  <DayTimeStart>sDayTimeStart</DayTimeStart>
  <DayTimeStop>sDayTimeStop</DayTimeStop>
  <DaysCallAllowed>sDaysCallAllowed</DaysCallAllowed>
```

```

<PortSelection>sPortSelection</PortSelection>
<CampaignName>sCampaignName</CampaignName>
<Priority>iPriority</Priority>
<OnAnswerLive>sOnAnswerLive</OnAnswerLive>
<OnAnswerMachine>sOnAnswerMachine</OnAnswerMachine>
<OnAnswerFax>sOnAnswerFax</OnAnswerFax>
<OnNotAnswered>sOnNotAnswered</OnNotAnswered>
<OnRetriesExhausted>sOnRetriesExhausted</OnRetriesExhausted>
<AnswerTimeout>iAnswerTimeout</AnswerTimeout>
<RetriesLeft>iRetriesLeft</RetriesLeft>
<RetriesDelay>iRetriesDelay</RetriesDelay>
<RV>sRV</RV>
<CallOptions>sCallOptions</CallOptions>
<Escalation>
    sEscalation
</Escalation>
</OutDialEntry>

```

Where:

Phone number to be dialed.

Prefix to the phone number to be dialed. Prepended to the phone number before dialing.

Time the call is to be made. Format is YYYY-MM-DD HH:NN:SS If this setting is blank or is omitted then the call will commence immediately. Format may also be: Now+X where X is the number of minutes from the current time that the call should be scheduled for.

Time of the day from which this number can be dialed. Format is: HHNN and the time is specified in 24-hour format. eg: 0800 would mean: this number may be called after 8am. If this setting is blank or is omitted then the call will be allowed to be made from midnight onwards.

Time of the day after which this number can no longer be dialed. Format is: HHNN and the time is specified in 24-hour format.

eg: 2100 would mean: do not dial this number after 9pm. If this setting is blank or is omitted then the call will be allowed to be made right up to midnight.

On which days of the week can this number be dialed. Format is : [Mo][Tu][We][Th][Fr][Sa][Su] eg: MoWe would mean: this number may only be called on Mondays and

Wednesdays. If this setting is blank or is omitted then the call will be allowed to be made on every day of the week.

If the outbound call may only be made on particular phone lines then the ports to which those lines are attached should be listed in this setting. The ports should be specified as a comma delimited list of physical ports (eg: 1, 2, 3). If any of the lines are allowed to to be used then this setting should be left blank.

eg: 3,4,5 would indicate that the outgoing call can only be made on third, fourth or fifth port.

User defined name campaign name associated with this call. Used to assist in segregating of outgoing calls based on their porpoise or source etc.

At what priority level the calls should be made. A lower number indicates a higher priority. ie: 1 is the highest priority, then 2 etc etc. Number of priority levels is unlimited.

VoiceGuide script or sound file which will be used when the call is answered by a real person The full path to this script or sound file must be specified. Specifying the starting module can be made using notation: *Script Filename|Module Title*

VoiceGuide script or sound file which will be used when the call is answered by an answering machine. The full path to this sound file or script must be specified. If this setting is blank or is omitted then the sOnAnswerLive setting will be used.

If the word **none** is specified, then no file will be played and the call will be hung up. The

call will still be considered as having been completed.

If the word **retry** is specified, then the no file will be played and the call will be hung up, with the call not considered completed - further call attempts will be made.

Fax to send. This can be a .PDF or a .TIFF file. Note that when this field is specified VoiceGuide will start playing the Fax Calling Tone immediately after the number is dialed. The OnAnswerLive and OnAnswerMachine will be ignored as only a fax send will be attempted during the call.

VBScript or .EXE/.COM/.BAT/etc. to run when the outgoing call has not been answered.

VBScript or .EXE/.COM/.BAT/etc. to run when the outgoing call has not been answered and there are no more retries left. This command is ran in addition to the OnNotAnswered command.

How many seconds VoiceGuide will wait for the call to be answered. If this setting is blank or is omitted then the answer timeout will be set to 60 seconds.

How many times VoiceGuide will re-dial the phone number, before abandoning trying to contact this number. If iRetriesLeft is set to 0 then only one call will be made. If this setting is blank or is omitted then a value of 2 will be used. (ie: up to 3 call attempts in total)

How many minutes VoiceGuide will wait between successive attempts (if iCallRetries is set to 1 or more). If this setting is blank or is omitted then a value of 5 will be used.

List of custom Result Variables which can be used by the VoiceGuide script above. The format of this field is [RvName]{RvValue}. Multiple Result variables can be specified by listing as many name-value pairs as needed. eg: [rv1]{val1}[rv2]{val2}[rv3]{val3}

Used for the specifying of custom call options. eg. CallerID to be sent.

Alternate numbers to be dialed if the call has not been answered. Once the number of retries calling sPhoneNumber is exhausted, then then alternative escalation number will be dialed immediately. Format is in <OutDialEntry>...</OutDialEntry> structure format - this allows for multiple escalation numbers to be specified.

Example 1 : Simple message delivery to a single number

Use > Any simple message / script delivery.

s :

Scenario: Call a telephone number 5553423, and run script d:\scripts\alarm.vgs when call is answered. Call is to be made immediately. The OutDial file should contain:

```
<OutDialEntry>
  <PhoneNumber>5553423</PhoneNumber>
  <OnAnswerLive>d:\scripts\alarm.vgs</OnAnswerLive>
</OutDialEntry>
```

If the call is not answered then the number will be called up to two more times at 5 minute intervals.

Example 2 : Scheduling a call for a particular time.

Uses > Wake up calls.
> Appointment reminder service.
: > Any announcements which need to be sent out at a particular time.

Scenario: Call the telephone number 5553328 at 6:30 in the morning on a 12th June 2008 and run the script d:\scripts\todays_specials.vgs - passing to the script Result Variables which that script uses (in this case it will be pricing of the items on special for this customer). If an Answering Machine answers then run a modified version of the script which just plays the prices and leaves contact details (d:\scripts\todays_specials_AM.vgs). The OutDial file should contain:

```
<OutDialEntry>
  <PhoneNumber>5553328</PhoneNumber>
```



```

<ActivateTime>2008-06-12 06:30:00</ActivateTime>
<OnAnswerLive>d:\scripts\todays_specials.vgs</OnAnswerLive>
<OnAnswerMachine>d:\scripts\todays_specials_AM.vgs</OnAnswerMachine>
<RV>[ClientID]{123456}[PriceWidget]{125}[PriceDelivery]{7.50}</RV>
</OutDialEntry>

```

Example 3 : Ensuring call is made during particular days/hours.

Uses > Surveys.
 : > Distributing information to a large client base.
 : > Whenever making a large number of calls and need to ensure that calls are only made between the hours allowed

Scenario: Call the telephone number 5553328 at between 10am and 6pm on any weekday, and run script d:\scripts\paymentoverdue.vgs - passing to the script Result Variables which that script uses (in this case the customer ID and the amount owing). If an Answering Machine answers then run a modified version of the script which just plays the amount owing and leaves contact details (d:\scripts\paymentoverdue_AM.vgs). The OutDial file should contain:

```

<OutDialEntry>
  <PhoneNumber>5553328</PhoneNumber>
  <DayTimeStart>1000</DayTimeStart>
  <DayTimeStop>1800</DayTimeStop>
  <DaysCallAllowed>MoTuWeThFr</DaysCallAllowed>
  <OnAnswerLive>d:\scripts\paymentoverdue.vgs</OnAnswerLive>
  <OnAnswerMachine>d:\scripts\paymentoverdue_AM.vgs</OnAnswerMachine>
  <RV>[ClientID]{4453566}[AmountOwing]{14325}</RV>
</OutDialEntry>

```

Specifying the days on which calls can be made and the time range during which the call can be made is useful if a large number of calls are queued and there is a need to ensure that the system will not make the calls on odd days or hours.

When specifying a large number of calls it is usually easier to use the Telephone Number Loader program, rather than specifying all the numbers using the Out Dial file.

Example 4 : Escalated calls

Uses : > Delivering information to on-call staff, with supervisor escalation if calls by main staff not answered.
 : > Follow-me message delivery - where a number of possible contact numbers is given.
 : > Whenever a message needs to be delivered to one of a range of telephone numbers.

Scenario 1: Call the number 5551111 immediately and run the script d:\scripts>alert.vgs - passing to the script Result Variables which that script uses (in this case it will be information on a problem reported by customer). If the number does not answer then wait 5 minutes and try calling it again. If after 3rd call attempt the number still does not answer then try calling 5552222 - again calling up to 3 times at 5 minute intervals. If an answering machine answers the call then start the d:\scripts>alert.vgs script as well (script should be designed to just play the information to caller). If on 3rd call attempt the second number still does not answer then run a VB Script d:\scripts\notanswered.vbs. The OutDial file should contain:

```

<OutDialEntry>
  <PhoneNumber>5551111</PhoneNumber>
  <OnAnswerLive>d:\scripts>alert.vgs</OnAnswerLive>
  <RetriesLeft>2</RetriesLeft>
  <RetriesDelay>5</RetriesDelay>
  <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
  <Escalation>
    <OutDialEntry>

```

```

    <PhoneNumber>5552222</PhoneNumber>

    <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>

    <RetriesLeft>2</RetriesLeft>

    <RetriesDelay>5</RetriesDelay>

    <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>

    <OnRetriesExhausted>d:\scripts\notanswered.vbs</OnRetriesExhausted>

</OutDialEntry>

</Escalation>

</OutDialEntry>

```

Scenario 2: Message delivery to a Line Person only : Call the number 5551111 immediately and run the script d:\scripts\alert.vgs - passing to the script Result Variables which that script uses (in this case it will be information on a problem reported by customer). If the number cannot be reached or if an answering machine answers the call then immediately call another number instead (5552222) and if that number is not answered by a live person then call a third number (5553333). If a live person could not be reached on any of these numbers then wait 10 minutes and try calling all 3 numbers again. If after the second call attempt to all 3 numbers no live person could be reached then run a VB Script d:\scripts\notanswered.vbs In this scenario it is acceptable to deliver the message to a Line Person only, calls answered by Answering Machines will not be regarded as successful message delivery calls and no message will be left on the answering machine. The OutDial file should contain:

```

<OutDialEntry>
    <PhoneNumber>5551111</PhoneNumber>
    <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
    <OnAnswerMachine>retry</OnAnswerMachine>

```

```

<RetriesLeft>0</RetriesLeft>
<RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
<Escalation>
  <OutDialEntry>
    <PhoneNumber>5552222</PhoneNumber>
    <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
    <OnAnswerMachine>retry</OnAnswerMachine>
    <RetriesLeft>0</RetriesLeft>
    <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
    <Escalation>
      <OutDialEntry>
        <PhoneNumber>5553333</PhoneNumber>
        <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
        <OnAnswerMachine>retry</OnAnswerMachine>
        <RetriesLeft>0</RetriesLeft>
        <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
        <Escalation>
          <OutDialEntry>
            <PhoneNumber>5551111</PhoneNumber>
            <CallTime>Now+10</CallTime>
            <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
            <OnAnswerMachine>retry</OnAnswerMachine>
            <RetriesLeft>0</RetriesLeft>
            <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
            <Escalation>
              <OutDialEntry>
                <PhoneNumber>5552222</PhoneNumber>
                <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
                <OnAnswerMachine>retry</OnAnswerMachine>
                <RetriesLeft>0</RetriesLeft>
                <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
              </Escalation>
            </Escalation>
          </Escalation>
        </Escalation>
      </Escalation>
    </Escalation>
  </Escalation>
</Escalation>
<RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
<Escalation>
  <OutDialEntry>
    <PhoneNumber>5553333</PhoneNumber>
    <OnAnswerLive>d:\scripts\alert.vgs</OnAnswerLive>
    <OnAnswerMachine>retry</OnAnswerMachine>
    <RetriesLeft>0</RetriesLeft>
    <RV>[ClientID]{554}[Severity]{2}[ContactTel]{5559743}</RV>
    <OnRetriesExhausted>d:\scripts\notanswered.vbs</OnRetriesExhausted>
  </OutDialEntry>
</Escalation>

```

