



Speech Developer's Tools Guide

IBM ViaVoice™ SDK for Windows®

Version 8.0

Printed in the USA

Note:

Before using this information and the product it supports, be sure to read the general information under Appendix A “Notices”.

Second Edition (June 2001)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM reseller or IBM marketing representative.

©Copyright International Business Machines Corporation 2000, 2001. All Rights Reserved.

Note to U.S. Government Users—Documentation related to restricted rights— Use, duplication or disclosure is subject to restrictions set forth in GS ADP Schedule Contract with IBM Corp.

Contents

About this Book		5
	Who Should Read This Document	5
	Related Publications	5
Chapter 1	SMAPI Grammar Development and Test Tools	7
	fsgenum	7
	fsgprint	9
	fsgtest	9
Chapter 2	SMAPI Grammar Test Tool	11
	Accessing the SMAPI Grammar Test Tool	11
	Using the Test Tool	13
	Selecting Word Lists, SMAPI Grammars, and Vocabularies to Test.	13
	Testing Word Lists, SMAPI Grammars, and Vocabularies	16
	Creating a Dynamic Vocabulary or External List	17
	Adding Words or Phrases to a Dynamic Vocabulary or External List.	18
	Displaying Informational and Error Messages	19
	Changing the Active User ID, Enroll ID, or Task	21
	Disconnecting from and Reconnecting to the Speech Engine.	21
	Requesting or Releasing Speech Focus	22
	Interpreting and Handling the Results	22
	Tips for Using the Tool.	23
Chapter 3	ViaVoice Runtime Interfaces	25
	ViaVoice Runtime Client Launcher APIs.	25
	VVRtkClients_Launch	26
	VVRTKCLIENT_LAUNCH_INFO Structure	27
	VVRtkClients_IsClientAvailable	30
	VVRtkClients_IsClientRunning	31

“LAUNCHER” Sample Application	32
ViaVoice Runtime Registry API's	33
VVRtkRegApi::GetFullPath	34
“REGISTRY” Sample Application	35
ViaVoice Audio Quality API's	36
QualityAudio::compute	37
“AUDIOQUAL” Sample Application	39
ViaVoice Runtime Deployment Information	40
ISV Runtime Licensing Guidelines	40
Installing ViaVoice Runtimes	41
Using ViaVoice Runtimes	46
Uninstalling ViaVoice Runtimes	47
Appendix A	
Notices	51
Trademarks	52

About This Document

This document provides detailed information on developing Windows® 95/98 or Windows NT™ speech-aware applications using the IBM ViaVoice™ Software Developer's Kit (SDK) for Windows and the IBM Speech Manager API (SMAPI) interface set. This document is prepared in Portable Document Format (PDF) to provide the advantages of text search and cross-reference hyperlinking and is viewable with the Adobe Acrobat Reader v.3.x. We recommend that you print all or part of this guide for quick reference.

Who Should Read This Document

Read this document if you are a programmer interested in writing Windows 95/98 or Windows NT 4.0 speech-aware applications that use ViaVoice for speech and are built using the IBM SMAPI interface set.

Related Publications

Refer to the following publications included with this version for additional programming, reference, and design information:

- *SMAPI Reference*

SMAPI Grammar Development and Test Tools

The ViaVoice SDK provides three command-line tools that can assist you as you design, test, and debug grammars. These tools can be used in conjunction with the SMAPI Grammar Compiler, Dictionary Builder of SMAPI, and the SMAPI Grammar Test Tool to verify the behavior of your grammars. These tools are:

- `fsgenum`
- `fsgprint`
- `fsgtest`

fsgenum

The `fsgenum` tool is used to generate the sentences accepted by a grammar. These generated sentences can be used as test scripts for the grammar or as a reference for what the grammar will accept. The command line parameters for `fsgenum` control how the sentences for the grammar will be generated. `fsgenum` can enumerate all the sentences accepted from a compiled grammar (FSG) to standard output. The command-line options specified below determine which sentences are generated. This tool can be used to produce test scripts for “live” grammar testing. The syntax of `fsgenum` is:

```
fsgenum [-r N | -R N | -f] [-a] [-p] [-l N] file.fsg
```

The parameters are:

-r N

Generates N random sets of sentences from the grammar. The same set of random sentences are always generated each time the command is executed with this option. The sentences are chosen in accordance with the probability that is assigned to them by the grammar.

-R N

Generates N a new random set of sentences from the grammar. A different set of random sentences is generated each time the command is executed with this option.

-f

Generates a complete set of sentences acceptable by the grammar. For some grammars, this set might be very large.

Note:

If none of the preceding options is specified, `fsgenum` will enumerate one sentence representing each possible state sequence in the FSG file.

- a**
Displays the annotations associated with each enumerated sentence in the format `word:annotation`.
- p**
Displays the probability associated with each enumerated sentence.
- l N**
Specifies how many times loops (repetition operators) are expanded in the enumeration. For example, with `-l 3` specified, the production `"very+"` generates “very,” “very very,” and “very very very” in the enumeration, while `"very*"` generates "", “very,” and “very very.” The default value for `N` is 2.

file.fsg

Specifies the FSG file containing the grammar to enumerate.

The following example enumerates 10 different, random sentences from the grammar `mygram.fsg` and shows the probability associated with each sentence:

```
fsgenum -R 10 -p mygram.fsg
```

fsgprint

The fsgprint tool prints a state-by-state description of a compiled grammar (FSG), with annotations and arc probabilities. This tool is used primarily for diagnostic purposes. The syntax of fsgprint is:

```
fsgprint [-v] file.fsg
```

The parameters are:

-v

Prints additional information about the contents of the FSG.

file.fsg

Specifies the FSG containing the grammar to print.

The following example prints the contents of the grammar mygram.fsg:

```
fsgprint mygram.fsg
```

fsgtest

The fsgtest tool accepts test sentences from standard input and, for each sentence, prints to standard output an indication of whether the sentence is accepted by the grammar. This tool is used to verify the expected behavior of a grammar. The syntax of fsgtest is:

```
fsgtest [-t] [-a] [-p] file.fsg
```

The parameters are:

-t

Prints to standard output translations of sentences read from standard input.

-a

Shows the annotations associated with each sentence in the format word:annotation.

-p

Shows the probability assigned by the grammar to each sentence.

A probability of "*****" indicates that the grammar does not accept the sentence. If -p is not specified, fsgtest prints each input sentence that is accepted by the grammar to standard output.

Any sentence that is rejected by the grammar is printed to standard output preceded by an asterisk "*".

file.fsg

Specifies the FSG containing the grammar to test.

The following example tests whether the sentence “what time is it” is accepted by the grammar mygram.fsg and shows the probability associated with that sentence:

```
fsgtest -p mygram.fsg
```

```
what time is it
```

You can use **Ctrl+C** or **Ctrl+Z** to end the session.

You might also want to create a file that contains all of the sentences you want to test. You can then redirect standard input to this file. In the following example, the file test.txt is used as input to fsgtest:

```
fsgtest <test.txt mygram.fsg
```

After you define your vocabulary, compile the grammar (if necessary), and create a dictionary of pronunciations for your vocabulary, you need to test it and the pronunciations to verify that all the words and phrases can be recognized by ViaVoice. The SMAPI Grammar Test Tool allows you to do this before you've written or modified your own application. If ViaVoice cannot recognize selected words or phrases, you might need to modify your vocabulary or the pronunciations of the words or phrases to resolve the problems.

The SMAPI Grammar Test Tool takes compiled grammar (FSG) files as input. It also supports command vocabularies; that is, you can specify word list (WDL) files as input, too. In addition, you can define dynamic command vocabularies and external lists from within the SMAPI Grammar Test Tool.

When testing your vocabulary, you should test both in-vocabulary and out-of-vocabulary words and phrases. If you have more than one vocabulary for your application, you should test each of the vocabularies separately, and then test them together.

