IBM ViaVoice™ Outloud API Reference

Version 5.0 - Beta

Printed in the USA

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

Fifth Edition (September 1999)

Contents

About This Document

This book provides information on incorporating IBM ViaVoice Outloud technology into other applications. It describes the programming interfaces available for developers to take advantage of these features within their applications. This book is prepared in Portable Document Format (PDF) to provide the advantages of text search and cross-reference hyperlinking and is viewable with the Adobe Acrobat Reader v.3.x or higher. We recommend that you print all or part of this guide for quick reference.

Who Should Read This Document?

Read this book if you are a developer interested in writing Linux applications that use IBM ViaVoice Outloud technology.

ViaVoice SDK-Related Publications

Programming, reference and design information needed to use this SDK is available in a variety of sources:

- *SAPI Reference*

Refer to the following sources for additional programming, reference, and design information:

- *Developer's Corner Web Page* for SDK downloads, updates, and other documentation at:
  http://www.ibm.com/ViaVoice/dev_home.html
- *IBM ViaVoice SDK Web Channel* at:
  http://www.software.ibm.com/ViaVoice/subscribe.html

How This Book Is Organized

This document is organized in the following manner:

- Chapter 1 contains user information about ViaVoice Outloud.
- Chapter 2 contains the reference information for using the ViaVoice Outloud functions.
- Appendix A notices and trademark information.
- Appendix B contains Annotations, which are special codes you can place in the input text to customize the output.
- Appendix C contains User Dictionaries, which allow you to specify explicit pronunciations for words, abbreviations, and acronyms.
- Appendix D contains tables of SPR Symbols that represent the phonetic spelling of a single word.
- Appendix E contains SPR symbols for British English, French, German, standard Italian, and Castilian Spanish.
- Appendix F contains a glossary of terms.

Document Conventions

The following conventions are used to present information in this document:

Text Format	**Applies to**
Monospace font	Code samples.
Bold	Function and callback names; data types.
Italics	Parameter names; references to other documentation; special information.
`UPPERCASE`	Property, ennumerator, mode, and state names.

Chapter 1
Application Programming Interface

Overview

ECI (Eloquence Command Interface) is a library that provides an interface between applications and the IBM ViaVoice Outloud text-to-speech system. Version 5.0 of ECI has been re-architected to provide support for multiple, concurrent speech synthesis threads, and a consistent interface on all supported platforms.

As in prior versions of ECI, text is appended to the input buffer. Each word takes its voice definition from the active voice. Speech is synthesized from the input buffer according to the associated voice parameters, placed in the output audio buffer, and sent to the appropriate destination. The active voice can be set from a number of built-in voices. The language, the dialect, and the voice parameters can be modified individually using either ECI function calls, or annotations inserted into the input buffer with the input text. As text is added to the input buffer, the active voice definition is stored with it, so that changes to the active voice do not affect text already in the input buffer.

Indices can be used to determine when the delimited text fragment has been synthesized. A message will be received when all text inserted before the index has been synthesized.

Output can be sent to one of three types of destinations: a callback function, a file, or an audio device. These destination types are mutually exclusive, so sending output to one of them turns off output to the previous destination. The default is to send to an available audio device.

Getting Started

Use the following general guidelines to create and organize the API:

- To synthesize text in a simple manner, use the high-level ECI function **eciSpeakText**.
- To create a new ECI instance, u**se eciNew.** You must call **eciNew** before calling any functions which are instance specific. All functions that are instance specific are passed a handle (type ECIHand) as the first parameter. These functions only affect the state of the ECI instance associated with the ECIHand that you provide. You can have several instances active at the same time. When you are finished using an instance, you must call **eciDelete** for that instance.

- To synthesize unannotated text with the default voice and parameters and speak them through the default audio output device, call **eciAddText.** Call **eciSynchronize** to wait until synthesis is complete.
- To synthesize text in line-oriented format, such as a table or list, call **eciAddText** and **eciSynthesize** for each line, to ensure that each line is spoken as a separate sentence.
- To synthesize annotated text, call **eciSetParam(*eciInstance*, eciInputType,** 1**)** before calling **eciAddText.** This lets ECI know that the text may contain annotations.
- To use one of the preset voices, call **eciCopyVoice** before calling **eciAddText.**The active voice (voice 0) is a set of voice characteristics, such as pitch baseline and pitch fluctuation, which are applied to all new text added to the input buffer. A more detailed discussion of voices and voice parameters follows later in this document.
- To receive indices, first call **eciRegisterCallback** to register a callback function. Then call **eciInsertIndex** after calling **eciAddText** with the text you want indexed. Your callback function will be called with the **eciIndexReply** message after the indexed text has been processed by the audio destination.
- To capture wave sample output, call **eciRegisterCallback** to register a callback function, and then call **eciSetOutputBuffer** to register a buffer where wave samples will be stored. Call **eciAddText** with the text you want synthesized. Your callback function will be called with the **eciWaveformBuffer** message when your registered wave sample buffer becomes full.

SDK Contents

The following file are included in the IBM ViaVoice Outloud SDK:

- ECI Interface Files
  IBMECI50.LIB – Link library for use in building C/C++ applications (Windows only)

  ECI.H – header file for use in building C/C++ applications

  BOOLEAN.H – header file for use in building C/C++ applications
- Sample Application Files
  SPEAK.CPP – Command line application
- Help Files and Documentation
  IBM ViaVoice Outloud – ECI Specification (this document) in Adobe ® Acrobat format.

  README – A readme text file.

Co-Requisite Software

Before you can run a speech-synthesis application program, created with this SDK, you need to obtain and install one of the IBM ViaVoice Outloud runtime products. If you are interested in using a particular synthesis language or language dialect on a specific operating system platform, please consult your software vendor for availability information.

Threading

You can write a single-threaded or multi-threaded application, on any of the supported platforms. If your application is multi-threaded, each ECI instance (ECIHand) that you use must be called from the same thread that created it. Do not create an instance on one thread, and use it on another. When you are done with the instance, delete it on the same thread that it was created on too. If you register a callback function with ECI, it will execute on the same thread that the instance is running on.

Callbacks

If you register a callback with ECI, it should be designed in such a way that it completes its execution quickly. You should not call an ECI function from a callback. When your callback completes, make sure it returns eciDataProcessed.

Code Samples

This first sample C program speaks a short phrase and then exits:

```
#include <eci.h>

int main(int argc, char *argv[])
  {
    eciSpeakText ("Hello World!", false);

    return 0;
  }
```

The second sample C program speaks a short phrase in English, then in Spanish, and then exits:

```
#include <eci.h>

int main(int argc, char *argv[])
{

  ECIHand eciHandle;
  char errorMsg[100];

  eciHandle = eciNew();//Create a new ECI Instance
  if (eciHandle != NULL_ECI_HAND) //Success?
  {
    //Give some text to the instance
    if (!eciAddText(eciHandle, "Hello World!"))
    {
      //We failed to add text
      //Print an error message
      printf( “eciAddText failed\n” );
    }
```

Continued on the next page

```
    Continued from previous page

    //Change the language to Castilian Spanish,
    //if available
    if (eciSetParam(eciHandle, eciLanguageDialect, eciCastilianSpanish)=
    -1)
    {
      //Error Changing to Spanish
      printf( “Could not change to Spanish\n” );
    }
    else
    {
      //Give some text in Spanish
      if (!eciAddText(eciHandle, "Uno. Dos. Tres."))
      {
        //We failed to add text
        //Print an error message
      printf( “eciAddText failed\n” );
      }
    }
    //Wait until ECI finishes speaking
    if (!eciSynchronize(eciHandle))
    {
      //We failed to synchronize
      //Print an error message
      printf( “eciSynchronize failed\n” );
    }
    //Delete our ECI Instance – deallocates memory
    eciDelete(eciHandle);
  }
  else
  {
    //We failed to create a new ECI Instance
    //Print an error message
    printf( “eciNew failed\n” );
  }

  return 0;
}
```

System State Variables

Input buffer

The input buffer contains text, entered with **eciAddText,** that has not yet been synthesized by ECI.

Text entered into the input buffer is associated with the active voice. Any change to the active voice (see below) affects text entered after the change.

Default: The input buffer is empty.

Speech Environment Parameters

eciSynthMode

0: Sentence: The input buffer is synthesized and cleared at the end of each sentence (default).

1: Manual: Synthesis and input clearing is controlled by commands only.

eciInputType

0: Plain: input consists of unannotated text. Any annotations will be spelled out (e.g. `v2 would sound like “backquote v 2”) (default).

1: Annotated: input text includes annotations. See Appendix B for more details on supported annotations.

eciTextMode

0: Default: no special interpretation is done (default).

1: AlphaSpell: letters and digits are spelled out while other characters are ignored (e.g. “This is a test.” would sound like “tea aych eye es eye es ay tea e es tea”).

2: AllSpell: all symbols are spelled out (e.g. “This is a test.” would sound like “like “tea aych eye es eye es ay tea e es tea period”).

3: IRCSpell: letters are pronounced using the International Radio Code (“alpha, bravo, charlie”); other characters are ignored (US English only).

eciDictionary

0: Abbreviatsions dictionaries are used (default)

1: Abbreviations dictionaries are not used.

See Appendix C for a description of the three types of user dictionaries.

eciSampleRate

0: 8000 samples per second

1: 11,025 samples per second (default)

eciWantPhonemeIndices

0: Phoneme indices are not generated.