```
        </Escalation>
    </OutDialEntry>
    </Escalation>
    </OutDialEntry>
    </Escalation>
    </OutDialEntry>
    </Escalation>
    </OutDialEntry>
    </Escalation>
</OutDialEntry>
```

The above script will result in the numbers being called in the following order : 55511111, then if no answer dial 55522222, and if no answer then dial 55533333, then wait for 10 minutes, and repeat calling 55511111 then 55522222 then 55533333. Dialing of further numbers would stop as soon as one of the calls is answered.

Please note that the above script will only work if Dialogic cards are used - detection of whether a Human or an Answering Machine answers the call can only be made by a Dialogic card - Voice modems report all outgoing calls as 'Connected to a Human' immediately after dialing the number.

Detect Call Answer

How VoiceGuide will detect that an outgoing call has been answered depends on what telephone lines/trunks are used.

Analog

Dialogic cards can detect when a call has been answered and can fairly reliably distinguish if a call has been answered by a live person or by an answering machine. As soon as the recipient of call starts saying something the Dialogic card can tell whether the call has been answered by a human or by a machine. VoiceGuide will then run the appropriate script.

If answering machine detection is disabled then the Dialogic card will just wait for any spoken sounds (live or machine), and once it hears anything (or ringback has gone away) then it will report call as connected. The ringback tone for which the Dialogic card is to listen out for needs to be correctly specified. The default value will work for most ringbacks, but sometimes it will be necessary to set the ringback tone definition in ConfigLine.xml file (if using VG for Dialogic). The TID_RNGB1 tone definition needs to be used. There is another ringback tone definition available - TID_RNGB2 - but we have found that many cards will not detect the ringback defined in TID_RNGB2 correctly.

If recipient of call does not say anything after answering the call (or there is just silence on the line) then the Dialogic card will deduce that the call has been answered only when it realizes that the 'ringback' signal has gone away.

If the Dialogic card does not correctly detect the ringback tone on the line, then after a timeout (about 8 seconds) it will report the outgoing call as connected even though the outgoing line is still ringing.

T1 and E1 ISDN

On ISDN connections the Dialogic card receives an indication from Telco the instant the call recipient picks up the handset.

If answering machine detection is disabled then then the digital card will report the call as connected immediately when the recipient of the call picks up their handset, and the script will be started.

If answering machine detection is enabled then, just like Dialogic Analog cards, the Dialogic Digital cards will be able to determine whether the call has been answered by a human or by a machine as soon as the recipient of call starts saying something. VoiceGuide will then run the appropriate script.

If answering machine detection is enabled and recipient of call does not say anything after answering the call then the Dialogic card will timeout out after a few seconds of receiving the 'handset is offhook' signal and report the call as connected.

Similarly, an indication that the destination telephone is busy or disconnected is sent back to the digital card from the network immediately - allowing for quick abandoning of those calls and dialing of another number if needed.

VoIP (SIP)

On VoIP connections VoiceGuide receives an indication whe the call recipient picks up the handset.

Answering machine detection is made once spoken sounds are heard.

The sound quality of VoIP connections is sometimes not as good as when Analog or Digital trunks are used, and hence the precision with which answering machines vs. live persons are detected is sometimes worse then on Analog or Digital systems.

Detecting Answering Machines

Dialogic cards ands SIP drivers can detect if a call has been answered by an answering machine or by a human. No other telephony drivers can do that and that is why Dialogic drivers are the best choice for systems which use outbound dialing.

VoiceGuide uses Dialogic's answering machine detection mechanism to distinguish whether a person or an answering machine has answered the call. Dialogic's answering machine detection uses frequency based analysis and detection of whether the speech is coming from live person or answering machine is made in less then a second. If Dialogic informs VoiceGuide that it has detected an answering machine then VoiceGuide will wait for the speaking on the line to finish before playing the answering machine message or starting the script.

Answering Machine detection is enabled by default. The following keywords can be used in the "Answering Machine" text field to modify this behaviour:

RETRY : VoiceGuide will hang up and dial again (up to the maximum number of redials allowed).

IGNORE : VoiceGuide will start the "Live Person Answer" script.

DISABLE VoiceGuide will disable Dialogic's detection of when the call has been answered.

If Answering Machine was detected, the `$RV[AmWelcMsg]` will store the legth of the Answering Machine message (in 100ms units).

False reports of an Answer Machine answer when a Live Peron actually answered the call:

On poor quality lines (usually low quality VoIP and mobile phone connections) the Dialogic card will sometimes mistakenly think an answering machine has answered the call when in fact the call was answered by a real person. The reason for this is that the low quality of the connection makes the voice of the person answering the call sound very similar to the sound quality usually expected from an answering machine.

Background noise can also play a factor in false detection of answering machines (eg: radio playing in background, machine noise, etc)

False reports of a Live Person answer when an Answer Machine actually answered the call:

If the answering machines/voicemails welcoming message is recorded in high quality then it is possible for the Dialogic card to mistake it for a real person answering the call. The normal every-day analog based answering machines are usually correctly detected by Dialogic.

One approach to work around this would be to first play in the "Live Answer" script a short message that says "Please wait" and then start recording. Set silence length to be short beforehand - say about half a second. In the record module use a "on {silence} goto..." path and point to start of the actual start of the script. When the recording finishes due to silence and the actual script starts then the first message played will be recorded on the answering machine, or heard by the caller. This approach effectively uses the record module will wait till answering machine finishes playing, but will result in a slightly delay in playing of the first message to live people. Some people do not place a Play module before this record module, so the live caller would just experience a half second delay before the main script is started.

Fine tuning Dialogic Answer Detection Parameters

With VoiceGuide configuration files it is possible to set all entries in the Dialogic DX_CAP structure, which includes Dialogic's low level settings that affect the Answering Machine / Live Answer detection parameters:

```
ca_cnosig
ca_noanswer
ca_pamd_failtime
etc.
```

There are about 30-40 settings in the DX_CAP structure, all of which are accessible, allowing total control (on a per-call basis) over how the Dialogic card performs the detection.

For more information on the fields in the DX_CAP structure please refer to Dialogic's documentation - the "Voice API" section.

To change settings from the default values used by VoiceGuide an XML style expression needs to be specified in the "Call Options" field. Example of the expression used is show below:

```
<DX_CAP><ca_cnosig>500</ca_cnosig><ca_pamd_failtime>200</ca_pamd_failtime></DX_CAP>
```

Outbound VoIP calls

When placing an outgoing call the IP address or domain of the destination or Switch/PBX or VoIP Provider relaying the call needs to be specified.

For example, when dialing another extension on the VoIP Switch/PBX the number dialed would be of the format: *extension@ip_address_of_voip_switch*

eg:

2043@10.1.1.11

According to how the VoIP switch is set up, the "*extension*" could also be an external phone number.

when dialing another number through a VoIP provider, the VoIP providers SIP server domain/ address needs to be specified: *telnumber@sipprovider*

eg:

15551231234@callcentric.com

When making an outgoing call on a VoIP line it is usually necessary to specify the CallerID to be used on the outgoing call. This advises the switch relaying the VoIP call as to which account/ subscriber is making the call. The CallerID on outgoing calls can be specified using the <CallerID> tag on the Options field when loading the outgoing call, and is in this format:

<CallerID>*accountnumber@voipprovider*</CallerID>

For example, to place a call through a FreeSWITCH PBX which is installed on a server with IP of 10.1.1.11, and with which the user/extension 1010 has been registered by VoiceGuide, the following entry would need to be placed in the Options field:

<CallerID>1010@10.1.1.11</CallerID>

Sometimes just user ID (or the registered telephone number) is sufficient, eg:

<CallerID>5625551234</CallerID>

And to place a call through a VoIP provider with which the account was registered by VoiceGuide, an entry that includes the domain name of the provider may need to be used. For example, with CallCentric, this may need to be used:

<CallerID>12345678@callcentric.com</CallerID>

Also, on outbound calls the VoIP Switch will often require to authenticate the user before allowing

the outbound call to be made. To allow HMP to process the authentication request it is necessary to specify the VoIP Line registration and authentication parameters. Please see the [VoIP Line Registraton](#) section.

If specific headers need to be added/modified in the outgoing SIP INVITE packet then these can also be specified in the Call Options field when loading the outgoing call. To add a SIP header a "<sip-header>" entry needs to be specified in the Call Options field.

eg: to add a "Remote-Part-ID" header, this entry could be added to the Call Options field:

```
<sip-header>Remote-Part-ID: "Flowroute" <sip:1234567890@sip.flowroute.com>
```

"sip-session-expires" and "sip-min-se" entries can also be specified, like this:

```
<sip-session-expires>999</sip-session-expires>
```

```
<sip-min-se>55</sip-min-se>
```

(or these could be just added using the "<sip-header>" approach)

Multiple headers can be specified. Below is a valid Call Options entry:

```
<CallerID>12121212</CallerID>
```

```
<sip-session-expires>999</sip-session-expires>
```

```
<sip-min-se>55</sip-min-se>
```

```
<sip-header>Remote-Part-ID: "Flowroute" <sip:123412341234@sip.flowroute.com;  
party=calling;screen=yes;privacy=on</sip-header>
```

```
<sip-header>X-Tag: mySpecialCallTag123</sip-header>
```

```
<sip-header>Diversion: <sip:333344445555@sip.mycarrier.com>
```

The WireShark protocol analyzer can be used to capture the traces of the actual SIP messages sent.

Predictive Dialers

Predictive Dialers are automated dialing systems designed to reach contacts and connect them to agents upon call answer.

VoiceGuide can be used to create all the various types of Predictive Dialing systems, by modifying the scripts used by the VoiceGuide dialer on outgoing calls. A VoiceGuide based system can blend inbound, outbound, and voice messaging campaigns as well as run multiple simultaneous campaigns.

Predictive Outbound IVR

The rate at which the Outbound IVR calls are made is dependant on how many agents are predicted to be available at the time when call recipients may request transfer to agent.

Click to Callback

Web page can include a "Call me" button to immediately initiate an outbound call from VoiceGuide, and once call answered and call recipient is verified VoiceGuide can connect the call to agent.

Predictive Dialer

Rate of calls is determined by number of agents predicted to be available when calls are predicted to be answered. If no agents are immediately available to connect to answered call then VoiceGuide can take contact through a personalized IVR script until an agent becomes available.

Power Dialer

Start multiple simultaneous calls per employee at the moment agents are ready. If more calls are answered then expected then VoiceGuide can take contact through an IVR script personalized with customer related information until an agent becomes available.

Progressive / Automated / Preview Dialer

At the moment agents are ready, a single record is displayed for employee review prior to connection and that single number is automatically called. This approach only eliminates the need for agent to manually enter the number to be called, but ensures that an agent will always be ready to speak to the the person answering the call.

External Database Source (v7)

VoiceGuide makes use of a database to store details of queued outgoing calls, CDRs, reportable statistics, etc.

VoiceGuide will by default use an SQLite database, which is sufficient for most applications. The SQLite database file is called vgDb.db and is located in VoiceGuide's \data\ subdirectory.

Sometimes it is desirable that VoiceGuide uses a different database engine instead.

To use another database engine:

1. Create the main Database object to be used by VoiceGuide.
2. Specify the connection string in VoiceGuide's Config.xml, in section <Dialer>

VoiceGuide will create all the necessary Tables/Indexes/etc the first time it uses the user specified database.

The <Dialer> section should be placed inside the Config.xml's <VoiceGuideConfig> section.

MS SQL Server

Example Config.xml <Dialer> section for MS SQL Server:

```
<Dialer>
<OutDialQue_ADODB_Provider>System.Data.SqlClient</OutDialQue_ADODB_Provider>
<OutDialQue_Database>vgDb</OutDialQue_Database>
<OutDialQue_ConnectString>Data Source=10.1.1.11,1402;Database=$DATABASE;User ID=someuser;
Password=somepassword;</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField></OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix></OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix></OutDialQue_SqlSuffix>
</Dialer>
```

After setting the Config.xml to appropriate values you will need to create the database (schema) object named vgDb (or whatever the name specified in the <OutDialQue_Database> section is). Use the Microsoft SQL Server Management Studio to create the database object.

Configuring MS SQL Server

If the SQL Server is on a system separate to VoiceGuide then it needs to be configured to allow external connections. Also the system's firewall needs to be set to pass through the external connection requests to the database.

MySQL

When using MySQL as the VoiceGuide backend DB you should first install the MySQL ADO.NET Data Provider.

Here is an example Config.xml <Dialer> section for MySQL:

```
<Dialer>
<OutDialQue_ADODB_Provider>MySQL.Data.MySqlClient</OutDialQue_ADODB_Provider>
<OutDialQue_Database>vgDb</OutDialQue_Database>
<OutDialQue_ConnectString>Database="$DATABASE";Data Source="10.1.1.9";User Id="someuser";
Password="somepassword";</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField></OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix>SELECT</OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix>LIMIT 1</OutDialQue_SqlSuffix>
</Dialer>
```

After setting the Config.xml to appropriate values you will need to create the database (schema) object named `vgDb` (or whatever the name specified in the `<OutDialQue_Database>` section is). You can use the MySQL Workbench to create the database schema object. You must ensure that the user specified in the Connection string has the rights to fully work with and manage the `vgDb` schema.

Best way to configure and test the ADO.NET connection is to just start the VoiceGuide "Outbound Call Loader" application. On startup the Outbound Call Loader will connect to the database object and run the `Db_Create_MySql.Data.MySqlClient.sql` script creating the Tables and Indexes. You can then use the Outbound Call Loader to load new calls into the system, or try loading the calls into the database yourself directly.

The `<OutDialQue_SqlPrefix>` and `<OutDialQue_SqlSuffix>` values are used when VoiceGuide constructs the SQL query to find the suitable call in the `PortToUse` table.

Oracle

When using Oracle as the VoiceGuide backend DB you should install Oracle's ADO.NET Data Provider (Oracle.DataAccess.Client).

Here is an example Config.xml <Dialer> section for Oracle:

```
<Dialer>
<OutDialQue_ADODB_Provider>Oracle.DataAccess.Client</OutDialQue_ADODB_Provider>
<OutDialQue_Database>vgDb</OutDialQue_Database>
<OutDialQue_ConnectString>Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=10.1.1.26)
(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=$DATABASE)));User Id=HR;
Password=hr;</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField>Disable</OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix>SELECT</OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix></OutDialQue_SqlSuffix>
</Dialer>
```

At this stage we would recommend disabling the PortToUse table if Oracle is used.

Postgres 9.2.x

Here is an example Config.xml <Dialer> section for Postgres 9.2.x (and later):

```
<Dialer>
<OutDialQue_ADODB_Provider>Npgsql</OutDialQue_ADODB_Provider>
<OutDialQue_Database>vgDb</OutDialQue_Database>
<OutDialQue_ConnectString>Server=10.1.1.36;User Id=postgres;Password=postgres;
Database=vgDb;</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField></OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix>SELECT</OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix>LIMIT 1</OutDialQue_SqlSuffix>
</Dialer>
```

For a Postgres installation on the same machine the IP address 127.0.0.1 should be used:

```
<Dialer>
<OutDialQue_ADODB_Provider>Npgsql</OutDialQue_ADODB_Provider>
<OutDialQue_Database>vgDb</OutDialQue_Database>
<OutDialQue_ConnectString>Server=127.0.0.1;Port=5432;User Id=postgres;Password=postgres;
Database=vgDb;</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField></OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix>SELECT</OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix>LIMIT 1</OutDialQue_SqlSuffix>
</Dialer>
```

If Postgres is installed on another server you will need to configure Postgres to allow remote access. This is done by editing Postgres pg_hba.conf configuration file.

Best way to configure and test the ADO.NET connection is to just start the VoiceGuide "Outbound Call Loader" application. On startup the Outbound Call Loader will connect to the database object and run the Db_Create_Npgsql.sql script creating the Tables and Indexes. You can then use the Outbound Call Loader to load new calls into the system, or try loading the calls into the database yourself directly.

SQLite

For completeness, here is an example Config.xml <Dialer> section which is used by default to connect to an SQLite database:

```
<Dialer>
<OutDialQue_ADODB_Provider>System.Data.SQLite</OutDialQue_ADODB_Provider>
<OutDialQue_Database>C:\Program Files\VoiceGuide\data\vgDb.db</OutDialQue_Database>
```

```

<OutDialQue_ConnectString>Data Source=$DATABASE</OutDialQue_ConnectString>
<OutDialQue_PortToUse_LinkField></OutDialQue_PortToUse_LinkField>
<OutDialQue_SqlPrefix>SELECT</OutDialQue_SqlPrefix>
<OutDialQue_SqlSuffix>LIMIT 1</OutDialQue_SqlSuffix>
</Dialer>

```

Call Prioritization

Priority ordering is specified by using the ORDER BY clause when selecting calls from the database. This clause is specified in the OutDialQue_SqlSuffix setting in the <Dialer> section of the Config.xml file:

```

<OutDialQue_SqlSuffix>ORDER BY Priority ASC</OutDialQue_SqlSuffix>

```

If you have a large number of calls loaded then ordering the retrieved calls by Priority can degrade call data retrieval speed. If you are seeing excessive call retrieval times from the database then removing this clause would speed up call retrieval.

If priority is being enabled and a large number of calls are being loaded then we'd recommend using a sever class database like SQL Server or MySql etc.

Loading Calls Directly

This section outlines how calls need to be loaded into the Dialer Database if you would like your own programs to load the calls instead of using the VoiceGuide Telephone Number Loader or the VoiceGuide WCF/COM function or XML file to load the calls.

The outbound calls database uses two tables: CallQue and PortToUse

Addition of new calls into the system involves placing new entries in both tables, with possible multiple entries in the PortToUse table, depending on what port selections need to be specified.

The PortToUse table is used to indicate on which ports the particular call can be made. If a call is allowed to be made on any of the systems ports then a single entry in the PortToUse table needs to be made, with the PortToUse.PortNumber field assigned a value of -1. Otherwise if call can only be made on some of the ports then a new row needs to be placed in PortToUse for each port on which the outgoing call is allowed to be made. The telephony ports on the VoiceGuide system are numbered from 1.

By default the PortToUse table is used when establishing which call is to be made next. Scheduling information is still included in the CallQue table as there is an option of turning off the PortToUse table use altogether if there is no need to limit the ports on which the calls can be made.

It's recommended to use the Outbound Call Loader application to load some calls into the database and then examine the database tables to how the information is placed in the tables.

The SQL statements used can be seen in the vgDialListLoad trace files (see VG's \log\ subdirectory), and it's recommend that these traces be looked at by anybody wanting to see how the Dial List

Loader is actually performing the inserts.

Here are the two main SQL statements used (with parameter placeholders):

```
INSERT INTO CallQue (GUID, PhoneNumber, PhoneNumberPrefix, ActivateTime, Priority,
TimeStart_Mon, TimeStart_Tue, TimeStart_Wed, TimeStart_Thu, TimeStart_Fri, TimeStart_Sat,
TimeStart_Sun, TimeStop_Mon, TimeStop_Tue, TimeStop_Wed, TimeStop_Thu, TimeStop_Fri,
TimeStop_Sat, TimeStop_Sun, CampaignName, OnAnswerLive, OnAnswerMachine, OnAnswerFax,
OnNotAnswered, OnRetriesExhausted, AnswerTimeout, RetriesLeft, RetriesDelay, RV,
CallOptions, EscalationCalls)
OUTPUT Inserted.ID VALUES (@guid, @strNbrToDial, @strPhoneNumberPrefix, @dateActivateTime,
@iPriority, @iTimeStart_Mon, @iTimeStart_Tue, @iTimeStart_Wed, @iTimeStart_Thu,
@iTimeStart_Fri, @iTimeStart_Sat, @iTimeStart_Sun, @iTimeStop_Mon, @iTimeStop_Tue,
@iTimeStop_Wed, @iTimeStop_Thu, @iTimeStop_Fri, @iTimeStop_Sat, @iTimeStop_Sun,
@strCampaignName, @strOnAnswerLive, @strOnAnswerMachine, @strOnAnswerFax,
@strOnNotAnswered, @strOnRetriesExhausted, @iAnswerTimeout, @iRetriesLeft, @iRetriesDelay,
@strRV, @strOptions, @strEscalationCalls);

INSERT INTO PortToUse (CallID, CallGUID, PortNumber, ActivateTime, Priority,
TimeStart_Mon, TimeStart_Tue, TimeStart_Wed, TimeStart_Thu, TimeStart_Fri, TimeStart_Sat,
TimeStart_Sun, TimeStop_Mon, TimeStop_Tue, TimeStop_Wed, TimeStop_Thu, TimeStop_Fri,
TimeStop_Sat, TimeStop_Sun)
VALUES (@iCallID, @guid, @iPortNumber, @dateActivateTime, @iPriority, @iTimeStart_Mon,
@iTimeStart_Tue, @iTimeStart_Wed, @iTimeStart_Thu, @iTimeStart_Fri, @iTimeStart_Sat,
@iTimeStart_Sun, @iTimeStop_Mon, @iTimeStop_Tue, @iTimeStop_Wed, @iTimeStop_Thu,
@iTimeStop_Fri, @iTimeStop_Sat, @iTimeStop_Sun);
```

All the call details are first inserted into the CallQue table, and then a subset of call information is inserted into the PortToUse table. After inserting a row into the CallQue table the value of the autogenerated ID field needs to be retrieved and the value of that ID field is used in the CallID field when inserting related rows into the PortToUse table. This can be done differently depending on what database is used.

In the `INSERT INTO CallQue` example above the ID retrieval is done using the SQL command **OUTPUT Inserted.ID**

The ID retrieval can also be done on some databases using this SQL command:<