Accessing the SMAPI Grammar Test Tool

From the ViaVoice SDK folder, under TOOLS, click SMAPI Grammar Test Tool. The SMAPI Grammar Test Tool main window appears.

You can also start the SMAPI Grammar Test Tool from the command line by typing:

```
gramtest
```

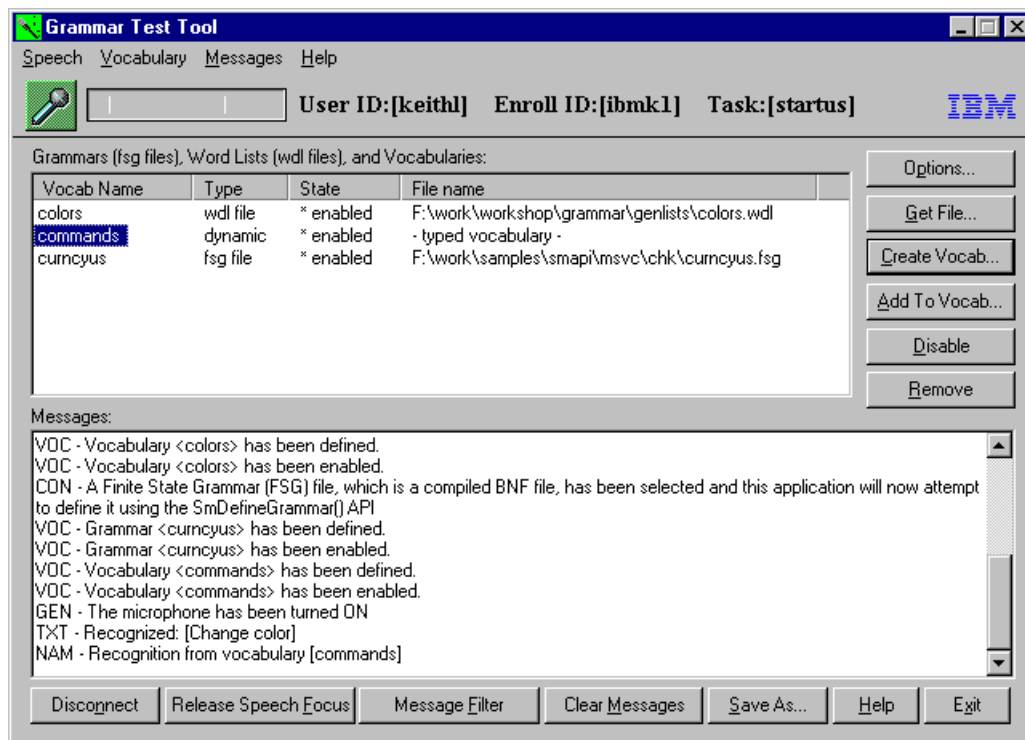


Figure 1. SMAPI Grammar Test Tool

This window shows the grammars you have selected to test in the Grammars (fsg files), Word lists (wdl files), and Vocabularies area. In the Messages area, you see informational and error messages, including messages that tell you whether the engine recognized what you said.

Using the Test Tool

With the Tool, you can:

- Select word lists, SMAPI grammars, and vocabularies to test
- Test word lists, SMAPI grammars, and vocabularies
- Filter, display, and save informational and error messages
- Create a dynamic vocabulary or external list
- Add words and phrases to a dynamic vocabulary or external list
- Change the active user ID, enroll ID, or task
- Disconnect from and reconnect to the speech recognition engine
- Request or release speech focus

Selecting Word Lists, SMAPI Grammars, and Vocabularies to Test

You use the Tool to test SMAPI grammars, word lists, dynamic vocabularies, and external lists. You can test multiple vocabularies of different types at the same time. The list of vocabularies available to test are shown in the Grammars (fsg files), Word lists (wdl files), and Vocabularies area.

To select a grammar (FSG) or word list (WDL) to test:

1. Click the Get File button.
2. Choose the file(s) you want to test. Click Open.
3. You will see the name of the file(s) you selected in the list of vocabularies to be tested.

To include a dynamic vocabulary or external list in the list of vocabularies to be tested, you must first create the vocabulary. Once it is created, it will be available for testing. Note that when you exit the SMAPI Grammar Test Tool, a dynamic vocabulary is lost, unless you saved the contents to a word list (WDL) file.

For each vocabulary in the list, you will see its type (fsg, wdl, or dynamic), whether it is enabled or disabled, and the full path name to the file. (For dynamic vocabularies, there is no path name, so you see an informational message that it is a typed vocabulary.)

When a vocabulary is enabled, it is used by the speech recognition engine when recognizing words and phrases. You can temporarily disable a vocabulary (that is, remove it from the set of vocabularies that the engine actually uses to recognize words but keep it in the list of vocabularies available to test) by first selecting the vocabulary and clicking Disable. You can later re-enable the vocabulary by selecting the vocabulary name and clicking Enable. (Note: The Disable and Enable buttons work as a toggle. The Disable button is available only when an enabled vocabulary is selected, and the Enable button is available only when a disabled vocabulary is selected.)

To remove a vocabulary from the list of vocabularies available to test, select the vocabulary name and click Remove. The vocabulary is removed from the list. When you remove a dynamic vocabulary, the contents of the vocabulary are lost.

Notes:

- SMAPI Grammars and vocabularies are automatically enabled by the Tool when they are defined (that is, when you get the file or you create the vocabulary.) You can choose not to have the Tool automatically enable SMAPI grammars and vocabularies. To do this, select Vocabulary from the menu, and then deselect Automatic Enable on Get or Create. Any SMAPI grammars and vocabularies that you subsequently select will be defined, but not enabled, by the Tool.
- When a grammar is compiled, you can specify whether to allow (or not allow) silence and/or mumbles between the words of a phrase. When the Tool defines a grammar, it uses the options compiled into the grammars by default. However, you can override the silence and mumble options set at compile time with the Tool. To do this, select Vocabulary from the menu, and then select Silence and Mumble Options.

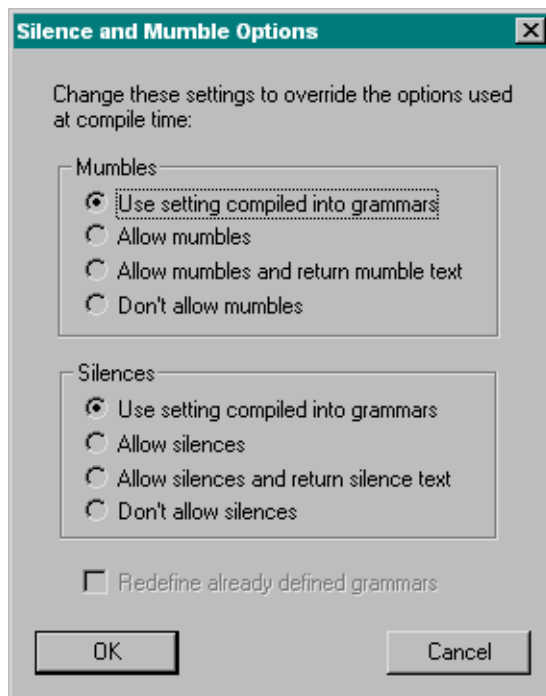


Figure 2. Silence and Mumble Options

Choose the options with which you would like to test your grammars. Click OK. Any grammars that you subsequently select will be defined with these new options. (You can redefine grammars that are already in the vocabulary list by selecting *Redefine already defined grammars*.) By varying the silence and mumble options, you can determine the optimum settings for your SMAPI grammars.

Testing Word Lists, SMAPI Grammars, and Vocabularies

Once you have selected the word lists, SMAPI grammars, and vocabularies you would like to test, you are ready to begin testing. To test speech vocabularies:

1. Turn the microphone on by clicking the microphone button.
2. Speak every word or phrase in the vocabulary, one at a time.

