Nuance Vocalizer
3.0

Developer's Guide

Nuance Vocalizer 3.0
Developer's Guide

1005 Hamilton Avenue, Menlo Park, California 94025 U.S.A.
Printed in the United States of America.
Last updated July 2004.

Contents

About this guide

Nuance Vocalizer™ provides text-to-speech (TTS) services integrated with the Nuance System's distributed architecture.

This guide describes how to use Nuance Vocalizer to incorporate text-to-speech in a Nuance speech application, including how to invoke TTS from your application code and how to set up your runtime configuration to most efficiently use Nuance Vocalizer services.

Audience

This document is for developers creating speech applications based on the Nuance Speech Recognition System, including applications built with Foundation SpeechObjects™ or applications built with VoiceXML running on the Nuance Voice Web Server™.

Organization

The guide is organized as follows:

Chapter 1 introduces Nuance Vocalizer's features.

Chapter 2 covers the installation process and system requirements for running Nuance Vocalizer.

Chapter 3 describes how to start and configure Nuance Vocalizer with the Nuance System.

Chapter 4 provides information on invoking TTS from your application code and describes Nuance Vocalizer's support for speech synthesis markup tags.

Chapter 5 explains how to start and modify configuration settings for Nuance Vocalizer using the Launcher, a graphical tool included with your installation.

Chapter 6 outlines the procedures for preprocessing text input for email reading and creating text replacement filters.

Chapter 7 explains how to generate audio files from text input through the Offline Audio Generator tool.

Chapter 8 describes how to use the Dictionary Editor tool to customize the pronunciation dictionary for your applications.

Chapter 9 describes how English text input is handled by the TTS engine.

Appendix A provides information on how Canadian French text input is processed.

Appendix B provides information on how American Spanish text input is processed.

Related documentation

The Nuance documentation contains a set of developer guides as well as comprehensive online API reference documentation.

In addition to this guide, the documentation set includes:

- *Introduction to the Nuance System*, which provides a comprehensive overview of the Nuance System architecture and features, the available tools and programming interfaces, and the speech application development process.
- *Nuance System Installation Guide*, which describes how to install and configure the Nuance Speech Recognition System.
- *Nuance Grammar Developer's Guide*, which describes the process of designing grammars and creating a recognition package.
- *Nuance Application Developer's Guide*, which describes how to develop, configure, and tune a Nuance speech application. This manual describes general application development and deployment issues such as setting Nuance parameters and launching recognition clients, servers, and other required processes.
- *Nuance System Administrator's Guide*, which describes topics related to monitoring the performance of a running speech application, including controlling Nuance process log output, using the Nuance Watcher to manage processes across a network, and using the runtime task adaptation feature.
- *Nuance Verifier Developer's Guide*, which describes how to use the Nuance Verifier™ to add security features to speech recognition and IVR applications.
- *Nuance Platform Integrator's Guide*, which describes the process and requirements for integrating the Nuance System with an existing IVR

(interactive voice response) platform or toolkit. It provides detailed information on the primary Nuance integration APIs, the RCEngine and NuanceSpeechChannel, and also discusses other specialized APIs for adding custom components such as audio providers to your integration.

- *Nuance System Glossary*, which defines terms and acronyms used in the Nuance documentation set.
- *Nuance API Reference*, which includes HTML-based documentation for Nuance APIs, parameters, command-line utilities, and GSL (grammar specification language) syntax. To access this documentation, open the file *%NUANCE%\doc\api\index.html*.

You may also refer to the Foundation SpeechObjects documentation, including the *SpeechObjects Developer's Guide*, which describes the Nuance SpeechObjects framework and how to use it to build a speech recognition application. SpeechObjects documentation is shipped with the Foundation SpeechObjects product.

To view the entire set of documentation online, open the file *%NUANCE%\doc\index.html* in any HTML browser. Install the *%NUANCE%\doc* directory on an internal web server or file system to minimize space requirements. This directory includes all Nuance developer guides in both HTML and PDF format, and HTML reference documentation.

Typographical conventions

Nuance manuals use the following text conventions:

italic text	Indicates variables, file and path names, program names, program options, web and email addresses, as well as terms introduced for the first time. For example: Edit the *ag.cfg* configuration file.
`Courier New`	Indicates method/member functions, parameter names and values, program constants, and onscreen program output. For example: Nuance recommends that you set the parameter `audio.OutputVolume` to `255`.

`> Courier New` Indicates commands or characters you type in and the responses that appear on the screen. The > character indicates the MS-DOS command prompt or Unix shell. Everything after this character is intended to be typed in. For example:

```
> resource-manager
```

In this example, the text that you actually type at the command line is "resource-manager"

`Courier New` Indicates a value that you replace. For example:

The usage for the *nlm* utility is:

```
> nlm license_key
```

In this example, *`license_key`* is a value that you replace, so the text you actually type could be, for example:

```
> nlm ncr8-16-100-a-b22-333c4d55eeff
```

Note: The Nuance System runs on both Windows and Unix platforms. Windows syntax is typically used throughout the documentation. If you are running on a Unix platform, substitute Unix syntax. For example, use *$NUANCE* wherever *%NUANCE%* appears, and use "/" in place of "\" in path names. Differences in usage or functionality on Windows and Unix platforms are noted where relevant.

Where to get help

If you have questions or problems, Nuance provides technical support through the Nuance Technical Support Online (Nuance Technical Support Online), a web-based resource center that includes online forums, technical support guides on specific topics, access to software updates, as well as a support request form. If you are a member, go to *extranet.nuance.com* to log on, or see the Nuance website *www.nuance.com* for information on how to become a member.

To submit comments on the documentation, please send email directly to *techdoc@nuance.com*. Note that no technical support is provided through this email address. Technical support is provided through the Nuance Technical Support Online.

Introducing Nuance Vocalizer

1

Nuance Vocalizer is Nuance's best-of-breed, text-to-speech (TTS) server, enabling automated generation of high quality speech synthesis with natural voice quality. Nuance Vocalizer integrates seamlessly with the Nuance Speech Recognition System and Nuance Verifier™ for a complete speech solution.

This chapter introduces Vocalizer's features, the types of applications that can implement TTS functionality, and the benefits of using TTS.

Overview

Nuance Vocalizer is designed to be used either as a standalone TTS server or as part of a single-tier or distributed architecture.

Vocalizer supports many features that allow you to customize speech output and enhance performance, including:

- Support for Windows 2000 and SPARC Solaris platforms
- Compliance to VoiceXML 1.0 and the World Wide Web Consortium's (W3C) April 5, 2002 Working Draft for Speech Synthesis Markup Language Specification
- Dynamic editing of entries and pronunciations in user dictionaries
- Customized text preprocessing for email reading and text replacement filters
- Generation of audio files in various formats from text files or direct user input
- Ability to run voice data from disk, thus reducing system memory requirements
- Support for these voice packs:

Voice pack	Language	Gender
Laurie Woods	North American English	Female
Reed Johnston	North American English	Male
Claire Kingston	U.K. English	Female
Tim Cooper	U.K. English	Male
Sarah Brown	Australian/New Zealand	Female
Josh Donnelly	Australian/New Zealand	Male
Julie Deschamps	Canadian French	Female
Catalina Romero	American Spanish	Female

The voice names correspond to the Nuance standard persona names. For information on selecting and running voices, see "Selecting voices" on page 12.

Caution: When using the non-English voice packs and entering text in SSML mode, you must specify the encoding type (ISO-8859-1) to avoid unpredictable behavior. See "Encoding" on page 39.

Installing Nuance Vocalizer

2

This chapter defines system requirements supported by Nuance Vocalizer and describes the procedure for installing Vocalizer and supported voices.

Supported platforms and system requirements

Nuance Vocalizer runs on both Windows 2000 and SPARC Solaris 2.8 platforms and can be installed either:

- With the Nuance System

  Nuance Vocalizer works with all audio/telephony systems supported on the underlying version of the Nuance System. For installation information, see the *Nuance System Installation Guide*.

- Standalone (without the Nuance System installed)

  You can install and run Vocalizer independently of the Nuance System. However, Vocalizer uses the Nuance License Manager (NLM). If you plan to run the *nlm* process from another machine where the Nuance System is installed, make sure you also have the appropriate Nuance System service pack installed on that machine.

See the *Release Notes* for specific requirements.

Memory requirements

Nuance recommends the following minimum memory requirements:

- For 1-20 channels, 512 MB
- For 20-50 channels, 768 MB
- For 50 channels, 1 GB

Note: These recommendations are approximate figures, as actual memory requirements depend on the maximum prompt duration used.

Third-party software requirements

The GUI utilities included with Nuance Vocalizer require the Java 2 Runtime Environment (JRE) version 1.3 (or higher).

- **On Windows**, the JRE version 1.3 is included as part of the installation package. See "Installing Vocalizer on Windows" on page 5.

  If you upgrade to a newer version of the JRE *after* installing Vocalizer and the GUI utilities, you must update the path pointing to the JRE within the *.bat* files for each of the utilities:

  a From *%VOCALIZER%\bin\win32* (on Windows), open each of these files with a text editor such as Notepad:

  - *DictionaryEditor.bat*
  - *EmailReaderConfigTool.bat*
  - *OfflineAudioGenerator.bat*
  - *VocalizerLauncher.bat*
  - *ReplacementFilterEditor.bat*

  b In each of the files, change the path to the *javaw* executable. For example, if your path is:

  "C:\Program Files\JavaSoft\JRE\1.3\bin\javaw"

  change this to the location of the latest version, for example:

  "C:\Program Files\JavaSoft\JRE\1.4\bin\javaw"

- **On Solaris**, you must download and install the JRE if you do not have it already. See the Sun website (*www.sun.com*) for download information.

Installing Nuance Vocalizer

The Nuance Vocalizer installation package is available for download from the Nuance Technical Support Online website (*support.nuance.com*). After downloading the appropriate installation package, follow the installation procedures described next.

Installing Vocalizer on Windows

To install Vocalizer on Windows:

1 Double-click the downloaded executable file to run InstallShield.

a The opening dialogue window offers the option of including these GUI tools and the JRE version 1.3 in your installation:

- Dictionary Editor
- Email Reader Configuration Tool
- Offline Audio Generator
- Text Replacement Filter Editor
- Vocalizer Launcher
- Java 2 Runtime Environment (JRE) version 1.3

In a multi-server configuration, you do not need to install these tools on every machine, since modifications made using these tools can be distributed to other machines. Before including them in your installation, consider:

- If you do not plan to use the GUI tools on this machine, you are not required to install them or the JRE
- If you already have the JRE version 1.3 (or higher) installed, do not include the JRE in your installation
- If you have an earlier version of the JRE, uninstall it before installing version 1.3

See "Third-party software requirements" on page 4.

b You also have the option of changing your installation path. By default, Vocalizer is installed in the *Program Files\Nuance\Vocalizer3.0* directory and the *%VOCALIZER%* environment variable is set accordingly. Vocalizer executables are installed in the *%VOCALIZER%\bin\win32* directory.

2 Reboot your machine once installation is complete.

Installing Vocalizer on Solaris

To install Vocalizer on SPARC Solaris:

1 Log on as the *root* user

2 Copy the downloaded installation package into a temporary directory, for example

3 Gunzip the file:

```
gunzip installation-package.tar.gz
```

4 Untar the file:

```
tar xvf installation-package.tar.gz
```

5 Run the *install.sh* script and follow the instructions:

```
./install
```

6 Set the $*VOCALIZER* environment variable to the installation path where Vocalizer was installed, with either `bash` or `tcsh`:

```
bash-$] export VOCALIZER=Vocalizer_path
tcsh-$] setenv VOCALIZER Vocalizer_path
```

Nuance recommends that this environment variable be set in a persistent place, such as in a *.profile* (`bash`) or *.cshrc* (`tcsh`) file.

7 Set the $*NUANCE* environment variable:

```
bash-$] export NUANCE=$VOCALIZER/core
tcsh-$] setenv NUANCE $VOCALIZER/core
```

Note: *$NUANCE* may be set to *$VOCALIZER/core* or to an existing Nuance System installation.

8 Source the *SETUP* scripts in *$NUANCE* and *$VOCALIZER/bin/sparc-solaris* to set up your *PATH* and *LD_LIBRARY* environment variables, with either `bash` or `tcsh`:

- With `bash`

```
bash-$] cd $NUANCE
bash-$] . SETUP sh
bash-$] cd $VOCALIZER/bin/sparc-solaris
bash-$] . SETUP sh
```

- With `tcsh`

```
tcsh-$] cd $NUANCE
tcsh-$] source SETUP
tcsh-$] cd $VOCALIZER/bin/sparc-solaris
tcsh-$] source SETUP
```

Nuance recommends that this environment variable be set in a persistent place, such as in a *.profile* (`bash`) or *.cshrc* (`tcsh`) file.

Environment variables

On Windows, the Vocalizer installation process modifies your system's *PATH* variable to include necessary libraries. Make sure the *PATH* variable includes *%NUANCE%\bin\win32* and *%VOCALIZER%\bin\win32*.

On Unix, *$VOCALIZER/bin/sparc-solaris* is added to the *PATH* and *LD_LIBRARY* environment variables when you source the *SETUP* script, as described in the installation procedure for Solaris (step 8 on page 6).

Nuance Vocalizer licensing

Nuance provides a license key that controls the number of concurrent channels that can be opened to the TTS server. You can run this license file using an instance of the Nuance License Manager (*nlm*), as described in "Starting a Vocalizer TTS server" on page 11.

Note: The default two-port license shipped with Vocalizer uses port 8471. When you buy your Vocalizer license, this port number may change, and you must set the `lm.Addresses` parameter accordingly.

Testing your installation

You can test your installation by sending text to Vocalizer to be synthesized through the Offline Audio Generator, a graphical tool included with your installation. See Chapter 7 for information on using this tool.

Uninstalling Vocalizer

When you uninstall Vocalizer, all user data, including custom dictionary files, email configuration files, voice configuration files, and custom filters, is backed up into a temporary folder, *vocalizer_files-<date>-<time>*, in either the *%TEMP%* or *%SYSTEMROOT%* directory.

Installing voice packs

The voice packs are available for download from the Nuance Technical Support Online website (*support.nuance.com*) and do not require separate licensing.

Vocalizer servers only support one voice at a time. When you start your Vocalizer server, you can select which voice to use through the *-voice* command-line option. See "Selecting voices" on page 12.

Installing voice packs on Windows

To install a voice pack on Windows:

1. Double-click the voice pack executable file to run InstallShield.
2. You will be prompted for the installation path.

   By default, InstallShield detects your Vocalizer installation and installs the voice pack files in the appropriate directory.

Installing voice packs on Solaris

To install a voice pack on Solaris:

1. Log on as the *root* user.
2. Copy the voice pack file into a temporary directory:

   ```
   cp VoiceName_pack.tar.gz tmp
   ```

3. From that directory, uncompress the file:

   ```
   gunzip VoiceName_pack.tar.gz
   ```

4. Untar the file:

   ```
   tar xvf VoiceName_pack.tar
   ```

5. Run the install script:

   ```
   ./install-VoiceName.sh
   ```

   The script detects your Vocalizer installation and installs the voice pack files in the appropriate directory.

Configuring Vocalizer with the Nuance System

3

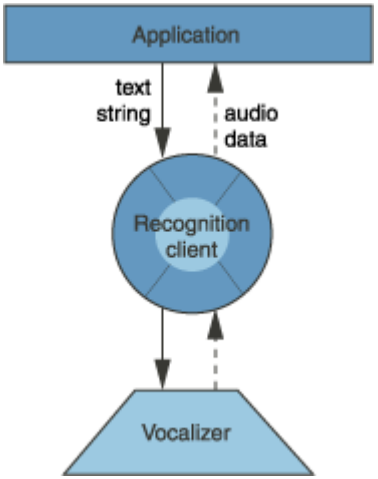
This chapter describes how to start the processes that let you use Nuance Vocalizer text-to-speech services from a Nuance speech application. You can start these processes directly from the command line, or via the Nuance Watcher tool.

Note: This chapter describes the runtime configuration for supporting text-to-speech. For information on Nuance programming interfaces for incorporating text-to-speech in your application code, see Chapter 4.

Including Vocalizer in a Nuance configuration

Vocalizer runs as an integrated part of a Nuance configuration, using functionality provided by the recognition client to issue TTS requests to Vocalizer and play the synthesized speech back to the caller.

The following diagram shows the general flow of data between the application and the TTS engine:



Vocalizer runs as a server process similar to other Nuance processes such as the recognition and compilation servers. You can set up a runtime configuration including Vocalizer TTS servers in one of two ways:

- Connect each recognition client directly to a single Vocalizer TTS server process
- Use a Nuance resource manager to mediate requests between recognition clients and one or more Vocalizer TTS servers

Nuance recommends that you manage Vocalizer server connections through a resource manager. The resource manager enables load balancing and scalability for large configurations. It also allows you to add, remove, and restart TTS server processes without restarting the entire configuration.

The following sections describe how to start a Vocalizer TTS server and how to integrate it within runtime configurations both with and without a resource manager.

Note: If you are not familiar with the resource manager and the overall Nuance System runtime architecture, see the *Introduction to the Nuance System* guide shipped with the Nuance System.

Starting a Vocalizer TTS server

To start a Vocalizer TTS server, you need to run *nlm* before running the *vocalizer* command-line program:

1. Start the license manager in a command-prompt window. From the *%VOCALIZER%* directory, enter:

   ```
   > nlm .\license.txt
   ```

2. Start *vocalizer* in a separate command-prompt window. From the *%VOCALIZER%\bin\win32* directory, include the hostname and port of the machine running the *nlm* process, setting the port to `8471`. For example:

   ```
   > vocalizer lm.Addresses=licenseServer:8471
   ```

   Note: The default two-port license shipped with Vocalizer uses port 8471. When you buy your Vocalizer license, this port number may change, and you must set the `lm.Addresses` parameter accordingly.

Command-line options

This section describes command-line options you can use when starting the *vocalizer* program. If you run *vocalizer* without including any options, default settings for speech synthesis are applied.

Specifying the number of channels

You can use the *--num_channels* option to specify the maximum number of simultaneous requests Vocalizer will handle. Requests exceeding the specified maximum experience higher latency (time to first audio) than other requests. For example, if you set *-num_channels 10*, and Vocalizer is already handling nine requests, the tenth request is handled immediately, but subsequent requests are blocked until a TTS request completes and Vocalizer can handle a new request.

Memory issues

Vocalizer's memory requirements are dependent upon the number of channels specified with the *-num_channels* option, so make sure you do not specify more channels than you need so that memory is not wasted. Even a relatively small number of channels (10-20) could affect performance on machines with small amounts of memory (512 MB) due to memory swapping.

License issues

Vocalizer allows you the option of having all licenses allocated at startup or having them allocated as they are required. By default, licenses are granted on a per request basis.

If you choose to preallocate your licenses, make sure *-num_channels* does not exceed the number of ports available with your license. When you start Vocalizer, it attempts to grab the specified number of licenses from the license server. If there are not enough licenses, Vocalizer will not start.

To enable preallocation of licenses, include the option *-preallocate_licenses* on the command line.

Selecting voices

By default, Vocalizer uses the North American English female voice, Laurie Woods.

To select a different voice when starting *vocalizer*, use the *-voice* option. For example, if you want to use the Reed Johnston voice pack, enter:

```
> vocalizer lm.Addresses=hostname:8471 -voice ReedJohnston
```

Vocalizer currently supports these voice packs:

Language	Gender	Voice pack	Command-line setting
North American English	Female	Laurie Woods	`lauriewoods`
North American English	Male	Reed Johnston	`reedjohnston`
U.K English	Female	Claire Kingston	`clairekingston`
U.K English	Male	Tim Cooper	`timcooper`
Australian English	Female	Sarah Brown	`sarahbrown`
Australian English	Male	Josh Donnelly	`joshdonnelly`
Canadian French	Female	Julie Deschamps	`juliedeschamps`
American Spanish	Female	Catalina Romero	`catalinaromero`

You can download these voice packs from the Nuance Technical Support Online website (*support.nuance.com*). See "Installing voice packs" on page 8.

Note: See Appendix A for information on Vocalizer's handling of Canadian French text input, and Appendix B for American Spanish information.