```
SELECT DISTINCT @@identity FROM CallQue
```

The above command can be issued at the same time as the Insert command, like this:

```
INSERT INTO CallQue (...) VALUES (...); SELECT DISTINCT @@identity FROM CallQue
```

An alternative approach instead of placing the CallQue.ID in the PortToUse.CallID column would be to use a GUID to link the PortToUse entries to the CallQue entry. Just generate a GUID and write the same GUID into the PortToUse.CallGUID column and into the CallQue.GUID column. A value of -1 can then be placed in the PortToUse.CallID column.

Other Notes

<OutDialQue_PortToUse_LinkField> options are:

ID	The ID value is used to link the entries in the CallQue and PortToUse tables. GUID field is not included in SELECT queries, which means that the CallQue and PortToUse tables do not even need to have the GUID column.
GUID	The GUID value is used to link the entries in the CallQue and PortToUse tables. The GUID would need to be created before insertion into CallQue and PortToUse tables.
Disable	Do not use PortToUse table at all. If PortToUse table is not used then VoiceGuide will make outbound calls on any available ports. Also when this setting is used the GUID field is not included in SELECT queries to the CallQue table, which means that the CallQue table does not even need to have the GUID column.
<i>not set</i>	If this field is empty or not included at all then VoiceGuide will default to the GUID setting

The <OutDialQue_SqlPrefix> and <OutDialQue_SqlSuffix> values are used when VoiceGuide constructs the SQL query to find the suitable call in the PortToUse table.

The default value used if <OutDialQue_SqlPrefix> is blank is: `SELECT TOP 1`

The actual SQL scripts used by VoiceGuide to automatically create the database tables used by it can be found in VoiceGuide's \system\setup\ subdirectory. The default scripts used is named Db_Create.sql

ADO.NET Provider Specific versions of this script can be created. To make a Provider-specific script file the Provider name needs to be specified as part of the filename:

`Db_Create_ProviderName.sql`

eg:

`Db_Create_System.Data.SqlClient.sql`

SugarCRM

VoiceGuide IVR can integrate with most CRM systems.

Integration is straightforward for CRMS that offer a REST interface.

Below example shows how VoiceGuide can perform OAuth2 authentication with SugarCRM, and then call the various functions in the SugarCRM REST API.

Version number of SugarCRM REST API that was tested was: v10

OAuth2 Authorisation

Voiceguide script would use a Web Service module to issue a POST request to the `/oauth2/token` REST API function, and provide the required data in JSON format.

The https address to POST to would be something like this:

```
https://YOURDOMAIN/rest/v10/oauth2/token
```

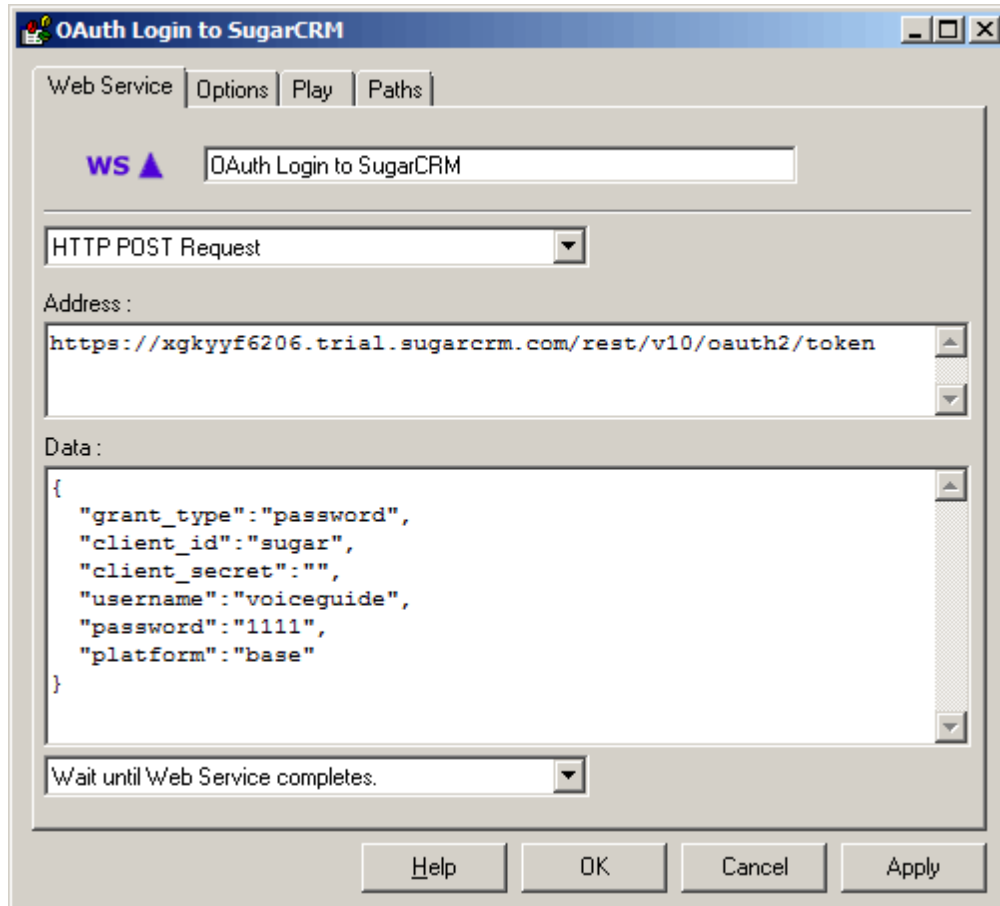
With the `YOURDOMAIN` part replaced with the domain name/IP of your SugarCRM installation.

The Data to be POSTed would be something like this:

```
{
  "grant_type": "password",
  "client_id": "sugar",
  "client_secret": "",
  "username": "MYUSERNAME",
  "password": "MYPASSWORD",
  "platform": "base"
}
```

With `MYUSERNAME` and `MYPASSWORD` replaced with a valid user/pass defined on your SugarCRM system.

Here is a screenshot of the Web Service module configuration screen:



When this module is ran SugarCRM will return data that will be saved into Result Variables. The OAuth2 access token will be saved in `$RV[access_token]`

This `$RV[access_token]` can then be used in subsequent modules that call other SugarCRM REST API functions to add/modify the various SugarCRM items (Accounts, Contacts, Leads, Tasks, Notes, Calls, Bugs, Cases etc.)

To add a new CALL object into SugarCRM a POST would be made to his https address:

`https://YOURDOMAIN/rest/v10/calls`

With the following data:

```
{
  "name": "New VoiceGuide IVR call",
  "description": "VoiceGuide IVR system received a new call",
  "assigned_user_name": "Assigned User Name",
  "contact_name": "Some Contact Name",
  "contact_phone": "5551112222",
  "contact_email": "support@voiceguide.com"
}
```

And the OAuth2 token would be set in the Headers options like this:

`oauth-token: $RV[access_token]`

Here are the screenshots showing module configuration:

Add New CALL

Web Service | Options | Play | Paths

WS ▲ Add New CALL

HTTP POST Request ▼

Address :
`https://xgkyyf6206.trial.sugarcrm.com/rest/v10/Calls`

Data :

```
{  
  "name": "New VoiceGuide IVR call",  
  "description": "VoiceGuide IVR system received a new call",  
  "assigned_user_name": "Assigned User Name",  
  "contact_name": "Some Contact Name",  
  "contact_phone": "5551112222",  
  "contact_email": "support@voiceguide.com"  
}
```

Wait until Web Service completes. ▼

Help OK Cancel Apply

Add New CALL

Web Service **Options** Play Paths

WS ▲

No Authentication ▼

Username Password

Headers

oauth-token: \$RV[access_token]

Options / Parameters

Help OK Cancel Apply

Similar approach is used when creating new Accounts, Contacts, Leads, Tasks, Notes, Bugs, Cases etc.

Sample scripts showing creation/addition of the various SugarCRM objects can be found in VoiceGuide's \Scripts\CRM\ directory.

Introduction

VoiceGuide v7 supports any MRCPv2 compliant speech recognition engine.

Speech Recognition is supported when any current Dialogic card is used - except the low end D/4PCIUF. Speech Recognition is also supported on HMP3.0 VoIP systems.

Please contact sales@voiceguide.com to discuss your Speech Recognition requirements.

Step 1 : Install the speech recognition engine recommended by VoiceGuide.

It is recommended that the Speech Recognition engine be installed on a separate server. Speech Recognition engines have high resource and CPU requirements and the high CPU load can degrade overall IVR performance if the Speech Recognition engine is installed on the same system.

- [Installing LumenVox](#)

Step 2 : Configure Dialogic cards

If using Analog Dialogic cards then a special "CSP Enabled" Firmware file will need to be selected. Go the Dialogic Configuration Manager, bring up the properties page for the analog card and on the Misc tab select the following Firmware file:

D/41JCT : d41jcsp.fwl

D/120JCT : d120csp.fwl

Similarly, when using the Dialogic JCT series T1 and E1 cards the "CSP Enabled" Firmware needs to be selected in the Dialogic Configuration Manager. Further changes in dxxx resource specification in Config.xml file also need to be made when using T1 and E1 JCT cards. Please consult support@voiceguide.com when deploying Speech Recognition on T1 or E1 JCT family cards.

No special settings are needed for the DMV cards.

Step 3 : Configure VoiceGuide

An mrcp.xml configuration file needs to be placed in VoiceGuide's \conf\ subdirectory.

A sample mrcp config file is placed in the \conf\ subdirectory on installation. It is called "sample_mrcp.xml" and needs to be renamed to mrcp.xml and its entries updated with the correct IP addresses. See comments in the file.

Step 4 : Test with Provided Sample Scripts

Some sample scripts have been provided as starting point. They can be downloaded from the Example Scripts section of the website.

Step 5 : Create Your Own Grammars

Speech recognition is right now enabled within VoiceGuide's Play modules only. To have VoiceGuide recognize speech during a play, a grammar associated with this play module must be defined. A grammar is defined by creating a text file which contains the grammar for the particular play module, and placing this file in the same directory where the script is located. The filename can be:

```
srgs_ModuleTitle.gram
```

```
srgs_ModuleTitle_PortNumber.gram
```

```
srgs_ModuleTitle.txt
```

```
srgs_ModuleTitle_PortNumber.txt
```

The *ModuleTitle* identifies for which play module the grammar file is for and the *PortNumber* can also be used to specify that this grammar should be used by this particular port only. The grammar files' contents are read in when the Play module starts. This approach allows user to dynamically update grammars as required. The order in which VoiceGuide searches for the grammar files is as listed above.

Step 5 : Modify Your Own Scripts to Handle Speech Recognition Responses

When VoiceGuide receives the response from the Speech Recognition Engine it will create the following Result Variables

<code>\$RV[ModuleTitle_ASR_Instance]</code>	contains the speech recognition engine's response. This is also called "Semantic Interpretation"
<code>\$RV[ModuleTitle_ASR_Input]</code>	Contains the text of the caller said.
<code>\$RV[ModuleTitle_ASR_Confidence]</code>	Contains the confidence level. Indicates how confident the speech recognition engine is that the recognition is correct range 0-100

VoiceGuide will first see if a path matching the value stored in `$RV[ModuleTitle_ASR_Instance]` is found. If the matching path is found then that path is taken.

Next VoiceGuide will first see if a path matching the value stored in `$RV[ModuleTitle_ASR_Input]` is found. If the matching path is found then that path is taken.

Otherwise the Success path is taken if some response was returned, and a Fail path is taken if no response was returned or the response was <nomatch/>

Install LumenVox

1. Purchase a LumenVox license through sales@voiceguide.com
2. You will receive an email with login and password which can be used to download LumenVox components.
3. Start the License manager.
4. Press "Connect" button to connect the license manager to the license server on your machine.
5. Press the "Create Server ID file" button and save the Info.bts file to somewhere you remember.
6. Press the "Connect to Website button".
7. In the launched LumenVox website go to the Account Information section.
8. Click on the License Upload link - on the right hand side of the Deployments section.
9. Fill out the Computer text box and browse to find the created Info.bts file.
10. Press upload.
11. Click on the License Download link - on the right hand side of the Deployments section where the License Upload link used to be.
12. Scroll to bottom and press I agree.
13. Soon you will be prompted to download and save the save the license file
14. Save the license file to somewhere you remember.
15. Go to LumenVox License Administrator and click on the "Install License" button
16. Browse to the license file and select it.
17. Your LumenVox license should now be installed. Click on the "View Current licenses" button to confirm that the license is installed.
18. If you have any problems with the above the copy a save the messages displaying in the Log Messages window and forward it to LumenVox.

The LumenVox MRCPv2 server by default uses port 5060 to accept incoming speech recognition requests. This is the same port used by most VoIP/SIP based applications. When running the LumenVox MRCPv2 server you will need to ensure that no other VoIP/SIP applications are running on the same server.

Alternatively the port can be changed to one of users choice. The change would need to be made in LumenVox's mrpc.config file as well as in the VoiceGuide config.xml file.

Additional Languages

If languages other than American English and American English Digits will be used (eg: Australian or Canadian etc) then the other used languages need to be copied from the `C:\Program Files\LumenVox\Engine\Lang\OtherLanguages` directory to the `C:\Program Files\LumenVox\Engine\Lang` directory.

Any of the languages located in the `Lang` directory can then be specified in the SRGS grammars.

Additional Configuration

LumenVox MRCPv2 Server does not stream audio into its speech recognition engine live. Instead, it buffers it, and sends it all in at once, after it has detected end of speech. The following two settings are important in regulating the end of speech detection:

1. **speech_complete_timeout**

speech_complete_timeout determines the length of pause within the speech that will then be regarded by LumenVox to indicate end of speech. **speech_complete_timeout** is best thought of as the maximum length of pauses allowed between words.

Eg: Imagine if a user is reading strings of digits. e.g. a telephone number: "Five-one-two [pause for a half second] three-one-four-four" is a normal sort of response. **speech_complete_timeout** controls how long we will consider pauses versus end of speech. If we set **speech_complete_timeout** to 700 (which represents 700 milliseconds = 0.7 second) then the engine will accept the half a second pause in the spoken numbers and recognize the numbers after the pause. But if we set **speech_complete_timeout** to 300 (which represents 300 milliseconds = 0.3 second) then the engine will assume that the half second pause between numbers can be regarded as end of speech, and only the first three digits will be recognized.

Note that setting **speech_complete_timeout** to a large value will result in the delay of speech recognition result returned to VoiceGuide, as LumenVox will wait for that amount of time after speech actually ends before realizing that speech has actually ended and that the recognition of what was spoken can begin.

2. **end_of_speech_timeout**

end_of_speech_timeout is the total amount of time the user can speak for. If end of speech is not detected by this time, LumenVox will time out.

The LumenVox MRCPv2 configuration file can be found at C:\Program Files\LumenVox\MRCPv2Server\config\mrCP.config

Here is an extract of the relevant sections of the .config file:

```
-----  
Vendor Specific parameters  
-----  
  
choose_model =1  
enable_lattice_scoring =1  
initial_audio_time =100  
wind_back_time =1000  
bargain_timeout =15000  
end_of_speech_timeout =10000  
snr_sensitivity_lvl =50
```

MRCPv2 Standard parameters

```
sensitivity_lvl = 0.5
nbest_length =1
confidence_thrsltd =0.45
no_input_timeout =20000
dtmf_termination_timeout =5000
recognizer_start_timers =true
recognition_timeout =20000
speech_complete_timeout =800
dtmf_inter_digit_timeout =5000
dtmf_buffer_time =5000
dtmf_term_char =#
save_waveform =false
waveform_url_location =
```

The MRCPv2 service needs to be restarted after the `mrcp.config` file is modified.

More information about each of the parameters can be found in LumenVox documentation and [MRCPv2 RFC](#)

Script Logs

Script Logs

VoiceGuide can store a log of what the caller did during the course of the call. The script activity logs contain information like:

- Date & time the call was made and when it completed.
- Caller ID details (name and number)
- All the numbers entered in the Play, Get Number Sequence and Say Number modules
- Filenames of all the recorded messages
- How many results were returned by the Database Query module
- Evaluate Expression module results
- Results of Run Program or Run Script modules
- etc.

The logs are saved in 4 different formats:

.vgl	VoiceGuide Log Format - saved as a string of Result Variables
.csv	Comma delimited format
.json	JSON document body format
.xml	XML document body format

Script Logs can be turned off by specifying `Scripts=OFF` in VG.IN in section [Log]

Subscripts

Each script which is used during the call will have a log entry created in the script's corresponding log files. If the script is visited multiple times during a call then a new log entry will be made each time the script is used. This means that if a subscript is called multiple times during one call, it's call log will have multiple entries for the same call - with each entry having different 'start_time' and 'end_time' timestamps.

Create your own Call Log

Run Program or Run VBScript modules can be used to create custom Log files. Please refer to Help file's sections on those modules to see how these modules can be used to append information to a text file.

CDR Logs

[Please see here for more information on CDR logs.](#)

vgEngine, ktTel, ktTTs Logs

VoiceGuide system internal application logs are saved in VoiceGuide's \log\ subdirectory.

System logs can be turned off by specifying:

VoiceGuide=0

ktTel=0

ktTts=0

NumberLoader=0

in VG.IN in section [Log]

Call Detail Records (CDRs)

VoiceGuide v7 will create a CDR for all the calls which arrived or were made by the system.

The CDR logs are automatically saved in VoiceGuide's \cdr\ subdirectory, and are also saved in a database. There is also an option to save the CDR data to a Syslog.

The call information is stored in the following CDR format:

- 1.**account:** An account code designation (string)
- 2.**src:** Caller*ID number (string)
- 3.**dst:** Destination extension (string)
- 4.**dcontext:** Destination context/submenu (string)
- 5.**clid:** Caller*ID with text (string)
- 6.**channel:** Channel used (string)
- 7.**dstchannel:** Destination channel if appropriate (string)
- 8.**lastapp:** Last application if appropriate (string)
- 9.**lastdata:** Last application data (arguments) (string)
- 10.**start:** Start of call (date/time)
- 11.**answer:** Answer of call (date/time)
- 12.**hangup:** End of call (date/time)
- 13.**duration:** Total time in system, in seconds (integer)
- 14.**billsec:** Total time call is up, in seconds (integer)
- 15.**disposition:** What happened to the call: ANSWERED, NO ANSWER, BUSY (string)
- 16.**amaflags:** Billing flags (string)
- 17.**user:** User data

The Result Variables defined during the call can be optionally saved in the **lastdata** field. To enable saving of \$RVS in CDR logs add this to the [Log] section of the VG.INI file:

```
CDR_LastData_SaveRv=1
```

Contents of \$RV[CDR_amaflags] will be placed in the 16th CDR field.

Contents of \$RV[CDR_user] will be placed in the 17th CDR field.

Temp Files

VoiceGuide v7 will save temporary files in \temp\ subdirectory.

Rec_Am_X.wav files are made during End Of Answering Machine Message detection.

X is the "LineID" of the line making the call.

There should be no more of these "Rec_Am" files then the number of lines on the system

TTS_X_Y.wav files are created during Text To Speech and contain the actual TTS generated sound that was played.

X is the "LineID" of the line and Y is the TTS generation counter during the call.

There should be no more of these "TTS" files then the number of lines on the system * max number of times the TTS is used during the call.

vbs_X_Y.wav files store the VBScript that was ran.

X is the "LineID" of the line and Y is the VBScript module counter during the call.

There should be no more of these "vbs" files then the number of lines on the system * max number of times the VBScript module is used during the call.

These files will only be created if SaveToFile option has been enabled in section [moduleRunScript] in VG.INI

js_X_Y.wav files store the Java Script that was ran.

X is the "LineID" of the line and Y is the VBScript module counter during the call.

There should be no more of these "js" files then the number of lines on the system * max number of times the JavaScript module is used during the call.

These files will only be created if SaveToFile option has been enabled in section [moduleRunScript] in VG.INI

The older versions of the files are overwritten by newer versions, so there is an upper limit of how many files can be in TEMP directory, and all the TEMP files store data related to last call on system made on that line only.

Admin_TraceLogAdd

Adds a log entry to VoiceGuide's event trace log.

Syntax

object.Admin_TraceLogAdd iLineId, iLogLevel, sLogInfo

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId</i>	Required. Identification number of the line with which this log entry is associated. Set to 0 if not used.
<i>iLogLevel</i>	Required. Compared against VoiceGuide's trace level setting to determine if the log entry should be printed. Set to 0 for the trace log entry to be always printed.
<i>sLogInfo</i>	Required. the information to be printed

Notes

VoiceGuide's trace level is set using VG.INI file, section "TraceLog", entry "TraceLevel"

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Admin_TraceLogAdd 0, 0, "This trace was printed from an external VB Script"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Dialer_MakeCall

Attempts to make a call on one of the available lines. If no lines are available to make the call immediately then an error will be returned.

This function does not save any call details in the OutDialQue database.

Syntax

object.Dialer_MakeCall sPhoneNumber, sLineSelection, sBridgeAfterDialing

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>sPhoneNumber</i>	Required. The telephone number to be called
<i>sLineSelection</i>	Required. If the outbound call may only be made on particular phone lines then these lines should be listed in this setting. If left blank then any available lines will be used. Line IDs may be specified as a comma delimited list of "LineIDs" (eg: 6,7,8) or using XML notation, (eg: <LineId>2</LineId><LineId>3</LineId>) or specified as a comma delimited list of the Dialogic line identifiers (eg:dxxxB1C2,dxxxB1C3).
<i>sBridgeAfterDialing</i>	Required. Line ID, or name of Dialogic channel to bridge the call with after the number has been dialed.
<i>sScriptToRunAfterDialing</i>	Required. Which script is to be started on the line after dialing the number. Leave blank to have the script currently assigned to the line to be started. If none is specified then no script will be used.

Returns

Returns the name of the LineID of line on which the call was made see the VoiceGuide'

Remarks

Please see the VoiceGuide's help file's section on the Auto Dialing for more information on the many ways in which outgoing calls can be scheduled in VoiceGuide.

Example 1

Scenario : Dial number 5551234 and after last digit is dialed bridge with call on current

```
line. set vg = CreateObject("vgServices.CommandLink")
vg.Dialer_MakeCall "0,5551234", "", "$RV_LINEID", ""
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Dialer_OutDialQueueAdd

Adds a new entry to Dialer's OutDialQueue database. The call will be made as outgoing lines become available, and the time before the call is made may depend on how many other entries are currently loaded in the OutDialQueue database. No return information about when the call will be actually made is provided when this function returns.