1: If a callback has been registered (see **eciRegisterCallback()** below), phoneme indices will be sent to the callback as each phoneme is being spoken. See also the **eciPhonemeIndexReply** message and the **ECIMouthData** type.

eciRealWorldUnits

Change the parameter ranges for specific voice parameters to Real World units.

eciLanguageDialect

A value specifying the language and dialect. These should be of type **ECILanguageDialect**. The language defaults to the "lowest-numbered" language installed on the system. Languages are numbered in the order specified by the **ECILanguageDialect** enum (see eci.h).

eciNumberMode

0: Pronounce 4-digit numbers as "nonyears" (e.g. "1984" would sound like "one thousand nine hundred eighty four").

1: Pronounce 4-digit numbers as "years" (e.g. "1984" would sound like "nineteen eighty four")(default)

Default:

- The input buffer is synthesized and cleared at the end of each sentence.
- The input does not contain annotations.
- No special interpretation is performed on the text.
- User dictionaries are used.
- The sample rate is 11,025 samples per second.
- Phoneme indices are not generated.
- ECI units are used for all voice definition parameters.
- The lowest-numbered language on the system is used.
- Four-digit numbers are pronounced as "years".

```
enum
   {
   eciGeneralAmericanEnglish,//0x00010000
   eciBritishEnglish,//0x00010001
   eciCastilianSpanish,//0x00020000
   eciMexicanSpanish,//0x00020001
   eciStandardFrench,//0x00030000
   eciStandardGerman,//0x00040000
   eciStandardItalian//0x00050000
   } ECILanguageDialect
```

Not all languages are available with all installations. See Appendix E for SPR tables for other languages.

Voice Definitions

There are eight preset voice definitions for each dialect of each language, of which three are reserved for future use. (Our runtimes support two English dialects, US and UK. Not all dialects are available on all operating system platforms. See Co-requisite Software ??->link) Each voice definition contains a set of parameters that control the attributes of the voice.

In addition to the preset voices, custom voices can be created by selecting unique combinations of these voice definition parameters.

The voice definition parameters and their values (ECI units / Real World units) are:

eciGender

0: male
1: female

eciHeadSize

0: tiny
100: huge

eciPitchBaseline

0 / 40: extremely low (Real World = 40 cycles per second)
100 / 422: extremely high (Real World = 422 cycles per second)

eciPitchFluctuation

0: completely monotonous
100: extreme fluctuation

eciRoughness

0: completely smooth
100: extremely rough

eciBreathiness

0: no breathiness
100: whispering

eciSpeed

0 / 70: extremely slow (Real World = 70 words per minute)
100 / 351: extremely fast (Real World = 351 words per minute)

eciVolume:

Log-scaled from **0** to **100** in ECI units, or linearly-scaled from 0 to 65535 in Real World units.

The preset voices for each of the languages are:

1. An adult male
2. An adult female
3. A child
4. Reserved for future use
5. Reserved for future use
6. Reserved for future use
7. An elderly female
8. An elderly male

Function Conventions

Use the following conventions to create and use ViaVoice Outloud functions.

Data Types

Boolean

Equivalent to an integer with values 0 (false) and 1 (true).

ECIHand

A handle to an instance of ECI.

ECIInputText

A const char* containing an ASCIIZ string using a system-dependent character set (currently ANSI for all platforms).

ECIMouthData

A structure containing a phoneme and its corresponding mouth position data. This is used by the **eciPhonemeIndexReply** message. Its members are:

```
char szPhoneme [eciPhonemeLength+1]; //ASCIIZ String containing SPR phoneme
ECILanguageDialect eciLanguageDialect; //Language of this phoneme
unsigned char mouthHeight; // 0-255
unsigned char mouthWidth; // 0-255
unsigned char mouthUpturn; // 0-128 (neutral)-255
unsigned char jawOpen; // 0-255
unsigned char teethUpperVisible; // 0 (hidden)-128 (teeth visible)-255 (teeth & gums)
unsigned char teethLowerVisible; // 0 (hidden)-128 (teeth visible)-255 (teeth & gums)
unsigned char tonguePosn; // 0 (relaxed)-128 (visible)-255 (against upper teeth)
```

```
    unsigned char lipTension; // 0-255
```

In addition to the phoneme symbols defined for SPR input (see Appendix D for a list), the symbol "\0\0\0\0"is also used to indicate end of utterance, and is sent with a set of neutral mouth position parameters.

ECIDictHand

A handle to an ECI dictionary set.

ECIDictVolume

enum {eciMainDict, eciRootDict, eciAbbvDict}. Identifies the volume within a dictionary set (see Appendix C).

Return Values

Unless otherwise specified, functions declared Boolean return true for success and false for failure.

Parameters

Unless otherwise specified, valid ECIHand parameters are assumed to be non-NULL (not equal to NULL_ECI_HAND), and pointers are assumed non-NULL. All strings are in ASCIIZ format.

Calling Conventions

On Win32 platforms, ECI functions are defined as __stdcall, formerly known as the PASCAL calling convention. Please see the Microsoft documentation for more information about what this convention means.

On UNIX platforms, ECI functions default to normal C functions.

Tags

Insert these tags into the input text to affect how it sounds when it is read out loud. If you are annotating an entry in the Main Dictionary, you must use the annotations equivalent of the tag.

Choosing a Speaker

Use the following tags to choose a speaker:

Tag	**Annotation**	**Description**
\Vce=Speaker=*name*\	- - -	Set speaker to the specified user-defined or built-in speaker.
\Vce=Speaker=Reed\	`l1.0 `v1	Set speaker to Reed, which uses an American English adult male voice and a sampling rate of 11 kHz.
\Vce=Speaker=Shelley\	`l1.0 `v2	Set speaker to Shelley, which uses an American English adult female voice and a sampling rate of 11 kHz.
\Vce=Speaker=Sandy\	`l1.0 `v3	Set speaker to Sandy, which uses an American English child voice and a sampling rate of 11kHz.
\Vce= Speaker=Grandma\	`l1.0 `v7	Set speaker to Grandma, which uses an American English older female's voice and a sampling rate of 11 kHz.
\Vce= Speaker=Grandpa\	`l1.0 `v8	Set speaker to Grandpa, which uses an American English older male's voice and a sampling rate of 11 kHz.

Although the tag labels are not case sensitive, the speaker names are, and must be typed in exactly as indicated above.

Choosing a Language and Dialect

Use the following annotations to choose a language and dialect (if available) (\Vce=Language=name\):

Annotation	Tag	Dialect or Language
`l1	\Vce=Language=English\	English
`l1.0	\Vce=Dialect=American\	American English (default dialect)
`l1.1	\Vce=Dialect=British\	British English
`l2	\Vce=Language=Spanish\	Spanish
`l2.0	\Vce=Dialect=Castilian\	Castilian Spanish (default dialect)
`l3	\Vce=Language=French\	French
`l3.0	\Vce=Dialect=Standard\	Standard French (default dialect)
`l4	\Vce=Language=German\	German
`l4.0	\Vce=Dialect=Standard\	Standard German (default dialect)
`l5	\Vce=Language=Italian\	Italian
`l5.0	\Vce=Dialect=Standard\	Standard Italian (default dialect)
`l6	\Vce=Language=Chinese\	Chinese
`l6.0	\Vce=Dialect=Standard\	Mandarin Chinese (default dialect)

Selecting a language will change the language only, not the voice characteristics. The last selected voice characteristics will remain in effect.

Choosing a Voice and Voice Characteristics

Use the following annotations to choose a stored voice (\Vce=StoredVoice="*name*"\):

Annotation	**Tag**	**Description**
- - -	\Vce=StoredVoice="*name*"\	Set voice to the specified user-defined or built-in voice.
`v1	\Vce=StoredVoice="Adult Male 1"\	Set voice to the default male voice. (Reed's voice)
- - -	\Vce=StoredVoice="Adult Male 2"\	Set voice to a higher-pitched male.
`v2	\Vce=StoredVoice="Adult Female 1"\	Set voice to the default female voice. (Shelley's voice)
`v3	\Vce=StoredVoice="Child 1"\	Set voice the default child voice. (Sandy's voice)
`v7	\Vce=StoredVoice="Elderly Female 1"\	Set voice the default older female voice. (Grandma's voice)
`v8	\Vce=StoredVoice="Elderly Male 1"\	Set voice to the default older male voice. (Grandpa's voice)

Although the tag labels are not case sensitive, the voice names are, and must be typed in exactly as indicated, including quotation marks.

Voice names are sensitive to the currently selected User Interface Language. If you have changed your UIL, you must use the voice names which correspond to the current UIL and are listed in the General dialog box. For example, if you have changed your UIL to Deutsch (German), and want to use the tag for "Adult Male 1", you must type in:

\Vce=StoredVoice="Erwachsener Mann 1"\.

Choosing a Voice by Gender or Age

Use the following tags to choose a voice by gender (\Vce=Gender=*gender*\) or age (\Vce=Age=*age*\):

Tag	**Description**
\Vce=Gender=Male\	Set voice to a male voice, which is either Adult Male, Child, or Elderly Male, depending on the age setting.
\Vce=Gender=Female\	Set voice to a female voice, which is either Adult Female, Child, or Elderly Female, depending on the age setting.
\Vce=Age=Child\	Set voice to the built-in Child voice.
\Vce=Age=Adult\	Set voice to an adult voice, which is Adult Male or Adult Female, depending on the gender setting.
\Vce=Age=Elderly\	Set voice to an elderly voice, which is Elderly Male or Elderly Female, depending on the gender setting.