Using the -region option

For American Spanish, Vocalizer supports various conventions for numbers, dates, and addresses. To indicate how Vocalizer should interpret Spanish text input, set the *-region* option to either `US` or `CALA`. For example:

```
vocalizer lm.Addresses=hostname:8471 -voice catalinaromero
-region CALA
```

For information on the differences between these settings, see "Regionalism" on page 104.

Running voice data from disk

Vocalizer also allows you to run the voice pack from a disk in order to reduce memory requirements by about 250 MB at startup.

To enable this feature, include the *-voices_from_disk* option. For example:

```
vocalizer lm.Addresses=hostname:8471 -voice reedjohnston
-voices_from_disk
```

Note: At least one voice pack must be installed on your system before using this option.

Using the -pruning option

The *-pruning* option provides another means of controlling CPU usage. The *-pruning* option can be set to a value between 80 to 330. Default pruning values are specific to voice packs.

Modifying the *-pruning* value enables you to increase or decrease the number of channels that can be obtained for the same CPU type or speed, such as on a dual 1.4 GHz Pentium III. Lower *-pruning* levels (towards 80) allow for more channels, but may produce slightly degraded output. If you require extremely high quality output, you can increase the *-pruning* level, which reduces the number of channels supported. Changing the value for *-pruning* has little impact on overall memory consumption.

Selecting the audio encoding

Vocalizer can generate speech output encoded in the following formats:

Format	Usage
mulaw	Mulaw encoding used as a standard in North America and Japan. This is the default.
alaw	Alaw encoding used as a standard in Europe.

To select an encoding, use the *-encoding* option:

```
> vocalizer lm.Addresses=hostname:8471 -encoding format
```

where *format* is either `alaw` or `mulaw`.

Note: Make sure that your audio provider supports the encoding that you specify.

Text format options

Vocalizer can be configured to automatically determine what type of text is being sent to it and treat it appropriately. To do so, you must give Vocalizer a hint as to what text types are expected by:

- Specifying the markup language, either plain text, VoiceXML, or SSML
- Setting up Vocalizer to accept email files

Setting the default markup type

Vocalizer lets you specify the text to be synthesized using either plain text, VoiceXML markup tags, or the W3C's Speech Synthesis Markup language (SSML) tags. By default, Vocalizer uses plain text.

Use the *-text_type* option to specify which format you are using. For example, if you are using SSML tags, start *vocalizer* with the following setting:

```
> vocalizer lm.Addresses=hostname:8471 -text_type ssml
```

See "Using TTS in a VoiceXML application" on page 29 for information on how markup tags are supported.

Enabling the email parser

Vocalizer can parse and read email messages encoded in the RFC 822 standard. To enable this feature, specify the *-filter* option on the *vocalizer* command line:

```
> vocalizer lm.Addresses=hostname:8471 -filter email
```

See "Passing email messages to Vocalizer" on page 47 for information on how you can configure the email parser.

Using custom filters

You can also use the *-filter* option to specify multiple filters, using a comma-separated list. (Do not include spaces between items.) For example,

```
>vocalizer -filter filter1,filter2,[other_filters]
```

See "Creating custom filters for text replacement" on page 52.

Expected text formats

Setting the *-text_type* and *-filter* options enables Vocalizer to expect the correct data format and treat it appropriately, as follows:

Command-line option	Expected format
`-text_type ssml`	Plain text or SSML
`-text_type vxml`	Plain text or VoiceXML
`-text_type plaintext`	Plain text
`-text_type ssml -filter email`	Plain text, SSML, or email
`-text_type vxml -filter email`	Plain text, VoiceXML, or email
`-text_type plain -filter email`	Plain text or email
`-filter email`	Plain text or email

For more information on creating custom filters, see "Creating custom filters for text replacement" on page 52.

Option summary

The full usage statement for the *vocalizer* command-line tool is:

```
> vocalizer lm.Addresses=hostname:8471
   [-num_channels number]
   [-preallocate_licenses]
```

```
[-text_type type]
[-voice voice_name]
[-region region]
[-voices_from_disk]
[-pruning value]
[-encoding format]
[-filter email]
[-dictionary_port port_number]
[-help]
```

This table summarizes the settings for each option:

Option	Description	Type	Values	Default
-num_channels	Specifies the number of engines this instance of Vocalizer will support. See “Specifying the number of channels” on page 11.	Integer	*number*	2
-preallocate_ licenses	Indicates that licenses are to be preallocated at startup. By default, this option is not included and licenses are granted per request as described in “License issues” on page 11.	String	n/a	n/a
-text_type	Specifies the markup language used in the text to be passed to the Vocalizer TTS engine. See “Selecting the audio encoding” on page 13.	String	plain ssml vxml	plain
-voice	Specifies the voice to use for synthesis. See “Selecting voices” on page 12.	String	laurieswoods reedjohnston clairekingston timcooper sarahbrown joshdonnelly juliedeschamps catalinaromero	LaurieWoods
-region	Specifies the convention to use when interpreting number, date, and address text input for American Spanish. See “Regionalism” on page 104.	String	US CALA	US
-voices_from_ disk	Specifies to run the voice pack from disk. See “Running voice data from disk” on page 12.	n/a	n/a	n/a

Option	Description	Type	Values	Default
-pruning	Specifies the pruning level for the trade-off between output quality and CPU usage.	Integer	Any value between 80 and 330 inclusive	Specific to voice pack
-encoding	Selects the audio encoding to be used to generate the speech output.	String	`mulaw` `alaw`	`mulaw`
-filter	Indicates how text input will be processed by the TTS engine. You can specify multiple filters with this option, including: ▪ `email`—To enable preprocessing of email messages for appropriate synthesis. See "Enabling the email parser" on page 14. ▪ Custom filters—To enable text replacement. See "Using custom filters" on page 14. For example, `>vocalizer -filter email, filter1,filter2`	String	`email` *`custom_filters`*	No default
-dictionary_port	Specifies the port for phoneme translation with the Dictionary Editor tool. See Chapter 8.	String	*`port_number`*	`22552`
-help *-h*	Displays the usage statement for the *vocalizer* program. When you include this option on the command line, *vocalizer* displays the usage statement and exits, regardless of other options you specify.	n/a	n/a	n/a

Nuance parameter settings

The *vocalizer* program can also accept a number of Nuance configuration parameters on the command line. These parameters control how the *vocalizer* process interacts with other Nuance processes, and include:

`rm.Addresses`

Defines the location (machine and port) of one or more resource managers for the TTS server to connect to. See "Connecting Vocalizer to a resource manager" on page 17 for more information.

`lm.Addresses`

Defines the location (machine and port) of the machine running the Nuance License Manager (*nlm*) process for Vocalizer. Set the port to `8471`. See "Starting a Vocalizer TTS server" on page 11.

`tts.Port`

Specifies the port the TTS server uses to listen for requests. The default is `32323`. You only need to set this if you want to use a specific port number other than the default.

If you are running two or more Vocalizer TTS servers on the same machine you must set this for all but one TTS server process. The first instance you start uses the default port, so you must specify a different port for additional servers.

`tts.ResourceName`

Provides a name for an instance of the Vocalizer TTS server. Clients can use this name to direct a TTS request to a specific class of servers. See "Directing requests to a specific TTS server" on page 24.

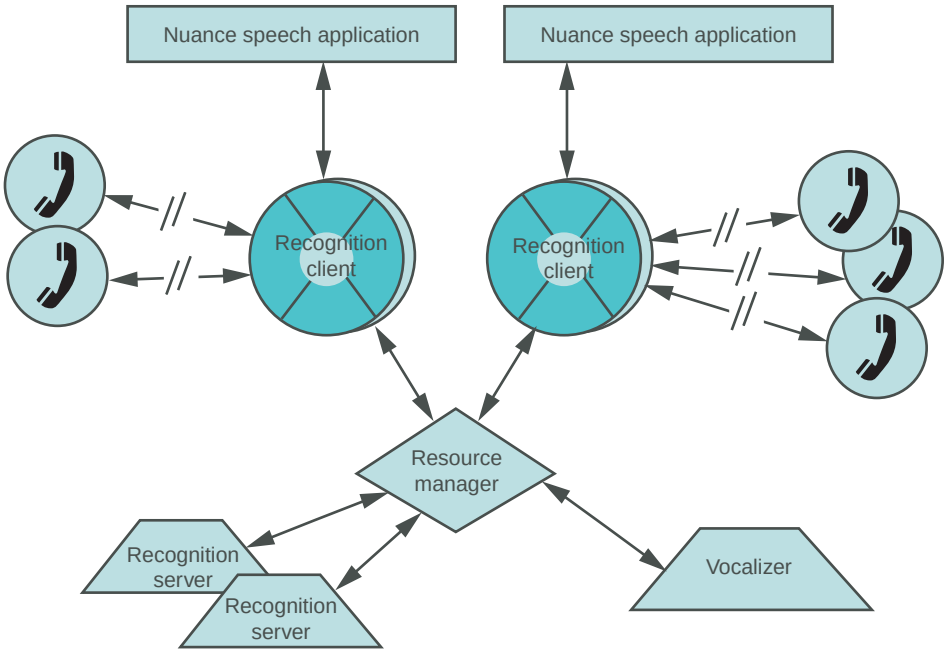
`config.LogFileRootDir` and related parameters

Set this parameter to cause the TTS server process to output log files. Other `config.LogFile*` parameters control how log files are serially numbered and how many files are produced. If you start *vocalizer* via the Nuance Watcher, you should enable this logging, as the program output is lost otherwise. See "Enabling Vocalizer logs" on page 23 for more information.

Connecting Vocalizer to a resource manager

Nuance recommends running Vocalizer in a configuration in which the Nuance resource manager controls the recognition client's access to the TTS server. This allows multiple recognition clients—each of which can support multiple ports—to use Vocalizer's TTS services through one or more Vocalizer TTS server processes. Whenever a recognition client requests a text-to-speech operation, the resource manager identifies the least-busy TTS server, and the client connects to that server for the duration of the operation. Nuance recommends this configuration even if you are only running a single TTS server, as it allows you to transparently add or swap out additional servers to the recognition clients.

A small-scale configuration of this type might look like this:



To start this type of configuration:

1 Start the *resource-manager* process:

```
> resource-manager
```

2 Start *nlm* with the Vocalizer license:

```
> nlm .\license.txt
```

3 Start the *vocalizer* process, specifying the location of the resource manager with the `rm.Addresses` parameter. For example, if you started the resource manager on a machine named "roadrunner":

```
> vocalizer rm.Addresses=roadrunner
  lm.Addresses=hostname:8471
```

If your configuration includes multiple resource managers on different machines, you can set `rm.Addresses` to a comma-separated list of all machines running resource managers, for example:

```
> vocalizer rm.Addresses=roadrunner,coyote
  lm.Addresses=hostname:8471
```

If you run the resource manager on a specific port (other than the default), include the port number in the `rm.Addresses` setting, for example:

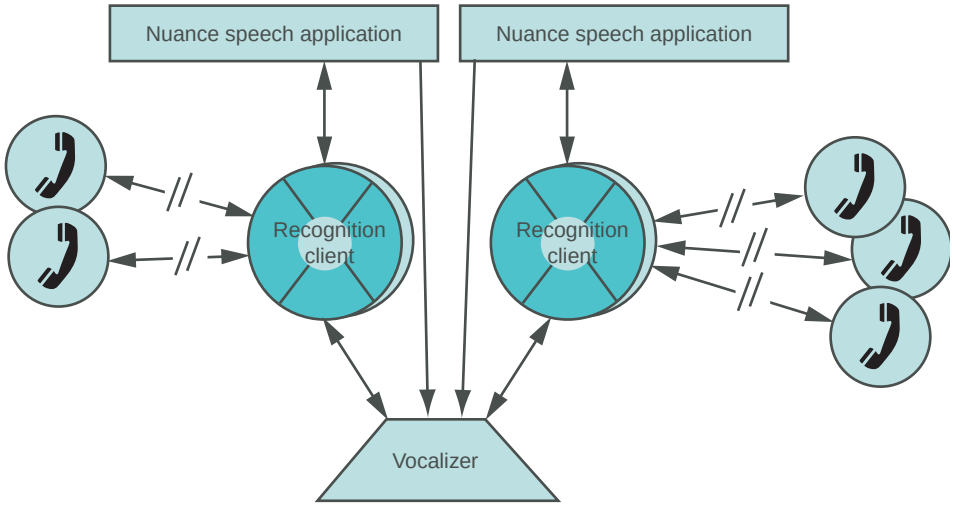
```
> resource-manager rm.Port=8584
> vocalizer rm.Addresses=roadrunner:8584
  lm.Addresses=hostname:8471
```

This causes the *vocalizer* process to register with the resource manager, and any application requests for TTS generation are directed to the Vocalizer server.

Note: See the Nuance System documentation for information on starting the other processes in your configuration, including the *recserver*. You can start *vocalizer* at any point after the resource managers have been started (that is, before or after the recognition servers and other processes).

Running Vocalizer without a resource manager

You can also run Vocalizer without a resource manager. In this case the recognition client sends all TTS requests directly to a specific Vocalizer TTS server. This type of configuration looks like this:



To start this type of configuration:

1 Start *nlm* with the Vocalizer license:

```
> nlm .\license.txt
```

2 Start the *vocalizer* TTS server process before starting your application or recognition client:

```
> vocalizer lm.Addresses=hostname:8471
```

3 Use the Nuance parameter `client.TTSAddresses` to specify the location of the running TTS server when you start your application. For example, if you have an application you start with the command-line program *my_app* that accepts Nuance parameters on the command line:

```
> my_app client.TTSAddresses=coyote
```

Changing the port setting

By default, Vocalizer uses the port 32323 to listen for direct requests from a recognition client. To use a different port:

1 Set the `tts.Port` parameter on the *vocalizer* command line to the new port:

```
> vocalizer lm.Addresses=hostname:8471 tts.Port=5353
```

2 Include the port number in the `client.TTSAddresses` setting:

```
> my_app client.TTSAddresses=coyote:5353
```

If you are running multiple TTS servers, you can specify them all as a comma-separated list in the `client.TTSAddresses` parameter, for example:

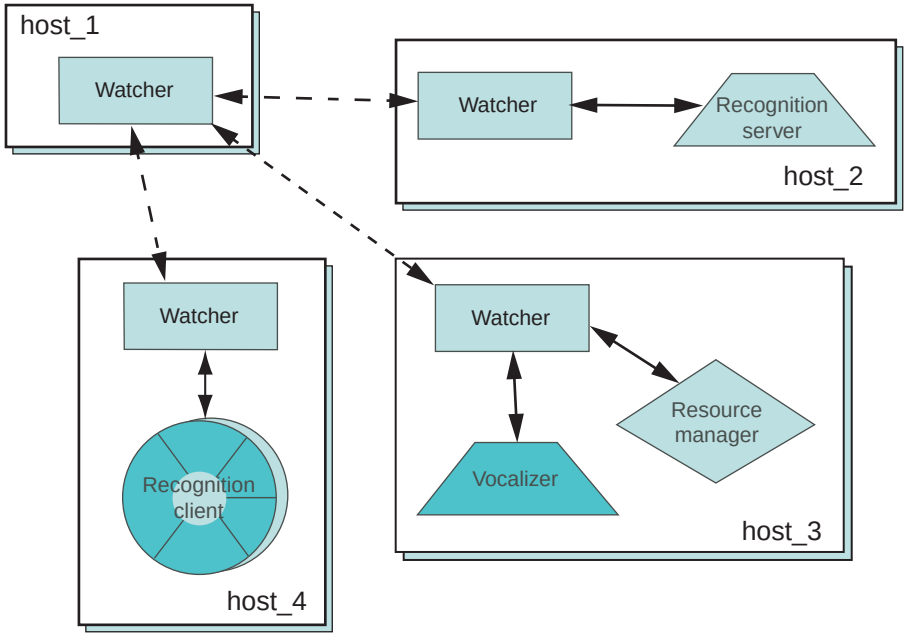
```
> my_app client.TTSAddresses=coyote,roadrunner
```

When the recognition client needs to issue a TTS request, it tries to connect to the first TTS server in the list, then the second, and so on, until it successfully connects to an available server.

Using the Nuance Watcher to start Vocalizer

The Nuance Watcher, shipped with the Nuance System, is an operations tool that monitors Nuance processes and can restart them if they fail. You can also use the Nuance Watcher to manually start and stop processes, and query the parameter information associated with a given process. Vocalizer is integrated with the Nuance Watcher, meaning that you can start and stop Vocalizer through the Watcher, and have the Watcher restart the Vocalizer TTS server process if it fails.

To use the Watcher, you run the Nuance *watcher-daemon* on each host machine in your Nuance configuration. That Watcher can detect and communicate with any Nuance process—including resource managers, recognition servers, and TTS servers—running on the machine. Each Watcher can also communicate with other Watchers on other machines in your network. Typically, you use one *master* Watcher to monitor all Nuance processes on all machines in your configuration:



Note: For complete information on the Nuance Watcher, see the *Nuance Application Developer's Guide*.

The following sections describe how to start Vocalizer through a Watcher startup file, and how to communicate with Vocalizer through a running Watcher.

Note: Nuance also recommends that you use the Nuance logging parameters when you start processes via the Watcher, otherwise the log output is lost. See "Enabling Vocalizer logs" on page 23 for information.

Starting Vocalizer automatically with a Watcher startup file

With the Watcher, you can create a startup file containing all the processes you want to start on a given machine. Each line of your startup file contains the command line for a Nuance process you want to start. You then pass this file to the *watcher-daemon* using the `watcher.DaemonStartupFilename`, and the Watcher attempts to start each process with the commands you specify.

To start Vocalizer through a Watcher startup file:

1 In your startup file, specify the process name *vocalizer*. You can pass in any options you can pass to the *vocalizer* command-line executable, for example:

```
vocalizer -num_channels 15 lm.Addresses=hostname:8471
```

You typically also specify:

- The `watcher.RestartOnFailure` parameter (this tells the Watcher to restart the process if it dies)
- Parameters to set up program logging

The following shows a simple startup file that starts a resource manager and a TTS server. Use forward slashes in path specifications (all platforms) and use a backslash to continue a command line on multiple lines. Make sure there are no spaces before or after "\".

```
# Start a resource manager
resource-manager lm.Addresses=acme\
  watcher.RestartOnFailure=TRUE\
  config.LogFileRootDir=d:/logs/rm/rmlog
  config.LogFileMaxNum=20
# Start the Nuance Vocalizer TTS server
vocalizer -num_channels 15
  watcher.RestartOnFailure=TRUE\
  config.LogFileRootDir=d:/logs/tts/ttslog
  config.LogFileMaxNum=20\
  rm.Addresses=hostname lm.Addresses=acme:8471
```

2 On the *watcher-daemon* command line, set `watcher.DaemonStartupFilename` to the name of your startup file:

```
> watcher-daemon watcher.DaemonStartupFilename=
  c:\scripts\watcher_startup.txt
```

Communicating with Vocalizer through the Watcher

You can also communicate with Vocalizer through a running Watcher process, including:

- Starting a Vocalizer TTS server
- Quiescing or killing a Vocalizer TTS server
- Getting general runtime information on the Vocalizer TTS server such as the process ID or the time the process was started

To start a Vocalizer TTS server through the Watcher Telnet interface, you could send the following command:

```
> startvp vocalizer watcher.RestartOnFailure=TRUE \
  config.config.LogFileMaxNum=d:/data/logs/tts
```

The Watcher lists Vocalizer TTS server processes as *vocalizer* and process parameters.

Note: For more information on information available through the Watcher, see the *Nuance Application Developer's Guide*.

Enabling Vocalizer logs

The Vocalizer TTS server process can generate logs similar to other Nuance runtime processes. If you are using the Watcher, you should enable logging, as standard program output is otherwise lost.

To enable logging:

1. Create a folder to store the log files.
2. On the *vocalizer* command line, set the parameter `config.LogFileRootDir` to the location at which you want log files to be written. This enables logging and sets both the location to which the files are written and the "root" name for the files. For example:

   ```
   > vocalizer config.LogFileRootDir=d:/data/logs/vocalizer/tts
   ```

   causes Vocalizer to write log files named *tts001*, *tts002*, and so on to the directory *d:\data\logs\vocalizer*.