If no other entries are in the OutDialQueue database and there are outgoing lines available then the call will be made immediately.

Please note that the parameters of this function have changed from VoiceGuide v6 to VoiceGuide v7.

Syntax v7

```
object.Dialer_OutDialQueueAdd sPhoneNumber, sPhoneNumberPrefix, sActivateTime,  
sDayTimeStart, sDayTimeStop, sDaysCallAllowed, sLineSelection, sCampaignName, iPriority,  
sOnAnswerLive, sOnAnswerMachine, sFaxToSend, sOnNotAnswered, sOnRetriesExhausted,  
iAnswerTimeout, iRetriesLeft, iRetriesDelay, sRV, sCallOptions, sEscalationCalls
```

Syntax v6

```
object.Dialer_OutDialQueueAdd sPhoneNumber, iActivateTime, iDayTimeStart, iDayTimeStop,  
sDaysCallAllowed, sLineSelection, iPriority, sAnnounceMessage, sOnAnswerLive,  
sAnswerMachineMsg, sRV, iAnswerTimeout, iCallRetriesLeft, iDelayBetweenRetries,  
sOnNotConnected, sOptionsXml, sEscalationCalls
```

Part	Description
<i>object</i>	VoiceGuide object
<i>sPhoneNumber</i>	The telephone number to be called
<i>sPhoneNumberPrefix</i>	Optional prefix dialed before the telephone number.
<i>ActivateTime</i>	<p>Date and Time when the call should be made.</p> <p>In v7 the time can be specified as a string in any locally valid date/time format. This format is recognized worldwide: YYYY-MM-DD HH:NN:SS AM/PM the AM/PM sign can be omitted if using 24-hour time notation. The YMMDDHHNN format can also be used.</p> <p>In v6 the time can be specified as a number in format YMMDDHHNN.</p>
<i>DayTimeStart</i>	<p>Time of the day before which the call to this number should not be made.</p> <p>In v7 the time can be specified as a string in any locally valid date/time format. This format is recognized worldwide: HH:NN:SS AM/PM the AM/PM sign can be omitted if using 24-hour time notation. The HHNN format can also be used.</p> <p>In v6 the time can be specified as a number in format HHNN. eg: 8am would be specified as the number 800. To allow this telephone number to be dialed from midnight onwards set to 0</p>
<i>DayTimeStop</i>	Time of the day after which the call to this number should not be made.

	<p>In v7 the time can be specified as a string in any locally valid date/time format. This format is recognized worldwide: HH:NN:SS AM/PM the AM/PM sign can be omitted if using 24-hour time notation. The HHNN format can also be used.</p> <p>In v6 the time can be specified as a number in format HHNN. eg: 9pm would be specified as the number 2100. To allow this telephone number to be dialed at any time of day set to 0</p>
<i>sDaysCallAllowed</i>	Days of the week when the call can be made. Specify each day using the first two characters of that day eg: to call on weekdays only specify "MoTuWeThFr". If not used (ie: can call on all days) set to empty string ""
<i>sLineSelection</i>	If the outbound call may only be made on particular phone lines then these lines should be listed in this setting. Line IDs may be specified as a comma delimited list of "LineIDs" (eg: 6,7,8) or using XML notation, (eg: <LineId>2</LineId><LineId>3</LineId>) or specified as a comma delimited list of the Dialogic line identifiers (eg: dxxxBlC2,dxxxBlC3). If left blank then any available lines will be used.
<i>sCampaignName</i>	Campaign name associated with this call. Used for users own categorizing of calls. Can be left as an empty string.
<i>iPriority</i>	At what priority level the calls should be made. A lower number indicates a higher priority. ie: 1 is the highest priority, then 2, then 3 ... etc. Number of priority levels is unlimited.
<i>sOnAnswerLive</i>	VoiceGuide script to run when the call is answered by a real person.
<i>sOnAnswerMachine</i>	Sound file (or VoiceGuide script) to be played if the call is answered by an answering machine. The sound file will be played after the answering machine's welcome message finishes. If not used set to empty string ""
<i>sFaxToSend</i>	.PDF/.TIF/.TIFF file to send, or VoiceGuide script to be started immediately after dialing. If this setting specified then the <i>sOnAnswerLive</i> and <i>sOnAnswerMachine</i> are effectively ignored, as when sending fax the fax calling tone has to be sent immediately after dialing. If not used set to empty string ""
<i>sOnNotAnswered</i>	.EXE/.COM/.BAT or .VBS (VBScript) to run when the call has not been answered. If not used set to empty string ""
<i>sOnRetriesExhausted</i>	.EXE/.COM/.BAT or .VBS (VBScript) to run when the call has not been answered and there are no more retries left. If not used set to empty string ""
<i>iAnswerTimeout</i>	How many seconds will the Dialer wait for the call to be answered. Default value of 60 will be used if this entry is set to 0
<i>iCallRetriesLeft</i>	How many more times will the Dialer attempt to call this number if the next call is unsuccessful. To only make one call attempt set to 0
<i>iRetriesDelay</i>	How many minutes will the Dialer wait before attempting to call again. If not used set to 0
<i>sRV</i>	Result Variables which will be available to the VoiceGuide script used on this call. If not used set to empty string ""
<i>sCallOptions</i>	XML formatted options string. If not used set to empty string ""
<i>sEscalationCalls</i>	Can be used to specify multiple escalations levels. XML format is used to specify the escalation calls. Please see the VoiceGuide's help file's section on the 'Out Dial' file for details on the XML format used to specify call details. If not used then set to empty string "".

Notes

Version 7 Examples:

Example 1

Straightforward example.

```
set vg = CreateObject("vgServices.CommandLink")
vg.Dialer_OutDialQueueAdd "ext1002@10.1.1.4", "", "2009-11-28 9:00:00 AM", "9:00 AM",
"5:00 PM", "", "", "BookingConfirmCampaign", 1, "c:\ConfirmBooking.vgs", "c:
\CallBackDetails.wav", "", "c:\ifNotAnswered.vbs", "c:\ifRetriesExhausted.vbs", 90, 2,
30, "[CustomerID]{44563}", "", ""
set vg = Nothing
```

Version 6 Examples:

Example 1

```
set vg = CreateObject("VoiceGuide.CommandLink")
vg.Dialer_OutDialQueueAdd "0,5551234", 0, 0, 0, "", "", 1, "c:\sendinfo\announce.wav",
"c:\sendinfo\InfoMenu.vgs", "c:\sendinfo\CallBackDetails.wav", "", 90, 2, 5, "", "",
""
set vg = Nothing
```

Example 2

Note: CStr() function needs to be used around a variable which needs to be passed in as a string and which is composed of numbers. If CStr() is not used VBScript will try to pass that variable as a number instead of as a string, which will generate an error.

Result Variables which will be available to scripts ran when the call is made are specified as well.

```
set vg = CreateObject("VoiceGuide.CommandLink")
sTelNumber = "5551234"
vg.Dialer_OutDialQueueAdd CStr(sTelNumber), 0, 0, 0, "", "", 1, "c:
\sendinfo\announce.wav", "c:\sendinfo\InfoMenu.vgs", "c:
\sendinfo\CallBackDetails.wav", "[CustomerID]{44563}", 90, 2, 5, "", "", ""
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Bridge_Connect

Connects two calls together, allowing callers on the two lines to speak to each other.

Syntax

```
sResult = object.Bridge_Connect(iLineId1, iLineId2)
```

or

```
sResult = object.Bridge_Connect(sChName1, sChName2)
```

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId1</i>	Required. Identification number of the first line
<i>iLineId2</i>	Required. Identification number of the second line
<i>sChName1</i>	Required. The name of the first Dialogic channel.
<i>sChName2</i>	Required. The name of the second Dialogic channel.
<i>sResult</i>	Optional. Empty string if function completed OK, otherwise it will contain the error description.

Notes

This command needs to be issued only once to connect two calls together. It does not matter which of the lines issues the call - both lines will be able to hear and speak to each other.

Examples

Example 1: Using 'Line ID' identifiers

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Connect $RV_LINEID, iSecondLineId
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Example 2: Using Dialogic channel descriptors

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Connect "dxxxB1T3", "dxxxB4T1"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Example 3: Using Dialogic channel descriptors

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Connect("dtiB1T21", "dtiB2T13")
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Bridge_Disconnect

Disconnects the two lines. Callers on the two lines will no longer be able to speak to each other.

Syntax

```
sResult = object.Bridge_Disconnect(iLineId1, iLineId2)
```

or

```
sResult = object.Bridge_Disconnect(sChName1, sChName2)
```

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId1</i>	Required. Identification number of the first line
<i>iLineId2</i>	Required. Identification number of the second line
<i>sChName1</i>	Required. The name of the first Dialogic channel.
<i>sChName2</i>	Required. The name of the second Dialogic channel.
<i>sResult</i>	Optional. Empty string if function completed OK, otherwise it will contain the error description.

Notes

This command needs to be issued only once to disconnect the two calls. It does not matter which of the lines issues the call - both lines will not be able to hear and speak to each other any more.

Examples

Example 1: Using line ID numbers

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Disconnect $RV_LINEID, iSecondLineId
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Example 2: Using Dialogic channel descriptors (Analog cards)

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Disconnect "dxxxB1T3", "dxxxB4T1"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Example 3: Using Dialogic channel descriptors (Digital cards)

```
set vg = CreateObject("vgServices.CommandLink")
vg.Bridge_Disconnect("dtiB1T21", "dtiB2T13")
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Line_Hangup

End the call on the specified line.

Syntax

object.Line_Hangup *iLineId*

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId</i>	Required. Identification number of the line. If set to 0 then the first device controlled by VoiceGuide will have it's call terminated. If set to -1 al devices controlled by VoiceGuide will have the calls terminated.

Notes

There must be an incoming call on the line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Line_Hangup $RV_LINEID
set vg = Nothing
```

Line_Pickup

Answer an incoming call and run VoiceGuide script.

Syntax

object.Line_Pickup iLineId, sVgScriptToRun

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId</i>	Required. Identification number of the line. If set to 0 then the first device controlled by VoiceGuide will answer the call.
<i>sVgScriptToRun</i>	Required. VG Script to run after answering call

Notes

There must be an incoming call on the line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Line_Pickup $RV_LINEID, "C:\Scripts\CallRec\RecCall11.vgs"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Play_Start

Plays a sound file on the specified line.

Syntax

```
object.Play_Start(iLineId, sSoundFiles)
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sSoundFiles</i>	The sound file to be played. Multiple sound files can be specified, separated by commas.

Notes

There must be an active call on the line. If any other sound files are being played or recorded on the line at the time they are stopped and the new file is played.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Play_Start $RV_LINEID, "c:\confirm.wav"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Play_Stop

Stop playing a sound file on the specified line.

Syntax

```
object.Play_Stop iLineId
```

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId</i>	Required. Identification number of the line

Notes

There must be an active call on the line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Play_Stop $RV_LINEID
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```


Record_Stop

Stop recording the sound file on the specified line.

