WHITE PAPER

Guidelines for Developing Windows Applications Compatible with Dragon® NaturallySpeaking and Dragon® Medical

NUANCE

INTRODUCTION

These guidelines explain how to make a Windows® application compatible with Dragon NaturallySpeaking and Dragon Medical (referred to collectively in this document as Dragon). The guidelines are intended for all Windows application developers. The primary goal of these guidelines is to allow applications to work with Dragon NaturallySpeaking (Dragon) "out of the box," that is, without the necessity of using the Dragon SDK to integrate speech into the application. However, following them will also make applications more compatible with other speech applications and accessibility tools.

You may be reading these guidelines because your customers or end users have asked you to make it possible to use your application with Dragon. They may not have provided you with specifics, so this paper explains the main reasons that users adopt Dragon and what features are important when they use Dragon with an application. You may need to ask your customers why they want to use Dragon and which of its features are most important to them. You may also be participating in the Nuance Dragon Ready program for electronic health records (EHR) systems and wanting information on how to make your application Dragon Ready.

The uses of Dragon fall into two major areas: dictation (also known as speech-to-text) and navigation (also known as command-and-control). This paper addresses both of them.

*Note: if you are developing a web-based application, see the white paper "Guidelines For Speech-Accessible HTML For Dragon NaturallySpeaking," available in tech note **3067** in the Nuance knowledge base at http://knowledgebase.nuance.com. That white paper is more relevant than this one.*

Why Do Your Users Need Speech Recognition (Productivity or Accessibility)?

Speech-recognition users generally adopt the technology for one of two basic reasons:

• **Productivity**: users want to accomplish tasks more quickly than they could by hand. Usually they are interested in the dictation capability of Dragon, which is much faster than typing. Dictation increases task speed in text-intensive practices such as creating physician notes and legal documents. (Another way to look at productivity is in terms of transcription cost; Dragon saves costs by allowing professionals to generate their own text when they would otherwise have a transcriptionist generate it.)

Sometimes Dragon can be used to improve productivity through its custom command (macro) capability to automate sequences of actions in the application. This capability is analogous to the macro capability in applications such as Microsoft Office or macro add-on software such as Macro Express, with the added feature that macros can be triggered by voice (or by the press of a button on a Nuance handheld microphone).

• **Accessibility**: they need to use computer applications with limited or no use of their hands, so both dictation and navigation are important. Often they are more concerned with reducing mouse usage than keyboard usage because the mouse is likelier to cause repetitive stress injuries. In other cases, they may be using their hands for tasks such as handling papers or medical specimens. Ideally the hands-free user should be as productive as a fully-abled user, but this may not always be possible, especially if the application involves more navigation than text entry.

Dragon features that address navigation include:

- automatic generation of voice commands for standard Windows controls (menus, buttons, etc.)
- automatic generation of voice commands for links and other web-page features in Internet Explorer and Mozilla Firefox
- the ability to generate any keystroke sequence by voice (useful in any application that can be navigated by keyboard)
- the ability to move the mouse pointer by voice (a cumbersome last resort in an application that requires the mouse for navigation).

DICTATION AND CORRECTION

Full Dictation Support ("Select-and-Say")

Dragon allows the user to dictate into any application. It also allows the user to correct and edit text by voice, a capability known to end users as "Select-and-Say™." A key command is the "select <text>" command, in which <text> can be any word or group of words in the control that currently has the input focus. A related capability is the ability to play back dictated text to assist in proofreading. For more information on the voice commands that users expect to be able to use during correction, see the *Correcting Recognition Errors* section of the Dragon Help; you can find this section under the Contents tab of the Help window.

For the Select-and-Say and playback capabilities to work properly in your application, Dragon must be able to set and get the text and the selection in any control in order to synchronize them with its internal dictation buffer. It can do this automatically if you follow these guidelines:

Use supported window classes for edit controls. Dragon 10 supports the following control classes. (Borland controls, TE Edit, and TX Text are supported only in Professional, Medical, Legal, and SDK editions of Dragon.)

Vendor	Control Class
Microsoft	Edit, Rich Edit, Rich Edit20A, RichEdit20W, Rich Edit50W, Inktextbox, Inkedit, RichTextBox .NET controls
Borland	TMemo, TEdit, TRichEdit
Sub Systems	TE Edit versions 10, 11, 12, 13, 14, 15 (not TE Edit .NET)
Text Control	TX Text versions 10, 11, 12, 13, 14 and TX Text .NET

Another way to take advantage of Select-and-Say and playback capabilities is for your application to use Microsoft Word as its text editor. Dragon NaturallySpeaking version 8 and higher enables dictation support in Microsoft Word even if Word is embedded in or launched from another application. In versions of Dragon previous to version 8, follow the directions in tech note **3402** at http://knowledgebase.nuance.com to enable support for Microsoft® Word in your application.