   Note: To set this parameter in a Watcher startup file, use forward slashes in your path name instead of back slashes. On a Windows DOS command line, you can use either format.

You can further control the behavior of the logging mechanism by setting these parameters on the *vocalizer* command line:

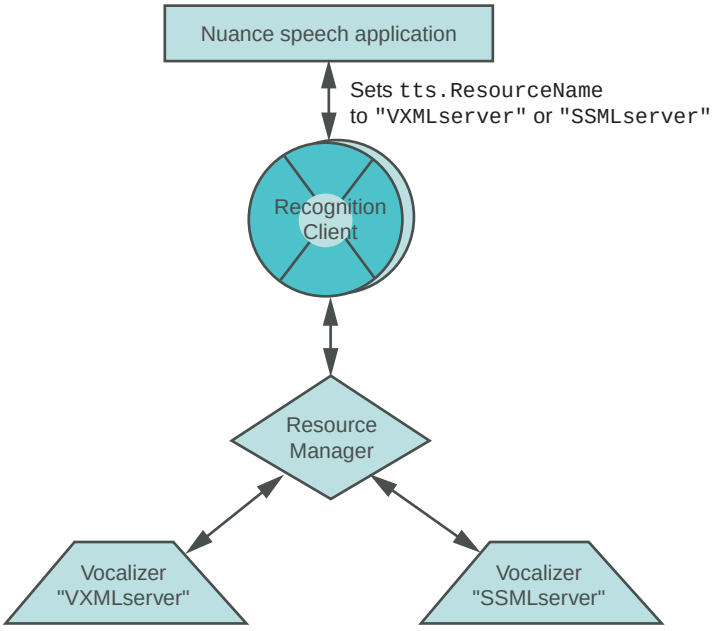
- `config.LogFileMaxNum` controls the number of sequential log files created. If you set this to `5`, for example, the logger creates *tts001* through *tts005* and then restarts with *tts001*. The last version of the *tts001* log file is replaced.
- `config.LogFileMaxSize` sets the maximum size of each log file, in megabytes. If you set this to `1`, for example, the logger writes to *tts001* until it reaches 1 MB, and then closes *tts001* and starts *tts002*.

Note: `config.LogFileRootDir` and related parameters must be set on the command line for each process you want to generate logs, including *vocalizer* and *watcher-daemon*. You cannot set these parameters globally, for example, in a *nuance-resources.site* file.

Directing requests to a specific TTS server

Vocalizer provides a mechanism that lets you direct a TTS request to a specific Vocalizer TTS server at runtime. This is useful if you want to run multiple TTS servers with different characteristics (for example, supporting different markup tags), and select the server to use for each individual request.

To do this, you give each server a name via the `tts.ResourceName` parameter as you start it, and use this name at runtime to indicate the server you want to use for each request:



Note: You must include a resource manager in your configuration to be able to hot swap TTS engines via the `tts.ResourceName` parameter at runtime.

To set up the configuration shown:

1 Start your resource manager:

```
> resource-manager
```

2 Start the Vocalizer TTS servers, providing a name for each by setting the `tts.ResourceName` parameter on the *vocalizer* command line, for example:

```
> vocalizer lm.Addresses=hostname:8471 -text_type vxml
  rm.Addresses=coyote
  tts.ResourceName=VXMLserver
```

```
> vocalizer lm.Addresses=hostname:8471 -text_type ssml
  rm.Addresses=coyote tts.ResourceName=SSMLserver
  tts.Port=32324 -dictionary_port 22553
```

Note: By default, Vocalizer opens two ports at startup, the TTS port (32323) and the dictionary port for phoneme translation (22552). If you are running more than one instance of Vocalizer on the same machine, only one instance can use the default port settings. Make sure to modify these port settings for other instances by setting the `tts.Port` and `-dictionary_port` on the *vocalizer* command line.

3 From the client side, use the parameter `tts.ResourceName` to select a specific TTS server.

If this parameter is set by the client, any TTS request is directed to a TTS resource with the same name. If it is not set, it is simply directed to the first TTS server identified by the resource manager. By changing the setting of the parameter before issuing a request, you can send the request to any named TTS resource running in your configuration.

Every Nuance API includes a mechanism for setting parameters at runtime. When programming with the NuanceSpeechChannel, for example, use the `setStringParameter` method:

```
sc.setStringParameter("tts.ResourceName", "SSMLserver");
CorePromptPlayer prompter = sc.getPromptPlayer();
prompter.appendTTS("Hello World");
prompter.play(false);
```

You can also set the parameter when you initialize your application. If most of your prompts include VoiceXML markup tags, for example, you can set the VoiceXML TTS resource as the default when you start your application. For example, if you start your application with the command-line executable *myapp*, you could run:

```
> myapp rm.Addresses=coyote tts.ResourceName=VXMLserver
```

All TTS requests go to the "VXMLserver" TTS server until `tts.ResourceName` is explicitly reset by your application.

Note: To set Nuance parameters from your application command line, your application's initialization code must pass command-line arguments through to a NuanceConfig object. The mechanism for doing this depends on which Nuance API you use to write your application.

You can start multiple Vocalizer TTS servers with the same name. If you need to run multiple servers with the same characteristics to handle the volume of TTS requests, start each with the same `tts.ResourceName` value. When the recognition client's `tts.ResourceName` parameter is set to that name and a TTS

request is issued, the resource manager locates the least-busy of the servers with that name.

Playing TTS prompts from your application

4

Vocalizer's TTS engine puts a human voice to your application interface, producing intelligible, synthesized speech from text input. This chapter describes how to:

- Invoke a TTS prompt from applications written with Nuance development APIs, including SpeechObjects, NuanceSpeechChannel, and the RCEngine
- Specify a TTS prompt using VoiceXML

Note: This chapter describes the APIs for specifying TTS prompts within your application code. At runtime, you must have a TTS server running and configured as described in Chapter 3 for the TTS API calls to work.

For information on the conventions used to synthesize the text based on the way you transcribe it, see Chapter 9.

Using Nuance prompt playback functions

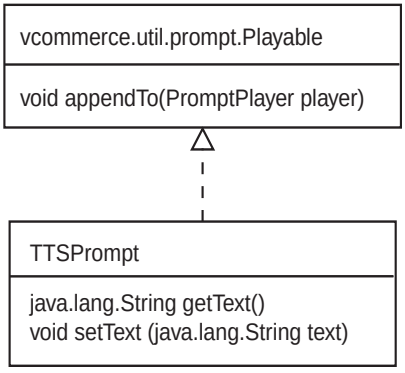
All Nuance development APIs include mechanisms for specifying TTS prompts. These mechanisms are similar to the way you work with prerecorded prompts, except that you specify the text to synthesize instead of the name of an audio file. You specify the text directly in your function call, for example, in the `RCEngine:PlayPrompts` call.

The following sections describe the mechanisms in each Nuance API, including:

- SpeechObjects
- NuanceSpeechChannel
- RCEngine

SpeechObjects

The Foundation SpeechObjects, Nuance's Java-based speech application development framework, defines classes that let you create specific TTS prompt objects from a simple text string. This class, TTSPrompt, implements the *Playable* interface, which allows it to be appended to a prompt queue and played directly:



To use the TTSPrompt class, just set the text string you want to be synthesized. You can then append the prompt to the prompt queue of your SpeechChannel™. For example:

```
TTSPrompt tts = new TTSPrompt("Hello world");
tts.appendTo(sc);
```

Note: In this example, *sc* is the SpeechChannel object being used by your application.

NuanceSpeechChannel

If you are programming with the NuanceSpeechChannel API directly instead of using the Foundation SpeechObjects, use the `appendTTS` method defined in the CorePromptPlayer interface to create TTS prompts. This method creates the prompt and appends it to the prompt queue.

The NuanceSpeechChannel class includes a default implementation of the CorePromptPlayer interface. For example, if *sc* is your NuanceSpeechChannel object, the following appends and plays a TTS prompt:

```
CorePromptPlayer player = sc.getPromptPlayer();
player.appendTTS("Hello world");
player.play(false);
```

For more information about the NuanceSpeechChannel API, see the online documentation shipped with the Nuance System.

Note: The TTSPrompt class described in “SpeechObjects” on page 28 does not work with the CorePromptPlayer class. You must use the SpeechChannel implementation provided by the SpeechObjects framework to use TTSPrompt.

RCEngine

The RCEngine is a C++ recognition client-level API targeted for use by developers who are integrating Nuance speech technology with an existing telephony platform. To play TTS prompts through a Nuance RCEngine, you use the same RCEngine `PlayPrompts` function you use to play a prerecorded prompt. Instead of specifying a set of audio file names, you can specify a TTS string by using the prefix `"-tts_text:"`. This indicates that the following text is the transcription for a TTS prompt, and not the name of a prerecorded audio file.

For example:

```
char const * prompts[2] = {"-tts_text:Hello world", NULL};
unsigned play_id = rce->GetUniqueID();
rce->PlayPrompts(prompts, play_id);
```

Note that the argument passed to `PlayPrompts` is actually a null-terminated array of strings representing prompts to be played. This means you can include multiple TTS prompt specifications, or mix TTS prompts with prerecorded prompts. You can also add a pause to the array of prompts by using the prefix `"-pause:"` followed by a number of milliseconds. For example:

```
char const * prompts[4] = {"welcome.wav", "-pause:500",
   "-tts_text:Mister Jones", NULL};
unsigned play_id = rce->GetUniqueID();
rce->PlayPrompts(prompts, play_id);
```

See the *Nuance Platform Integrator’s Guide* shipped with the Nuance System for more information on the RCEngine API.

Using TTS in a VoiceXML application

You can use Vocalizer with the Nuance Voice Web Server to generate TTS prompts from VoiceXML applications.

Vocalizer supports the following markup languages:

- VoiceXML 1.0, in accordance with the W3C’s May 5, 2000 Note (*www.w3c.org/TR/voicexml*)
- SSML 1.0, based on the W3C’s April 5, 2002 Working Draft for Speech Synthesis Markup Language Specification (*www.w3.org/TR/2002/WD-speech-synthesis-20020405*)

Text input with markup elements *must* be well-formed in order to be recognized as XML markup rather than plain text. For example, the following text is valid SSML:

```
<speak> Hello <break/> World</speak>
```

However, this text:

```
Hello <break/> World
```

would be synthesized as:

"Hello angle bracket break slash angle bracket world."

Note that attributes specified as required in VoiceXML 1.0 and SSML 1.0 are not enforced by Vocalizer. If a required attribute is missing from a tag, Vocalizer still attempts to parse and handle that tag in a best-effort manner. Both VoiceXML 1.0 and SSML 1.0 translators treat all numeric attribute values as integers.

VoiceXML 1.0 elements

When using VoiceXML 1.0 elements, make sure Vocalizer is running in `vxml` mode:

```
> vocalizer lm.Addresses=hostname:8471 -text_type vxml
```

The primary VoiceXML element for specifying a TTS prompt is the `<prompt>` element. Any character data within a `<prompt></prompt>` block that is not nested in another child element is treated as text to be synthesized by the TTS server. For example:

```
<prompt>
   Welcome to the Acme Travel Company.
</prompt>
```

Note: Text sent to Vocalizer must be well-formed XML in order to be recognized as XML markup rather than plain text.

VoiceXML defines a number of child elements you can nest within a `<prompt>` element to control the characteristics of the synthesized text, including those listed in Table 1.

Table 1: Nested elements

Element	Description
`<break>`	Inserts a pause.
`<emp>`	Increases or reduces the emphasis put on specific words.
`<div>`	Allows you to specify either a sentence or a paragraph.

Table 1: Nested elements *(continued)*

Element	Description
<audio>	Plays an audio file within a prompt. Supports the following encoding: Supports the following encoding: For WAV files: ■ 8 kHz, mulaw, unheadered ■ 8 kHz, mulaw, Sphere header ■ 8 kHz, PCM, Windows WAV format For AU files: ■ 8 kHz, mulaw (type 1), AU header ■ 8 kHz, PCM (type 3), AU header See “URI formats for the SSML <audio> element” on page 40.
<pros>	Changes prosodic characteristics of synthesized speech, including volume, pitch, range, and rate (speed).

<sayas>	Provides a number of attributes for controlling how text is synthesized.
	For example, you can define a phrase to spell out an acronym or pronounce an unusual word:
	`<prompt>`
	`  Contact the`
	`  <sayas sub="world wide web consortium">W3C</sayas>`
	`  for more information.`
	`</prompt>`
	This generates the prompt:
	“Contact the World Wide Web Consortium for more information.”

<value>	Inserts the current value of a variable into the string to be sent to the TTS engine.
	This shows a very simple example:
	`<var name="x" expr="12"/>`
	`<prompt>`
	`  You have <value expr="x"> new email messages.`
	`</prompt>`
	In a real application, of course, you would determine the value of the variable programmatically rather than defining a static value.

For more information on writing VoiceXML applications, see the Nuance Voice Web Server documentation.

Note: The elements listed above may not all be supported on all versions of the Voice Web Server.

SSML 1.0 elements

When using SSML 1.0 elements, make sure text input to Vocalizer is well-formed XML.

Note: Unless you are using the `<speak>`, `<paragraph>`, or `<sentence>` tags, do not include punctuation marks that denote the end of a sentence (".", "?", "!", or "...") in tagged data, as these cause the SSML element to be ignored.

Table 2: Supported SSML 1.0 elements

Element	Attribute	Value	Notes
`<speak>`	—	—	Root element. **Note:** Only a single `<speak>` element is allowed per request as this is a root element.
`<paragraph>` `<p>`	—	—	Identifies the enclosed text as a paragraph.
`<sentence>` `<s>`	—	—	Identifies the enclosed text as a sentence.

`<phoneme>`	`alphabet`	`native`
		`ipa`
		`worldbet`
		`xsampa`
			Provides a phonetic pronunciation for the enclosed text. To use Vocalizer's native phoneme set, change the value of alphabet to `native`. See "Resolving XML parsing errors with predefined entities" on page 39.
			Default is `ipa`

	`ph`	—	Specifies the phoneme string.
`<sub>`	`alias`		Replaces contained text with the value of the `alias` attribute. The `alias` attribute is required. If it is missing, an error is logged and the `<sub>` element is ignored. **Note:** This element replaces the `sub` attribute of the `<say-as>` element, which is no longer supported.

Table 2: Supported SSML 1.0 elements *(continued)*

Element	Attribute	Value	Notes
<emphasis>	level	strong moderate reduced none	Specifies that the contained text be spoken with emphasis. Default is moderate
<break>	size	small (500ms) medium (1000ms) large (2000ms) none	Controls the pausing or other prosodic boundaries between words. For example: <speak> Take a deep breath <break/> then continue. </speak> Default is medium **Note:** The <break> element must be enclosed by surrounding text when used as a child of the <audio> element.
	time	—	Indicates the duration of the pause in milliseconds or seconds, for example: 2000ms, 2000 or 2s.
<say-as>			Specifies the type of text construct contained within the element.
	type	acronym	Indicates contained text is an acronym. Vocalizer capitalizes the contained text and attempts to look it up in the acronym dictionary. If no suitable match is found, the acronym is spoken letter by letter.
	type	spell-out	Indicates contained text should be spoken letter by letter.
	type	number number:ordinal	Ignores all non-digit characters except Roman numerals. Interpretation of intermixed Roman numerals and digits is based on the first character. If the first character is a Roman numeral, the entire sequence is treated as a Roman numeral. Otherwise, Vocalizer ignores the non-digit characters and treats the data as a digit sequence.
	type	number:digits	Ignores all non-digit characters.
	type	date	Indicates contained text is a date.

Table 2: Supported SSML 1.0 elements *(continued)*

Element	Attribute	Value	Notes
<say-as>		date:ymd date:mdy date:dmy date:ym date:my date:md date:y date:m date:d	Specifies the expected order for the components of the date input. For example: <say-as type="date:ymd"> 02/03/04</say-as> and <say-as type="date:dmy"> 04/03/02</say-as> would both be interpreted as: “March fourth two thousand and two” if using North American English voice packs.
	type	time:hms time:hm time:h	Supports data in a 24-hour clock format only. Data is converted into 12-hour clock time. For example: ■ 13:30:10 is converted to “one thirty and ten seconds p.m.” ■ 12:00:00 is converted to “twelve o’clock noon” ■ 00:00:00 is converted to “twelve o’clock midnight”
	type	currency	Supports values which can be interpreted as dollars and cents only. International currency indicators, such as USD or CAD, are not supported.
	type	name	Indicates contained text is a proper name.
	type	net:email net:uri	Indicates contained text is an internet identifier.
	type	telephone	Supports North American style phone numbers of 7, 10, and 11 digits. All punctuation is stripped and ignored.

Table 2: Supported SSML 1.0 elements *(continued)*

Element	Attribute	Value	Notes
<say-as>	type	voicexml:date	Correspond to the VoiceXML 2.0 built-in grammar types. For details on the required input formats for these types, see the VoiceXML 2.0 specification at *www.w3.org/TR/2001/WD-voicexml20-20011023/#dml2.3.1.1*
		voicexml:time	
		voicexml:boolean	
		voicexml:phone	
		voicexml:currency	
		voicexml:digits	
		voicexml:number	
<prosody>	—	—	Specifies prosodic information for the enclosed text. For all <prosody> attribute tags, if user-specified values are above or below the supported range of values, the maximum or minimum value is substituted for the user-specified value.
	pitch		Supported values: ■ Descriptive: ■ high (190) ■ medium (155) ■ low (140) ■ Absolute hertz: 50 to 300 ■ Relative hertz: -50 to +100 ■ Relative percentages: -30% to +30% ■ Relative semitones: +5st to -5st **Note:** If % or st is omitted, the value is interpreted as relative hertz. If - or + is omitted, the value is interpreted as absolute hertz, even if % or st is present. Default is 155

Table 2: Supported SSML 1.0 elements *(continued)*

Element	Attribute	Value	Notes
<prosody>	contour		Specifies the pitch contour for the contained text using the syntax: <prosody contour="(*position*,*target*) [(*position*,*target*) ...]" > Where: ■ *position* is the percentage of the period of the contained text, where "0%" corresponds to the beginning of the contained text and "100%" corresponds to the end. ■ *target* is the target pitch at that position. See the description of the pitch attribute for supported target values. For an example of using contour, see the "Prosody and contour" on page 38.
	range		Supported values: ■ Descriptive: ■ high (100) ■ medium (50) ■ low (10) ■ Absolute hertz: 0 to 100 **Note:** If relative values are specified, the attribute is ignored. Default is 50
	rate		Supported values: ■ Descriptive: ■ fast (80) ■ medium (100) ■ slow (130) ■ Fixed percentages: 75% to 150% **Note:** If the % character is omitted, the value is ignored. Default is 100

Table 2: Supported SSML 1.0 elements *(continued)*

Element	Attribute	Value	Notes
<prosody>	duration		Specifies milliseconds or seconds, for example: 3000ms, 3000 or 3s.
	volume		Supported values: ■ Descriptive: loud, medium, soft, silent ■ Fixed percentages: 0% to 100% **Note: Note**: If the % character is omitted, the value is ignored.
<audio>	src	*URI*	Lets you specify the audio files to be played and text to be spoken, using the *file:* or *http:* formats. Supports the following encoding: ■ For WAV files: ■ 8 kHz, mulaw, unheadered ■ 8 kHz, mulaw, Sphere header ■ 8 kHz, PCM, Windows WAV format ■ For AU files: ■ 8 kHz, mulaw (type 1), AU header ■ 8 kHz, PCM (type 3), AU header See "URI formats for the SSML <audio> element" on page 40.

SSML sample text

The following text demonstrates how SSML tags can be used to modify TTS output. If you are running Vocalizer in ssml mode, you can hear the results by copying the following text into the Offline Audio Generator (see Chapter 7).

Note: The text below can be copied in its entirety. If you want to test individual paragraph samples, make sure to enclose the text in a <speak></speak> block so that it is well-formed XML.