There are no equivalent annotations for gender and age tags.

Defining Voice Characteristics

Use the following annotations or tags to define voice characteristics:

Annotation	**Tag**	**Description**
`vgN	\xVct=*gender*\	Set vocal tract configuration to male or female. Tag \xVct=male\ or \xVct=female\. Values are 0=male, 1=female.
`vbN	\Pit=N\	Set **pitch baseline** to N. Tag range is 40-422 Hz. Annotation range is 0-100.
`vhN	\xHsz=N\	Set **head-size** to N. Range is 0 (very small head) to 100 (very large head).
`vrN	\xRgh=N\	Set **roughness** to N. Range is 0 (smooth) to 100 (rough).

`vyN	\xBth=N\	Set **breathiness** to N. Range is 0 (not breathy) to 99 (very breathy). Setting breathiness to 100 yields a whisper.
`vfN	\xPfl=N\	Set **pitch fluctuation** to N. Range is 0 (narrow=monotonic) to 100 (wide).
`vsN	\Spd=N\	Set **speed** of the utterance to N. Tag range is 70-350 words per minute. Annotation range is 0-100.
`vvN	\Vol=N\	Set speech **volume** to N. Tag range is 1 to 65535 (linear). Annotation range is 0-100.
(no annotation)	\Rst\	**Reset** the voice to the original characteristics for the selected speaker.

Once a voice characteristic annotation is invoked, the voice will continue to use that characteristic until another value for that voice characteristic is given or the \Rst\ (reset) command is invoked.

Choosing a Speaking Style

Use the following annotations or tags to choose a speaking style (\Chr="*style*"\):

Annotation	**Tag**	**Description**
`vy100	\Chr=Whisper\	Set voice to a whisper.
`vf0	\Chr=Monotone\	Set voice to a monotone.
varies	\Chr=Normal\	Restore voice to the speaking style in use before the \Chr\ command was invoked. The equivalent annotation will vary depending on the original breathiness or pitch fluctuation values for the selected speaker.

The reset command does not apply to the \Chr\ tag. The \Chr=Normal\ tag is used instead.

Adding Emphasis, Tone, Intonation and Pauses

Use the following annotations or tags to choose the emphasis of words (\xWac=*n*\):

Annotation	**Tag**	**Description**
`00	\xWac=00\	reduced emphasis
`0	\xWac=0\	no emphasis
`1	\xWac=1\	normal emphasis
`2	\xWac=2\	added emphasis
`3	\xWac=3\ or \Emp\	heavy emphasis
`4	\xWac=4\	very heavy emphasis

Assigning Tones to Words

Use the following annotations or tags to assign tones to words (\xWac=*tone*\):

Annotation	**Tag**	**Description**
`al	\xWac=Low\	Low Tone
`ah	\xWac=High\	High Tone (this is the default for content words)
`af	\xWac=Falling\	Falling Tone
`ar	\xWac=Rising\	Rising Tone
`as	\xWac=Scooped\	Scooped Tone
`ad	\xWac=Downstep\	Downstepped Tone

Windows-Specific Note:

Emphasis and tone can also be combined in a single tag:
\xWac=*n, tone*\ or \xWac=*tone, n*\

Modifying Phrase-Final Intonation

Use the following annotations or tags to modify the intonation (\xPhf=*intonation*\ ):

Annotation	Tag	Description
`%	\xPhf=SmallRise\	Small pitch rise at the end of the phrase
`%%	\xPhf=ContinuationRise\	Continuation rise at the end of the phrase and low pitch on the nuclear accented word of the phrase.
`%%%	\xPhf=HighFlat\	Flat, high pitch at the end of the phrase.
`/	\xPhf=LargeFall\	Large pitch fall, as at the end of a paragraph. More perceived finality than at the end of a sentence.

Windows-Specific Note:

The \xPhf\ tag or annotation must be immediately followed by the punctuation ending the intonation phrase: either a period, comma, exclamation point, question mark, colon, or semicolon. If the required punctuation is missing, the tag is ignored.

Inserting Pauses

Use the following annotation or tag to insert a pause (\Pau=*n*\) :

Annotation	Tag	Description
`pN	\Pau=n\	Create a pause n milliseconds long.

Choosing Alternative Pronunciations

Use the following annotations or tags to set the character spelling mode (\xSpl=<*value*>\):

Annotation	Tag	Description
`ts0	\xSpl=off\	No special interpretation (default setting).
`ts1	\xSpl=alphanumeric\	Pronounce only alphanumeric characters by name.
`ts2	\xSpl=allchars\	Pronounce all characters individually by name.
`ts3	\xSpl=radio\	Pronounce alphabet characters according to the International Radio Alphabet.

Pronouncing Numbers and Years

Use the following annotations or tags to pronounce numbers and years (\xYr=<*value*>\):

Annotation	Tag	Description
`ty0	\xYr=off\	Pronounce 4-digit numbers as "nonyears."
`ty1	\xYr=on\	Pronounce 4 digit numbers as "years" (default setting).

Dictionary Processing of Abbreviations

Use the following annotations or tags to process abbreviations:

Annotation	Tag	Description
`da0	\xAbb=off\	Don't use the abbreviation dictionaries (the Internal Abbreviations Dictionary and the User's Abbreviations Dictionary)
`da1	\xAbb=on\	Use the abbreviation dictionaries (default setting).

Entering Symbolic Phonetic Representations

Use the following annotation or tag to enter SPRs:

Annotation	Tag	Description
`[*SPR*]	\xSPR=`*[SPR]*\	Pronounce the word phonetically as contained in `*[SPR]*.

Unlike the other tags, the **\xSPR** tag does not modify the word or words which follow it. Instead, it is used in place of the word for which it specifies the pronunciation.

The Engine-Specific Tag

The "Engine Specific" or \Eng\ tag provides a way to use annotations rather than tags. The tag must first contain "Eng;" then the ViaVoice Outloud engine identifier enclosed in curly brackets:

```
{F063EDA0-8C65-11CF-8FC8-0020AF14F271}
```

Follow this by a colon, followed by an annotation that is enclosed in double quotes. For example:

`\Eng;{F063EDA0-8C65-11CF-8FC8-0020AF14F271}:" `vr80 "\`	Set voice roughness to 80.
`\Eng;{F063EDA0-8C65-11CF-8FC8-0020AF14F271}:" `3 "\`	Put heavy emphasis on the following word.

Once you have issued the full engine identifier in a given text, you may eliminate it from all following **Eng** tags in that text, leaving just the **Eng** label and the annotation.

For example (note the colon after **Eng**):

```
\Eng: " `vy80 "\
\Eng: " `3 "\
```

Classification of Functions

The next chapter contains an alphabetical list of ECI functions. The following lists contain ECI functions according to their classification:

System Control

`eciDelete`
`eciNew`
`eciReset`
`eciSpeakText`

Synthesis Control

`eciAddText`
`eciClearInput`
`eciGeneratePhonemes`
`eciGetIndex`
`eciInsertIndex`
`eciPause`
`eciSpeaking`
`eciStop`
`eciSynchronize`
`eciSynthesize`
`eciSynthesizeFile`

Output Control

All of the **`eciSetOutput...()`** functions fail if they are called while **`eciSpeaking()`** is true.

`eciSetOutputBuffer`
`eciSetOutputDevice`
`eciSetOutputFilename`

Speech Environment Parameter Selection

`eciGetParam`
`eciSetParam`

Voice Parameter Control

eciCopyVoice
eciGetVoiceName
eciGetVoiceParam
eciSetVoiceParam

Dynamic Dictionary Maintenance

eciDeleteDict
eciDictFindFirst
eciDictFindNext
eciDictLookup
eciGetDict
eciGetIndex
eciLoadDict
eciNewDict
eciSaveDict
eciSetDict
eciUpdateDict

Diagnostics

eciTestPhrase
eciVersion

Callback

eciRegisterCallback

Chapter 2 Functions

eciAddText

Appends new text to the input buffer.

Syntax

```
Boolean eciAddText(
  ECIHand eciInstance,
  ECIInputText text
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. The value returned by **eciNew**.

text
: Pointer to the text you want synthesized. Your text is not modified by this function. A pointer to a null-terminated C-string. Should not be NULL.

Return Values

true
: A copy of your text was added to the input buffer.

false
: Failure. Input buffer out of memory, or internal speech synthesis engine error.

Remarks

The added text will be synthesized with the voice definition that is active at the time this call is made. The added text is appended to the end of the input buffer.

By default, when you create a new ECI instance, text that is added to the input buffer is automatically transferred to the synthesis engine and synthesized. This call returns while the engine is asynchronously synthesizing the text. You do not need to call **eciSynthesize** to start synthesis.

You can add more text while the engine is still synthesizing previously added text. Sentences may be split between calls to **eciAddText,** but words may not. When the engine empties the input buffer, it has to assume that the last word completes a sentence, even if it does not end with proper punctuation. If the input buffer is empty because your application is not adding text fast enough, and the last word is not at the end of a sentence, the user will hear an abnormal pitch contour.

You can delay the transfer of the input buffer to the synthesis engine by setting the **eciSynthMode** to manual. Added text is queued up in the input buffer until you call **eciSynthesize**. If your application adds text without regard to sentence ends, this mode may preserve correct prosody more reliably than the default, automatic mode.

Text can contain embedded annotations. Annotations control the state of the ECI instance, and are not synthesized.

Each ECI instance has a separate input buffer

Example

```
#include <stdio.h>
#include “eci.h”

//print a string to stdout and wait for any key
void showMessage( char *msg )
{
 printf( msg );
 getchar();
}

int main( int argc, char *argv[] )
{
 ECIHand myECI;
 FILE *myFP;

 myECI = eciNew();//create a new ECIHand
 if ( NULL_ECI_HAND == myECI )
  showMessage( “eciNew failed.\n” );
 else
 {
  if ( NULL != (myFP = fopen( argv[1], “rt” )) )
  {
   char buffer[1000];
   eciSetParam( myECI, eciSynthMode, 1 ); //set manual mode
   while( fgets(buffer, 1000, myFP) ) //read entire file
   {
    if ( !eciAddText(myECI, buffer) )
     showMessage( “eciAddText failed.\n” );
   }
   eciSynthesize( myECI ); //start synthesis
   eciSynchronize( myECI ); wait for synthesis complete
   bclose( myFP );
  }
  eciDelete( myECI ); //clean up
 }
}
```

See Also

eciCopyVoice, eciSetParam, eciSetVoiceParam, eciSynchronize, eciSynthesize

eciClearInput

Clears the input buffer.

Syntax

```
Boolean eciClearInput(
 ECIHand eciInstance
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. This is the value returned by **eciNew**.