Syntax

```
object.Record_Stop iLineId
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line

Notes

There must be an active call on the line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Record_Stop $RV_LINEID
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Record_Start

Start Recording the sound on the specified line.

Syntax

```
object.Record_Start(iLineId, sSoundFile, iPlayBeep, sOptionsXml)
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sSoundFile</i>	The filename of the sound file to be recorded
<i>iPlayBeep</i>	1 if a 'beep' is to be played before recording, 0 is no 'beep' is to be played
<i>sOptionsXml</i>	Record options

Notes

There must be an active call on the line. If any other sound files are being played or recorded on the line at the time they are stopped and the new file is recorded.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Record_Start $RV_LINEID, "c:\newfile.wav", 1, ""
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Record_2Lines_Start

Start Recording the sound from two lines into the specified line. Usually used to record the two sides on conversation on a conferenced call.

Available in VoiceGuide v7.x only.

To use this function in VoiceGuide v6 please contact sales@voiceguide.com

Syntax

object.Record_2Lines_Start(*iLineIdRec*, *iLineId1*, *iLineId2*, *sSoundFile*, *sOptionsXml*)

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineIdRec</i>	LineID of the line handling the recording process.
<i>iLineId1</i>	Identification number of the first line to be recorded
<i>iLineId2</i>	Identification number of the second line to be recorded
<i>sSoundFile</i>	The filename of the sound file to be recorded
<i>sOptionsXml</i>	Record options

Notes

There must be an active call on both lines. If any other sound files are being played or recorded on either line at the time they are stopped and the new file is recorded.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Record_2Lines_Start $RV_LINEID, $RV_LINEID, $RV[Transfer_2ndLeg], "c:\newfile.wav", ""
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Run_ResultReturn

Returns information from the called Program or script back to VoiceGuide - indicating what path VoiceGuide should take next from the "Run Program" or "Run VBScript" modules.

Syntax

object.Run_ResultReturn iLineId, sResult

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sResult</i>	Result returned to VoiceGuide. Can be "success", "fail", "true", "false" or a string of Result Variables. If set to empty string then the Fail path will be taken

Notes

Used to return results if VoiceGuide is currently awaiting results from a called Program or Script.

If a list of \$RVs is returned then a Success path will be taken.

When used in the Run VBScript modules: the jump to next module is taken immediately when Run_ResultReturn is called, so Run_ResultReturn usually should be the last function in the script (before 'set vg = Nothing')

Example 1

```
set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Example 2

```
set vg = CreateObject("vgServices.CommandLink")
vg.Run_ResultReturn $RV_LINEID, "[MarketDow]{9,321}[MarketNasdaq]{1,702}[MarketSP500]{990}"
set vg = Nothing
```

RvGet

Returns value of the specified Result Variable.

Syntax

```
sValue = object.RvGet(iLineId, sRv)
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sRv</i>	Result Variable who's value is to be returned. The leading dollar sign can be omitted. eg. in case of \$RV[GetCardDetails] just the RV[GetCardDetails] can be specified, and in case of a system RV like \$RV_CALLLENGTH just the RV_CALLLENGTH can be specified. Omitting the dollar sign is necessary if RvGet() is used from within VoiceGuide, as any fully specified RVs will be replaced with their value before the script is ran.
<i>sValue</i>	Value of the specified Result Variable. If this Result Variable is not currently defined on the line then an empty string is returned.

Notes

Result Variables are reset at the beginning of a new call. Hence they are available for querying after the call has finished, but before a new call has begun on the same line.

RvGet() should only really be used from VoiceGuide's VB Script if that VB script polls for a value of a particular Result Variable in a loop. If the value of the Result Variable is not expected to change during the running of the script then the actual Result Variable should be used directly in the script, without the need of a RvGet function.

Example 1

Running from an external application.

```
set vg = CreateObject("vgServices.CommandLink")
'see how long the call on line who's LineID is 6 has been going on for:
sReturnValue = vg.RvGet(6, "$RV_CALLLENGTH")
set vg = Nothing
MsgBox sReturnValue
```

Example 2

Running from an external application.

```
set vg = CreateObject("vgServices.CommandLink")
'get the RV [GetCardDetails] defined on LineID 6:
sReturnValue = vg.RvGet(6, "$RV[GetCardDetails]")
set vg = Nothing
MsgBox sReturnValue
```

Example 3

Running from VoiceGuide's VB Script module. Note that the \$ has been omitted in the \$RV, as if it is included then the \$RV will be replaced with value of \$RV[GetCardDetails] on the current line, before the VBScript is ran.

Example below will read \$RV_CALLLENGTH from a call on another line:

```
set vg = CreateObject("vgServices.CommandLink")
sReturnValue = vg.RvGet($RV[SecondLineId], "RV_CALLLENGTH")
vg.Run_ResultReturn $RV_LINEID, "[OtherLineCallLength]{" & sReturnValue & "}"
set vg = Nothing
```

Example 4

Running from VoiceGuide's VB Script module. Note that the \$ has been omitted in the \$RV, as if it is included then the \$RV will be replaced with value of \$RV[GetCardDetails] on the current line, before the VBScript is ran.

Example below will read \$RV[GetCardDetails] from a call on another line:

```
set vg = CreateObject("vgServices.CommandLink")
sReturnValue = vg.RvGet($RV[SecondLineId] "RV[GetCardDetails]")
vg.Run_ResultReturn $RV_LINEID, "[OtherLineCardDetails]{" & sReturnValue & "}"
set vg = Nothing
```

RvGet_All

Returns all the Result Variables current for the specified line.

Syntax

```
sResult = object.RvGet_All(iLineId)
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sResult</i>	Result Variables currently defined for the selected line.

Notes

Result Variables are reset at the beginning of a new call. Hence they are available for querying after the call has finished, but before a new call has begun on the same line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
sReturnedData = vg.RvGet_All($RV_LINEID)
vg.Run_ResultReturn $RV_LINEID, sReturnedData
set vg = Nothing
```

RvGet_AllXml

Returns all the Result Variables current for the specified line in XML-ish format

Syntax

```
sResult = object.RvGet_AllXml(iLineId)
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sResult</i>	Result Variables currently defined for the selected line. Returned on XML-ish format

Notes

The returned data defines each variable in XML format, but no XML headers or footers are supplied.

Result Variables are reset at the beginning of a new call. Hence they are available for querying after the call has finished, but before a new call has begun on the same line.

Example

```
set vg = CreateObject("vgServices.CommandLink")
'see what Result Variables are currently defined on current line
sReturnValue = vg.RvGet_AllXml($RV_LINEID)
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
MsgBox sReturnValue
```


RvSet

Sets the value of a Result Variable. If the Result Variable does not exist then it is created.

Syntax

object.**RvSet** *iLineId*, *sRv*, *vValue*

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sRv</i>	Name of the Result Variable
<i>vValue</i>	New value of the Result Variable

Notes

Result Variables are reset at the beginning of a new call.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.RvSet $RV_LINEID, "ThisCallersRating", "medium"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

RvSet_RvList

Sets a number of Result Variables. If any of the the Result Variables do not exit then they are created.

Syntax

```
object.RvSet_RvList iLineId, sRvList
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sRvList</i>	List of Result Variables and their values. Each RV-Value pair is in format [<i>Name</i>]{ <i>Value</i> } a number of RV-Value pairs can be specified one after another.

Notes

Result Variables are reset at the beginning of a new call.

Example

```
set vg = CreateObject("vgServices.CommandLink")
'sets two Result variables - which will be able to be accessed
'later as $RV[CallersRating] and $RV[CallersPreference]
vg.RvSet_RvList $RV_LINEID, "[CallersRating]{medium}[CallersPreference]{blue}"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Script_Gosub

Call a VoiceGuide subscript, specifying the script filename and start module. Specify which script and module should be ran once the subscript returns.

Syntax

```
object.Script_Gosub iLineId, sDestScript, sDestModule, sRvList, sReturnScript,  
sReturnModule
```

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sDestScript</i>	The filename of the VoiceGuide script where the subscript is located. If the subscript is in the same VoiceGuide script as the calling module then this entry can be left blank.
<i>sDestModule</i>	Name of the starting module.
<i>sRvList</i>	List of Result Variables and their values. Each RV-Value pair is in format [<i>Name</i>]{ <i>Value</i> } a number of RV-Value pairs can be specified one after another.
<i>sReturnScript</i>	The filename of the VoiceGuide script where the system should return to after finishing the subscript. If the subscript is in the same VoiceGuide script as the calling module then this entry can be left blank.
<i>sReturnModule</i>	Name of the return module.

Notes

If the module names are blank then the default starting modules in the script are used.

When used in the Run VBScript modules: the jump to next module is made immediately when Script_Gosub is called, so Script_Gosub usually should be the last function in the script (before 'set vg = Nothing')

Example 1

```
set vg = CreateObject("vgServices.CommandLink")  
'Gosub to a new script and return to module VmMmMenuPlay in the current script  
vg.Script_Gosub $RV_LINEID, "vmAdmin.vgs", "", "", "", ""  
set vg = Nothing
```

Example 2

```
set vg = CreateObject("vgServices.CommandLink")  
'Gosub to a new script and return to module VmMmMenuPlay in the current script  
vg.Script_Gosub $RV_LINEID, "vmAdmin.vgs", "VmAdminWelcMsgMenuPlay", "", "",  
"VmMmMenuPlay"  
set vg = Nothing
```

Script_Goto

Jump to new VoiceGuide script and/or module.

Syntax

object.Script_Goto iLineId, sDestScript, sDestModule, sRvList

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sDestScript</i>	The filename of the VoiceGuide script where the module subscript is located. If the subscript is in the same VoiceGuide script as the calling module then this entry can be left blank.
<i>sDestModule</i>	Name of the starting module. If left blank then the script's default starting module is used.
<i>sRvList</i>	List of Result Variables and their values. Each RV-Value pair is in format [<i>Name</i>]{ <i>Value</i> } a number of RV-Value pairs can be specified one after another.

Notes

If the module names are blank then the default starting modules in the script are used.

When used in the Run VBScript modules: the jump to next module is made immediately when Script_Goto is called, so Script_Goto usually should be the last function in the script (before 'set vg = Nothing')

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Script_Goto $RV_LINEID, "GetPayment.vgs", "", ""
set vg = Nothing
```

Script_Return

Returns control of the call to the calling script.

Syntax

object.Script_Return iLineId, sRvList

Part	Description
<i>object</i>	VoiceGuide object
<i>iLineId</i>	Identification number of the line
<i>sRvList</i>	List of Result Variables and their values. Each RV-Value pair is in format [<i>Name</i>]{ <i>Value</i> } a number of RV-Value pairs can be specified one after another.

Notes

Effectively pops the return destination script/module of the call stack and start execution at that script/module.

When used in the Run VBScript modules: the jump to next module is made immediately when Script_Return is called, so Script_Return usually should be the last function in the script (before 'set vg = Nothing')

Example 1

```
set vg = CreateObject("vgServices.CommandLink")
vg.Script_Return $RV_LINEID, ""
set vg = Nothing
```

Example 2

```
set vg = CreateObject("vgServices.CommandLink")
vg.Script_Return $RV_LINEID, "[PersonalDataVerified]{True}"
set vg = Nothing
```

Serial_Tx

Transmits data over a serial port.

Syntax

object.Serial_Tx iSerialPortNumber, sTx

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iSerialPortNumber</i>	Required. Serial port number.
<i>sTx</i>	Required. String to be sent on the Serial port.