Say-as

```
<speak>This is a demonstration of the Vocalizer <say-as
type="acronym">TTS</say-as> engine, showcasing a range of SSML
compatible features.
```

Prosody and volume	<prosody volume="60%"> We can change </prosody> <prosody volume="100%"> the volume </prosody> <prosody volume="70%"> of ranges of words </prosody>.
Emphasis	We can also apply emphasis, for example: Do you like this one, or do you prefer <emphasis level="strong"> this </emphasis> one. Alternatively, we can add emphasis using a combination of volume pitch and duration markup flags. <prosody volume="50%"> For example, how do you like </prosody><prosody rate="150%" contour="(0,+0)(50,+50)(90,-10)" volume="100%"> this </prosody><prosody volume="50%" contour="(0,+0)(100,-10)"> one </prosody>.
Prosody, rate, and duration	We can change the duration of an utterance by using the <prosody rate="150%"><emphasis level="strong"> rate </emphasis></prosody> and <prosody rate="150"> duration </prosody> flags. For example, <prosody rate="150%"> This sentence is spoken 50% slower than usual </prosody>. <prosody duration="4s"> Whereas, this sentence is exactly 4 seconds in duration </prosody>.
Prosody, pitch, and phoneme	We can make absolute and relative pitch changes to the speech, for example, <prosody pitch="170"> this sentence is spoken with an average frequency of 170 hertz </prosody>. <prosody pitch="-20%"> This sentence's average frequency has been <phoneme ph="l o *r d"/> by 20 Hertz </prosody>.
Prosody and contour	An arbitrary pitch contour can be applied using the <emphasis level="high"> contour </emphasis> flag. <prosody contour="(0,-30)(90,+30)"> For example, this sentence interpolates a relative frequency change from 30 hertz below normal at the beginning to 30 hertz above normal at the end</prosody>.
Misapplied elements	Multiple target frequencies can be specified across the range of the contour flag, allowing an unskilled user to easily create prosodic monstrosities. For example, <prosody contour="(0,+20)(70, +0)"> question </prosody> type intonations can be badly <prosody rate="130%" contour="(0,+40)(50,-10)(100,+90)"> applied </prosody>. However, in skilled hands and applied responsibly, these flags provide a user with an impressive level of prosodic control.</speak>

Resolving XML parsing errors with predefined entities

When using Vocalizer in `vxml` or `ssml` mode, certain characters may cause XML parsing to fail so that no audio is rendered. For example, the Worldbet and XSAMPA phonetic alphabets include characters which may cause parsing errors if embedded within the `ph` attribute of the `<phoneme>` element.

To resolve parsing errors, use XML predefined entities to represent these special characters, as shown in the table:

Reserved character	Replace with predefined entity:
& (ampersand)	&amp;
“ (double quote)	&quot;
< (less than)	&lt;
> (greater than)	&gt;
‘ (apostrophe)	&apos;

As an example, the XSAMPA alphabet uses “`g_<`“ to indicate an implosive “g”. However, the “<” and “>” characters are reserved in XML-based markup languages, so the following text will be parsed incorrectly:

```
I like <phoneme ph="g_<oldfIS" alphabet="xsampa">goldfish
</phoneme>.
```

To resolve the parsing error, you would replace “<” with “`<`” as follows:

```
I like <phoneme ph="g_&lt;oldfIS" alphabet="xsampa">goldfish
</phoneme>.
```

See the W3C’s website for more information on predefined entities (*www.w3.org/TR/2000/REC-xml-20001006#dt-escape*).

Encoding

By default, Vocalizer markup processors expect any XML markup to be UTF-8 encoded, per RFC 2279. A request without an XML header must be encoded in UTF-8. For example:

```
<speak> This message should be in UTF-8 </speak>
```

For non-English languages with accented characters, text requires UTF-8 encoding of the ISO-8859-1 characters before being sent to Vocalizer. In SSML

mode, Vocalizer can also interpret ISO-8859-1 input character if you provide an appropriate XML header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<speak> This message is in iso 8859 1 format </speak>
```

For most English text, encoding is not an issue because of the direct correspondence between UTF-8 and 7-bit ASCII; characters numbered 0 to 127 in 7- or 8-bit ASCII are the same as the characters numbered 0 to 127 in UTF-8. However, if you enter non-English expressions in SSML mode, such as words like “coup d'état,” you must specify the encoding. Failure to enter the encoding causes unpredictable behavior. For example, entering text in the Offline Audio Generator with at least one accented character produces no output.

Caution: When using non-English voice packs and entering text in SSML mode, you must specify the encoding type, ISO-8859-1, to avoid unpredictable behavior. For example, entering text in the Offline Audio Generator with at least one accented character produces no output. See “Text encoding” on page 88.

The following encoding values are recognized by Vocalizer:

- `utf-8`
- `utf-16`
- `iso-8859-1`
- `us-ascii`

When using the `<phoneme>` element in SSML, if you select IPA as the alphabet, the IPA characters can be represented by multi-byte values only, and the entire document passed to Vocalizer must therefore be encoded using either UTF-8 or UTF-16.

URI formats for the SSML <audio> element

Vocalizer handles a specific set of *http:* and *file:* URI formats (per RFC s 2616 and 2396) for the VoiceXML `<audio>` element. For example, this table shows illegal formats:

URI	Error
C:/foo.wav	Invalid format per RFC 2396
file://C:/foo.wav	Uses only two forward slashes (/) instead of three between the colon (:) and the “C”. The correct form is: *file:///C:/foo.wav*

Table 3 and Table 4 demonstrate the correct formats for absolute and relative URIs, and show how these URI formats are handled by Vocalizer.

Some notes:

- You can use either "/" or "\" for path components
- The only authority Vocalizer handles for file URIs is `localhost` or `127.0.0.1`
- Syntax marked as Unix will not work on Windows; examples containing "C:" will not work on Unix
- The base URI for relative file URIs is:

  file://localhost/%VOCALIZER%/data/audioFiles/

  This means:

  - If the scheme is not specified it will be assumed to be *file:*
  - If the authority is not specified, it will be assumed to be `localhost`
  - If the path portion of the URI is relative, the file is searched for in the *%VOCALIZER%/data/audioFiles/* directory

Table 3: Absolute URIs

URI	Reference
file://localhost/C:/foo/bar.wav	*C:\foo\bar.wav*
file://localhost/foo/bar.wav	*/foo/bar.wav* (Unix)
file:///C:/foo/bar.wav	*C:\foo\bar.wav*
file:///foo/bar.wav	*/foo/bar.wav* (Unix)
file:/C:/foo/bar.wav	*C:\foo\bar.wav*
file:/foo/bar.wav	*/foo/bar.wav* (Unix)
http://localhost/bar.wav *http://audio-server/audiofiles/bar.wav* *http://audio-server:8080/audiofiles/bar.wav*	

Table 4: Relative URIs (*file:* format only)

URI	Reference
//localhost/C:/foo/bar.wav	*C:\foo\bar.wav*

Table 4: Relative URIs (*file:* format only)

URI	Reference
//localhost/foo/bar.wav	*/foo/bar.wav* (Unix)
///C:/foo/bar.wav	*C:\foo\bar.wav*
///foo/bar.wav	*/foo/bar.wav* (Unix)
C:/foo/bar.wav /	*C:\foo\bar.wav*
/foo/bar.wav	*/foo/bar.wav* (Unix)
foo/bar.wav	*%VOCALIZER%\data\audioFiles\foo\bar.wav*
./foo/bar.wav	*%VOCALIZER%\data\audioFiles\foo\bar.wav*
../foo/bar.wav	*%VOCALIZER%\foo\bar.wav*
bar.wav	*%VOCALIZER%\data\audioFiles\bar.wav*
./bar.wav	*%VOCALIZER%\data\audioFiles\bar.wav*
../bar.wav	*%VOCALIZER%\bar.wav*

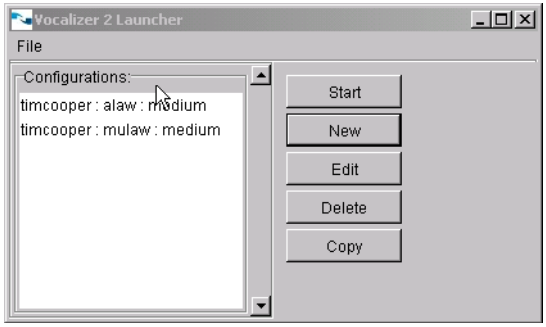
Starting Vocalizer through the Launcher

5

The Vocalizer Launcher is graphical tool that allows you to create, edit, and save start-up configurations for running Nuance Vocalizer.

Using the Vocalizer Launcher

From the Vocalizer Launcher interface, you can configure and start *vocalizer*, along with the *nlm* process. You can also start any of the other GUI tools provided with Vocalizer.



To use the Launcher:

1 From the Windows Start menu, go to Programs Nuance→ Nuance Vocalizer 3.0→ Executables→ Vocalizer Launcher

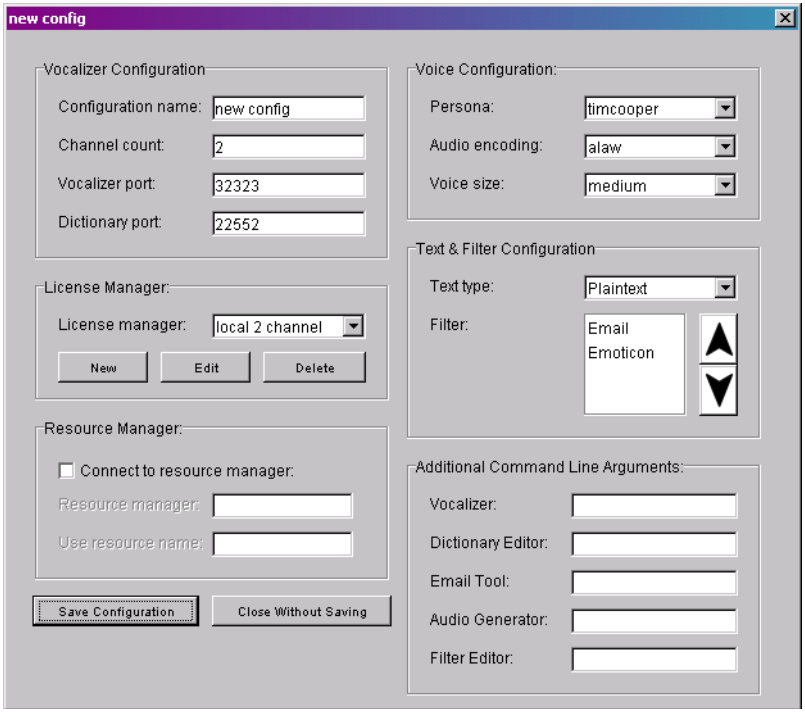
The Configurations window lists the voice packs and configurations available on your system.

2 After selecting a configuration, choose either:

- **Start**—Opens the Status window and starts *vocalizer* with the default options for the selected configuration.
- **New**—Opens the Configuration Editor for you to create a new configuration.
- **Edit**—Opens the Configuration Editor and allows you to modify the selected configuration.
- **Delete**—Removes the selected configuration.
- **Copy**—Duplicates an existing configuration, which you can then open and modify. This is useful if you want to create a similar configuration with only a few adjustments.

Configuration editor

The configuration editing window lets you specify your configuration options that you would normally set on the command line.



The window groups configurations options into these sections:

- Vocalizer configuration
  - Configuration name—Allows you to rename your configuration
  - Channel count—Allows you to enter the number of channels you want to use, up to the maximum specified in your license
  - Vocalizer port—Allows you to change the TTS port setting for audio generation
  - Dictionary port—Allows you to change the dictionary port setting for phonetic generation.
- License manager
  - License manager—Allows you to enter and select the license manager running your *nlm* process
  - New/Edit—Opens a dialogue window in which you can either specify a different license manager host and port to run your *nlm* process or start the local license manager
  - Delete—Removes the license manager running your *nlm* process
- Resource manager
  - Connect to a resource manager—Enables connection to a resource manager
  - Resource manager—Specifies hostname and port setting
  - Use resource name—Sets the `tts.ResourceName` parameter to the name by which your Vocalizer server identifies itself
- Voice configuration
  - Persona—Lists the voice packs available from which you can make a selection
  - Audio encoding—Lets you choose from mulaw or alaw audio encoding, depending on the voice pack installed
  - Voice size—Displays the type of amount of compression, depending on the voice pack installed
- Text and filter configuration
  - Text type—Lets you select Plaintext, VXML, or SSML
  - Filter—Allows you to select the text filter (or filters) you want to use and specify the order in which they are applied

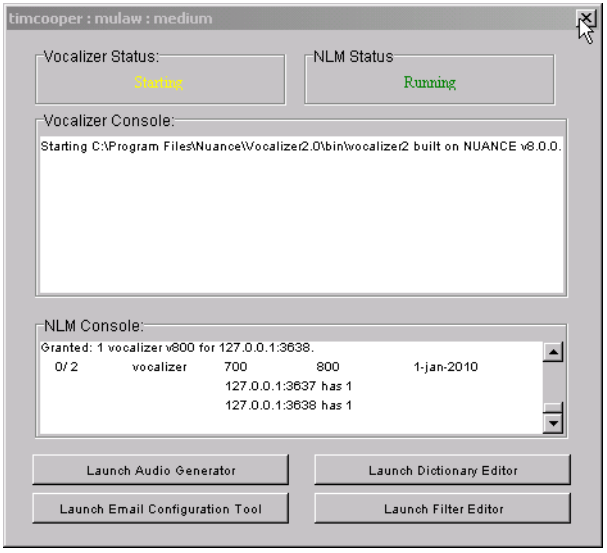
- Additional command-line options—Accepts a single command line option (with no white spaces). For example, you can enter the *-preallocated_licenses* or *-voice_from_disk* option.

  Note: Do not enter options that take arguments (such as "`-text_type ssml`") because white spaces are not accepted in this field.

You can name the set of configuration options and save them for reuse.

Status window

Selecting Start from the main window opens the Status window, which displays the current state of the *vocalizer* process.



Once the *vocalizer* process has been started, you can start these GUI tools:

- Offline Audio Generator—See Chapter 7
- Email Configuration Tool—See Chapter 6
- Filter Editor—See Chapter 6
- Dictionary Editor—See Chapter 8

Closing this window stops the *vocalizer* process.

Preprocessing text input

6

Vocalizer allows you to control the way email messages and other text input is processed. This chapter explains how to do this using the Email Reader Configuration tool and the Text Replacement Filter Editor.

Passing email messages to Vocalizer

Vocalizer can parse and read email messages encoded using the RFC 822 standard. To enable this feature, add the *-filter* option on the *vocalizer* command line:

```
-filter email
```

This allows you to pass email message text directly to Vocalizer.

Note: Vocalizer currently cannot parse email in Encrypted, MIME, or any type of 8- to 7-bit character encoding such as quoted-printable, UUEncode, or base 64 encoding.

Features of the Email Reader Configuration Tool

You can use the Email Reader Configuration Tool to configure the way email messages are converted to speech and save your configuration options. The email handler is able to switch between multiple configuration files.

The Email Reader Configuration Tool allows you to do the following:

- Load, edit, and save email configurations
- Specify which email fields to read and in what order
- Specify text to be spoken before and after each field

- Specify a maximum number of names to be read from the To and Copied fields
- Normalize email addresses and remove separator lines from signatures

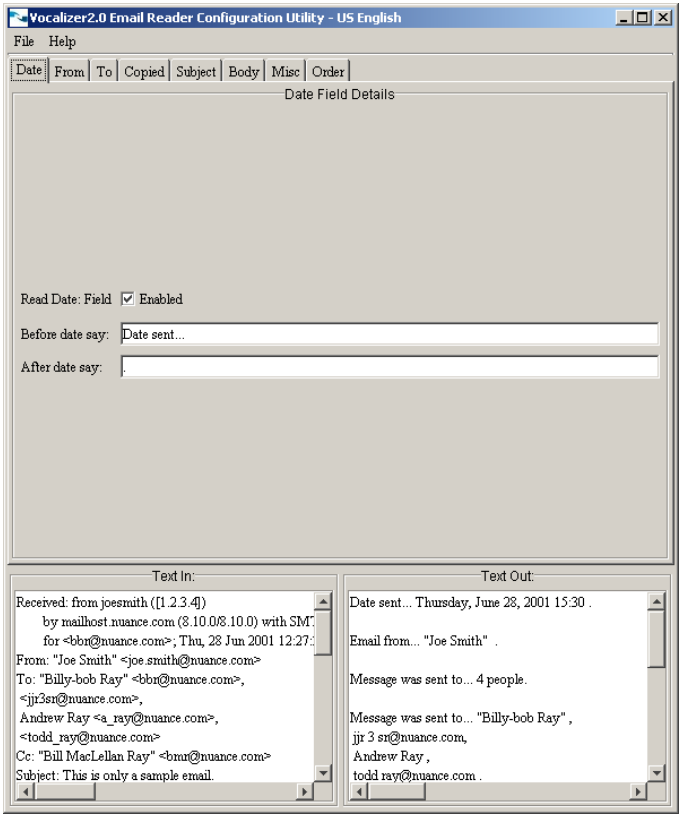
The Email Reader is a Java application built on Java 2. The Java Runtime Environment (JRE) 1.3 is installed with Vocalizer to allow you to run this utility. See “Third-party software requirements” on page 4.

Using the Email Reader Configuration Tool

To start this tool:

- **On Windows**, from the Programs group in your Windows Start menu, select Nuance→ Nuance Vocalizer 3.0→ Executables→ Email Reader Configuration Tool
- **On Solaris**, enter:

```
> EmailReaderConfigTool.sh
```



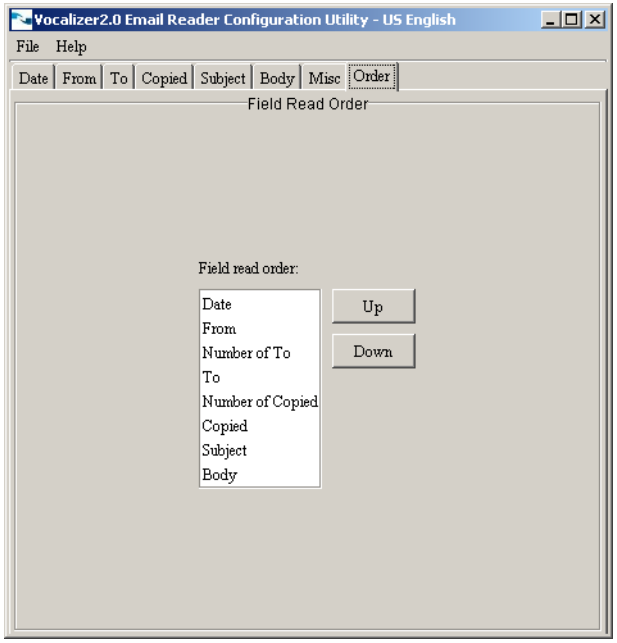
The Text In box displays a sample RFC 822 email. The Text Out box allows you to see the results of configuration choices, displaying how the email messages will be read. Use the tabs to choose which field you want to edit. You can see your updates in the Text Out box as you make your changes.

Specifying the fields to read

Under each tab is a checkbox labeled "Read *field_name* field." Check the Enabled box to specify whether or not you want information from that field included in the synthesized output. For example, under the Date tab in the previous graphic, "Read Date field" is enabled.

Specifying order

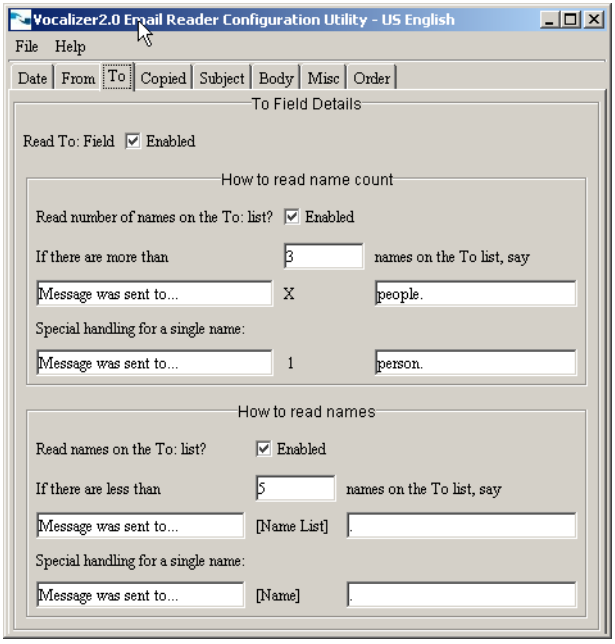
The Field Read Order list box lists all the email fields in the order they will be read. To change the order, select the field you want to move, then click the Up or Down button.



Specifying text output

Each field of the email can be prefaced or followed by user-defined comments. To edit this text, select the tab for the particular field, either Date, From, To, Copied, Subject, or Body. Enter your comments in the appropriate text box.

The To and Copied fields often include several names. You can decide if you want Vocalizer to tell you the total number of names listed, and if you want Vocalizer to read each name.



To avoid having a long list of names synthesized, you can specify a maximum number, *X*, so that Vocalizer only reads the list if there are fewer than *X* names. For example, if you set the "If there are less than" field to 5, the application reads all the names that are on the list *only* if there are fewer than five. If there are five or more names, nothing is read.

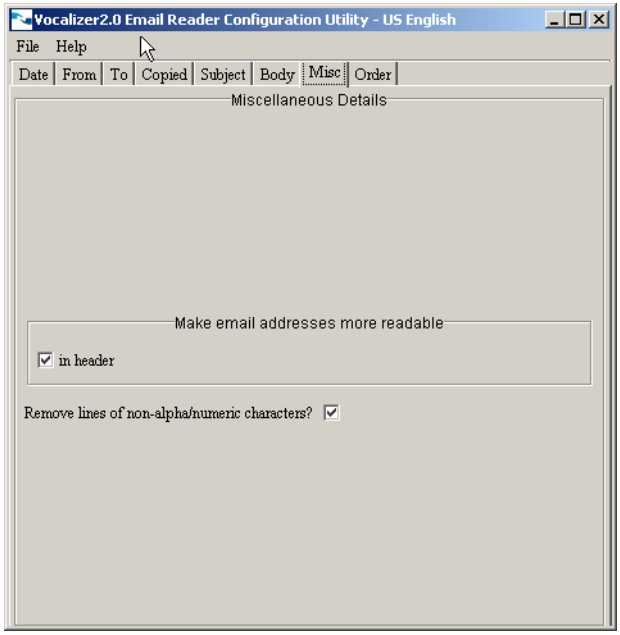
If you want Vocalizer to tell you the total number of names listed, specify a minimum number, *Y*, so that Vocalizer only tells you the total when there are more than *Y* names. For example, if you set the "If there are more than" field to 1, the application tells you the total number of names *only* if there is more than one name. If there is only one or zero names, nothing is said.

Address and signature handling

From the Miscellaneous tab, you can enable the settings for normalizing email addresses and signature handling.

Normalizing addresses uses white space to replace underscore characters (_) and to separate digits and characters within a strings. For example:

Address input	Normalized result
a_name@company.com	*a name@company.com*
ted3@hotmail.com	*ted 3@hotmail.com*



Some email signatures include a separator line. Check the "Remove non alpha/numeric characters?" to remove lines which do not contain alpha or numeric characters.

Distributing changes

You can edit the email configuration files for all your Vocalizer servers on a single development machine. When you save changes in the Email Reader Configuration Tool, the configuration file is compiled and put in the *%VOCALIZER%**data*\`locale`*email* directory. You can distribute this file to your other TTS server machines by copying the contents of the *%VOCALIZER%**data*\`locale`*email* directory to the same path on the machine(s) where you want to distribute the changes.

Creating custom filters for text replacement

The Text Replacement Filter Editor allows you to create custom filters to modify how specific text input is rendered by the TTS engine.

Text replacement filters allow you to specify a set of related translations, mapping text input to a replacement value. For example, the emoticon filter includes several replacements, such as:

Input	Replacement text
:-)	smiling face
;-)	winking face
:-(	sad face

Each filter is stored in a single file. There is no limit to the number of filters you can create and use.

To enable this feature, add the *-filter* option on the *vocalizer* command line:

```
lm.Addresses=localhost:8471 -filter your_custom_filter
```

Applying filters

You can specify multiple filters and choose the order in which they are applied. Enter the filter names using a comma-separated list with no spaces. For example, if you have a filter named *medical*, which expands medical acronyms, you can include it with the emoticon filter like this:

```
lm.Addresses=localhost:8471 -filter emoticon,medical
```

Filters are applied in the order in which they appear. The order specified can affect the output if replacement text generated by one filter matches a transform for a following filter. For example, if you input this text:

“My computer is broken. :-(“

the emoticon filter outputs:

“My computer is broken. sad face”

If your medical filter recognizes “sad” as an acronym to be expanded, then it would output:

“My computer is broken. Seasonal Affective Disorder face”

Reversing the order of the filters yields the following output:

"My computer is broken. sad face"

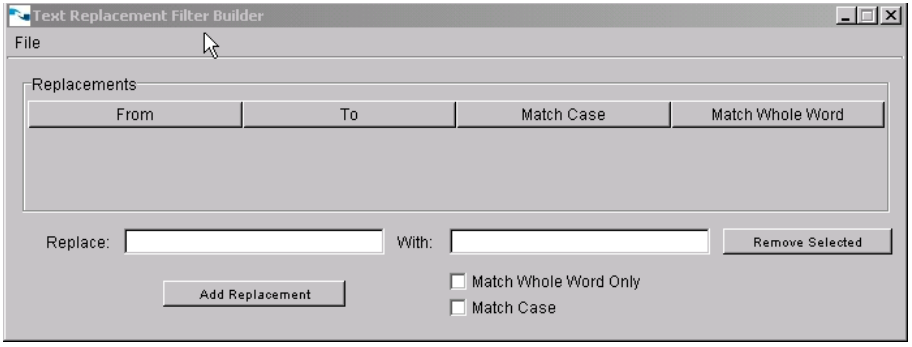
Using the Text Replacement Filter Editor

To use the Text Replacement Filter Editor:

1 Start the tool:

- **On Windows**, from the Programs group in your Windows Start menu, select Nuance→ Nuance Vocalizer 3.0→ Executables→ Text Replacement Filter Editor
- **On Solaris**, enter:

```
> ReplacementFilterEditor.sh
```



2 In the Replace text box, enter the text you want recognized. You can also choose to:

- Match whole word only

  By default, Vocalizer tries to match the text string as a word, and looks for word delimiters, including spaces, sentence terminators, hyphens, and parentheses. If the text string is found within another word, it is ignored.

  If you want the text string to always be replaced, uncheck this box. For example, if you create a filter to block obscenities, you would probably want the text filtered, even if it appears as part of other text strings.

- Match case

  By default, Vocalizer ignores the case for text entered. Selecting Match Case allows a case-sensitive search for the specified string.

3 In the With text box, enter the text you want Vocalizer to output.

4 Click Add Replacement.

When you're done, save the list of replacements as a file in the *%VOCALIZER%\data\replace-filter* directory. You can use the Text Replacement Filter Editor to open and edit any of your text replacement files.

Distributing changes

You can create the filter files for all your Vocalizer servers on a single development machine. After saving your changes in the Text Replacement Filter Editor, the filter files are stored in the *%VOCALIZER%\data\replace-filter* directory. You can distribute this file to your other TTS server machines by copying the contents of the *%VOCALIZER%\data\replace-filter* directory to the same path on the machine(s) where you want to distribute the changes.

Generating audio files from text input

7

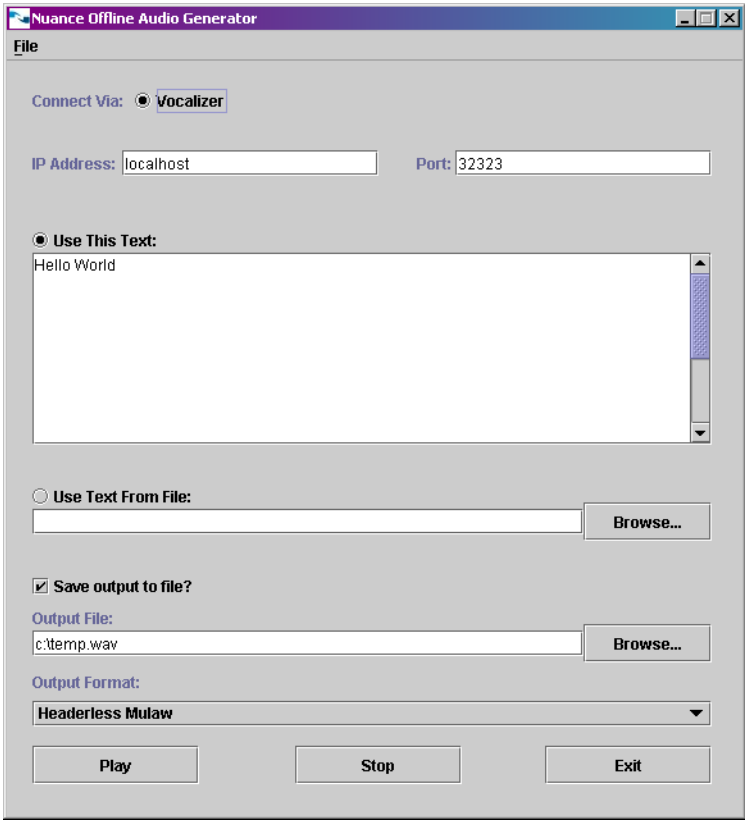
Vocalizer can generate audio streams directly from user input or from text files. The resulting audio stream can be saved to files for playback. This chapter explains how to do this using the Offline Audio Generator.

Using the Offline Audio Generator

The Offline Audio Generator is a GUI tool which accepts text input either directly from users or from text files.

The text input is sent to Vocalizer, which returns an audio stream. The audio is played back to the user and can be written to a file in one of these formats:

- Headerless mulaw
- Mulaw with Sphere Header (playable with Nuance's *playwav* utility)
- Headerless Linear
- Linear with Sphere Header (playable with Nuance's *playwav* utility)
- Linear Riff (Windows default *.wav* format)



To use the Offline Audio Generator:

1 Make sure the *vocalizer* process is running.

2 Start the tool, either using the Launcher (see Chapter 5) or:

- **On Windows**, from the Programs group in your Windows Start menu, select Nuance→ Nuance Vocalizer 3.0→ Executables→ Offline Audio Generator
- **On Solaris**, enter:

```
> OfflineAudioGenerator.sh
```

The GUI will appear.

3 Enter your IP address and the TTS port number. (The default TTS port is 32323).

4 Specify the text to be rendered by selecting either:

- Use this text—Enter text in the input field
- Use text from file—Select a text file through the Browse button

5 Choose whether or not to save audio output to a file. If you select Save output to file:

   a Specify the file to which audio output should be written, by either entering the filename in the output file field or selecting an existing audio file to overwrite.

   b Specify the audio format for saving output from the drop-down menu.

   By default, audio files are generated as *.wav* files. Choose the Sphere headered for use with Nuance applications, or headerless formats for use with other audio editors.

6 Click Play.

   You can stop playback at anytime by clicking the Stop button. The Exit button also stops playback before quitting.

Validating multi-server configurations

If you want to test the Vocalizer installation on other servers in your configuration, enter the IP address of the appropriate machine. The text is sent to the Vocalizer server on that machine.

Chapter 8

Customizing your application's dictionary

8

Vocalizer includes the Dictionary Editor, a graphical tool that allows you to dynamically modify the dictionary files used by the TTS engine.

This chapter describes how to use the Dictionary Editor to add words and acronyms to your dictionary, or modify the pronunciation of existing entries. It also explains how to distribute changes made on one machine to other Vocalizer servers.

Features of the Dictionary Editor

The Dictionary Editor is a Java application built on Java 2. The Java Runtime Environment (JRE) 1.3 installed with Vocalizer allows you to run this utility. See "Third-party software requirements" on page 4.

The Dictionary Editor's main menu includes File and Options:

- From the File menu you can either:
  - Save Changes Locally—Saves your modifications to your machine only
  - Save Changes to Server—Compiles your changes and uploads them to Vocalizer

  Note: Selecting Save Changes to Server blocks all TTS requests during the time it takes to compile the changes, so this option should not be used during peak periods. You can only use this option if the edited dictionary files are for the same language as the voice pack you are running.
- From the Options menu, you can either:
  - Modify the port settings for Vocalizer audio generation. The default TTS port is 32323, and the default port for phonetic generation is 22552.

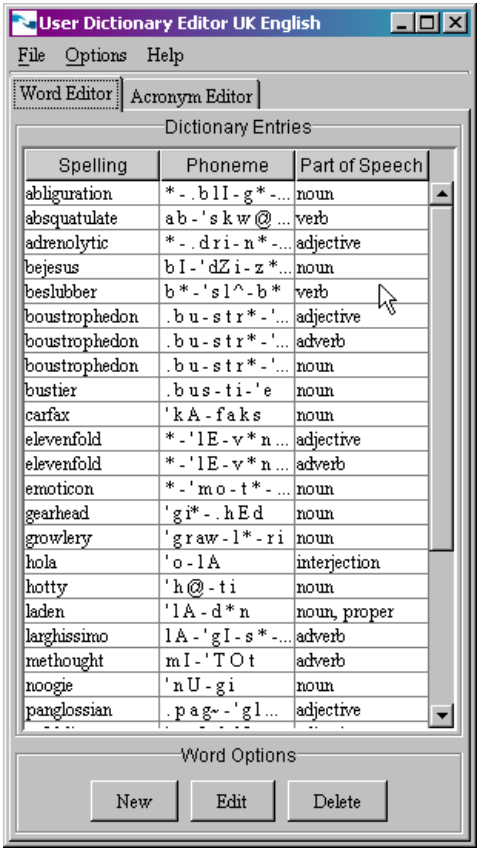
- Select the dictionary file to modify according to language, either:
  - North American English
  - U.K. English
  - Australian/New Zealand English

  By default, North American English language dictionary files are loaded.

  Because dictionary files are language-specific, if you are running a North American English voice, for example, but modify the U.K. English dictionary, you can save your changes, but you will only hear the effect the next time you run a U.K. English voice pack.

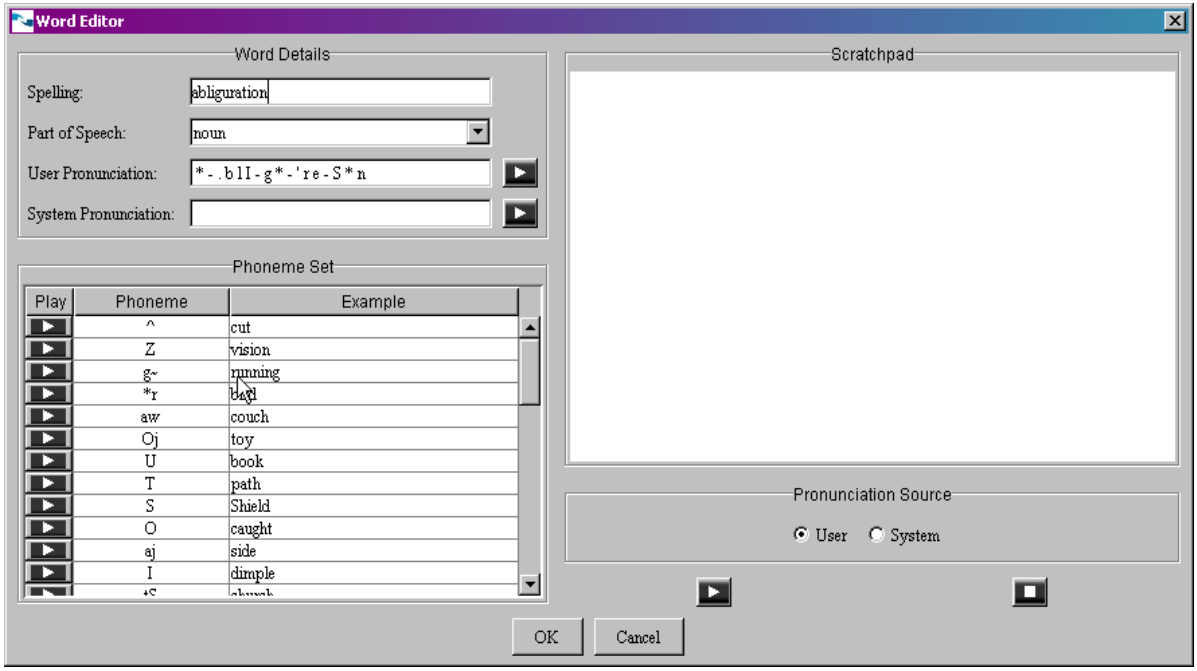
The main window of the Dictionary Editor has two tabbed panes:

- **Word Editor**—Lets you view the current dictionary entries, add new words, modify the way existing entries are pronounced, or remove entries
- **Acronym Editor**—Lets you view all acronyms known to Vocalizer in the dictionary entries, enter new acronyms, modify the way existing acronyms are expanded, or remove entries



Depending on which tabbed pane you have selected, clicking New or Edit in the Word Options section opens either the Word Editor or Acronym Editor dialogue window. Selecting Delete removes whichever entry is currently selected in the Dictionary Entries.

Note: If you close the Dictionary Editor without saving, your changes will not take effect.



Audio feedback

Vocalizer provides immediate feedback on your entries in the Word Details and Scratchpad sections.

The System Pronunciation field displays the current phonetic translation for a selected word, and you can use the Play button next to that field to hear the TTS engine's pronunciation. When you change the pronunciation of an existing entry, you can hear your modified version, using the Play button next to the User Pronunciation field.

The Scratchpad lets you enter a text string using a dictionary entry, then play either the user-specified pronunciation or the system's pronunciation in that context. See "Audio scratchpad" on page 64.

Word details

The Word Details section defines the usage and phonetic characteristics of your entry through the following fields:

- Spelling—Displays the entry
- Part of speech—Displays the entry's linguistic role
- Say As (in the Acronym Editor only)—Displays the text expansion for acronyms
- User Pronunciation—Displays the user-defined phonetic translation
- System Pronunciation—Displays the system's phonetic translation

Phoneme set

Pronunciation is built with phonemes. Vocalizer's phonetic translation is based on the Computer Phonetic Alphabet (CPA). The CPA is also used by the Nuance System to build pronunciations in Nuance recognition dictionaries.

The Phoneme Set contains a complete list of phonemes, along with stress markers and a syllable separator, which you can use to build pronunciations for your entries. You can use the Play button beside each phoneme to hear the system's pronunciation and determine your selection. Clicking on the phoneme inserts it at the current cursor position of the User Pronunciation.

Auto-phonetic transcription

When you enter new words and new expansions for acronyms in the Say As field, Vocalizer automatically provides a phonetic translation for your entry. Based on the TTS engine's pronunciation, you can modify the phonetic translation appropriately.

Inserting stress markers

In addition to the phonetic translation, your pronunciation definition can include stress markers for improved prosody:

Marker	Symbol
Primary Stress	Single quote (')
Secondary Stress	Period (.)
Syllable separator	Dash (-)

Enter stress markers immediately preceding the syllable to be accented. For example:

Word	Pronunciation encoding
emoticon	i ' m o - t I . k A n
laden	' l A - d * n

Audio scratchpad

The Scratchpad lets you send text to Vocalizer to test pronunciations in the context of your application. You can enter text strings in which your modified entries may be used, then click the Play button below the Scratchpad to hear your text synthesized.

Choosing either User or System determines if the Scratchpad plays your entry according to the system's pronunciation or the phonetic translation you've entered.

See "Testing your entries in Scratchpad" on page 67.

Using the Dictionary Editor

To use the Dictionary Editor:

1 Make sure the *vocalizer* process is started.

2 Start the Dictionary Editor:

- **On Windows**, from the Programs group in your Windows Start menu, select Nuance→ Nuance Vocalizer 3.0→ Executables→ Dictionary Editor
- **On Solaris**, enter:

```
> DictionaryEditor.sh
```

Note: Changes are not saved automatically when you close the editing tool. After following the procedures for adding or modifying dictionary entries, make sure to save your changes before closing the Dictionary Editor.

Enabling logging

You can edit the *Dictionary.bat* file to enable debug information to be sent to a log file (*dictionaryEditor.log*) in the *%VOCALIZER%\bin\win32* directory:

1 From the *%VOCALIZER%\bin\win32* directory, open *Dictionary.bat* as a text file.

2 Before `-D"INSTALL_PATH"`, add the parameter setting `-D"debug=true"`. For example:

```
"C:\Program Files\JavaSoft\JRE\1.3\bin\java" -jar -cp
"C:\Program Files\Nuance\Vocalizer3.0\java"
-D"debug=true"
-D"INSTALL_PATH=C:\Program Files\Nuance\Vocalizer3.0"
"C:\Program Files\Nuance\Vocalizer3.0\java\
DictionaryEditor.jar"
```

Adding new entries

To add a new word or acronym to the dictionary:

1 In Word Options, click New to open the Word Editor or Acronym Editor.

2 In the Spelling field, enter the new word or acronym.

Make sure to enter the correct spelling for text transcriptions sent to the TTS engine; dictionary entries are not case-sensitive.

If you are entering a new word, the User Pronunciation field will automatically display a phonetic translation based on the text in the Spelling field.

3 From the Part of Speech drop-down menu, make the appropriate selection, for example, “noun, proper” or “adverb.”

4 If you are entering a new acronym, in the Say As field, enter the expansion. For example, with the acronym “TTS” you would enter “text-to-speech.”

The User Pronunciation field automatically displays a phonetic translation based on the text in the Say As field.

5 In the User Pronunciation, build or modify the phonetic pronunciation for your word:

a This field is filled automatically, but you can modify the phonetic translation using the Phoneme Set. See “Phoneme set” on page 63.

b Add Primary or Secondary Stress symbols *before* the syllable to be accented; see “Inserting stress markers” on page 63.

6 Press the Play button next to User Pronunciation or use the Scratchpad to test your phonetic translation; see "Testing your entries in Scratchpad" on page 67.

7 When you are happy with the pronunciation, click OK to add the word to the dictionary.

8 Save your changes. From the File menu, select either:

- Save Changes Locally—Saves your changes, but does not dynamically update Vocalizer.
- Save Changes to Server—Saves and dynamically updates Vocalizer.

  You can only use this option if the edited dictionary files are for the same language as the voice pack you are running.

Modifying dictionary entries

To modify the pronunciation associated with an existing dictionary entry:

1 Select one of the Dictionary Entries:

- From the Word Editor, you can modify word pronunciation
- From the Acronym Editor, you can modify acronym expansion and pronunciation

2 In Word Options, click Edit to open up the Word Editor (or Acronym Editor).

3 Modify the appropriate Word Details, as described in "Adding new entries" on page 65.

4 Click OK when you are done.

5 Save your changes. From the File menu, select either:

- Save—Saves your changes, but does not dynamically update Vocalizer.
- Save and Upload—Saves and dynamically updates Vocalizer.

  You can only use this option if the edited dictionary files are for the same language as the voice pack you are running.

Testing your entries in Scratchpad

When you add or edit a dictionary entry, you can use the Scratchpad to test your modifications:

1. Select a dictionary entry to edit; phonetic translation for the entry appears in both the User Pronunciation and System Pronunciation fields.
2. Edit the User Pronunciation.
3. In the Scratchpad, enter a text string including the word you have edited.
4. In the Pronunciation Source, select User or System.
   - Select User to play your text string using the modified pronunciation; if you've changed the Say As field for an acronym, Scratchpad plays the modified expansion.
   - Select System to play your text string using the system's version of that entry's pronunciation.

Deleting entries

To delete from the Dictionary Entries:

1. In Dictionary Entries, select the appropriate word or acronym.
2. In Word Options, select Delete.
3. Save your changes. From the File menu, select either:
   - Save—Saves your changes, but does not dynamically update Vocalizer.
   - Save and Upload—Saves and dynamically updates Vocalizer.

     You can only use this option if the edited dictionary files are for the same language as the voice pack you are running.

You may be able to increase synthesis time by removing unused words from your dictionary.

If you remove all Dictionary Editor entries, all pronunciations will be determined solely by the TTS engine.

Distributing dictionary changes

You can edit the user dictionaries for all your Vocalizer servers on a single development machine. When you save changes in the Dictionary Editor, the dictionary files are compiled and put in the *%VOCALIZER%**data*\`locale`*Lexicon* directory. You can distribute these files to your other server machines as follows:

1. From the machine on which your edits were made, copy the contents of the *%VOCALIZER%**data*\`locale`*Lexicon* directory to the same path on the machine(s) where you want to distribute the changes.
2. Optional. The dictionary source files are in the *%VOCALIZER%**data*\`locale`*dictionary* directory. If you wish to keep the source files on each of the machines to which changes were distributed, copy the contents of *%VOCALIZER%**data*\`locale`*dictionary* to the same path on the destination machine(s). These files are not used during regular Vocalizer server execution.

Techniques for enhancing audio output

9

The layout, punctuation, and capitalization of text sent to Vocalizer affect speech output. Certain types of characters and phrases—for example, digits, dates, and email addresses—are handled in particular ways. By changing the way you transcribe phrases and characters, you can alter the speech synthesis. This chapter outlines how Vocalizer processes your text input.

This information applies to text processing for the English language in North American, U.K., and Australian/New Zealand dialects. Note that date formats for North America differ from U.K. and Australian/New Zealand formats. See "Dates" on page 84.

For information on text processing for Canadian French, see Appendix A.

Phrasing and punctuation

Vocalizer recognizes punctuation used to end phrases, sentences, and paragraphs. Vocalizer strips all layout from input text and uses it, in conjunction with punctuation, to determine phrasing, thereby enabling natural-sounding intonation and pausing at the ends of phrases, sentences, and questions. Text that is laid out and punctuated in a natural fashion, as it would be for normal written text, will be read in a natural predictable way.

TIP: For the most natural-sounding speech, try to break very long sentences into multiple sentences. Sending input such as "turn left and go 1 mile turn right and go 2 miles," will not sound right because there is no punctuation in the input text.

Table 5 explains how Vocalizer interprets punctuation:

Table 5: Interpretation of punctuation

Punctuation mark	Interpretation
▪ Comma (,) ▪ Semicolon (;) ▪ Colon (:)	End of phrase
Either a: ▪ Period (.) ▪ Question mark (?) ▪ Exclamation mark (!) ▪ Ellipsis (...) Followed by a single space	End of sentence
Period (.)—not immediately following an abbreviation or a single character—followed by a single space See “Hyphens and dashes” on page 71	End of sentence
Either a: ▪ Period (.) ▪ Question mark (?) ▪ Exclamation mark (!) ▪ Ellipsis (...) *Followed* by a: ▪ Single quote (‘) ▪ Double quote (“) ▪ Closing parenthesis ”) ” *and* either a single space, a new line, or the end of input	End of sentence
Period (.) preceded by a non-alphabetic character	End of sentence
Two or more consecutive new lines	End of paragraph
End of a sentence followed by a new line *and* either another new line, a tab, or a space	End of paragraph
End of input	End of paragraph

Note: The last character in a sentence does not necessarily have to be a sentence terminator. In some cases, sentences are enclosed inside quoted or parenthesized sections, for example:

"Then he said: 'Sorry, I can't do that'."

Periods

Periods can be used as sentence terminators, abbreviation markers, decimal points, IP address dividers, and date dividers. Vocalizer determines the correct interpretation from the context:

- **Sentence terminators**—See Table 5 on page 70
- **Abbreviation markers**—The period is considered an abbreviation marker if it has not been identified as a sentence terminator and it follows either a single character or a known abbreviation. See "Abbreviations" on page 73.

Note: As a general rule, allow two spaces to follow a sentence terminator, and use a period and an optional space to terminate an abbreviation.

- **IP address dividers**—See "Periods in digit sequences" on page 78
- **Decimals**—See "Decimal fractions" on page 79
- **Date dividers**—See "Dates" on page 84

Hyphens and dashes

Since hyphens, dashes, and minus signs are represented by the same ASCII character, ambiguous situations may occur at times. To avoid ambiguous situations, Vocalizer uses a set of disambiguation rules. Specifically, Vocalizer considers the "-" character to be:

- A *minus sign* if it is immediately followed by a digit.

  When a minus sign is identified, it is replaced by the word "minus."

- A *dash* if it has a whitespace on either side or if it is the first character in a sentence.

  The dash is removed, and the output is marked to indicate the punctuation.

- A *soft hyphen* if it follows a letter and is followed by a carriage return and a letter.

  The words on either side of the hyphen are joined and handled as a single word.

- A *hard hyphen* if it is used in any other situation, for example, if it appears within a word.

  Hyphenated words are split into constituent words. Also, the fact that hyphenated words are part of a single unit is recorded by the system for later reference and processing.

TIP: A hyphenated score, such as a football score of 37-27, should be sent to Vocalizer as "37 to 27" or "37, 27"so the output is not rendered as "thirty seven minus twenty seven."

Parentheses and double quotes

Vocalizer handles parentheses and double quotes. If a closing parenthesis or double quote occurs without any preceding matching symbol, for example, an open parenthesis, Vocalizer reads the close parenthesis or double quote as text. If two double quotes appear consecutively, they will also be read as text. For example:

Input text	Output text
(hi)	hi
hi)	hi parenthesis
"hi"	hi
""	quote quote

Apostrophes and single quotes

Since apostrophes and single quotes are represented by the same ASCII character, ambiguous situations may occur at times. For example, consider a case where two or more words ending in "s" followed by an apostrophe (') character are found consecutively in the same sentence:

'Sorry, we do not deal with customers' complaints.'

In cases such as this one, Vocalizer considers the last (') character as the closing of a quoted section (if previously opened) and the remaining (') characters as apostrophes.

To avoid ambiguous situations, Vocalizer uses the following set of disambiguation rules regarding the (') character:

- An *apostrophe* if the character appears within a word
- An *open single quote* if the character is the first character of a word

- A *closing single quote* if the character is the last character of a word whose last letter is not "s"
- An *apostrophe* if the character is the last character of a word whose last letter is "s" and a quoted section has not been opened previously
- Ignored in all other cases

Abbreviations

Abbreviations may consist of one or more alphabetic characters, which are either all lowercase, all uppercase, or have an uppercase first letter followed by lowercase letters. You can separate abbreviations from words by adding:

- A period and a space—For example, *Vocalizer, v. 3*
- A period only—For example, *Vocalizer, v.3*
- A space only—For example, *Vocalizer, v 3*

Single-letter abbreviations, ambiguous abbreviations, homographic abbreviations, and measurement abbreviations require additional work, as described in the following subsections for detailed information.

Single-letter abbreviations

Ambiguous situations may occur between single-letter abbreviations and initials. Where a possible single-letter abbreviation is found, Vocalizer interprets the letter as:

- An *abbreviation* if it is followed by one or more digits and it appears in the list of abbreviations
- An *abbreviation* if it follows one or more digits and it is a measurement or numeric abbreviation
- An *initial* in all other cases

Ambiguous abbreviations

Ambiguous abbreviations and their long form must be specified in the abbreviations file. At present, Vocalizer only disambiguates titles and road types, for example St. (saint or street) and Dr. (doctor or drive). Vocalizer resolves ambiguity by the following set of rules in the order prescribed:

1. The abbreviation is a *title* if it is the first word in a sentence.
2. The abbreviation is a *road type* if it is followed by a comma, a semi-colon, a colon, or a period.
3. The abbreviation is a *title* if it follows a comma, a semi-colon, or a colon.
4. The abbreviation is a *title* if the previous word is a road type.
5. The abbreviation is a *title* if the first letter of following word is capitalized.
6. The abbreviation is a *road type* if the first letter of the previous word is capitalized.
7. The abbreviation is a *title* in all other cases.

Homographic abbreviations

Homographic words are words that are spelled the same way but have different meaning, for example, the noun “lead” and the verb “lead”. *Homographic abbreviations* are abbreviations that are spelled the same way as a word or letter but have different meaning, for example, the abbreviation “Sat.” and the verb “sat” or “in” and “in.” (inch).

Vocalizer expands homographic abbreviations if they are followed by a period and a single space, or a period and a punctuation mark, for example:

Input text	Output text
sat. 4th dec	saturday fourth december
sat.. Then he went	saturday (pause) then he went
he sat. Then he went	he sat (pause) then he went

Homographic abbreviations can also be acronyms if they are written with at least one uppercase letter (not including the first letter of the word). For example, “SUN” and “sUn” can either be the abbreviation for “Sunday” or an acronym.

As mentioned earlier, Vocalizer expands homographic abbreviations if they are followed by a period and a single space, or a period and a punctuation mark, for example:

Input text	Notes
SUN computers	Interpreted as an acronym
SUN. 1st June	Converted to “Sunday” because ”SUN” is followed by a period and a single space
we are coming on SUN.. Computers are here.	Converted to “Sunday” because ”SUN” is followed by two periods
sun.1st June	Converted to “Sunday” because ”sun” is all lowercase

TIP: Be careful when creating a custom text replacement filters that replace homographic abbreviations. For example, a filter that replaces “me” with “Maine” could incorrectly render the word “me” when encountered in regular text.

Measurement and numeric abbreviations

Measurement and numeric abbreviations are only recognized if they follow a digit (with or without a space), for example, “1M” and “4 cm”.

With measurement abbreviations, the following rules apply:

- If the preceding word is the number “1” (entered as a digit), Vocalizer uses the singular version of the long form; otherwise, it uses the plural version.
- When a number is followed by an alphabetic character, Vocalizer checks whether the characters are specified in the list of measurement abbreviations:
  - If the characters are specified, it expands the abbreviation to its long form.
  - If the characters are not specified, Vocalizer treats the characters as separate letters.

For example:

Input text	Output text
3 kg	three kilograms
3 cm	three centimeters

Input text	Output text
1M	one million
1m	one meter
5yt	five y t

Acronyms and initials

Vocalizer also handles acronyms and initials. Vocalizer defines acronyms and initials as follows so that the pronunciation tool can treat them appropriately.

- A string of letters including one or more uppercase letters (not including the first letter)
- A string of letters separated by periods
- A single letter, except where the letter is an abbreviation or “I”, “i”, “A”, or “a”

  Since “I”, “A”, and “a” can be considered as words or letters, Vocalizer distinguishes them as follows:

  - “I”, “A”, or “a” is a *letter* if followed by a period and a single space
  - “A” is a *letter* if capitalized and not the first word in the sentence
  - “I”, “A”, and “a” are considered *words* in all other cases

Note: Periods are removed from the acronyms.

Note on capitalization

Vocalizer’s sophisticated handling of capitalized characters can affect interpretation of input if words are capitalized incorrectly. When sending proper names or other words through the system, make sure they are sent to Vocalizer the way they would appear normally, with appropriate capitalization, rather than using all capital letters.

TIP: When you know in advance that text input includes the names of places, make sure you capitalize them appropriately. Sending “New York” instead of “new york” gives Vocalizer a better chance of rendering the name successfully. This rule also applies to your custom text replacement filters (see “Creating custom filters for text replacement” on page 52).

Numbers

Vocalizer has a set of defined conventions for handling digits and common components that contain numbers such as dollar amounts and telephone numbers. Use the rules defined in this section to ensure the most accurate and natural synthesis for specific types of numbers and number-based phrases.

Cardinals and ordinals

Vocalizer converts cardinal numbers (which indicate quantity) and ordinal numbers (which indicate position in a series) to their textual form.

Using a comma within the number does not change speech output:

Input text	Output text
56,734	fifty six thousand seven hundred and thirty four
56734	fifty six thousand seven hundred and thirty four
1st	first
345th	three hundred and forty fifth
1,103rd	one thousand one hundred and third

However, if the comma is followed by one or more spaces, Vocalizer treats the digits as separate numbers:

Input text	Output text
56, 734	fifty six (pause) seven hundred and thirty four
1, 103rd	one (pause) one hundred and third

Digit sequences

Digit sequences are spoken in groups based on the way you transcribe them. To have digits voiced individually instead of as a single number, include either:

- A space between digits
- A comma followed by a space
- A period followed by a space

These examples show how various digit strings are converted to speech:

Input text	Output text
1234567	one million two hundred thirty four thousand five hundred sixty seven
1 2 3 4 5 6 7	one two three four five six seven (No pauses)
1, 2, 3, 4, 5, 6, 7	one two three four five six seven (Short pauses)
1. 2. 3. 4. 5. 6. 7	one two three four five six seven (Long pauses)

Periods in digit sequences

When periods (not followed by a space) are used as separators in a digit sequence, Vocalizer interprets the string as either a date, if the sequence makes sense as a date, or an IP address:

Input text	Output: North American format	Output: U.K. and Australian/New Zealand format
1.20.2000	the twentieth of january two thousand	one point two zero point two zero zero zero
1.32.1999	one point three two point one nine nine nine	one point three two point one nine nine nine
13.1.2001	thirteen point one point two zero zero one	the thirteenth of january two thousand and one

For more information, see "Dates" on page 84.

Leading zeros

Leading zeros in a number are omitted from the speech output:

Input text	Output text
0123	one hundred and twenty three
01	one

Note: The digit "0" is always said as "zero."

Fractions

Vocalizer recognizes fractions, where numerators and denominators are separated by a forward slash, with no intervening spaces:

Input text	Output text
1/2	one half
3/4	three quarters
42/357	forty two three hundred and fifty sevenths
1 / 2	one slash two

Decimal fractions

Vocalizer reads decimal numbers naturally, incorporating the decimal point in the phrase.

To ensure decimal values between 0 and 1 are read correctly, you must use a leading zero. There is no limit to the number of permissible decimal places. For example:

Input text	Output text
24,563.75	twenty four thousand five hundred and sixty three point seven five
3.141	three point one four one
0.76	zero point seven six
.50	dot fifty

If the input text begins with a decimal point and includes letter characters, the decimal point is read as "dot" and the digits are read as a single number. If the decimal point is internal, it is omitted from the output. For example:

Input text	Output text
.9k	dot nine k
p.50	p fifty

Percentages

The percent sign ("%") is always read as "percent." Space between the number and the percentage sign does not matter. For example:

Input text	Output text
75%	seventy five percent
75 %	seventy five percent
99.9%	ninety nine point nine percent
0.99%	zero point nine nine percent

Equations

Vocalizer correctly reads equations containing arithmetic operators such as plus ("+"), minus ("-"), and equals ("="):

Input text	Output text
3+4	three plus four
18-4=14	eighteen minus four equals fourteen

Account and social security numbers

To have account numbers or social security numbers read in segments instead of as a single digit, you must include hyphens between the segments:

Input text	Output text
555-00-9800	five five five zero zero nine eight zero zero
5405-9870-8000	five four zero five nine eight seven zero eight zero zero zero

Combining letters and numbers

Combining letters and numbers in a single words generates varied results. Typically the characters are read individually and the digit groups are treated as a single number:

Input text	Output text
h1	h one

Input text	Output text
hello123	h e l l o one two three
45z45	four five z four five
23b	twenty three b

Currency

Vocalizer recognizes the following currency symbols:

- Dollar sign ($)
- Pound sterling sign (£)
- Euro symbol (€)
- French franc abbreviation (FF)
- Deutschmark abbreviation (DM)

Numbers preceded by a recognized currency symbol (possibly with intervening spaces) are interpreted as currency amounts and expanded appropriately, as shown:

Input text	Output text
$12.34	twelve dollars and thirty four cents
$ 12.34	twelve dollars and thirty four cents
£12.34	twelve pounds and thirty four pence
£ 12.34	twelve pounds and thirty four pence
€12.34	twelve euros and thirty four cents
€ 12.34	twelve euros and thirty four cents
FF12.34	twelve francs and thirty four centimes
FF 12.34	twelve francs and thirty four centimes
DM12.34	twelve deutschmarks and thirty four pfennigs
DM 12.34	twelve deutschmarks and thirty four pfennigs
$1599.99	one thousand five hundred and ninety nine dollars and ninety nine cents

Input text	Output text
$2 million	two million dollars
$1000.896	one thousand dollars and eighty nine point six cents
$0.99	ninety nine cents
£.99	ninety nine pence
-$.99	minus ninety nine cents
-$1.95	minus one dollar and ninety five cents

Telephone numbers

Vocalizer recognizes local, national, and international telephone number formats used in North America and the United Kingdom, including area codes, access prefixes, and the international indicator prefix (+). If a three-digit segment of the number ends with exactly two zeros, it is read as "hundred."

The digits are grouped as they are presented, in order that pauses may be introduced:

Input text	Output text
North American formats	
847-5900	eight four seven (pause) five nine zero zero
847 9999	eight four seven (pause) nine nine nine nine
800 9999	eight hundred (pause) nine nine nine nine
650-847-9999	six five zero (pause) eight four seven (pause) nine nine nine nine
650.847.9999	six five zero (pause) eight four seven (pause) nine nine nine nine
[650]847.9999	six five zero (pause) eight four seven (pause) nine nine nine nine
(800)555-0202	eight hundred (pause) five five five (pause) zero two zero two
1-800-555-1212	one eight hundred (pause) five five five (pause) one two one two
1 888 847 9999	one eight eight eight (pause) eight four seven (pause) nine nine nine nine
1888 847 9999	one eight eight eight (pause) eight four seven (pause) nine nine nine nine
+1 888 847 9999	plus one eight eight eight (pause) eight four seven (pause) nine nine nine nine

Input text	Output text
U.K. formats	
456 161	four five six (pause) one six one
456161	four five six one six one
222 1234	two two two (pause) one two three four
8222 1234	eight two two two (pause) one two three four
020 8222 1234	zero two zero (pause) eight two two two (pause) one two three four
(020) 8222 1234	zero two zero (pause) eight two two two (pause) one two three four
01603 456 161	zero one six zero three (pause) four five six (pause) one six one
(01603) 456 161	zero one six zero three (pause) four five six (pause) one six one
+44 20 8222 1234	plus four four (pause) two zero (pause) eight two two two (pause) one two three four
Australian/New Zealand formats	
9818 7882	nine eight one eight (pause) seven eight eight two
02 9818 7882	zero two (pause) nine eight one eight (pause) seven eight eight two
0407 060 757	zero four zero seven (pause) zero six zero (pause) seven five seven
+61 2 9818 7882	plus six one (pause) two (pause) nine eight one eight (pause) seven eight eight two
+61 407 060 757	plus six one (pause) four zero seven (pause) zero six zero (pause) seven five seven

Addresses

Addresses are handled as follows:

Input text	Output text
1380 Willow Road	thirteen eighty willow road
180 Park Avenue	one hundred and eighty park avenue
42 St James's St	forty two saint james's street

Zip codes

A five-digit number, optionally followed by a hyphen and a four-digit number, is treated as a zip code if it is immediately preceded by the two-character postal

abbreviation of a state name. The numbers in a zip code are always read individually. For example:

Input text	Output text
94025	ninety four thousand and twenty five
CA 94025	california nine four zero two five
Menlo Park, CA 94025	menlo park california nine four zero two five
Florham Park, NJ 07932-0971	florham park new jersey zero seven nine three two zero nine seven one
Menlo Park, 94025	menlo park ninety four thousand and twenty five
California 94025	california ninety four thousand and twenty five

Note: Vocalizer handles Canadian postal codes and UK post codes as described in "Combining letters and numbers" on page 80.

Dates

Vocalizer recognizes dates specified in both text ("December 9, 1997") and digit ("12/9/1997") formats, and reads them accordingly. The digit format can be delimited with forward slashes ("/"), commas(","), periods ("."), or hyphens ("-").

Note: If you are using North American English voice packs, dates are parsed using the American date format, *mm/dd/yyyy*. If you are using either U.K. or Australian/New Zealand English voice packs, dates are parsed using the *dd/mm/yyyy* format.

Vocalizer recognizes month abbreviations as well as the full month name. Certain month names and month abbreviations—such as Jan, Mar, May, and June—are treated specially because they are also words. Based on the context of the overall phrase, these might be read as dates or read literally:

Input text	Output: North American format	Output: U.K. and Australian/New Zealand format
Feb 5	february five	february five
Jan 1, 2000	january one two thousand	january one two thousand
February 5, 1997	february five nineteen ninety seven	february five nineteen ninety seven

Input text	Output: North American format	Output: U.K. and Australian/New Zealand format
1/1/2000	the first of january two thousand	the first of january two thousand
3/12/97	the twelfth of march ninety seven	the third of december ninety seven
3.12.97	the twelfth of march ninety seven	the third of december ninety seven
1/6/99	the sixth of january ninety nine	the first of june ninety nine
1/6/00	the sixth of january two thousand	the first of june two thousand
1/6/01	the sixth of january two thousand and one	the first of june two thousand and one
3/12	three twelfths	three twelfths
on 3/12	on three twelfths	on three twelfths
12/1	twelve fslash one	twelve slash one
4-19-1966	the nineteenth of april nineteen sixty six	four one nine one nine six six
4-19	four minus nineteen	four minus nineteen
19-4	nineteen minus four	nineteen minus four
Jan 1	january one	january one
1 Jan	one january	one january

Times

Vocalizer recognizes time-of-day phrases in the formats *hh*:*mm* and *hh*:*mm*:*ss*, where the hour is between 0 and 24. It also recognizes time phrases that are succeeded by the following abbreviations: *am*, *pm*, *AM*, or *PM*. Space between the number and the abbreviation does not matter.

Time-of-day phrases in the format *X:00* are converted to "*X o'clock*." Hours from 13 to 24 are read using military time. If the hour value is greater than 24, the phrase is not read as a time. For example:

Input text	Output text
1:30	one thirty
1:30:31	one thirty and thirty one seconds
6:30 am	six thirty am

Input text	Output text
6:30am	six thirty am
12:00	twelve o'clock
12:00:05	twelve o'clock and five seconds
05:45	five forty five
23:00	twenty three hundred hours
14:35	fourteen thirty five
34:34	thirty four thirty four

Email and web addresses

Vocalizer reads email and web addresses naturally, including symbols such as "@" (expanded to "at") and "." (expanded to "dot"). Vocalizer determines when address segments can be read as words or need to be spelled out. For example:

Input text	Output text
mail@nuance.com	mail at nuance dot com
abc@nuance.com	a b c at nuance dot com
www.nuance.com	w w w dot nuance dot com
http://www.nuance.com	h t t p colon slash slash w w w dot nuance dot com
http://extranet.nuance.com /developers/index.html	h t t p colon slash slash extranet dot nuance dot com slash developers slash index dot h t m l

Handling 8-bit characters

If you expect to enter foreign language words or expressions in `ssml` mode, you must specify the encoding as described in "Encoding" on page 39.

Appendix A

Text processing for Canadian French

This appendix outlines how Nuance Vocalizer processes text input for Canadian French.

Topics include:

- Text encoding
- Phrasing and punctuation
- Abbreviations
- Capitalization
- Numbers
- Currency
- Telephone numbers
- Addresses
- Dates
- Times
- Email and web addresses

For detailed information on text normalization rules used for English dialects, see Chapter 9.

Text encoding

Nuance Vocalizer supports the Microsoft CP1252 (WinLatin1) and ISO-8859-1 If you are using Vocalizer in SSML mode, Vocalizer can interpret ISO-8859-1 input characters if you provide an appropriate XML header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<speak> Les lettres accentuées telles que: é,à,è,...sont
fréquentes en français </speak>
```