Return Values

true
: The input buffer has been cleared.

false
: An error occurred. Could not clear the buffer.

Remarks

The input buffer can be cleared only if the ECI instance is in manual mode. In automatic mode, the input buffer is transferred immediately to the synthesis engine and cannot be cleared.

Other functions that clear the input buffer: **eciDelete**, **eciReset**, **eciStop**.

When this function succeeds, text that has been added to the input buffer when in manual mode, that has not already been sent to the engine via a call to **eciSynthesize**, is removed from the buffer. All resources associated with the input buffer are returned to the system.

Example

```
#include <stdio.h>
#include “eci.h”

//print a string to stdout and wait for any key
void showMessage( char *msg )
{
 printf( msg );
 getchar();
}

int main( int argc, char *argv[] )
{
 ECIHand myECI;
 FILE *myFP;

 myECI = eciNew(); //create a new ECIHand
 if ( NULL_ECI_HAND == myECI )
  showMessage( “eciNew failed.\n” );
 else
 {
  if ( NULL != (myFP = fopen( argv[1], “rt” )) )
  {
   char buffer[1000];
   eciSetParam( myECI, eciSynthMode, 1 ); //set manual mode
   while( fgets(buffer, 1000, myFP) ) //read entire file
   {
    if ( !eciAddText(myECI, buffer) )
     showMessage( “eciAddText failed.\n” );
   }
   if  (  ferror(myFP) )
   {
    showMessage( “Error reading input file\n” );
    if ( !eciClearInput(myECI) )
     showMessage( “Error clearing input buffer\n” );
   }
```

continued on next page

continued from previous page

```
      else
      {
        eciSynthesize( myECI ); //start synthesis
        eciSynchronize( myECI ); //wait for synthesis complete
      }
      fclose( myFP );
    }
    eciDelete( myECI ); //clean up
  }
}
```

See Also

eciDelete, eciReset, eciStop

eciCopyVoice

Makes a copy of a set of voice parameters.

Syntax

```
Boolean eciCopyVoice(
  ECIHand eciInstance,
  int voiceFrom,
  int voiceTo
);
```

Parameters

eciInstance

Handle to the synthesis engine instance. The value returned by **eciNew**.

voiceFrom

Voice to copy. You can copy a preset voice, or a user-defined voice that you have created, or the active voice. One of the preset voices:

1 (adult male)
2 (adult female)
3 (child)
7 (elderly female)
8 (elderly male)
Or one of the user-defined
voices 9–16.
Or 0, the active voice.

voiceTo

Voice to store the copy of *voiceFrom*. Either 0 (the active voice), or 9–16 (a user-defined voice).

Return Values

true

The voice was copied.

false

Failure. Check *voiceFrom* and *voiceTo*.

Remarks

A “voice” is a set of voice parameters. Voice 0 indicates the active voice. When you add text to the input buffer, it is synthesized with voice 0.

Each language that we support comes with five preset voices. The default voice is voice 1. When you create a new ECI instance, voice 1 is automatically copied to voice 0 and becomes the active voice. The user-defined voices are initially undefined.

You can change the parameters of the active voice, and of the user-defined voices, by calling **eciSetVoiceParam**. You cannot change any of the preset voices with **eciSetVoiceParam**. If you want to change any of the preset voices, then you must first use **eciCopyVoice** to copy it to either the active voice, or to one of the user-defined voices.

Example

```
#include <stdio.h>
#include “eci.h”

//print a string to stdout and wait for any key
void showMessage( char *msg )
{
  printf( msg );
  getchar();
}
```

continued on next page

continued from previous page

```
int main( int argc, char *argv[] )
{
  ECIHand myECI;
  int voice;
  char buffer[64];

  myECI = eciNew();
  //create a new ECIHand
  if ( NULL_ECI_HAND == myECI )
    showMessage( “eciNew failed.\n” );
  else
  {
    eciAddText(myECI, “Default voice.” );
    for( voice = 1; voice <= ECI_PRESET_VOICES; voice++ )
    {
      if ( !eciCopyVoice(myECI, voice, 0 ) )
      {
        sprintf( buffer, “Cannot not copy voice %d to 0\n”, voice );
        showMessage(  buffer );
      }
      else
      {
        sprintf( buffer, “Preset voice %d.”, voice );
        eciAddText( myECI, buffer );
      }
    }
    eciSynchronize( myECI ); //wait for synthesis complete
    eciDelete( myECI ); //clean up
  }
}
```

See Also

eciGetVoiceName, eciGetVoiceParam, eciSetVoiceParam

eciDelete

Stops and deletes an ECI instance.

Syntax

```
ECIHand eciDelete(
 ECIHand eciInstance
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. This is the value returned by **eciNew**.

Return Values

NULL_ECI_HAND
The synthesis engine instance was successfully destroyed.

Remarks

This function closes and returns to the system, all resources associated with this instance of ECI, including memory, handles, etc. If synthesis is underway when this function is called, it is terminated immediately.

Example

```
#include <stdio.h>
#include “eci.h”

void showMessage( char *msg )
{
  printf( msg );
  getchar();
}

int main( int argc, char *argv[] )
{
  ECIHand myECI;

  myECI = eciNew();
  if ( NULL_ECI_HAND == myECI )
    showMessage( “eciNew failed.\n” );
  else
  {
    showMessage( “eciNew succeeded!\n” );
    eciAddText( myECI, “This is a test.” );
    eciSynchronize( myECI );
    eciDelete( myECI );
  }
}
```

See Also

eciNew, eciReset, eciStop

eciDeleteDict

Deletes the dictionary set referenced by dictHandle.

Syntax

```
ECIDictHand eciDeleteDict(
 ECIHand eciInstance,
 ECIDictHand dictHandle
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. This is the value returned by **eciNew**.

dictHandle
Handle to the dictionary set. This is the value returned by **eciNewDict** or **eciGetDict**.

Return Values

NULL_DICT_HAND
Requested dictionary set was successfully deleted.

Remarks

On success, returns NULL_DICT_HAND. Deleting the currently active dictionary set and deactivates all dynamic dictionary lookups for this ECI instance.

Example

```
Not provided in this release.
```

See Also

eciGetDict, eciNewDict

eciDictFindFirst

Finds the first entry in a dictionary.

Syntax

```
ECIDictError eciDictFindFirst(
 ECIHand eciInstance,
 ECIDictHand  dictHandle,
 ECIDictVolume whichDictionary,
 const char** key,
 const char** translationValue
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. This is the value returned by **eciNew**.

dictHandle
Handle to the dictionary set. This is the value returned by **eciNewDict** or **eciGetDict**.

whichDictionary
Enumeration representing the particular volume of a dictionary set:

eciMainDict – main dictionary
eciRootDict – root dictionary
eciAbbvDict – abbreviation dictionary

key
Pointer to a key of the first entry in this dictionary. This is the address of the variable where a pointer to a constant C-string is returned.

translationValue
Pointer to the value of the first entry in this dictionary. This is the address of the variable where a pointer to a constant C-string is returned.

Return Values

DictNoError
    The call executed properly

DictNoEntry
    The dictionary is empty.

DictFileNotFound
    The specified file could not be found.

DictOutOfMemory
    Ran out of heap space when creating internal data structures.

DictInternalError
    An error occurred in the internal text-to-speech engine.

DictAccessError
    An error occurred when claiming operating specific resources for dictionary access.

Remarks

Starts scanning through the dictionary from the beginning and retrieves the first entry. The key and *translationValue* will receive pointers to their corresponding strings in the dictionary. These should not be modified (or deallocated), as the dictionary may become corrupted and synthesis may fail. **ECIDictError** indicates whether any errors occurred in the call to **eciDictFindFirst**.
Refer to the sections on User Dictionaries and SPR's for more information about the key and value parameters.

Parameter and return code enumerations are declared in eci.h

Example

```
Not provided in this release.
```

See Also

eciDictFindNext, eciNewDict, eciSetDict, eciUpdateDict,
Appendix C "User Dictionaries", Appendix D "SPR Symbols"

eciDictFindNext

Retrieves the next entry in the dictionary.

Syntax