As you say a word or phrase, a message appears in the Messages area, telling you whether the word or phrase was recognized by the engine. For example, if the word you said is recognized, you will see a message like “TXT - Recognized [word].”

The speech recognition engine can return three types of recognition results: recognized, rejection, or substitution. For example, the word “orange” is in one of the vocabularies available to the speech recognition engine. If the engine recognizes the word “orange” when it is spoken, you will see the following message:

TXT - Recognized [orange]

Rejections occur when you say a word that is in one of the vocabularies, but is not recognized by the engine. A rejection can also occur if you say a word that is not in any of the enabled vocabularies. In these cases, you will see the following message:

TXT - Recognized ?? [different word]

The “??” indicate that the speech recognition engine rejected the utterance, with the word in brackets being the engine's closest guess.

Substitution occurs if the engine recognizes a word, but it is not the word you said (the speech recognition engine substitutes another word for the word you said). In this case, you should see a message like:

TXT - Recognized [different word]

For tips on interpreting and handling these types of messages, refer to [“Interpreting and Handling the Results” on page 22](#).

Creating a Dynamic Vocabulary or External List

With the Tool, you can define a vocabulary “on the fly” as you’re testing it. This is how you specify a dynamic command vocabulary or external list to be tested. It is also a useful way of testing alternative words and phrases for an existing vocabulary. To define a dynamic vocabulary, follow these steps:

1. Click the Create Dynamic button.

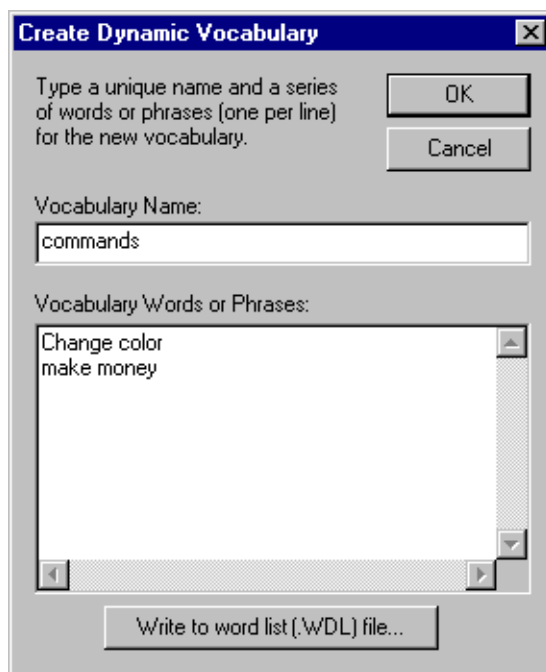


Figure 3. Create Dynamic Vocabulary

2. The Create Dynamic Vocabulary window appears. Type a name for your dynamic vocabulary in the Vocabulary Name field. This name must be unique.
3. Enter all of the words and phrases for this vocabulary in the Vocabulary Words or Phrase field. Use one word or phrase per line.
4. When you’re done, click OK. You will see the name of the vocabulary in the list of vocabularies to be tested. Its type will be dynamic.

Notes:

- You can optionally save a dynamic vocabulary as a word list (WDL) file. To do this, click the Write to word list (.WDL) file... button.
- The Tool interprets more than one word on the same line of a WDL file as a phrase to be recognized, rather than as individual words to be recognized. For example, the line:

open file

indicates that the phrase “open file” is to be recognized, rather than the individual words “open” and “file.” The Tool recognizes phrases even if pronunciations exist only for the individual words in a phrase. Developers are encouraged to create dictionary files with the Dictionary Builder for all of the words and phrases in their applications before testing with the Tool. This ensures the best recognition performance for all words and phrases.

- If you are defining an external list, you must name it exactly as it is specified in the grammar that refers to it.

Adding Words or Phrases to a Dynamic Vocabulary or External List

To add words or phrases to a dynamic vocabulary or external list:

1. Select the vocabulary to which you want to add a word or phrase. (It must have a type of dynamic.)
2. Click the Add to Vocab button.
3. Enter the words and phrases for this vocabulary in the Vocabulary Words or Phrases field. Use one word or phrase per line.
4. When you are done, click OK.

Displaying Informational and Error Messages

The Messages area of the Tool displays a variety of informational and error messages that can help you as you test your vocabularies. You can choose the amount and type of information to be displayed in the Messages area by clicking the Message Filter button.

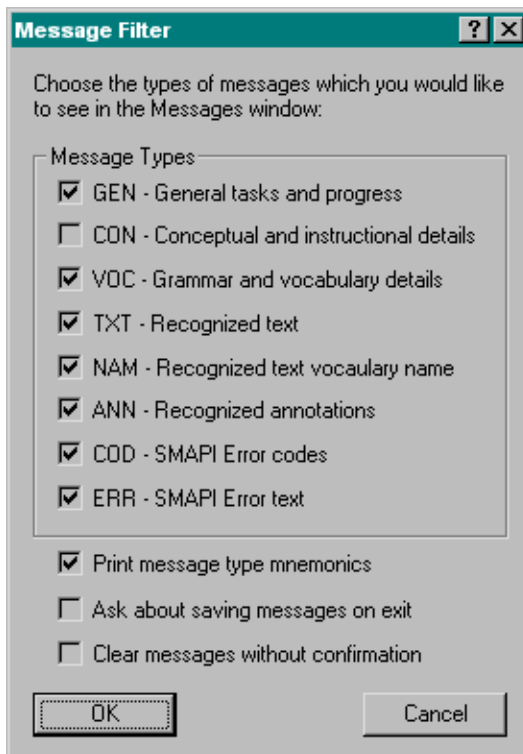


Figure 4. Message Filter

The types of messages that can be displayed are:

- General (such as connecting and disconnecting from the speech recognition engine, requesting and releasing speech focus, and errors in vocabulary definition)
- Conceptual and instructional (detailed information about what is happening, such as “Attempting to connect to ViaVoice” and “Checking connect status”)
- Vocabulary details (when vocabularies are defined, enabled, disabled, undefined, or removed)
- Recognized Text (the actual text that was recognized by the engine)
- Vocabulary name (the name of the vocabulary in which the recognized text was found)
- Annotations (any text or numeric annotations associated with the recognized text)
- SMAPI error codes
- SMAPI error text

By default, all message types except “Conceptual and Instructional” are selected. Experiment with displaying different combinations of messages to determine which are most useful to you.

Hint:

You might want to display “Conceptual and Instructional” messages if this is the first time you are using the Tool, or if you are new to developing SMAPI applications using ViaVoice.

Notes:

- Each message can be prefaced with its type (such as “TXT” or “ANN”). You can turn this feature on or off by selecting the Print message type mnemonics button. (This feature is on by default.)
- You can also save any messages that are in the Messages area to a file. When you exit the Tool, you will be asked if you want to do this. To turn this feature on or off, select the Ask about saving messages on exit button. (This feature is off by default.)
- You can clear the information in the Messages area by clicking Clear messages from the main Tool window. If there is a significant amount of text to clear, you will be asked to confirm that you want to clear the Messages area. To turn this feature off, select the Clear messages without confirmation button. (This feature is off by default.)
- You can edit the information in the Messages area. For example, you might want to annotate the information with notes to yourself to clarify the results of your testing. Any text that you add to the Messages area is saved along with the messages produced by the Tool, if you choose to save the messages information.

Changing the Active User ID, Enroll ID, or Task

The Tool displays the active user ID, enroll ID, and task. You might need to vary these settings to fully test your vocabularies. To change any of these settings, click the Options button. The ViaVoice Options in Control Panel is displayed.

It is important to test grammars under a variety of conditions, including variations in recognition sensitivity, recognition performance, users, and enrollments.

Note:

Changing certain settings (such as user ID) will cause the engine to send an SM_REQUEST_DETACH to all speech-aware applications. The Tool honors this request by disconnecting. You will need to reconnect to the speech recognition engine after changing the user ID.