For more information, see "Encoding" on page 39.

Phrasing and punctuation

Vocalizer recognizes punctuation used to end phrases, sentences, and paragraphs. Vocalizer strips the layout from input text and uses it in conjunction with punctuation to determine phrasing, thereby enabling natural-sounding intonation and pausing at the ends of phrases, sentences, and questions. Text that is laid out and punctuated in a natural fashion, as it would be for normal written text, will be read in a natural predictable way.

TIP: For the most natural-sounding speech, try to break very long sentences into multiple sentences.

This section outlines text normalization rules for:

- Periods
- Hyphens and dashes
- Parentheses and double quotes

For general text normalization rules used to interpret punctuation, see the "Phrasing and punctuation" on page 69.

Periods

Vocalizer determines the correct interpretation for periods from the context. Periods can be used as:

- Sentence terminators—As a general rule, allow two spaces to follow a sentence terminator
- Abbreviation markers—See "Abbreviations" on page 89
- Decimals—See "Decimal fractions" on page 94

Hyphens and dashes

Since hyphens, dashes, and minus signs are represented by the same ASCII character, ambiguous situations may occur at times. To avoid ambiguous situations, Vocalizer uses a set of disambiguation rules. Specifically, Vocalizer considers the "-" character to be:

- A *trait d'union* if located between letters and consider each side as being part of the same word if in dictionary.
- A *signe de moins* if followed by a number, indicating a negative number.
- A *tiret* and pronounced "à" if located between two numbers without a space as in number intervals or game scores.
- A *trait d'union* if preceded by a letter and followed by a carriage return plus a letter. The words on either side of the hyphen are joined and handled as a single word.

Parentheses and double quotes

Vocalizer handles parentheses and double quotes. If a closing parenthesis or double quote occurs without any preceding matching symbol, for example, an open parenthesis, Vocalizer reads the close parenthesis or double quote as text. For example:

Input text	Output
(salut)	salut
salut)	salut parenthèse
"salut"	salut
«salut»	salut

Abbreviations

Abbreviations may consist of one or more alphabetic characters, which are either all lowercase, all uppercase, or have an uppercase first letter followed by lowercase letters. You can separate abbreviations from words by adding:

- A period and a space—For example, *Vocalizer, v. 3*
- A period only—For example, *Vocalizer, v.3*
- A space only—For example, *Vocalizer, v 3*

Single-letter abbreviations, homographic abbreviations, and measurement abbreviations are described next.

Note: As a general rule, use a period and an optional space to terminate an abbreviation.

Single-letter abbreviations

Ambiguous situations may occur between single-letter abbreviations and initials. Where a possible single-letter abbreviation is found, Vocalizer interprets the letter as:

- An *abbreviation* if it is followed by one or more digits and it appears in the list of abbreviations
- An *abbreviation* if it follows one or more digits and it is a measurement or numeric abbreviation
- An *initial* in all other cases

Homographic abbreviations

Homographic words are words that are spelled the same way but have different meaning, for example, the noun "lead" and the verb "lead." *Homographic abbreviations* are abbreviations that are spelled the same way as a word or letter but have different meaning, for example, the abbreviation for jeudi ("jeu.") and "jeu" the noun.

Vocalizer expands homographic abbreviations if they are followed by a period and a single space, or a period and a punctuation mark, for example:

Input text	Output
jeu. 4 décembre	jeudi quatre décembre
jeu.. 4 décembre	jeudi (pause) quatre décembre.
Va au jeu. Alors, nous partirons.	Va au jeu (pause) alors …

TIP: Be careful when creating a custom text replacement filters that replace homographic abbreviations. For example, a filter that replaces "on" with "Ontario" could incorrectly render the word "on" when encountered in regular text.

Measurement and numeric abbreviations

Measurement and numeric abbreviations are only recognized if they follow a digit (with or without a space). For example:

Input text	Output
5m	cinq mètres
+6.6cm	plus six point six centimètres
7 km	sept kilomètres
-40kg	moins quarante kilogrammes
250ml	deux cent cinquante millilitres
23s	vingt trois secondes
30° C	trente degré Celsius
-15 degré F	moins quinze degré Fahrenheit

Acronyms and initials

Vocalizer also handles acronyms and initials. Vocalizer defines acronyms and initials as follows so that the pronunciation tool can treat them appropriately:

- A string of letters including one or more uppercase letters (not including the first letter)
- A string of letters separated by periods
- A single letter, except where the letter is an abbreviation or “A”, “a”, “À“, “à“, “Y”, and “y”

See “Single-letter abbreviations” on page 90.

Note: Periods are removed from the acronyms.

Capitalization

Vocalizer’s sophisticated handling of capitalized characters can affect interpretation of input if words are capitalized incorrectly. When sending proper names or other words through the system, make sure they are sent to Vocalizer the way they would appear normally, with appropriate capitalization, rather than using all or no capital letters.

TIP: When you know in advance that text input includes the names of places, make sure you capitalize them appropriately. Sending "New York" instead of "new york" gives Vocalizer a better chance of rendering the name successfully. This rule also applies to your custom text replacement filters, which are discussed in the "Creating custom filters for text replacement" on page 52.

Directions

Use appropriate capitalization to prevent ambiguity when text input includes directions. For example, the verb "est" and the direction "Est" would be rendered incorrectly in the following text:

"La circulation est congestionnée en direction est sur le boulevard Maisoneuve est."

Numbers

Vocalizer has a set of defined conventions for handling digits and common components that contain numbers such as dollar amounts and telephone numbers. Use the rules defined in this section to ensure the most accurate and natural synthesis for specific types of numbers and number-based phrases.

Cardinals and ordinals

Vocalizer converts cardinal numbers (which indicate quantity) and ordinal numbers (which indicate position in a series) to their textual form.

The metric system is the main convention supported, but the imperial system is also supported when there is no conflict with the metric system. The metric convention uses single spaces to separate digit triplets within integers (which indicate hundreds, thousands, millions, and so on). While Vocalizer accepts commas as separators for digit triplets (imperial system), Nuance recommends using spaces instead to avoid confusion with commas as decimal separators.

Input text	Output
5600	cinq mille six cents
5 600	cinq mille six cents
4500000	quatre millions cinq cents mille
4ième	quatrième
102e	cent deuxième

Input text	Output
1234ième	mille deux cent trente-quatrième

However, if the comma is followed by one or more spaces, Vocalizer treats the digits as separate numbers:

Input text	Output
56, 734	cinquante-six (pause) sept cent trente-quatre
1, 103ième	un (pause) cent troisième

Digit sequences

Digit sequences are spoken in groups based on the way you transcribe them. To have digits voiced individually instead of as a single number, include either:

- A comma followed by a space
- A period followed by a space

While a single space or a comma with no space can be used to enumerate, they are not recommended as they could be interpreted as triplet separators within a single number.

These examples show how to various digit strings are converted to speech:

Input text	Output
1234567	un million deux cent trente-quatre mille cinq cent soixante-sept
4, 500, 200	quatre (pause) cinq cents (pause) deux cents
1, 2, 3, 4, 5, 6, 7	un deux trois quatre cinq six sept (Short pauses)
1. 2. 3. 4. 5. 6. 7	un deux trois quatre cinq six sept (Long pauses)
4 5 6 7	quatre cinq six sept
4 500 200	quatre millions cinq cents mille deux cents **Note:** This example illustrates the disadvantage of using single spacing as a separator, since the digits are processed as a single number.

Leading zeros

Leading zeros in a number are omitted from the speech output:

Input text	Output
044	quarante-quatre
02	deux
00	zéro

Fractions

Vocalizer recognizes fractions, where numerators and denominators are separated by a forward slash, with no intervening spaces:

Input text	Output
1/2	une demie
-13/5	moins treize cinquième
1 / 2	un barre oblique deux

Decimal fractions

By default, Vocalizer interprets metric decimal number convention for Canadian French, which uses a comma to separate decimals from the integer part. Optionally, spaces could be used to separate the digit triplets (hundreds, thousands, millions, and so on). Usually, the decimal part is a digit enumeration, except for digit *pairs* higher than "09". For example:

Input text	Output
+1300321	plus un million trois cents mille trois cent vingt et un
4,15	quatre virgule quinze
4.66	quatre point soixante-six
-55,345	moins cinquante-cinq virgule trois quatre cinq
6,05	six virgule zéro cinq

The imperial decimal system, using a period to separate the integer from the decimal part, is also supported when there is no possible ambiguity. For example, the number 1,234 would be considered by default as a metric decimal number and would be synthesized as "un virgule deux trois quatre." However, combining the use of a comma separator *and* a period, (1,234.5) forces the

number to be recognized as an imperial decimal number, and the output would be “mille deux cent trente quatre point cinq.” For example:

Input text	Output
13,400,777	treize millions quatre cents mille sept cents
-4.72	moins quatre point soixante-douze
56,400.924	cinquante-six mille quatre cents point neuf deux quatre
56,466	cinquante-six virgule quatre six six **Note:** Parsed by default as a metric decimal.

Percentages

The percent sign (“%”) is always read as “pour cent.” Space between the number and the percentage sign does not matter. For example:

Input text	Output
90%	quatre-vingt-dix pour cent
-5.125 %	moins 5 point un deux cinq pour cent
+700 %	plus sept cents pour cent
9/4%	neuf quart pour cent

Account numbers

To have account numbers or social security numbers read in segments instead of as a single digit, you must include hyphens or slashes between the segments. The character separator is pronounced to help determine segment boundaries:

Input text	Output
2222/3333/4444/5555	deux deux deux deux barre oblique trois trois trois trois barre oblique quatre quatre quatre quatre barre oblique cinq cinq cinq cinq
44/33	quarante-quatre trente-troisieme **Note:** This example is interpreted as a fraction rather than an account number because there are only two numbers separated by a slash with no intervening spaces. See “Fractions” on page 94.

Input text	Output
78-122-2	sept huit trait d'union un deux deux trait d'union deux
78-122	soixante-dix-huit à cent vingt-deux **Note:** This example is interpreted as an interval or a game score rather than an account number.

Combining letters and numbers

Combining letters and numbers in a single words generates varied results. Typically the characters are read individually and digits groups are treated as a single number if at the beginning:

Input text	Output
c20b	c deux zero b
d4df3	d quatre d f trois
20b	vingt b

Currency

Vocalizer recognizes the following currency symbols:

- Dollar sign ($), including:
  - Canadian dollars ($CAD)
  - American dollars ($USD)
- Cent sign (¢)
- Pound sterling sign (£)
- Euro symbol (€)
- French franc abbreviation (FF)
- Deutschmark abbreviation (DM)
- Yen symbol (¥)

Vocalizer supports the official French convention of currency symbols following the digits and will also accept currency symbols preceding the digits, as in English, which allows the abbreviation for millions or billions suffix to be added.

Input text	Output
50$	cinquante dollars
12$ CAD	douze dollars canadien
12 $USD	douze dollars américain
$25.50	vingt-cinq dollars et cinquante cennes
10,40 $ CAD	dix dollars canadiens et quarante cennes
$0,99	quatre-vingt-dix-neuf cennes
65,66¢USD	soixante-cinq virgule soixante-six cennes américain
$50M	cinquante millions de dollars
£ 10.20	dix livres et vingt penny
12,34 €	douze virgule trente-quatre euros
30,75FF	trente francs et soixante-quinze centimes
12,34 DM	douze virgule trente-quatre deutsche mark

Input text	Output
¥100	cent yen

Telephone numbers

Vocalizer recognizes local and national telephone number formats used in North America, including area codes.

The digits are grouped as they are presented, in order that pauses may be introduced:

Input text	Output
256 9866	deux cinq six (pause) neuf huit six six
514-732-4619	cinq un quatre (pause) sept trois deux (pause) quatre six un neuf
(819) 623-4455	huit un neuf (pause) six deux trois (pause) quatre quatre cinq cinq
1 800 256 9866	un (pause) huit cents (pause) deux cinq six (pause) neuf huit six six
1-514-732-4619	un (pause) cinq un quatre (pause) sept trois deux (pause) quatre six un neuf

Addresses

Vocalizer recognizes zip and postal codes and address abbreviations, such as “ave” or “st”. Addresses are handled as follows:

Input text	Output
Alexandre LeGrand 101 3e ave. Mtl, Qc H5T 2v2	Alexandre LeGrand cent et un troisieme avenue (pause) Montréal (pause) Québec h cinq t deux v deux
111 Duke, Mtl, Qc, Canada, H8T2V7	cent onze Duke (pause) Montréal (pause) Québec (pause) Canada (pause) h huit t deux v sept
CA 22333	Californie deux deux trois trois trois

Input text	Output
CO. 44666-1234	Colorado quatre quatre six six six trait d'union un deux trois quatre

Note: Pauses are added when there is punctuation between address fields.

Dates

Vocalizer recognizes dates specified in both text ("9 décembre, 2003") and digit ("2003/12/09") formats, and reads them accordingly. The digit format can be delimited with forward slashes ("/"), periods ("."), or hyphens ("-"). Normally, the digit order is year first, followed by month then day, but the reverse order is also supported (day/month/year) when no ambiguity is possible with the main convention. Standard abbreviations for months are supported.