```
ECIDictError eciDictFindNext(
 ECIHand eciInstance,
 ECIDictHand  dictHandle,
 ECIDictVolume whichDictionary,
 const char** key,
 const char** translationValue
);
```

Parameters

eciInstance
Hndle to the synthesis engine instance. This is the value returned by **eciNew**.

dictHandle
Handle to the dictionary set. This is the value returned by **eciNewDict** or **eciGetDict**.

whichDictionary
Enumeration representing the particular volume of a dictionary set:

eciMainDict – main dictionary
eciRootDict – root dictionary
eciAbbvDict – abbreviation dictionary

key
Pointer to a key of the next entry in this dictionary, if any. This is the address of the variable where a pointer to a constant C-string is returned.

translationValue
Pointer to the value of the next entry in this dictionary, if any. This is the address of the variable where a pointer to a constant C-string is returned.

Return Values

DictNoError
: The call executed properly.

DictNoEntry
: No more entries.

DictFileNotFound
: The specified file could not be found.

DictOutOfMemory
: Ran out of heap space when creating internal data structures.

DictInternalError
: An error occurred in the internal text-to-speech engine.

DictAccessError
: An error occurred when claiming operating specific resources for dictionary access.

Remarks

Retrieves the next entry in the dictionary. The input and output parameters have the same meaning as they do for the **eciDictFindFirst** function. This function returns DictNoEntry if there are no more entries in the dictionary. The first call to this function should be preceded by a call to **eciDictFindFirst**. Entries are not returned in any particular order. An **ECIDictError** is returned which indicates any errors that occurred in the call to **eciDictFindNext**.

Parameter and return code enumerations are declared in eci.h.

Example

```
Not provided in this release.
```

See Also

eciDictFindFirst, eciNewDict, eciUpdateDict

eciDictLookup

Returns a pointer to the translation value for key.

Syntax

```
const char* eciDictLookup(
  ECIHand eciInstance,
  ECIDictHand dictHandle,
  ECIDictVolume whichDictionary,
  const char* key
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. The value returned by **eciNew**.

dictHandle
Handle to the dictionary set. The value returned by **eciNewDict** or **eciGetDict**.

whichDictionary
Enumeration representing the particular volume of a dictionary set:

eciMainDict – main dictionary
eciRootDict – root dictionary
eciAbbvDict – abbreviation dictionary

key
Pointer to a key of the entry whose value you want. This is a null-terminated C-string containing the key.

Return Values

NULL
The key is not in the dictionary.

Remarks

Returns a pointer to the translation value for key or NULL if the key is not in the dictionary. The string referenced by the return value should not be modified, as the dictionary may become corrupted and synthesis may fail.

Parameter and return code enumerations are declared in eci.h.

Example

```
Not provided in this release.
```

See Also

eciNewDict, eciUpdateDict

eciGeneratePhonemes

Converts text to phonemes.

Syntax

```
Boolean eciGeneratePhonemes(
  ECIHand eciInstance,
  int size,
  char* phonemeBuffer
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. This is the value returned by **eciNew**.

size
: Size of buffer (in bytes). 0 cancels phoneme generation

phonemeBuffer
: Pointer to the buffer to receive phonemes. NULL cancels phoneme generation.

Return Values

true
: Phoneme generation was successfully performed.

false
: Failure of callback, or synthesis engine busy, prevented generation of phonemes.

Remarks

The **eciSynthMode** should be set to manual before text is added to the input buffer. If you fail to do this, text may be synthesized automatically (i.e., dequeued and processed) before you call **eciGeneratePhonemes**. The **eciSynthMode** is set with **eciSetParam**. Text is added to the input buffer with **eciAddText**.

A callback should be registered with **eciRegisterCallback** before you call **eciGeneratePhonemes**. If a callback has not been registered, or synthesis is already underway, this function returns false; otherwise, phoneme generation is performed for the text in the input buffer.

The text in the synthesis engine's input buffer is converted to phonemes and placed in the phoneme buffer that you designate. When your buffer is full, or all the text has been converted, whichever comes first, your callback is called with an **eciPhonemeBuffer** message. If your phoneme buffer cannot hold all the generated phonemes, your callback is called repeatedly. You need to process the contents of your phoneme buffer every time your callback is called; otherwise, the contents may be overwritten as the input text continues to be converted to phonemes.

This function should not be used to get mouth position data for facial animation, because indices are not reported while phonemes are generated. Use normal synthesis instead, and the **eciPhonemeIndexReply** message.

eciGeneratePhonemes returns synchronously when phoneme conversion is complete**.**

Example

```
Not provided in this release.
```

See Also

eciAddText, eciRegisterCallback, eciSetParam, eciSpeaking

eciGetDict

Returns the handle of the active dictionary set.

Syntax

```
ECIDictHand eciGetDict(
 ECIHand eciInstance
);
```

Parameters

eciInstance
    Handle to the synthesis engine instance. This is the value returned by **eciNew**.

Return Values

Non-null
    A valid dictionary set handle.

NULL_DICT_HAND
    There is no active dictionary set for this instance.

Remarks

Gets the handle for the currently active dictionary set, or NULL_DICT_HAND if there is no active set. The synthesis engine will not perform any dynamic dictionary lookups until a dictionary set is established as the current set.

Example

```
Not provided in this release.
```

See Also

eciNewDict, eciSetDict

eciGetIndex

Returns the last index reached in the output buffer.

Syntax

```
int eciGetIndex(
 ECIHand eciInstance
);
```

Parameters

eciInstance
    Handle to the synthesis engine instance. This is the value returned by **eciNew**.

Return Values

0
    No indices have been encountered yet.

non-zero
    The last index encountered in the output buffer.

Remarks

Returns the last index reached in the output buffer, or 0 if no index has been encountered. All inserted indices must be nonzero integer values; thus, the return value of **eciGetIndex** is unambiguous.

Example

```
Not provided in this release.
```

See Also

eciInsertIndex

eciGetParam

Returns the value of an environment parameter.

Syntax

```
int eciGetParam(
 ECIHand eciInstance,
 ECIParam eciParameter
);
```

Parameters

eciInstance
  Handle to the synthesis engine instance. This is the value returned by **eciNew**.

eciParameter
  The parameter whose value you want. See the ECIParam enumeration in eci.h:

  eciSynthMode
  eciInputType
  eciTextMode
  eciDictionary
  eciSampleRate
  eciWantPhonemeIndices
  eciRealWorldUnits
  eciLanguageDialect
  eciNumberMode

Return Values

>= 0
  The eciParameter value.

-1
  An error. The eciParameter is out of range.

Remarks

Gets the value of an environment parameter. Returns the value on success or -1 on failure.

Example

```
Not provided in this release.
```

See Also

Speech Environment Parameters

eciGetVoiceName

Copies the voice name to name buffer.

Syntax

```
Boolean eciGetVoiceName(
 ECIHand eciInstance,
 int voiceNum,
 char* nameBuffer
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. This is the value returned by **eciNew**.

voiceNum
: Number of the voice whose name you want. 1, 2, 3, 7, 8, 9-16.

nameBuffer
: Pointer to the buffer where a null-terminated C-string containing the name will be copied. Non-NULL.

Return Values

true
: The text string was successfully copied to the nameBuffer.

false
: An error occurred preventing the text string from being copied.

Remarks

Your *nameBuffer* should be ECI_VOICE_NAME_LENGTH + 1 bytes long. Otherwise, a long voice name may corrupt memory.

Example

```
Not provided in this release.
```

See Also

eciSetVoiceParam, Voice Definitions

eciGetVoiceParam

Returns a voice parameter.

Syntax

```
int eciGetVoiceParam(
 ECIHand eciInstance,
 int voiceNum,
 ECIVoiceParam voiceParameter
);
```

Parameters

eciInstance

Handle to the synthesis engine instance. This is the value returned by **eciNew**.

voiceNum

The number of the voice whose parameter value you want. 0, 1, 2, 3, 7, 8, 9-16

voiceParameter

The voice parameter whose value you want. See the ECIVoiceParam enumeration in eci.h:

eciGender
eciHeadSize
eciPitchBaseline
eciPitchFluctuation
eciRoughness
eciBreathiness
eciSpeed
eciVolume

Return Values

>= 0

The voice parameter value you requested.

-1

The voice number or voice parameter was out of range.

Remarks

Gets the specified voice parameter value for the specified voice. A voice of 0 indicates the active voice. Parameter values range from 0 to 100, except when using eciRealWorldUnits. See Voice definitions for more information.

Example

```
Not provided in this release.
```

See Also

eciSetVoiceParam, Voice Definitions.

eciInsertIndex

Inserts an index into the input buffer.

Syntax

```
Boolean eciInsertIndex(
  ECIHand eciInstance,
  int indexNum
);
```

Parameters

eciInstance

Handle to the synthesis engine instance. This is the value returned by **eciNew**.

indexNum

Unique index number that you want to insert. If it is not unique, you may be unable to later determine which index is being returned.

Return Values

true

Index inserted successfully.

false

Error inserting index. Out of memory, or bad *eciInstance* handle.

Remarks

Appends an index, with the specified number, to the input buffer. After all the text prior to this index has been synthesized, an **eciIndexReply** message containing this index number is sent to your callback function. If you are synthesizing to an audio device, then the index reply message is sent at about the same time the text is being heard on the speakers.

Indices must be nonzero integer values.

If no callback has been registered, the index reply message cannot be sent. You can still query the latest index with **eciGetIndex**.