Disconnecting from and Reconnecting to the Speech Engine

The Tool automatically connects to the speech recognition engine when it starts. You can explicitly disconnect from the speech recognition engine. This allows testing across repeated program executions (simulated by disconnecting and reconnecting) without interrupting a test session. It is also convenient for testing interactions (coexistence) with other speech-aware applications. To disconnect from the speech recognition engine:

1. Click the Disconnect button.
2. If any vocabularies appear in the vocabulary list, they will be undefined and removed from the list. Click OK to continue or Cancel to stop the disconnect.
3. If you click OK, the Tool is disconnected from the speech recognition engine, the vocabularies are undefined and removed, and the microphone, Get File, Create Dynamic, Add to Vocab, Enable/Disable, Remove, and Get Speech Focus buttons are disabled.

To reconnect to the speech recognition engine:

1. Click Connect.
2. The microphone, Get File, Create Dynamic, Add to Vocab, Enable/Disable, Remove, and Get Speech Focus buttons are now available.

3. You will need to select the vocabularies you want to test again.

The Disconnect and Connect buttons work as a toggle. If the Tool is connected to the speech recognition engine, the Disconnect button is available. If the Tool is not connected to the speech recognition engine, the Connect button is available.

Requesting or Releasing Speech Focus

With the Tool, you can explicitly request or release speech focus. This is useful for testing interactions (coexistence) with other speech-aware applications. To release speech focus, click the Release Speech Focus button. To request speech focus, click the Request Speech Focus button.

The Get Speech Focus and Release Speech Focus buttons work as a toggle. If the Tool has speech focus, the Release Speech Focus button is available. If another speech-aware application has speech focus (or the Tool has released speech focus), the Get Speech Focus button is available.

Interpreting and Handling the Results

The following examples illustrate how you might interpret and handle some of the more common results that you might encounter when testing your vocabulary. This is not a complete list, nor is there a prescribed approach to testing and verifying your vocabulary and its associated pronunciations. However, this section is meant to provide some insight into the testing process, and it should guide you in determining when you need to change your vocabulary and when you need to change or add a pronunciation to fix a problem.

- You experience frequent misrecognitions of a short phrase. For example, you defined the word “blue” in your vocabulary to select the color blue. When you test it using the Grammar Test utility, it is consistently misrecognized as “new.” You probably need to provide a longer alternative; for example, you might want to define “choose blue” or “new color blue.”
- You defined the phrases “Add gray pen” and “Add green pen” in your vocabulary. You find that “Add gray pen” is consistently misrecognized as “Add green pen” (“green” is substituted for “gray”). Look at the pronunciations for “gray” and “green.” You might need to provide an alternative pronunciation for “gray” using the Dictionary Builder.

- You have defined two functions, “Add <color>” and “Get <color>,” which do two very different things (they are not synonyms). However, when you say “Add blue,” it is frequently recognized as “Get blue.” You might want to reword one of the phrases; for example, change “Get blue” to “Mix blue.”

Tips for Using the Tool

This section describes some hints and tips for using the Tool:

- If your application supports multiple vocabularies, you need to test each one separately first, and then test them together.
- Test your vocabulary with a reasonable population of users. This should help you identify if there are words that are consistently misrecognized over a number of users; if there are, you might need to change the word or phrase or add multiple pronunciations. You should test your vocabulary against a variety of factors, including people with accents, males and females, people who speak the language natively and those who speak it as their secondary language, and users who have enrolled and those who have not enrolled.
- Test “out-of-vocabulary” words and phrases as well as those that are in your vocabulary. For example, you might want to test things that users might say when they are interrupted and forget to turn the microphone off (such as “Good morning” and “Hello”). The goal of testing out-of-vocabulary words and phrases is to ensure that ViaVoice is not inadvertently recognizing these out-of-vocabulary words and causing functions to be performed that the user did not specify.
- Remember that testing your vocabulary is an iterative process. As you make changes to your vocabulary or the pronunciations, you should go back and retest the entire vocabulary to verify that all of the words and phrases in the vocabulary can still be recognized.
- Use word list files (WDL) for large dynamic vocabularies. This provides you with a mechanism for editing the vocabulary, as well as enabling easier input to the Dictionary Builder and the Tool.
- You can edit the information in the Messages area. Whatever text you add or change in the Messages area is saved along with the messages produced by the Tool. This can be used, for example, to add notes to yourself to clarify the results of your testing.

ViaVoice Runtime Client Launcher APIs

The **VVRtkClient** library, which is part of SDK helps you launch different tools that are included with runtime. Each tool (application) expects different number of parameters to be passed in. For example, the User Options tool expects **StartPage** with the page number (defined in `vvrkclients.h` file) to display the specified page as the first panel, such as “StartPage=4”.

Note:

Some runtime clients, such as Dictation Macro Editor or Vocabulary Manager, require the user to have successfully completed at least one enrollment script before being able to use them. If you try launching one of these clients and User Wizard (or Enrollment Wizard) is launched instead, it means that the current default user is not properly configured its voice model yet. Please have the users complete the enrollment process before launching these tools.

The following are three APIs in Client Launcher library that are of importance.

```
VVRtkClients_Launch (VVRTKCLIENT_LAUNCH_INFO *pcliInfo,  
                    VVRTKCLIENT_PROCESS_INFO *pcpiProcInfo,  
                    int nReserved)
```

```
VVRtkClients_IsClientAvailable(VVRTKCLIENT_ID nClientID)
```

```
VVRtkClients_IsClientRunning(VVRTKCLIENT_ID nClientID)
```

VVRtkClients_Launch

The **VVRtkClients_Launch()** method is used to start (run) the specified ViaVoice runtime client (passed through the `m_ClientID` field in the **VVRTKCLIENT_LAUNCH_INFO** structure).

Syntax:

```
VVRTKCLIENT_RC VVRtkClients_Launch(VVRTKCLIENT_LAUNCH_INFO *pcliInfo,
                                     VVRTKCLIENT_PROCESS_INFO *pcpiProcInfo,
                                     int nReserved)
// Client launching information record (used to launch clients).
typedef struct
{
    VVRTKCLIENT_ID m_ClientID;
    char *m_pszOptions;
} VVRTKCLIENT_LAUNCH_INFO;
```

Parameters:

`pcliInfo`

`VVRTKCLIENT_LAUNCH_INFO`

Runtime client passed through the `m_ClientID` field in the **VVRTKCLIENT_LAUNCH_INFO** structure. Client specific parameters passed through `m_pszOptions`.

`pcpiProcInfo`

`VVRTKCLIENT_PROCESS_INFO`

This parameter (**VVRTKCLIENT_PROCESS_INFO** structure) is optional and you can pass NULL (or 0) if the return information is not needed. Otherwise, pass a pointer to an already allocated structure of type **VVRTKCLIENT_PROCESS_INFO** and its fields will contain the pertinent information upon return from call.

`Reserved`

`int`

This parameter is reserved for future use and should always be set to 0.

VVRTKCLIENT_LAUNCH_INFO Structure

Parameters:

m_ClientID
VVRTKCLIENT_ID

The m_ClientID member variable specifies the ViaVoice runtime client to launch. The values must be from VVRTKCLIENT_ID list.

m_pszOptions
char *

The m_pszOptions member variable takes a (null-terminated) string buffer, which contains different options depending on the runtime client being launched. Each parameter in the options line should be separated by a blank space. The user interface language parameter is required for all runtime clients (please see the table below for a list of supported languages) in order to display the correct resources depending on the configured ViaVoice language. The language parameter takes the following form:

Language=<lang> where <lang> can be any of the ViaVoice supported languages

TABLE 1. List of Supported ViaVoice Languages

Language	Language Key
U.S. English	En_US
U.K. English	En_UK
French	Fr_FR
German	Gr_GR
Italian	It_IT
Spanish	Es_ES
Brazilian Portuguese	Pt_BR
Arabic	Ar_AR
Japanese	Ja_JP
Chinese (Simplified)	Zh_CN
Chinese (Traditional)	Zh_TW

Remarks:

There are other client-specific parameters that must be specified when launching certain runtime clients. Below is a list of each supported ViaVoice runtime client and its set of custom parameters:

- **User Wizard**

`m_ClientID = VVRTKCLIENT_USER_WIZARD`

`m_pszOptions` = string buffer which contains any of the following parameters:

-User=<value> where <value> may be,
N - User Name filled & may be typed in
D - User Name empty & may be typed in
S - User Name filled & may be selected only(no typing) existing Users only.