Notes

Default parameters for the serial connection can be set in VG.INI file, using section [Serial]:

```
[Serial]
CommPort=1
Handshaking=0
InBufferSize=1024
InputMode=1
NullDiscard=False
ParityReplace=?
RThreshold=1
RTSEnable=False
Settings=9600,N,8,1
```

Calls to `Serial_Tx` function will then send the specified data over the Serial port. The Serial port will be opened during the sending of data and closed immediately afterwards.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Serial_Tx 1, "This is a test sent from VoiceGuide."
set vg = Nothing
```

Vm_Event

Indicate an event has occurred affecting a particular voicemail box, or an action should be taken affecting a particular voicemail box.

Syntax

object.Vm_Event iLineId, sEvent, sVmbId, sParam1, sParam2, sParam3

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>iLineId</i>	Required. Identification number of the line
<i>sEvent</i>	Required. Event/Action type. Allowed values are: FORWARD, MSG_NEW, MSG_NONEWMSGGS, MSG_NOSAVEDMSGGS, LOGIN, LOGOUT
<i>sVmbId</i>	Required. The ID of the Voicemail box to which the Event/Action applies.
<i>sParam1</i>	Required. Event/Action specific parameter. Set to blank if not used.
<i>sParam2</i>	Required. Event/Action specific parameter. Set to blank if not used.
<i>sParam3</i>	Required. Event/Action specific parameter. Set to blank if not used.

For examples of use please see the VoiceGuide Scripted Voicemail System script files.

Vm_VmbConfig_Get

Retrieves a property value for a particular voicemail box.

Syntax

```
sCurrentValue = object.Vm_VmbConfig_Get(sVmbId, sTag)
```

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>sVmbId</i>	Required. The ID of the Voicemail box who's property is to be retrieved.
<i>sTag</i>	Required. The name of the property to be retrieved.
<i>sCurrentValue</i>	Required. The current value of the property.

Notes

Returns the information stored for the specified property. The actual information returned will be the contents in the VmBoxList.xml file between `<sTag>` and `</sTag>` for the selected Voicemail Box.

Example

```
set vg = CreateObject("vgServices.CommandLink")
sPin = vg.Vm_VmbConfig_Get("0001" "Pin")
vg.Run_ResultReturn $RV_LINEID, "[ThisUserPin]{" & sPin & "}"
set vg = Nothing
```


Vm_VmbConfig_Set

Set a property for a particular voicemail box.

Syntax

object.Vm_VmbConfig_Set sVmbId, sTag, sNewValue

Part	Description
<i>object</i>	Required. VoiceGuide object
<i>sVmbId</i>	Required. The ID of the Voicemail box who's property is to be set.
<i>sTag</i>	Required. The title of the property to be set.
<i>sNewValue</i>	Required. The new property value.

Notes

Sets the current value of the specified property. The new Value will be inserted in the VmBoxList.xml file between *<sTag>* and *</sTag>* for the selected Voicemail Box.

Example

```
set vg = CreateObject("vgServices.CommandLink")
vg.Vm_VmbConfig_Set "0001", "Pin", "1234"
vg.Run_ResultReturn $RV_LINEID, "Success"
set vg = Nothing
```

Inband Signaling

Many PBXs and Switches can send information about the call using Inband Signaling. This information may contain CallerID, number dialed, extension from which the call was transferred, reason for the transfer etc. Different PBXs and Switches support provision of different information via Inband Signaling.

VoiceGuide can be configured to detect Inband Signaling sent by the PBX at the beginning of the call and make this information available to scripts as Result Variables.

Inband Signaling is sent using a series of DTMF tones immediately after the call is answered. It is only sent to the recipient of the call and it does not matter after how many rings the call is answered - the signaling tones are sent immediately after call answer.

It is easy to check if Inband Signaling is sent on the line: place an extension headset against your ear and keep the 'handset on hook' switch depressed so that that handset can accept calls, then make a call to that extension and when the call arrives lift the hand off the 'handset on hook' switch - if there is any Inband Signaling sent you will hear it then a quick series of DTMF tones. (You will not be able to hear the tones usually if you just normally pick up the handset as the playing of the tones would have finished by the time you raise the handset to your ear)

To capture what tones are being sent on the line just set up a VoiceGuide script whose first module is a "Get Numbers" type module - that module will then capture the sent digits and what it captures will be stored in the Log File.

To enable inband signaling detection and interpretation in VoiceGuide an "Inband Signaling Definition File" needs to be created. This file defines what Inband Signaling VoiceGuide can expect to receive from the PBX and how it should be interpreted.

Creating an "Inband Signaling Definition File"

Creating an Inband Signaling definition files requires precise knowledge of what PBX will send in various circumstances.

Inband Signaling detection is only supported in the Enterprise (and Evaluation) version of VoiceGuide.

An example file is shown below. Comments within the file explain the purpose of the "Pattern" and "RV" entries.

```
Pattern=[*][*][*]????????[*][1-4]
RV=Inband_TOC,4,1,Inband_Calling,5,5,Inband_Called,10,5,Inband_Info,17,1
Pattern=[*][*][*]????[*][1-4]
RV=Inband_TOC,4,1,Inband_Calling,5,5,Inband_Called,10,0,Inband_Info,11,1
```

When using the file above the following Result Variables will be available to the script:

\$RV[Inband_TOC]	4the character sent
\$RV [Inband_Calling]	5 characters starting from position 5
\$RV[Inband_Called]	In case of forwarded call the 5 characters starting from position 10, or an empty string case of a direct call
\$RV[Inband_Info]	Last character of the data string.

The "Pattern" entries are defined according to the rules Visual Basic for Applications LIKE operator pattern definition, major part of which is quoted below:

Selecting which "Inband Signaling Definition File" to use

To indicate that a particular "Inband Signaling Definition File" is to be used by the system the full path to the file must be specified in the VG.INI file, section [PBX], entry InbandSignalConfig. Eg:

```
[PBX]
InbandSignalConfig=InbandSignaling_SiemensHiPath3000_v1.2.txt
```

Inband Signaling Call Answering and Timings

If Inband Signaling is defined, VoiceGuide will wait for up to 2 seconds for DTMF tones to arrive. Maximum time between successive DTMF tones is 0.5 seconds. Once DTMF signaling stops VoiceGuide will perform the pattern matching and start running the script

If Inband Signaling does not arrive within 2 seconds of answering the call VoiceGuide will start running the script and the Result variables which would have been defined as a result of any pattern matches will be left undefined.

The 2 sec timeouts were found to be suitable for most systems, but it can be changed using the settings in VG.INI file. To change that setting please see the [PBX] section, entry InbandSignalWait.

Please contact support@voiceguide.com if an interdigit timeout other than 0.5 seconds is needed on your system

Signal Patterns Definition

The pattern-matching features allow you to use wildcard characters, character lists, or character ranges, in any combination, to match strings. The following table shows the

characters allowed in *pattern* and what they match:

Characters in <i>pattern</i>	Matches in <i>string</i>
?	Any single character.
*	Zero or more characters.
#	Any single digit (0–9).
[<i>charlist</i>]	Any single character in <i>charlist</i> .
[!<i>charlist</i>]	Any single character not in <i>charlist</i> .

A group of one or more characters (*charlist*) enclosed in brackets (**[]**) can be used to match any single character in *string* and can include almost any character code, including digits.

Note To match the special characters left bracket (**[**), question mark (**?**), number sign (**#**), and asterisk (*****), enclose them in brackets. The right bracket (**]**) can't be used within a group to match itself, but it can be used outside a group as an individual character.

By using a hyphen (–) to separate the upper and lower bounds of the range, *charlist* can specify a range of characters. For example, **[A-Z]** results in a match if the corresponding character position in *string* contains any uppercase letters in the range A–Z. Multiple ranges are included within the brackets without delimiters.

Other important rules for pattern matching include the following:

- An exclamation point (**!**) at the beginning of *charlist* means that a match is made if any character except the characters in *charlist* is found in *string*. When used outside brackets, the exclamation point matches itself.
- A hyphen (–) can appear either at the beginning (after an exclamation point if one is used) or at the end of *charlist* to match itself. In any other location, the hyphen is used to identify a range of characters.
- When a range of characters is specified, they must appear in ascending sort order (from lowest to highest). **[A–Z]** is a valid pattern, but **[Z–A]** is not.
- The character sequence **[]** is considered a zero-length string ("").

Signal Patterns Examples

In the example file shown above the pattern:

```
[*][*][*]????????[*][1-4]
```

specifies a string which has a total length of 16 characters, beginning with "****" followed by 11 characters, followed by another "*" and then followed by a digit between 1 and 4.

The pattern of:

```
[*][*][*]?????[*][1-4]
```

specifies a string which has a total length of 11 characters, beginning with "****" followed by 6 characters, followed by another "*" and then followed by a digit between 1 and 4.

In the pattern definition we need to enclose "*" characters in square brackets, otherwise they would be interpreted as a special "Zero or more characters." Indicator. The section Signal Patterns Definitions above has more information on how the various patterns can be defined.

Copyright & Disclaimer

VoiceGuide is Copyrighted by Katalina Technologies Pty. Ltd.

VoiceGuide is a Registered Trademark owned by Katalina Technologies Pty. Ltd.

The software is provided AS IS without warranty of any kind, whether express or implied, including, without limitation, warranties of merchantability or fitness for a particular purpose. The use of this software is entirely at the users own risk, and under the users own responsibility. Katalina Technologies shall have no liability to customer or any third party for any claim, loss or damage of any kind whether direct or indirect, including but not limited to lost profits, punitive, incidental, consequential or special damages, arising out of or in connection with the use or performance of the software and accompanying documentation.

An unlicensed evaluationcopy of VoiceGuide may be used for evaluation purposes only. This covers every application component of VoiceGuide including but not limited to VoiceGuide Engine, VoiceGuide Script Designer and VoiceGuide Dialer.

No part of the VoiceGuide distribution package may be modified or distributed separately without express permission of Katalina Technologies Pty. Ltd.

Parties found breaching above conditions will be prosecuted to the maximum extent possible under the law.

Index

A

Admin_TraceLogAdd 241

C

Call Finish 91
Call Start 89
Copyright & Disclaimer 273

D

Database Query 119
Dialer_OutDialAdd 243

E

Evaluate Expression 166

G

Get Numbers 109
Graphical Script Design Environment 75

H

Hangup Call 185

I

Inband Signaling 269
Introduction 74, 192

L

Loading Numbers to Call 205

M

Module Types 77
Multilanguage Systems 93

O

ODBC Data Sources 36

P

Paths 78
Play 101
Play_Start 250
Play_Stop 251
Protected Scripts 95

R

Record 105
Record_Start 253
Record_Stop 252
Registering VoiceGuide 69
Result Variables 82
Run Program 126
Run VB Script 131
Run_ResultReturn 255
RvGet 256
RvGetAll 258
RvGetAllXml 259
RvSet 260
RvSet_RvList 261

S

Say Numbers 114
Script_Gosub 262
Script_Goto 263
Script_Return 264
Send Email 116
Send Pager Message 183
Send Phone Message 180
Sound files 96
System Requirements 3

T

Time Switch 156
Transfer Call 158

U

Unique System Identifier 71

V

Vm_ForwardMsg 266
Voicemail Menus 200
Voicemail System Manager 195