Example

```
Not provided in this release.
```

See Also

eciGetIndex, eciRegisterCallback

eciLoadDict

Loads the dictionary volume.

Syntax

```
ECIDictError eciLoadDict(
 ECIHand eciInstance,
 ECIDictHand dictHandle,
 ECIDictVolume whichDictionary,
 const char* filename
);
```

Parameters

TeciInstance
  Handle to the synthesis engine instance. This is the value returned by **eciNew**.

dictHandle
  Handle to the dictionary set. This is the value returned by **eciNewDict** or **eciGetDict**.

whichDictionary
  Enumeration representing the particular volume of a dictionary set:

  eciMainDict – main dictionary
  eciRootDict – root dictionary
  eciAbbvDict – abbreviation dictionary

filename
  Pointer to the name of a dictionary file, which may contain a path. This is a null-terminated C-string containing the name of a dictionary file.

Return Values

DictNoError
  The call executed properly

DictFileNotFound
  The specified file could not be found.

DictOutOfMemory
  Ran out of heap space when creating internal data structures.

DictInternalError
  An error occurred in the internal text-to-speech engine.

DictAccessError
  An error occurred when claiming operating specific resources for dictionary access.

Remarks

Loads the dictionary volume identifed by whichDictionary from the file named by *filename*. The input file consists of ASCII text with one dictionary entry on each line. Each input line contains a key and the corresponding translation value, separated by a tab character. The translation value may be one or more words or phonembes. An **eciDictError** is returned, which indicates any errors that occurred in the call to **eciLoadDict**.

Example

```
Not provided in this release.
```

See Also

eciDeleteDict, eciDictFindFirst, eciDictFindNext, eciDictLookup, eciNewDict, eciSaveDict, eciSetDict, Appendix C “User Dictionaries”, Appendix D “SPR Symbols”

eciNew

Creates a new instance of ECI and returns a handle to it.

Syntax

```
ECIHand eciNEW(
 void
);
```

Parameters

None.

Return Values

ECIHand

A handle representing one instance of the synthesis engine. This same value must be used in all subsequent calls to this instance of the synthesis engine, such as **eciAddText**, **eciSynthesize**, **eciSynchronize**, **eciDelete**, and so on.

NULL_ECI_HAND

Indicates that an unrecoverable error occurred.

Remarks

This is typically the first call you will make to the ECI API. It creates a new ECI instance with default attributes. This includes setting the synthesis language to the lowest-numbered language and voice installed on the system.

Example

```
#include <stdio.h>
#include “eci.h”

void showMessage( char *msg )
{
  printf( msg );
  getchar();
}

int main( int argc, char *argv[] )
{
  ECIHand myECI;

  myECI = eciNew();
  if ( NULL_ECI_HAND == myECI )
    showMessage( “eciNew failed.\n” );
  else
  {
    showMessage( “eciNew succeeded!\n” );
    eciAddText( myECI, “This is a test.” );
    eciSynchronize( myECI );
    eciDelete( myECI );
  }
}
```

See Also

eciAddText, eciDelete, eciSynchronize, System State Variables

eciNewDict

Creates a new dictionary set for the given ECI handle.

Syntax

```
ECIDictHand eciNewDict(
 ECIHand eciInstance
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. This is the value returned by **eciNew**.

Return Values

eciDictHand
: Handle to the dictionary set. Use this handle in subsequent dictionary calls that require an eciDictHand.

NULL_DICT_HAND
: The dictionary set could not be created.

Remarks

Creates a new dictionary set for the given eci handle. The main, root, and abbreviation dictionaries are empty. ViaVoice Outloud will not look up entries in the new dictionary until it is activated with a call to **eciSetDict**. Returns NULL_DICT_HAND if the dictionary set could not be created.

Example

```
Not provided in this release.
```

See Also

eciDeleteDict, eciLoadDict, Appendix C “User Dictionaries”, Appendix D “SPR Symbols”

eciPause

Pauses or unpauses speech systhesis and playback.

Syntax

```
Boolean eciPause(
  ECIHand eciInstance,
  Boolean bPauseon
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. The value returned by **eciNew**.

bPause
Boolean value that indicates whether to pause or resume.

Return Values

true
Synthesis thread and output devices are successfully paused.

false
Speech resumes where it left off.

Remarks

If the variable *bPause* is set to true, the synthesis thread is paused and the output device is paused. When paused, no output is sent to the audio device or to your callback function. If the variable *bPause* is set to false, the speech resumes where it left off.

Example

```
Not provided in this release.
```

See Also

N/A

eciRegisterCallback

Registers your callback function with the API.

Syntax

```
void eciRegisterCallback(
 ECIHand eciInstance,
 ECICallback pCallback
 void* data
);
```

Parameters

eciInstance
Handle to the synthesis engine instance. This is the value returned by **eciNew**.

bCallback
Nname of your callback function. A function pointer, or NULL.

data
Pointer to an arbitrary value, or a key to the size of a void pointer. All values are allowed.

Return Values

None.

Remarks

This function registers your callback function with an instance of the synthesis engine. if *pCallback* is NULL, the current callback is removed. The supplied data pointer is associated with your callback. it is passed back to your callback function on entry, so that your function can use it, if desired, for identification purposes, such as a class pointer or an instance reference. Only one callback function can be registered at a time with each ECI instance.

Your callback must be registered with **eciRegisterCallback** before any function that creates messages is called. The functions that cause messages to be sent to your callback are **eciGeneratePhonemes**, **eciInsertIndex**, and **eciSetOutputBuffer**.

For any given ECI Instance, your callback will be called from the same thread on which your applicsation calls ECI. This is achieved by passing control from your application to the API. When you call eciSynchronize, the API will retrieve all messages and execute your callback for each one, until synthesis is complete. If you call **eciSpeaking**, the API will retrieve just those messages that are ready, execute your callback for each one, and then return. If you choose to use **eciSpeaking**, instead of **eciSynchronize**, you must keep calling it until it returns false.

The syntax of your callback is as follows:

```
ECICallbackReturn callback(
 ECIHand eciInstance,
 ECIMessage msg,
 long lparam,
 void* data
);
```

Parameters

eciInstance
: Handle to the synthesis engine instance. This is the value returned by **eciNew**.

msg
: Enumeration indicating the type of message (see eci.h):

  eciWaveformBuffer
  eciPhonemeBuffer
  eciIndexReply
  eciPhonemeIndexReply

lparam
: A long whose value and interpretation depends on the ECIMessage type. See discussion below.

data
: An arbitrary value, or key, which is the size of a void pointer. You specify this value in your call to **eciRegisterCallback**. All values are allowed.

Return Values

eciDataProcessed

You have processed the message and any associated data in your output buffer. Subsequent messages may be sent to your callback.

eciDataNotProcessed

You could not process the message, or associated data in your output buffer. The same message will be sent to your callback later.

If your callback processes the ECIMessage, and does not wish to see that same message again, it should return eciDataProcessed. If your callback function cannot process the message, and would like to see the same message again, it should return eciDataNotProcessed; your callback will be called with the same message at a later time. This is particularly useful if an eciWaveformBuffer message can not be processed because the sink you are writing to is temporarily full. No new ECIMessage will be sent if eciDataNotProcessed is returned. If your application continues to return eciDataNotProcessed, synthesis will stop.

All callbacks should return quickly to ensure that there is no interruption of output.

The value and interpretation of lparam is dependent on ECMessage.

ECIMessage eciWaveformBuffer:

lparam indicates the number of samples (not bytes) that have just been added to your output buffer. Your output buffer is specified in a call to **eciSetOutputBuffer**.

This message is sent with lparam = 0 when all the waveform samples for the text in the input buffer have been synthesized. When phoneme indices are also being generated, this message is sent for the samples for each phoneme.

Samples are 16-bit signed PCM values and are centered at 0.

Once your callback returns eciDataProcessed, the data in your output buffer is no longer protected; therefore, your callback should only return eciDataProcessed when it has processed all the data in your buffer. No more data will be added to your buffer until eciDataProcessed is returned.

ECIMessage eciPhonemeBuffer:

lparam indicates the number of characters (bytes) that have just been added to your phoneme buffer. Your phoneme buffer address was given to the API in your call to **eciGeneratePhonemes**.

Once your callback returns eciDataProcessed, the data in your phoneme buffer is no longer protected; therefore, your callback should only return eciDataProcessed when it has processed all the data in the buffer. More data will not be added to your phoneme buffer until eciDataProcessed is returned.

ECIMessage eciIndexReply:

lparam is an index that was reached during synthesis and playback of the input text buffer. Indices are inserted into the input text buffer with **eciInsertIndex**.

Your callback should return immediately to ensure that there is no interruption of output.

ECIMessage eciPhonemeIndexReply:

lparam is a pointer to an **ECIMouthData** structure.

This message is sent only when the eciWantPhonemeIndices environment parameter is set to 1. One of these messages is sent for each phoneme spoken, just before the phoneme starts playing on the audio device (or just before the associated waveform audio is placed in your output buffer, if you have called **eciSetOutputBuffer**).

In addition to the language-specific phoneme symbols , the symbol '¤' is used to indicate the end of an utterance, and is sent with a set of neutral mouth position parameters.

Receiving phoneme notifications this way is appropriate for synchronizing facial animation or other graphics. If your application only needs to synchronize with individual words or larger units, use word indices (eciIndexReply messages). To convert text to phonemes, use the **eciGeneratePhonemes** function and the eciPhonemeBuffer message will be sent.