-LaunchAudioSetup
-LaunchVocabExpander
-LaunchQuickTraining
-LaunchUnsupervisedEnrollment
-DisplayLastPanel

- **Audio Setup Wizard**

`m_ClientID = VVRTKCLIENT_AUDIO_SETUP_WIZARD`

`m_pszOptions` = string buffer which contains any of the following parameters:

-RestoreMicLevel

- **Enrollment Wizard (Analyze My Voice)**

`m_ClientID = VVRTKCLIENT_USER_WIZARD`

`m_pszOptions` = string buffer which contains any of the following parameters:

-User=S
-LaunchEnrollment

- **Vocabulary Expander Wizard (Analyze My Documents)**

`m_ClientID = VVRTKCLIENT_VOCABULARY_EXPANDER_WIZARD`

No optional parameters.

- **ViaVoice Options**

m_ClientID = VVRTKCLIENT_VIAVOICE_OPTIONS

m_pszOptions = string buffer which contains any of the following parameters:

-StartPage=<value> where <value> can be:

4 - User page

16 - Voice page

1024 - Formatting 1 page

2048 - Formatting 2 page

- **Dictation Macro Editor**

m_ClientID = VVRTKCLIENT_DICTATION_MACRO_EDITOR

No optional parameters.

- **Vocabulary Manager**

m_ClientID = VVRTKCLIENT_VOCABULARY_MANAGER

No optional parameters.

- **User Migration Tool**

m_ClientID = VVRTKCLIENT_USER_MIGRATION_TOOL

No optional parameters.

- **Vocabulary and Topic Installer**

m_ClientID = VVRTKCLIENT_VOCABULARY_AND_TOPIC_INSTALLER_TOOL

No optional parameters.

VVRtkClients_IsClientAvailable

The **VVRtkClients_IsClientAvailable()** method is used to detect whether the specified ViaVoice runtime client application or tool is currently installed. A return value of **VVRTKRC_OK** (or zero) means that the specified client is available, any other return value means that is not installed in the user's system.

Syntax:

```
VVRTKCLIENT_RC VVRtkClients_IsClientAvailable(VVRTKCLIENT_ID nClientID)
```

Parameter:

nClientID
VVRTKCLIENT_ID

This parameter can be any of the predefined client ID values included in the API header file.

VVRtkClients_IsClientRunning

The **VVRtkClients_IsClientRunning()** method is used to detect whether the specified ViaVoice runtime client application or tool is currently running or not. A return value of **VVRTKRC_OK** (or zero) means that the specified client is currently running, any other return value means not currently running.

Syntax:

```
VVRTKCLIENT_RC VVRtkClients_IsClientRunning(VVRTKCLIENT_ID nClientID)
```

Parameter:

nClientID
VVRTKCLIENT_ID

This parameter can be any of the predefined client ID values included in the API header file.

“LAUNCHER” Sample Application

LAUNCHER is a sample application included in the SDK, which demonstrates how to use **VVRtkClient** library API's to launch the different ViaVoice runtime tools. By passing the above listed parameters in the command line field, you can launch different runtime tools and different options (try them out). You can check which runtime clients are running by selecting the client from the drop-down list and clicking "Is Client Running" button.

ViaVoice Runtime Registry API's

VVRtkReg library helps you to find ViaVoice installed paths. This library contains one relevant API, which allows you to find the ViaVoice runtime install paths. For example, the directory where all executables and libraries are installed, the path for all the help files, and the ViaVoice install root.

The following is the API in Runtime Registry library.

```
VVRtkRegApi::GetFullPath(VVPath vvPath, LPTSTR lpszFullPath)
```

VVRtkRegApi::GetFullPath

VVRtkRegApi::GetFullPath() method returns the requested ViaVoice runtime full path. This method takes different constants as parameters, which are defined in `vvrkreg.h` file. It returns TRUE for success or FALSE for failure.

Syntax:

```
BOOL GetFullPath(VVPath vvPath, LPTSTR lpszFullPath)
```

Parameters:

`vvPath`

VVPath

This parameter specifies the requested ViaVoice runtime directory.

The following are the constant values that can be used as parameters.

RUNTIME_BASE

Returns the ViaVoice base directory, e.g. C:\Program Files\ViaVoice.

RUNTIME_BIN

Returns the path where all the runtime libraries and executables are installed.

RUNTIME_HELP

Returns the path where all runtime help files are installed.

RUNTIME_TEMP

Returns the path where ViaVoice engine creates temp files including log files.

`lpszFullPath`

LPTSTR

This parameter is used to return a fully qualified path for the requested ViaVoice runtime path. Caller must allocate enough space for storing the return path (usually `_MAX_PATH` size.)

“REGISTRY” Sample Application

There is a sample application included in the ViaVoice SDK called REGISTRY, which demonstrates the use of the API's described in the previous section. It is a simple MFC based application that allows you to query and see the results of each parameter constant described through a simple user interface.

ViaVoice Audio Quality API's

The SDK ships with a library called IBMSnQA, which measures the audio quality of PCM data. During recognition, if the audio quality gain is not set properly, there are chances that you will get a lot of misrecognition. You will use the audio quality library to determine the audio quality. Depending on the results, you will need to re-adjust the audio input gain from your application.

The following is the main API in this library.

```
void QualityAudio::compute(double frequency,  
                           long          samples,  
                           short*       pcm,  
                           int          reset,  
                           QualityStats* quality);
```

QualityAudio::compute

The **QualityAudio::compute** method computes the quality of the input PCM audio. It returns a reply structure, **QualityStats** with detailed information filled in. The **QualityStats** object has the results of the test. It has several access methods to find out, if more pcm data is needed to compute the quality. Once enough data is fed to compute API, you can access the results by calling `is_excellent()`, `is_very_good`, `is_good()` and other methods of **QualityStats** object.

Syntax:

```
void compute(double      frequency,
             long        samples,
             short*      pcm,
             int         reset,
             QualityStats* quality);
```

Parameters:

`frequency`
Double

Frequency of the PCM measured in Hertz.

`samples`
Long

Number of samples in the PCM vector.

`pcm`
Short*

Pointer to the vector of PCM.

`reset`
Int

0 - continue the computation, appending this PCM to the PCM from previous calls.

1 - compute using only the PCM given in this call.

quality
QualityStats*

Pointer to a **QualityStats** instance into which the results will be written.

Remark:

The calling program must provide the **QualityStats** instance. When calling with `reset==1`, it is not necessary to use the same **QualityStats** structure, but you may if you choose.

“AUDIOQUAL” Sample Application

The AUDIOQUAL sample demonstrates the use of audio quality library included in the SDK.

This MFC based sample comprises of **CRecordAudio** class, which abstracts the multimedia device APIs: **waveInOpen**, **waveInStart**, **waveInStop**, etc. Using these APIs the sample captures the raw PCM data by setting chunk size to 1sec from the microphone and saves it in a local buffer. On the callback message handler **WIM_DATA**, we post ourselves a message **WM_TEST_RECORD_RESULT** to compute asynchronously. On the handler of this message, we call **compute**, the audio quality library API, with the saved PCM data and passing the **QualityStats** object, which will be filled with the result. From the returned object, we determine if the library needs more data to compute. If no more data is required then we find out if the quality is excellent, very good, good, fair or poor from the **QualityStats** object.