NaturalText (Global Dictation), Non-standard Windows

Dragon can still be used even when dictation support is not enabled, which includes these cases:

- The application uses nonstandard edit controls, and dictation support has not been enabled for those controls using the Dragon SDK.
- The application is running on a server through a client/server system such as Citrix or Microsoft Windows Terminal Server, and the user is running Dragon on the client system.

This capability is called NaturalText or "global dictation." NaturalText sends keystrokes to the application in order to enter text and change the selection, and has no way to detect what text is present in the edit control (unlike dictation support, which does all of those things programmatically). The end user sees a message in the Dragon Bar that says "dictating into a non-standard window." The Select-and-Say indicator in the DragonBar is black instead of green. The Select-and-Say and playback capabilities function in a limited fashion. For details of the limitations of nonstandard windows and how end users can work within their limitations, see tech note **4247** or the topic *"Dictating to non-standard windows"* in the Dragon Help; you can find this topic by entering "non-standard windows" in the Search tab of the Help window.

Many end users find it frustrating to work in nonstandard windows, though others can accept the limitations or may not even notice them. For the greatest user satisfaction, it is best to implement your application so that it benefits from full dictation support as described in the previous section.

Additional Guidelines for Dictation

The following guidelines are equally applicable whether your application uses dictation support or NaturalText.

- Place the input focus in the appropriate text window, with an insertion point that makes it obvious to the user where typed or dictated text will appear. Make sure the visible focus is in the same control that would be returned by a call to a Windows® GetFocus call, because the latter is where Dragon places dictated text. This is because users can become very frustrated when their dictated does not appear where they expect it to, or does not appear at all.
- Return the focus to that window, at the same insertion point, if the application is deactivated and then activated again. The reason for this requirement is that the user needs to use commands that display the Dragon Spell dialog, such as "Spell That," to correct recognition errors. Your application window becomes deactivated when Dragon displays the Spell dialog and reactivated when the user closes the Spell dialog. If the focus has changed, Dragon cannot place the corrected word in the text and the user cannot continue dictating without first clicking into the text window to restore the focus; this situation causes frustration.
- If your application needs to take action as a result of changes to the text, do not assume that changes can come only from the keyboard or by pasting from the clipboard. Dragon dictation support changes the text by sending messages such as WM_SETTEXT and EM_REPLACESEL.
- Try not to enter a state where the text cannot be changed for an extended period of time. For example, do not change a read-write window into a read-only window. This is because the user will try to dictate into the window and become frustrated.

NAVIGATION BY VOICE

Say What You See™ (Tracking)

Background: Dragon implements a feature called "Say What You See." The Nuance engineering term for this capability is "tracking." Tracking allows the user to say the name of any menu (such as "File"), menu command (such as "Open"), button, or static control that is used to set the focus to an edit control. The user can precede the command with the word "click" if necessary to distinguish it from dictation. (A user-settable option requires the user to say "click" before every such command.) For more information on what users expect to be able to do with tracking, see the topic *"Controlling programs by voice quick reference"* in the Dragon Help.

Tracking commands are built at the start of every utterance (that is, when the user starts speaking after a pause), at which point Dragon enumerates the child windows of the active window. For each one that is a menu, button, or text label with an accelerator key, it builds a voice command for that control. The voice command for a menu or button generates a mouse click on that control. The voice command for a static text label generates the corresponding accelerator key.

Dragon does not attempt to build commands for individual items in a list box or combo box (except in Internet Explorer; see the HTML white paper referenced above). Users can select items by saying commands that generate keystrokes, for example "move down 2" or "press B". You can also create commands for these items using the voice-command control of the Dragon SDK.

Therefore, in order for tracking to work properly in your application, you must do the following:

- Give each control in the application a label that is descriptive, unique, and short enough to be used as a voice command.
- If the label must be long, shorten the corresponding voice command by enclosing part of the text in parentheses; the user does not have to say the part of the label that is between the parentheses.
- Do not label controls with images rather than text.
- Make it visually clear which label corresponds to each control.
- Use standard window classes for controls, such as "Button."
- If you create custom controls, make sure they have the appropriate class and also respond appropriately to the WM_GETDLGCODE and WM_GETTEXT messages.
- Always use accelerator keys in text control labels that are associated with (that is, used for navigation to) captionless controls such as edit controls and list boxes.