As with other ECIMessage's, your callback function must return quickly. If significant processing is required, as with complex graphics, your application should spawn a new thread, and marshal the callback messages to the new thread. Your application is responsible for skipping ECIMessage's, if it receives them faster than it can process them.

Examples

The following example C++ function converts an input string to phonemes and writes them to cout:

```
#include <eci.h>
#include <iostream.h>

const int bufferSize = 100;
static char buffer[bufferSize];

static ECICallbackReturn callback(ECIHand eciHand,
      ECIMessage msg, long lparam, void* data)
{
  if (msg == eciPhonemeBuffer)
  {
    cout << buffer;
  }
  return eciDataProcessed;
}

void showPhonemes(const char* text)
{
  //This function demonstrates the proper use of
  //eciGeneratePhonemes.  The standard error checking of
  //ECI functions which return error codes is left out for
  //the sake of simplicity
  ECIHand eciHand = eciNew();

  eciRegisterCallback(eciHand, callback, 0);
  eciSetParam(eciHand, eciSynthMode,1);
  eciAddText(eciHand, text);
  eciGeneratePhonemes(eciHand, bufferSize, buffer);
  eciDelete(eciHand);
  return;
}
```

This C++ example illustrates how to capture the waveform samples in order to process them using an alternate method, rather than sending them directly to the audio device:

```
#include <eci.h>
#include <iostream.h>

const int bufferSize = 100;
static short buffer[bufferSize];

// This function performs some kind of processing on the samples
// It returns true if it is done with those samples,
// and false if it wants to be called with the same samples again.
extern Boolean handleSamples(const short* samples, long count);

static ECICallbackReturn callback(ECIHand eciHand,
      ECIMessage msg, long lparam, void* data)
{
  Boolean retval = true;
  if (msg == eciWaveformBuffer)
  {
    retval = handleSamples(buffer, lparam);
  }
  return (retval?eciDataProcessed:eciDataNotProcessed);
}
void collectSamples(const char* text)
{
  //This function demonstrates the proper use of the
  //eciWaveformBuffer message.  The standard error checking
  //of ECI functions which return error codes is left out
  //for the sake of simplicity

  ECIHand eciHand = eciNew();

  eciRegisterCallback(eciHand, callback, 0);
  eciSetOutputBuffer(eciHand, bufferSize, buffer);
  eciAddText(eciHand, text);
  eciSynthesize(eciHand);
  //Wait until synthesis is complete
  eciSynchronize (eciHand);
  eciDelete(eciHand);
}
```

This C++ program uses word indices to synchronize a visible countdown with an audible one:

```
#include <iostream.h>
#include <stdio.h>
#include <eci.h>

static ECICallbackReturn callback(ECIHand eciHand,
      ECIMessage msg, long lparam, void* data)
{
 if (msg == eciIndexReply
 {
  cout << param << "..." << endl;
 }
 return eciDataProcessed;
}

int main(int argc, const char* argv[])
{
 //This function demonstrates the proper use of the
 //eciIndexReply message.  The standard error checking
 //of ECI functions which return error codes is left out
 //for the sake of simplicity

 ECIHand eciHand = eciNew();

 eciRegisterCallback(eciHand, callback, 0);
 // count from 10 to 0, inserting an index at each count
 for (int i = 10; i > 0; i--)
 {
  char buf[10];
  sprintf(buf, "%d, ", i);
  eciInsertIndex(eciHand, i);
  eciAddText(eciHand, buf);
 }
 eciInsertIndex(eciHand, 0);
 eciAddText(eciHand, "go!");
```
continued on next page

```
  continued from previous page
  // synthesize, and wait until speaking is finished
  eciSynthesize(eciHand);
  //Wait until synthesis is complete
  eciSynchronize (eciHand);
  cout << "Gone." << endl;

  eciDelete(eciHand);
  return 0;
}
```

Phoneme indices may be used in a similar way, except that there is no need for calls to **eciInsertIndex**. The callback function might look like this:

```
static ECICallbackReturn callback(ECIHand eciHand,
      ECIMessage msg, long lparam, void* data)
{
  if (msg == eciPhonemeIndexReply
  {
    const ECIMouthData* mouthData = (ECIMouthData*)lparam;

    // process the mouth position data--here, we'll just
    // print the phoneme name
    cout << ((ECIMouthData*)param)->phoneme << endl;
  }
  return eciDataProcessed;
}
```

See Also

eciSpeaking, eciSynchronize, Data Types, Appendix D “SPR Symbols”

eciReset

Resets the ECI instance to the default state.

Syntax

```
Boolean eciReset(
 ECIHand eciInstance
);
```

Parameters

eciInstance
 A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

true
 Success

false
 Failure

Remarks

This function resets the ECI instance to the default state. This is the same state that a new instance returned by **eciNew** has. It includes setting the synthesis language to the lowest-numbered language and voice installed on the system.

If synthesis is underway when this function is called, synthesis is terminated immediately.

Example

```
Not provided in this release.
```

See Also

eciNew, System State Variables

eciSaveDict

Writes the contents of a dictionary volume to a file.

Syntax

```
ECIDictError eciSaveDict(
 ECIHand eciInstance,
 ECIDictHand dictHandle,
 ECIDictVolume whichDictionary,
 const char* filename
);
```

Parameters

eciInstance.
: A handle to the synthesis engine instance. The value returned by **eciNew.**

dictHandle
: A handle to the dictionary set. The value returned by **eciNewDict** or **eciGetDict.**

whichDictionary
: An enumeration representing the particular volume of a dictionary set:

  eciMainDict -- main dictionary
  eciRootDict -- root dictionary
  eciAbbvDict -- abbreviation dictionary

filename
: The name of a dictionary file. A pointer to a null-terminated C-string containing the name of a dictionary file.

Return Values

DictNoError
: Call executed properly

DictFileNotFound
: The specified file could not be found.

DictOutOfMemory
  Ran out of heap space when creating internal data structures.

DictInternalError
  Error occurred in the internal text-to-speech engine.

DictAccessError
  Error occurred when claiming operating system specific resources for dictionary access.

Remarks

Writes the contents of the dictionary volume to the named file. The ASCII file will be in a format suitable for reloading with **eciLoadDict**. The entries are listed in no particular order and will generally be different from the order entered or loaded. An ECIDictError is returned, which indicates any errors that occurred in the call to **eciSaveDict**.

Example

```
Not provided in this release.
```

See Also

eciLoadDict, Appendix C “User Dictionaries”

eciSetDict

Sets the dictionary set referenced by dictHandle as the current dictionary set for the given ECI instance.

Syntax

```
ECIDictError eciSetDict(
 ECIHand eciInstance,
 ECIDictHand dictHandle
);
```

Parameters

eciInstance
  A handle to the synthesis engine instance. The value returned by **eciNew**.

dictHandle
  A handle to the dictionary set. The value returned by **eciNewDict** or **eciGetDict**.

Return Values

DictNoError
  Call executed properly.

DictOutOfMemory
  Ran out of heap space when creating internal data structures.

DictInternalError
  Error occurred in the internal text-to-speech engine.

DictAccessError
  Error occurred when claiming operating system specific resources for dictionary access.

Remarks

Sets the dictionary set referenced by dictHandle as the current dictionary for the given ECI handle. The main, root, and abbreviation dictionaries will become active. Passing this function a NULL_DICT_HAND for the dictHandle deactivates all dynamic dictionary lookups. Maintenance on the dictionary set may be performed at any time. An ECIDictError is returned which indicates any errors that occurred in the call to **eciSetDict**.

Example

```
Not provided in this release.
```

See Also

eciGetDict, eciLoadDict, eciNewDict, Appendix C “User Dictionaries”

eciSetOutputBuffer

Sets an output buffer as the synthesis destination.

Syntax

```
Boolean eciSetOutputBuffer(
 ECIHand eciInstance,
 int size,
 short* buffer
);
```

Parameters

eciInstance
: A handle representing the synthesis engine instance. The value returned by **eciNew**.

size
: The size of your buffer, in samples. Use 0 to cancel waveform buffer callbacks.

buffer
: A pointer to the buffer where you want to receive PCM audio samples. Use NULL to cancel waveform buffer callbacks.

Return Values

true
: Successfully set the output buffer.

false
: A callback has not been registered.

Remarks

Registers your output buffer to receive 16-bit signed PCM audio samples. Signed PCM samples are centered on 0. Calling this function with a size of 0 or a NULL buffer pointer reverts to the default destination and cancels waveform buffer callbacks. Otherwise, when the buffer is full or speech is finished, the eciWaveformBuffer message will be sent to your callback.

If a callback has not been registered, returns false. When an output buffer is successfully registered, device and file output are cancelled.

Example

```
Not provided in this release.
```

See Also

eciRegisterCallback, eciSetOutputDevice, eciSetOutputFilename

eciSetOutputDevice

Sets an audio output hardware device as the synthesis destination.

Syntax

```
Boolean eciSetOutputDevice(
 ECIHand eciInstance,
 int deviceNum
);
```

Parameters

eciInstance
: A handle to the synthesis engine instance. The value returned by **eciNew**.

deviceNum
: The hardware device number. The default device. Values: >= 0 or -1

Return Values

true
: The synthesis destination was successfully set.

false
: An error occurred that prevents output from being sent to this audio device.

Remarks

Sets the specified audio device as the destination for synthesis. On Windows, the number of devices is returned by the **waveOutGetNumDevs** call, and devices are numbered sequentially starting with 0. If **deviceNum** is -1, the default destination is used. When a device is successfully set, buffer and file output are cancelled.