The timer calls repetitively the **Record** method in case if the audio quality library needs more data to compute. When **QualityStatus** returns false on **need_more_data** method, we stop collecting data and kill the timer.

ViaVoice Runtime Deployment Information

ISV Runtime Licensing Guidelines

Following proper guidelines ensures that the ViaVoice Runtime is properly counted, maintained for other applications, and removed from the system when not used anymore. You, as an ISV, must comply with the following guidelines in your software to comply with the licensing associated with this ViaVoice Runtime:

- Your application must copy the Runtime in its entirety to your product CD.
- Your application's installation **must** install this ViaVoice Runtime when your application is installed. Similarly, your application must uninstall this ViaVoice Runtime when your application is uninstalled. More information about this is located below in the INSTALLING VIAVOICE RUNTIMES and UNINSTALLING VIAVOICE RUNTIMES.
- Your application must create a return area prior to the ViaVoice Runtime installation so the ViaVoice Runtime can return information. Your application installation program must remove this return area upon completing execution. More information about this is located below in the INSTALLING VIAVOICE RUNTIMES.
- Your application must invoke the ViaVoice Runtime installation with at least the minimum number of command line arguments as outlined below.
- Your application must register itself as a client of the ViaVoice Runtime. More information about this is located below in the INSTALLING VIAVOICE RUNTIMES.
- Your application's installation must inspect the return codes from the ViaVoice Runtime installation. These return codes must be handled appropriately. More information about this is located below in the INSTALLING VIAVOICE RUNTIMES.
- Your application must not add or remove files from the ViaVoice directories. More information is located below in USING RUNTIMES.
- Your application must resolve path information for ViaVoice binaries through VVRTKREG.DLL. More information about this is located below in USING RUNTIMES.
- Your application must launch ViaVoice executables through the VVRTKCLIENTS.DLL. More information about this is located below in USING RUNTIMES.
- Your application must create a return area prior to uninstalling the ViaVoice Runtime so the ViaVoice Runtime can return information. Your application uninstallation program must remove

this return information area upon completing execution. More information about this is located below in UNINSTALLING VIAVOICE RUNTIMES.

- Your application's uninstall must uninstall this ViaVoice Runtime through a helper DLL, VV70U_XX.DLL (where XX is a language code - US, UK, FR, GR, SP, etc.). More information about this is located below in UNINSTALLING VIAVOICE RUNTIMES.
- Your application's uninstall must unregister itself as a client of the ViaVoice Runtime. More information about this is located below in UNINSTALLING VIAVOICE RUNTIMES.
- Your application's uninstall must inspect the return codes from the ViaVoice Runtime uninstall. These return codes must be handled appropriately. More information about this is located below in UNINSTALLING VIAVOICE RUNTIMES.

This runtime includes features to prevent accidental removal of the ViaVoice Runtime while a ViaVoice Runtime-dependent application is still installed on the system. It requires a little more implementation effort for installation and uninstallation of this runtime, but the end result is a better user experience with your application.

To simplify your implementation, sample code has been provided. Keep in mind that this is sample code and it may have errors although IBM tries very hard to provide “bug-free” code. Examples outlining proper procedures are provided under the \SAMPLES directory on your SDK installation. Developer support files are provided under the \INCLUDE and \LIB directories on your installation.

Installing ViaVoice Runtimes

Your application install must install this version of the ViaVoice Runtime. Sample code executing the ViaVoice Runtime Installation can be found under \SAMPLES on your installation media.

The ViaVoice Runtime allows you to customize the directory into which it installs. The installation directory, however, must be limited to 60 characters due to backward-compatibility issues under Windows 95. Also, in order for the runtimes to co-exist with other IBM speech products, the custom directory will be ignored if ViaVoice is already installed on the machine.

Special considerations are necessary when installing the ViaVoice Runtime. More details on each step are provided in the bulleted section after the steps.

To install on a machine where no previous or current versions of your product exist, follow these steps:

1. Register your application as a client of the ViaVoice Runtime by setting registry keys.
2. Create the key '\HKEY_LOCAL_MACHINE\IBM\ViaVoice Installation' for return codes.
3. Launch the SETUP.EXE with the proper command line parameters for the ViaVoice Runtime.

To install on a machine where any version of your application is installed (either upgrading or over-installing), follow these steps:

1. Uninstall the ViaVoice Runtime. More information is located in UNINSTALLING VIAVOICE RUNTIMES.
 2. Register your application as a client of the ViaVoice Runtime by setting registry keys.
 3. Create the key '\HKEY_LOCAL_MACHINE\IBM\ViaVoice Installation' for return codes.
 4. Launch the SETUP.EXE with the proper command line parameters for the ViaVoice Runtime.
- To register your application as a client of the ViaVoice Runtime, set the following values under a new key called HKEY_LOCAL_MACHINE\Software\IBM_REGISTER with the following values:

Value	Description
szVisibleName	The name of your application (may be displayed)
szMajor	Major version number of your application
szMinor	Minor version number of your application
szUnique	Any unique identifier you wish to add

You must then delete this key after the ViaVoice Runtime installation returns to your application. Sample code to create this key, fill in the values, and delete it can be found under the directory \SAMPLES on your installation media.

- To launch the ViaVoice Runtime installation, you must invoke SETUP.EXE for the ViaVoice Runtime on the media you are distributing. SETUP.EXE takes several command line parameters. These are described below.

Simplified BNF:

```
<Parse> ::= <installation-media-dir>SETUP.EXE [<installpath><FORWARDSLASH> <space>  
<options>];  
<installation-media-dir> ::= <drive><colon><Valid-Path><BACKSLASH>;
```

<installpath> ::= <destination-directory>
<destination-directory> ::= <Valid-Path of less than 61 characters>;
<username> ::= <QUOTE> <first name> [[<space>] <surname>]<QUOTE> ;
<first name> ::= 1..40::<ALPHA-NUMERIC> || <space> || "(" || ")" || "_" || "." || "-";
<surname> ::= 1..40::<ALPHA-NUMERIC> || <space> || "(" || ")" || "_" || "." || "-";
<options> ::= <option> <space> <option>
<option> ::= "-SMS" || "/nq" || "/nn" || "/nu" <QUOTE> <username> <QUOTE> || /nSRMike8
|| /nSRMike11 || /nSRMike22 || /nr || /nl || /ns
<space> ::= ASCII 0x20 (decimal 32)
<QUOTE> ::= ASCII 0x22 (decimal 34)
<Valid-Path of less than 61 characters> ::= 1..60::<Valid-Path-Char>
<Valid-Path> ::= 1.._MAX_PATH::<Valid-Path-Char>
<Valid-Path-Char> ::= <RESTRICTED ASCII>

Option**Description**

-SMS	Ignore Network Timeouts (might be necessary with slower networks or CD-ROM drives).
/nq	Run in quiet mode showing only a thermometer.
/nn	Display the new orthography panel for Austrian German version.
/nu "firstname surname"	New user to be created.
/nSRMike8	Set the default sampling rates to 8 KHz.
/nSRMike11	Set the default sampling rates to 11 KHz.
/nSRMike22	Set the default sampling rates to 22 KHz. If none are selected, this is the default.
/nr	Disable Reboot screen.
/ns	Disable Sound Check.

Example:

SETUP.EXE C:\Program Files\MyInstViaVoice/ /nr /nq /ns /nu "John Smith" -SMS

Notes:

- If ViaVoice is already installed, the custom directory will be ignored.
- The minimum arguments to successfully install under another application are the destination directory, the user name (/nu "username") and the IBM_Register registry key (and its subkeys).