In addition to (or instead of) following the above requirements, you can use Microsoft Active Accessibility (MSAA). MSAA makes information about your application's UI elements available to Dragon as well as to other client programs that use MSAA, which include accessibility aids and automated testing tools. MSAA is a component of the Windows XP, Windows Server 2003, and Windows Vista operating systems, and can be installed on Windows 2000, Windows 98, Windows ME, and Windows NT. For more information, search for the *Active Accessibility* start page at www.microsoft.com.

Keyboard Accessibility

In general, applications that will be used by voice should not require the use of the mouse.

There are three reasons:

1. *Generating mouse actions by voice is slower than generating keystrokes by voice.*
2. *When the user wants to build a voice macro to automate a sequence of actions programming keystrokes into the macro is far more reliable than capturing mouse movements and also results in quicker macro execution.*
3. *If the application is running on a server through a client/server system such as Citrix or Microsoft Windows Terminal Server, and the user is running Dragon on the client system, tracking does not work and the user must rely on voice commands that generate keystrokes for navigation.*

To reduce your users' dependence on the mouse, follow the following guidelines:

- Make sure that all controls have keyboard shortcuts or can be reached with the Tab key.
- If the Tab key is used for navigation, make sure there is at least one control in a predictable location (such as the first control on the screen) that can be reached by a keyboard shortcut. Also make sure the tab order corresponds to the visual layout of the screen.
- If your application uses toolbars (which Dragon does not track), make sure each toolbar function can also be accessed through the application's menus and that the menus can be reached by keystrokes.
- Support the standard Windows keyboard shortcuts that are meaningful in your application. The Microsoft web site has several documents listing Windows keyboard shortcuts, such as knowledge base article 301583 ("List of the keyboard shortcuts that are available in Windows XP.") In particular, if your application supports formatted text, it should support the standard Windows keyboard shortcuts for bold (Ctrl+b), italics (Ctrl+i) and underline (Ctrl+i). Users expect to be able to build voice commands for these formatting functions, and these keyboard shortcuts allow them to do so.
- Make sure all the keyboard shortcuts in your application are documented or that you provide visual cues to tell the user what they are.
- If your application places a shortcut on the Windows Start menu and/or desktop, make the name of the shortcut pronounceable and short enough to use in a voice command. This is because Dragon uses MSAA to generate a "Start <application>" command for every shortcut on the Start menu and desktop.

SYSTEM INTERFACE

Here are some additional, general requirements.

- Include a VERSIONINFO block in your executable files, making it possible for other programs to query the version of your application. This allows Dragon to behave differently in different versions of your application, should the need arise.
- Do not allow your application to go into a spin loop that consumes 100% of the CPU, even though it may be invisible to your application's performance except when using instrumentation. The reason for this restriction is that Dragon needs to use 100% of the CPU while it is processing speech; if your application consumes the CPU then Dragon becomes unusable. The task manager tray icon is useful for detecting this problem. Pop-up menus and mouse-capture modes are the most common offenders.
- For keystroke input, if your application needs to check the current state of a key (such as the CTRL key VK_CONTROL), use GetKeyState and avoid GetASyncKeyState. This is because GetAsyncKeyState reads the current state of the physical keyboard, which is not meaningful if Dragon is in the process of generating a keystroke sequence. GetKeyState reads the keyboard state after it has been manipulated by Dragon.

CONCLUSION

Dragon attempts to adapt itself to your application, and you can help it by following the guidelines above. Still, it is best to perform realistic test scenarios with your application, using voice only, without touching the mouse or the keyboard.

It is a good idea to follow general guidelines for accessibility (and also Section 508 compliance in the United States). For reasons listed earlier, even fully abled users can benefit when an application is made accessible. Microsoft provides accessibility guidelines for application developers on their web site.

For assistance in developing your application, you can sign up for Nuance developer support. Developer support is not only for developers using the Dragon SDK; it is for any developer whose application needs to work well with Dragon and who needs help from Nuance engineers.

For more information, see http://www.nuance.com/naturallyspeaking/sdk/support/.

The experience speaks for itself™

NUANCE COMMUNICATIONS, INC.
ONE WAYSIDE ROAD
BURLINGTON, MA 01803
888 350 4836
NUANCE.COM

NUANCE