Input text	Output
1999/1/5	le cinq janvier mille neuf cent quatre-vingt-dix-neuf
13-12-2003	le treize décembre deux mille trois
99/03/04	le quatre mars mille neuf cent quatre-vingt-dix-neuf
01/02/03	le trois février deux mille un
02/4/5	le cinq avril deux mille deux
6 jan, 2002	le six janvier deux mille deux
4 fev	le quatre février
2003-04-02	le quatre février deux mille trois
14/14/2000	un quatre barre oblique un quatre barre oblique deux zéro zéro zéro **Note:** This example can not be a valid date so it is interpreted as an account number. See "Account numbers" on page 95.

Times

Vocalizer recognizes time-of-day phrases using “h“ and “min“ to delimit the hour and minute fields. It also recognizes time phrases that are succeeded by the abbreviations “am,” “pm,” “AM,” or “PM” with a space between the number and the abbreviation.

Input text	Output
5h30min30sec	cinq heures trente minutes et trente secondes
13h45	treize heures quarante-cinq minutes
4 h 20 min	quatre heures vingt minutes
20h	vingt heures
20h00	vingt heures
0h30	zéro heures trente minutes
6h pm	six heures p m
3h 30min am	trois heures trente minutes a m
9h AM	neuf heures a m

Email and web addresses

Vocalizer reads email and web addresses naturally, including symbols such as “@” (expanded to “a commercial”) and “.” (expanded to “point”). Vocalizer determines when address segments can be read as words or need to be spelled out. For example:

Input text	Output
courrier@nuance.com	courrier a commercial nuance point com
abc@nuance.com	a b c a commercial nuance point com
www.nuance.com	triple w point nuance point com
http://www.nuance.com	h t t p deux points barre oblique barre oblique triple w point nuance point com

Input text	Output
http://support.nuance.com /developers/index.html	h t t p deux points barre oblique barre oblique support point nuance point com barre oblique developers barre oblique index point h t m l

Appendix B

Text processing for American Spanish

B

This appendix outlines how Nuance Vocalizer processes text input for American Spanish.

Topics include:

- Text encoding
- Regionalism
- Phrasing and punctuation
- Abbreviations
- Capitalization
- Numbers
- Currency
- Telephone numbers
- Dates
- Times
- Addresses
- Email and web addresses

For detailed information on text normalization rules used for English dialects, see Chapter 9.

Text encoding

Nuance Vocalizer supports the Microsoft CP1252 (WinLatin1) and ISO-8859-1 character encoding for plaintext input of American Spanish characters.

If you are using Vocalizer in SSML mode, Vocalizer can interpret ISO-8859-1 input characters if you provide an appropriate XML header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<speak> Las letras acentuadas son muy usadas en español </speak>
```

For more information, see "Encoding" on page 39.

Regionalism

For American Spanish, Vocalizer supports various conventions for numbers, dates, and addresses. For example, in the United States, decimal numbers use the form 4,123.5, while most Central and Latin American countries use the form of 4.123,5. Since these variations cause conflicting interpretations, Vocalizer supports the *-region* command-line option to determine the normalization convention. Set *-region* to either `US` or `CALA` to specify the set of conventions required. See "Using the -region option" on page 12.

The main distinction between `CALA` and `US` mode are:

`US`	`CALA`
Periods indicate decimal points; digit triplets are separated with commas	Commas indicate decimal points; digit triplets are separated with periods
Dates are interpreted using the *mm/dd/yyyy* format	Dates are interpreted using the *dd/mm/yyyy* format
"$" indicates dollars	"$" indicates pesos
Address abbreviations are interpreted in English	Address abbreviations are interpreted in Spanish

Phrasing and punctuation

Vocalizer recognizes punctuation used to end phrases, sentences, and paragraphs. Vocalizer strips the layout from input text and uses it in conjunction with punctuation to determine phrasing, thereby enabling natural-sounding intonation and pausing at the ends of phrases, sentences, and questions. Text

that is laid out and punctuated in a natural fashion, as it would be for normal written text, will be read in a natural predictable way.

TIP: For the most natural-sounding speech, try to break very long sentences into multiple sentences.

This section outlines text normalization rules for:

- Periods
- Parentheses and double quotes

For general text normalization rules used to interpret punctuation, see the "Phrasing and punctuation" on page 69.

Periods

Vocalizer determines the correct interpretation for periods from the context. Periods can be used as:

- Sentence terminators—As a general rule, allow two spaces to follow a sentence terminator
- Abbreviation markers—See "Abbreviations" on page 106
- Decimals—See "Cardinals and decimal fractions" on page 108

Parentheses and double quotes

Vocalizer handles parentheses and double quotes. If a closing parenthesis or double quote occurs without any preceding matching symbol, for example, an open parenthesis, Vocalizer reads the close parenthesis or double quote as text. For example:

Input text	Output
(buenos días)	buenos días
buenos días)	buenos días cierra paréntesis
"buenos días"	buenos días
«buenos días»	buenos días

Abbreviations

Abbreviations may consist of one or more alphabetic characters, which are either all lowercase, all uppercase, or have an uppercase first letter followed by lowercase letters. You can separate abbreviations from words by adding:

- A period and a space—For example, *Vocalizer, v. 3*
- A period only—For example, *Vocalizer, v.3*
- A space only—For example, *Vocalizer, v 3*

Single-letter abbreviations, homographic abbreviations, and measurement abbreviations are described next.

Note: As a general rule, use a period and an optional space to terminate an abbreviation.

Single-letter abbreviations

Ambiguous situations may occur between single-letter abbreviations and initials. Where a possible single-letter abbreviation is found, Vocalizer interprets the letter as:

- An *abbreviation* if it is followed by one or more digits and it appears in the list of abbreviations
- An *abbreviation* if it follows one or more digits and it is a measurement or numeric abbreviation
- An *initial* in all other cases

Homographic abbreviations

Homographic words are words that are spelled the same way but have different meaning, for example, the adjective “dedicada” and the verb “dedicada.” *Homographic abbreviations* are abbreviations that are spelled the same way as a word or letter but have different meaning, for example, the abbreviation for marzo (“mar.”) and “mar” the noun.

Vocalizer expands homographic abbreviations if they are followed by a period and a single space, or a period and a punctuation mark, for example:

Input text	Output
4 mar. 2004	cuatro de marzo de dos mil cuatro
Vivo cerca del mar.	Vivo cerca del mar.

TIP: Be careful when creating a custom text replacement filters that replace homographic abbreviations. For example, a filter that replaces “mar” with “marzo” could incorrectly render the word “mar” when encountered in regular text.

Measurement and numeric abbreviations

Measurement and numeric abbreviations are only recognized if they follow a digit (with or without a space). For example:

Input text	Output
5m	cinco metros
+6.6cm	más seis punto seis centímetros
7 km	siete kilómetros
-40kg	menos cuarenta kilogramos
250ml	doscientos cincuenta mililitros
23s	veintitrés segundos
30° C	treinta grados centígrados
-15 grados F	menos quince grados Fahrenheit

Acronyms and initials

Vocalizer also handles acronyms and initials. Vocalizer defines acronyms and initials as follows so that the pronunciation tool can treat them appropriately:

- A string of letters including one or more uppercase letters (not including the first letter)
- A string of letters separated by periods
- A single letter, except where the letter is an abbreviation or “A”, “a”, “E“, “e“, “O”, “o”, “U”, “u”, “Y”, and “y”

See “Single-letter abbreviations” on page 106.

Note: Periods are removed from the acronyms.

Capitalization

Vocalizer’s sophisticated handling of capitalized characters can affect interpretation of input if words are capitalized incorrectly. When sending proper names or other words through the system, make sure they are sent to Vocalizer the way they would appear normally, with appropriate capitalization, rather than using all or no capital letters.

TIP: When you know in advance that text input includes the names of places, make sure you capitalize them appropriately. Sending “New York” instead of “new york” gives Vocalizer a better chance of rendering the name successfully. This rule also applies to your custom text replacement filters, which are discussed in the “Creating custom filters for text replacement” on page 52.

Numbers

Vocalizer has a set of defined conventions for handling digits and common components that contain numbers such as dollar amounts and telephone numbers. Use the rules defined in this section to ensure the most accurate and natural synthesis for specific types of numbers and number-based phrases.

Cardinals and decimal fractions

Vocalizer’s interpretation of cardinal numbers depends on the value of the *-region* option (see “Regionalism” on page 104). In `US` mode, interpretation of the text input is based on the English system, which uses commas to separate digit triplets (indicating hundreds, thousands, millions, and so on) and periods

indicate decimals. In `CALA` mode, periods are used to separate digit triplets and commas indicate decimals. When there is no ambiguity, the appropriate convention is used regardless of the *-region* setting.

For example:

Input text	US mode	CALA mode
5601	cinco mil seiscientos uno	cinco mil seiscientos uno
5,601	cinco mil seiscientos uno	cinco coma seis cero uno
5.601	cinco punto seis cero uno	cinco mil seiscientos uno
-4.500,05	menos cuatro mil quinientos coma cero cinco	menos cuatro mil quinientos coma cero cinco
4.12	cuatro punto doce	cuatro punto doce
4.123	cuatro punto uno dos tres	cuatro mil ciento veintitrés

If the comma is followed by one or more spaces, Vocalizer treats the digits as separate numbers:

Input text	Output
56, 734	cincuenta y seis, setecientos treinta y cuatro

Ordinals

Vocalizer interprets the two frequent forms of Spanish ordinals: the “ésimo” form and the “avo” form. The ordinal suffix determines the form, gender, and number associated with the ordinal. You can use suffixes with 1 to 4 letters.

Input text	Output
1o	primero
1os	primeros
1ro	primero
1ero	primero
1eros	primeros
1a	primera
7o	séptimo

Input text	Output
7os	séptimos
7mo	séptimo
7imo	séptimo
7imos	séptimos
3o	tercero
5as	quintas
17o	decimoséptimo
32a	trigésima segunda
32avo	treinta y dosavo
17avo	diecisieteavo
17ava	diecisieteava
17avos	diecisieteavos
17avas	diecisieteavas

Digit sequences

Digit sequences are spoken in groups based on the way you transcribe them. To have digits voiced individually instead of as a single number, include either:

- A comma followed by a space
- A period followed by a space

While a single space or a comma with no space can be used to enumerate, they are not recommended as they could be interpreted as triplet separators within a single number.

These examples show how to various digit strings are converted to speech:

Input text	Output
1234567	un millón doscientos treinta y cuatro mil quinientos sesenta y siete
4, 500, 200	cuatro (pause) quinientos (pause) doscientos

Input text	Output
1, 2, 3, 4, 5, 6, 7	uno dos tres cuatro cinco seis siete (Short pauses)
1. 2. 3. 4. 5. 6. 7	uno dos tres cuatro cinco seis siete (Long pauses)
4 5 6 7	cuatro cinco seis siete
4,500,200	cuatro millones quinientos mil doscientos (US mode) **Note:** This example illustrates the disadvantage of using single coma as a separator, since the digits are processed as a single number.

Leading zeros

Leading zeros in a number are omitted from the speech output:

Input text	Output
044	cuarenta y cuatro
02	dos
00	cero

Fractions

Vocalizer recognizes fractions, where numerators and denominators are separated by a forward slash, with no intervening spaces:

Input text	Output
1/2	un medio
-13/5	menos trece quintos

Percentages

The percent sign ("%") is always read as "por ciento." Space between the number and the percentage sign does not matter. For example:

Input text	Output
90%	noventa por ciento
-5.125 %	menos cinco mil ciento veinticinco por ciento
+700 %	más setecientos por ciento

Input text	Output
9/4%	nueve cuartos por ciento

Account numbers

To have account numbers or social security numbers read in segments instead of as a single digit, you must include hyphens or slashes between the segments. The character separator is pronounced to help determine segment boundaries:

Input text	Output
2222/3333/4444/5555	dos dos dos dos barra oblicua tres tres tres tres barra oblicua cuatro cuatro cuatro cuatro barra oblicua cinco cinco cinco cinco
44/33	cuarenta y cuatro treinta y tresavos **Note:** This example is interpreted as a fraction rather than an account number because there are only two numbers separated by a slash with no intervening spaces. See “Fractions” on page 111.
78-122-2	siete ocho guión uno dos dos guión dos
78-122	setenta y ocho (pause) a (pause) ciento veintidós **Note:** This example is interpreted as an interval or a game score rather than an account number.

Combining letters and numbers

Combining letters and numbers in a single words generates varied results. Typically the characters are read individually and digits groups are treated as a single number if at the beginning:

Input text	Output
c20b	c dos cero b
d4df3	d cuatro d f tres
20b	veinte b

Currency

Vocalizer recognizes currency symbols, including:

- Dollar sign ($, US, US), including:

  Note: The interpretation of "$" depends of the region specified (see regionalism above section) being "dólares" for US mode and "pesos" for CALA mode.

- Cent sign (¢)
- Pesos symbols ($, P, Ps, PS)
- Nuevo pesos (N$, N$U)
- ISO currency symbols, such as MXN, CLP, ARS, USD, and so on

Vocalizer accepts currency symbols either preceding or following digits, and abbreviations for million or billion suffix.

Input text	Output
$50	US mode: cincuenta dólares CALA mode: cincuenta pesos
US $22M	veintidós millones de dólares
$25.50	veinticinco dólares con cincuenta centavos
$0,99	noventa y nueve centavos
21 P	veintiún pesos
101 MXN	ciento un pesos mexicanos

Telephone numbers

Vocalizer recognizes local and national telephone number formats used in North America, including area codes.

The digits are grouped as they are presented, in order that pauses may be introduced:

Input text	Output
256 9866	dos cinco seis (pause) nueve ocho seis seis
514-732-4619	cinco uno cuatro (pause) siete tres dos (pause) cuatro seis uno nueve
(232) 623-4455	dos tres dos (pause) seis dos tres (pause) cuatro cuatro cinco cinco

Input text	Output
1 800 256 9866	uno (pause) ochocientos (pause) dos cinco seis (pause) nueve ocho seis seis
2335-4445	dos tres tres cinco (pause) cuatro cuatro cuatro cinco
233-44-45	dos tres tres (pause) cuatro cuatro (pause) cuatro cinco
(23)(23) 3453456	dos tres (pause) dos tres (pause) tres cuatro cinco tres cuatro cinco seis
Tel: 623-4455	teléfono (pause) seis dos tres (pause) cuatro cuatro cinco cinco
TEL 2334545	teléfono (pause) dos tres tres cuatro cinco cuatro cinco

Dates

Vocalizer recognizes dates specified with abbreviation (6 ene. 2004) or digit format (2004/01/06).

The expected digit format depends on the *-region* specified (see "Regionalism" on page 104). In US mode, date input is interpreted using the format *mm/dd/yyyy* (or *mm/dd/yy*). In CALA mode, date input is interpreted using the format *dd/mm/yyyy* (or *dd/mm/yy*). You can delimit digit formats with forward slashes ("/"), hyphens ("-") or periods (".").

When date input is not entered in the expected order, alternative forms are considered. For example, if Vocalizer is in CALA mode, Vocalizer recognizes that a date such as 02/30/04 cannot be processed as *yy/mm/dd* but should be interpreted as *mm/dd/yy*.

Input text	Output
1999/1/5	cinco de enero de mil novecientos noventa y nueve
13-12-2003	trece de diciembre de dos mil tres
99/03/04	cuatro de marzo de mil novecientos noventa y nueve
01/02/03	US mode: dos de enero de dos mil tres CALA mode: primero de febrero de dos mil tres
02/4/5	US mode: cuatro de febrero de dos mil cinco CALA mode: dos de abril de dos mil cinco

Input text	Output
6 Ene. 2004	seis de enero de dos mil cuatro
12 Ago., 2000	doce de agosto de dos mil
2003-04-02	dos de abril de dos mil tres
14/14/2000	uno cuatro barra oblicua uno cuatro barra oblicua dos cero cero cero **Note:** This example can not be a valid date so it is interpreted as an account number. See “Account numbers” on page 112.

Times

Vocalizer recognizes time-of-day phrases using “h” and “min” to delimit the hour and minute fields or the semicolon form *hh:mm:sec*. It also recognizes time phrases that are succeeded by the abbreviations “am,” “pm,” “GMT,” or “EST” with a space between the number and the abbreviation.

Input text	Output
5h30min30sec	cinco horas treinta minutos y treinta segundos
13h45	trece horas y cuarenta y cinco
4 h 20 min	cuatro horas y veinte minutos
20h	veinte horas
20h00	veinte horas
0h30	cero hora y treinta
6h pm	seis horas postmeridiano
3h 30min am	tres horas y treinta minutos antemeridiano
9h AM	nueve horas antemeridiano
5 :30 :20	cinco horas treinta minutos y veinte segundos
4 :20	cuatro horas y veinte
12 :00	mediodía
8h GMT	ocho horas (pause) hora media de Greenwich

Input text	Output
9h30 EST	nueve horas y treinta (pause) hora estándar del este
10:40 PST	diez horas y cuarenta (pause) hora estándar del pacífico

Addresses

Vocalizer recognizes address elements like country abbreviations, state abbreviations, postal codes, and address abbreviations, such as “ave.”, “st”, “blvd.”, “N.”, “Nte”, and “P.O.”

Typical abbreviations for both English and Spanish address formats are supported. Interpretation of some abbreviations depends on the *-region* setting specified (see “Regionalism” on page 104). For example “N.” is translated as “North” in US mode but as “Norte” in CALA mode.

To facilitate the interpretation of the address rendering, short pauses are introduced to separate the most important element of the address. Addresses are handled as follows:

Input text	Output
Miami, FL	Miami (pause) Florida
FL, USA	Florida (pause) Estados Unidos
232 El Camino Real Blvd. Menlo Park, CA	doscientos treinta y dos (pause) El Camino Real boulevard (pause) Menlo Park (pause) California
444 25th Ave Santa Claus, CA	cuatrocientos cuarenta y cuatro (pause) twenty fifth Avenue (pause) Santa Claus (pause) California
12 N Mormon St. Salt Lake City UT 12345-6666 USA	US mode: Doce (pause) North Mormon Street (pause) CALA mode: Doce (pause) Norte Mormon Street (pause) Salt Lake City (pause) Utah (pause) uno dos tres cuatro cinco (pause) seis seis seis (pause) Estados Unidos
40 E Cactus Dr. Suite 20c Tucson, AZ	Cuarenta (pause) East Cactus Drive (pause) suite viente c (pause) Tucson (pause) Arizona
1000 Redwood PO BOX 101 Seattle, 90001 USA	Mil (pause) Redwood (pause) Post Office Box ciento uno (pause) Seattle (pause) nueve cero cero cero uno (pause) Estados Unidos

Input text	Output
Mrs. Chandler 7449 Rupert Avenue SAINT LOUIS, MO USA 63117	Misses Chandler (pause) siete mil cuatrocientos cuarenta y nueve (pause) Rupert Avenue (pause) Saint Louis (pause) Missouri (pause) Estados Unidos (pause) seis tres uno uno siete
MEX	CALA mode: (pause) México
Santiago, CHL	CALA mode: Santiago (pause) Chile
Sr. Francisco Ave Providencia N 112 6º piso, Las Condes Santiago, CHL 63117	CALA mode: Señor Francisco (pause) avenida Providencia norte (pause) ciento doce (pause) sexto (pause) piso (pause) Las Condes (pause) Santiago (pause) Chile (pause) seis tres uno uno siete

Email and web addresses

Vocalizer reads email and web addresses naturally, including symbols such as ″@″ (expanded to ″arroba″) and ″.″ (expanded to ″punto″). Vocalizer determines when address segments can be read as words or need to be spelled out. For example:

Input text	Output
carlos@nuance.com	carlos arroba (pause) nuance punto (pause) com
carlos2b@nuance.com	carlos dos b (pause) arroba (pause) nuance punto (pause) com
abc@nuance.com	a (pause) b (pause) c (pause) arroba (pause) nuance punto (pause) com
www.nuance.com	w w w (pause) punto (pause) nuance punto (pause) com
http://www.nuance.com	h t t p (pause) dos punto (pause) barra barra (pause) w w w (pause) punto (pause) nuance punto (pause) com

Index

R

S

T

U

V

W

X