- After installing the ViaVoice Runtime, your application's installation must inspect the return codes from the ViaVoice Runtime installation and handle them appropriately. The return codes for the ViaVoice Runtimes install and uninstall are described below.

RETURN CODE VALUES FROM VIAVOICE RUNTIME INSTALL AND UNINSTALL

Errors encountered during both the installation and uninstallation are returned to your software via registry keys. The return information can be found under “HKEY_LOCAL_MACHINE\Software\IBM\ViaVoice Installation”. Under this key, the following values can be found:

"ErrorCode" - This is a string value and is “0” for success, otherwise an error was encountered. This number is a decimal representation of a 32-bit hexadecimal bit sensitive return code. A return code may consist of more than one bit being set.

In order to determine the nature of the error, the string value must be converted into an integer and the resulting number must be ANDED with mnemonic from the VVINSRC.H file. For example, suppose an application wants to install the ViaVoice Runtime to support a speech interface on an NT machine. If the user launching the installation does not have administrative privileges, the ViaVoice Runtime installation would return a value other than “0” through the “ErrorCode” registry entry. The application installation program regains execution control after the Runtime installation aborted. The application loads the Registry value from HKEY_LOCAL_MACHINE\Software\IBM\ViaVoice Installation and notices it is not “0”, but it is “67”. To report errors properly to the user the application converts the string into an integer, I, which equals 67. The application checks to see if (I & RT_CONFIG_APP_ACTIVE) is true. It is not (but if it is, this means a speech application was running so the installation is aborted). Next the application checks to see if (I & RT_CONFIG_BAD_OS) is true. It is not (but if it is, it means a minimum requirement for the operating system is not met and the installation is aborted). Next, the application checks to see if (I & RT_CONFIG_NOT_ADMIN) is true. It is TRUE, so the application reports the reason of the failure to the user.

ERROR CODES

Value	Mnemonic	Description
0x00000000	RT_CONFIG_SUCCESS	Operation was successfully completed.
0x00000001	RT_CONFIG_ERROR	General error encountered.
0x00000002	RT_CONFIG_INSTALL	Install set the error.
0x00000004	RT_CONFIG_UNINSTALL	Uninstall set the error.
0x00000010	RT_CONFIG_APP_ACTIVE	Speech application(s) was running.
0x00000020	RT_CONFIG_OS_BAD	Operating system check failed.
0x00000040	RT_CONFIG_NOT_ADMIN	User did not have administrative authority.
0x00000080	RT_CONFIG_DISK_SPACE	Disk space was not sufficient.
0x00000100	RT_CONFIG_NO_SOUND	Sound check was not successful.
0x00000200	RT_CONFIG_DIFF_LANG	Older version of a different language must be uninstalled before installing this version.
0x00000400	RT_CONFIG_REBOOT_REQ	Reboot is required before using this runtime.
0x00000800	RT_CONFIG_BADCMDLNE	Command line arguments were incorrect.
0x00001000	RT_PRODUCTS_EXIST	Installed applications depend upon the runtime.
0x00002000	RT_DPRODUCTS_EXIST	Discrete Products using runtimes exist on the machine (these products will not be installed over until they are removed).
0x40000000	RT_CONFIG_USER_QUIT	User cancelled the operation.

- "ErrorString" - A description of the error. Currently only install provides language-specific descriptions. The uninstall of ViaVoice runtimes provides English-only descriptions, so it cannot be passed along to the user.

After installation, remove the key 'HKEY_LOCAL_MACHINE\Software\IBM\ViaVoice Installation' and the IBM_Register registry key.

If the return code indicates that locked system files prevented installation of files, a reboot will be required. Your product's installation must present the user the option of rebooting after your application has completed installing if the return code indicates a reboot is necessary. If the return code from ViaVoice Runtime 8.0 installation indicates a reboot is required, do not attempt to use any of the ViaVoice files until after reboot.

After inspecting the return codes, the return code area must be removed from the registry. Remove the key '\HKEY_LOCAL_MACHINE\IBM\ViaVoice Installation.' Sample code to inspect the return value and remove this key can be found under \SAMPLES on your installation media.

Using ViaVoice Runtimes

Developer support files are provided in SDK. All required header files used for development are located there. A partial list of files included are listed below:

DEVELOPER FILES (US)

File	Description
VVRTKClients.lib	File resolving calls to VVRTKClients to functions in library.
VVRTKClients.h	File defining interface to VVRTKClients.
VVRTKReg.lib	File resolving calls to VVRTKReg to functions in library.
VVRTKReg.h	File defining interface to VVRTKReg.
VV70u_us.lib	File resolving calls to VV70u_us to functions in library.

Sample Files

File	Description
INSTALL.RUL	A code snippet written in InstallShield 5.5 to install Runtimes.
UNINSTALL.C	A code snippet written in C++ to uninstall Runtimes.
INSTALL.H	A header defining the error codes for install/uninstall of the Runtime.

Your application must not add or remove files from the ViaVoice directories. ViaVoice might remove files in these directories. Furthermore, your application should make no assumptions about the location of ViaVoice files on the hard disk. Launching ViaVoice executables should be done through the VVRTKCLIENTS.DLL, which will be placed into the system directory of the operating system. Getting Registry information about ViaVoice should be done through the VVRTKREG.DLL, which is also placed into the system directory of the operating system. The interface to these dynamic linked libraries is outlined in the UNINSTALLING VIAVOICE RUNTIMES.

If you want to use a USB microphone or a USB adaptor for a microphone, you must install the USB audio driver and select the USB audio device as your preferred recording device. (Your computer must have an USB port, and your Windows version must be Window 98 or higher.)

To set up the USB Audio Device:

1. Plug the USB connector into the USB port on your computer.
2. Follow the on-screen instructions to install the USB driver on your system.
3. Select the USB Audio Driver as your recording device using one of the following methods:
 - Click Start > Settings > Control Panel. Double-click **Multimedia**. Open the Audio tab on the Multimedia Properties window. Under Recording, open the Preferred Device menu, and **select USB Audio Device**. Click **OK** or **Apply**.
 - Alternatively, you can select the USB Audio Device during the Audio Setup wizard after installing the ViaVoice Runtime. In the Selecting a Sound Card window (second window of the wizard), open the Input menu, and select **USB Audio Device**, and then complete the rest of the wizard.

Uninstalling ViaVoice Runtimes

Your program's uninstall must call the uninstall for the ViaVoice Runtime.

Sample code for uninstalling the ViaVoice Runtime can be found under in SDK's samples installation. The sample code is also described below.

Note:

The ViaVoice Runtime uninstall allows users to completely remove ViaVoice from their system if they choose to do so. By default, all user data remains on the machine unless the proper command line switches are added to call to the Runtime uninstall program.

INTERFACES VV70U_US.DLL

Only the functions stated below are supported. Any others are IBM-proprietary and are subject to change at any time.

- VV70U_US.DLL is designed to help you safely uninstall the ViaVoice Runtime. The main functions you will be calling are:

IsRemovable (LPCSTR szProductType)

UninstallQuietly (LPCSTR szProductType, LPCSTR szOptions)

CreateAddRemoveEntry (LPCSTR szProductType)

UnregisterClient (LPCSTR szProductName, LPCSTR szMajor, LPCSTR szMinor, LPCSTR szUnique)

Of particular interest is the `szOptions` argument to `UninstallQuietly`. Currently three options are supported: “remain”, “remove”, and “deleteusers”. The argument “remain”, instructs the ViaVoice Runtime to stay on the machine even if its reference count reaches 0; “remove” instructs the ViaVoice Runtime to delete itself even if the reference count has not reached 0. “deleteusers” instructs the runtime to remove all user data from the machine. In order to pass these options to the uninstall for the ViaVoice Runtime, simply space-delimit them to the second argument of `UninstallQuietly`.

For example, if you were uninstalling an application that uses the ViaVoice Runtime, you would use the following pseudo-code in your uninstall:

```
Function RemoveRuntimes
```

```
Set bProceed variable to TRUE;
```

```
Set ID_PRODUCT_RTDCICT to 4.
```

```
Set ID_PRODUCT_RTCONTROL to 5.
```

```
If Uninstalling RT_DICTATION then set RuntimeType to ID_PRODUCT_RTDCICT.
```

```
  If Uninstalling RT_COMMAND then set RuntimeType to ID_PRODUCT_RTCONTROL.
```

```
  Load the VV70u_us.DLL using the VVRtkReg.DLL and the Win32 Call LoadLibrary.
```

```
  If Loading was not successful then Set bProceed to FALSE.
```

```
Endif.
```

```
If bProceed is TRUE then:
```

```
  Call the Win32 API call GetProcAddress to find the function UnregisterClient.
```

```
  Call the Win32 API call GetProcAddress to find the function IsRemovable.
```

```
  Call the Win32 API call GetProcAddress to find the function UninstallQuietly.
```

```
  Call the Win32 API call GetProcAddress to find the function CreateAddRemoveEntry.
```

```
  If GetProcAddress was not successful for all four functions then:
```

```
    Set bProceed to FALSE.
```

```
  Endif.
```

```
Endif.
```

```
If bProceed is TRUE then:
```

```
  Call UnregisterClient to unregister your application as a user of the runtime.
```

```
  Call Function IsRemovable returning IRResult.
```

```
  If IRResult <> 0 then:
```

```
    Query user if they wish to delete user data.
```

```
    If user said yes:
        Call Function UninstallQuietly with RuntimeType and "remove deleteusers".
    Else:
        Call Function UninstallQuietly with RuntimeType and "remove".
Else:
    Call Function UninstallQuietly with RuntimeType and "remain".
    Call Function CreateAddRemoveEntry with RuntimeType.
Endif_then_else.
Endif.

If bProceed is TRUE then:
    Call function CheckDictRTUninstStatus (see sample code).
Endif.

End Function
```

For more explicit (code-oriented) information, please see the samples located in the directory \SAMPLES on your installation media.

Uninstalling the ViaVoice Runtime requires using a helper DLL. For the US English version of ViaVoice, this dynamic-linked library is called VV70u_us.dll. Since you can make no assumptions about the location of the ViaVoice binary files, it is necessary to locate this dynamic-linked library using the function call GetFullPath() in VVRTKReg.dll, located in the system directory of your operating system. For more information about using this library, refer to USING VIAVOICE RUNTIMES and INTERFACES TO VVRTKREG, VVRTKCLIENTS, AND VV70U_US.DLL above.

The helper DLL facilitates the following steps in the uninstall procedure:

- Unregister your application as a ViaVoice Runtime client.
- Ensure that other speech applications will not be crippled by the uninstallation.
- Uninstall the ViaVoice Runtime quietly.
- Create an Add/Remove program entry for the ViaVoice Runtime (when needed).
- Clear the registry from residual entries.
- Allow the user to completely remove all data from the machine.

In addition to using these helper functions, your uninstall should inspect the return codes set during the uninstall of the ViaVoice Runtime. Finally, your installation should use a helper function to clean up the return area used for return codes.

Users who want to uninstall the IBM ViaVoice Runtime can do so by using the **Add/Remove Programs** option under **Start > Settings > Control Panel**. However, in some cases, such as when a user first uninstalls IBM ViaVoice for Windows 8.0, the ViaVoice Runtime might not appear in the Add/Remove Programs menu, but the files still appear in the ViaVoice directory. If this occurs, to uninstall the U.S. IBM ViaVoice Runtime files, the user should invoke the **rmdictUS.bat** file from the Bin directory. Users should also run the language-specific batch file for each installed language of the ViaVoice Runtime, such as **rmdictUK.bat** (U.K. English), **rmdictIT.bat** (Italian), **rmdictFR.bat** (French), **rmdictBR.bat** (Brazilian Portuguese), **rmdictGR.bat** (German), and **rmdictJP.bat** (Japanese).

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only that IBM product, program, or service may be used.

Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service.

The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Asia-Pacific users can inquire, in writing, to the IBM Director of Intellectual Property and Licensing, IBM World Trade Asia Corporation, 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106, Japan.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Department T01B, 3039 Cornwallis Road, Research Triangle Park, NC 27709-2195, USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM

ViaVoice

VoiceType

Adobe Acrobat is a trademark or registered trademark of Adobe Systems Incorporated.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.

Microsoft, Windows, Windows NT, Windows 95, Windows 98, and the Windows 2000 logo are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

Numerics

-1 N parameter, 8

A

-a parameter, 8, 9

acceptable sentences, generating, 7

accessing

Grammar Test Tool, 11

active

user ID, changing, 21

adding

phrases to dynamic vocabulary, 18

phrases to external list, 18

words to dynamic vocabulary, 18

words to external list, 18

annotations

associated with sentences, 8, 9

C

changing

active User ID, Enroll ID, or Task, 21

creating

dynamic vocabulary or external list, 17

D

diagnostic tools, fsgprint, 9

disconnecting

changing active settings, 21

speech engine, 21

displaying

enumerated sentence annotations, 8

error messages, 19

informational messages, 19

dynamic vocabulary

creating, 17

E

Enroll ID, changing, 21

enumerated sentences, displaying
annotations, 8

external list

adding words to, 18

creating, 17

F

-f parameter, 7

file.fsg parameter, 8, 9, 10

focus

requesting speech, 22

fsgenum

description of tool, 7

parameters, 8

-a, 8

-f, 7

-f N, 7

-p, 8

-R N, 7

syntax, 7

fsgprint

description of tool, 9

parameters

file.fsg, 9

-v, 9

syntax, 9

fsgtest

description of tool, 9

parameters

-a, 9

file.fsg, 10

-p, 9

-t, 9

syntax, 9

G

grammar

- generating sentences, 7
- printing, 9
- selecting for testing, 13
- testing, 16
- verifying behavior of, 9

Grammar Test Tool

- accessing, 11
- tips for, 23
- using, 13

I

informational messages, displaying, 19

interpreting and handling results, 22

M

message

- informational, displaying, 19

P

-p parameter, 8, 9

parameters

- l N, 8
- a, 8, 9
- f, 7
- file.fsg, 8, 10
- p, 8, 9
- R N, 7
- r N, 7
- t, 9
- v, 9

printing

- grammar, 9
- standard output translations, 9

probability parameter, 9

R

-R N parameter, 7

-r N parameter, 7

recognition

- results
- types of, 16

reconnecting speech engine, 21

releasing

- speech focus, 22

repetition operators, specifying, 8

requesting

- speech focus, 22

results

- interpreting and handling, 22
- types, 16

S

selecting

- grammars to test, 13
- vocabularies to test, 13
- word lists to test, 13

sentence

- annotations, 9
- generated by grammar, 7

set of acceptable sentences, generating, 7

speech engine

- disconnecting, 21
- reconnecting to, 21

speech focus

- requesting or releasing, 22

steps

- dynamic vocabulary, creating, 17
- external list, creating, 17
- grammar, selecting, 13
- reconnecting speech engine, 21
- word lists, selecting, 13

syntax

- fsgenum tool, 7
- fsgprint, 9
- fsgtest, 9

T

- t parameter, 9
- tasks
 - changing, 21
- testing
 - grammars, 16
 - vocabularies, 16
 - word lists, 16
- time loops, specifying, 8
- tips for Grammar Test Tool, 23
- tools
 - fsgenum, 7
 - fsgprint, 9
 - fsgtest, 9
- translations, printing standard output of, 9
- types of recognition results, 16

U

- using Grammar Test Tool, 13

V

- v parameter, 9
- verifying
 - behavior of grammar, 9
- vocabularies
 - testing, 16

W

- word lists
 - selecting for testing, 13
 - specifying as input, 11
 - testing, 16