Example

```
Not provided in this release.
```

See Also

eciSetOutputBuffer, eciSetOutputFilename

eciSetOutputFilename

Sets a file as the synthesis destination.

Syntax

```
Boolean eciSetOutputFilename(
  ECIHand eciInstance,
  const char* filename
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

filename
The name of the file where audio samples will be stored. May contain a path. A pointer to a null-terminated C-string.

Return Values

true
The output file was successfully set.

false
An error occurred that prevented setting the output file.

Remarks

Sets the output file as the synthesis destination. If the filename is NULL, or the C-string is empty, the synthesis engine reverts to the default destination. The extension of the name determines the format of the audio samples written to the file:

.

.WAV:	CM Sound in Windows Format
.AU:	u-law with header
.RAU:	raw u-law without header

If the file does not exist, it is created. If the file already exists, output is appended to the existing contents. ECI should only be used in append mode if it is appending to a file that was created by ECI, since it does not currently support all features of complex “.wav” file formats.

When a file is successfully set, buffer and device output are cancelled.

Example

```
Not provided in this release.
```

See Also

eciSetOutputBuffer, eciSetOutputDevice

eciSetParam

Sets an environment parameter.

Syntax

```
int eciSetParam(
  ECIHand eciInstance,
  ECIParam eciParameter,
  int value
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

eciParameter
The parameter whose value you want to set. See the ECIParam enumeration in eci.h:

eciSynthMode
eciInputType
eciTextMode
eciDictionary
eciSampleRate
eciWantPhonemeIndices
eciRealWorldUnits
eciLanguageDialect
eciNumberMode

value
The value to want to set the parameter to. Usually 0 or 1. See Speech Environment Parameters.

Return Values

>=0

Success. Returns the previous value.

-1

Failure. Check eciParameter and value.

Remarks

Sets an environment parameter. Does not affect text already in the input buffer. Returns the previous value on success, -1 on failure.

Example

```
Not provided in this release.
```

See Also

eciAddText, eciGetParam

eciSetVoiceParam

Sets a voice parameter.

Syntax

```
int eciSetVoiceParam(
  ECIHand eciInstance,
  int voiceNum,
  ECIVoiceParam voiceParameter,
  int value
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

voiceNum
The number of the voice whose parameter value you want to set.

0, 9-16 (see Voice Definitions).

voiceParameter
The voice parameter whose value you want. See the ECIVoiceParam enumeration in eci.h:

eciGender
eciHeadSize
eciPitchBaseline
eciPitchFluctuation
eciRoughness
eciBreathiness
eciSpeed
eciVolume

value
The value that you want the voice parameter to have. 0 – 100, except for speed, and except Real World units.

Return Values

>=0

Success. Returns the previous value.

-1

Error.

Remarks

Sets the specified voice parameter value for the specified voice. A voice of 0 indicates the active voice. Parameter values range from 0 to 100, except when using eciRealWorldUnits. See Voice definitions for more information.

Returns the previous parameter value, or -1 for error. On error, check the voice number, parameter number, and value.

Example

```
Not provided in this release.
```

See Also

eciGetParam, Voice Definitions

eciSpeaking

Returns true if synthesis is in progress.

Syntax

```
Boolean eciSpeaking(
 ECIHand eciInstance
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

true
Synthesis is in progress.

false
Synthesis is not in progress.

Remarks

Use this function to poll the ECI instance for synthesis in progress. This function provides an alternative to blocking your thread's execution on **eciSynchronize** during synthesis.

When your application calls **eciSpeaking**, it gives the API an opportunity to check for messages from the synthesis engine. If you have registered a callback, and there are one or more callback messages from the engine, your callback will be executed on your thread from within this function.

If accurate timing and synchronization are issues for your application, then you should call this function frequently enough that messages and callbacks from the engine can be serviced in a timely manner.

Example

```
Not provided in this release.
```

See Also

eciRegisterCallback, eciSynchronize

eciSpeakText

Synthesizes text to the default audio device.

Syntax

```
Boolean eciSpeakText(
  ECIInputText szTextPhrase,
  Boolean bAnnotations
);
```

Parameters

szTextPhrase.
  The text that is to be spoken. A pointer to a null-terminated C-string

bAnnotations
  A Boolean value that indicate whether annotations are embedded in szTextPhrase:

  True indicates annotations are in szTextPhrase and are to be interpreted.
  False indicates annotations are not in szTextPhrase.

Return Values

true
  Successfully spoke the requested string.

false
  An error occurred and the requested string was not spoken.

Remarks

Creates a new ECI instance, speaks the null-terminated C-string szTextPhrase to the default output device, and destroys the ECI instance it created.

When bAnnotations is true, annotations within the text are processed, and the voice and speaking characteristics are changed as appropriate (see Appendix C “User Dictionaries” for a list of annotations).

For example, if you have English and French installed, and the text in szTextPhrase is in French, you can insert an annotation indicating that the French TTS engine should be used. See the example.

If you set bAnnotations to false, but there are annotations in szTextPhrase, then the synthesis engine will attempt to read them as normal text.

Example

```
#include <stdio.h>
#include “eci.h”

void showMessage( char *msg )
{
  printf( msg );
  getchar();
}

int main( int argc, char *argv[] )
{
  ECIInputText szTextPhrase = “Hello world. `l3.0 Bonjour le monde.”
  if ( eciSpeakText(szTextPhrase, true) )
    showMessage( “Success!\n” );
  else showMessage( “Failed.\n” );
  return 0;
}
```

See Also

eciAddText, eciDelete, eciNew, eciSynchronize

eciStop

Stops synthesis.

Syntax

```
Boolean eciStop(
 ECIHand eciInstance
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

true
Synthesis is successfully stopped, and the input and output buffers are cleared.

false
An error occurred that failed to stop synthesis.

Remarks

Aborts any synthesis in progress, clears the input buffer, clears the output buffer, and releases the audio device if it has been claimed.

This function is synchronous, so ECI will have stopped processing before **eciStop** returns.

If the active voice changed during synthesis, and synthesis is preempted by a call to **eciStop**, the state of the active voice is undefined when **eciStop** returns. The active voice should be reset by your application to an appropriate setting.

Example

```
Not provided in this release.
```

See Also

eciPause, eciSpeaking, eciSynthesize

eciSynchronize

Waits until synthesis is complete.

Syntax

```
void eciSynchronize(
  ECIHand eciInstance
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

None.

Remarks

Waits in an efficient state until all synthesis is finished. If the synthesis destination is a device, this function does not return until the audio has been played on the device.

While waiting, the API calls your callback, if you have registered one, and if the synthesis engine produces any messages for your callback.

Alternatives to **eciSynchronize** are:

- Insert an index using **eciInsertIndex** and wait in a message loop until your callback function is called with that index.
- Wait in a loop that polls **eciSpeaking** and processes your own application messages.

Example

```
Not provided in this release.
```

See Also

eciInsertIndex, eciSpeaking

eciSynthesize

Starts synthesis of text in the input buffer.

Syntax

```
Boolean eciSynthesize(
 ECIHand eciInstance
);
```

Parameters

eciInstance

A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

true

Synthesis has started.

false

An error occurred that prevented synthesis from starting.

Remarks

Starts synthesis of all text in the input buffer. Returns immediately. It is important to call this function after the last text of an utterance has been passed to **eciAddText,** if the synthesis mode is manual**,** so that the synthesis process will begin.

Example

```
Not provided in this release.
```

See Also

eciAddText, eciSynchronize, Speech Environment Parameters.

eciSynthesizeFile

Synthesizes the contents of a file.

Syntax

```
Boolean eciSynthesizeFile(
  ECIHand eciInstance,
  const char* filename
);
```

Parameters

eciInstance
: A handle to the synthesis engine instance. The value returned by **eciNew**.

filename
: The name of the text file whose contents you want to synthesize. May contain a path. A pointer to a null-terminated C-string.

Return Values

true
: Successfully opened the file.

false
: The file does not exist.

Remarks

Opens the named file and starts reading its contents. **eciSynthesizeFile** returns immediately. If the file does not exist, it returns false with no other error flags set.

This function is equivalent to sending all the text in the named file using **eciAddText,** followed by **eciSynthesize.**

Your application should wait for synthesis to complete before terminating. Use **eciSpeaking** or **eciSynchronize** for this purpose.

Example

```
Not provided in this release.
```

See Also

eciSpeaking, eciSynchronize

eciTestPhrase

Synthesizes a test phrase.

Syntax

```
Boolean eciTestPhrase(
 ECIHand eciInstance
);
```

Parameters

eciInstance
: A handle to the synthesis engine instance. The value returned by **eciNew**.

Return Values

true
: Success

false
: Failure

Remarks

Aborts synthesis if it is underway, clears the input and output buffers, sets the active voice to preset voice 1, loads the sentence “1 2 3. ” starts synthesis, and returns.

Your application should wait for synthesis to complete before terminating. Use **eciSpeaking** or **eciSynchronize** for this purpose.

Example

```
Not provided in this release.
```

See Also

eciSpeaking, eciSynchronize

eciUpdateDict

Updates a dictionary volume with a key/translation pair.

Syntax

```
ECIDictError eciUpdateDict(
  ECIHand eciInstance,
  ECIDictHand dictHandle,
  ECIDictVolume whichDictionary,
  const char* key,
  const char* translationValue
);
```

Parameters

eciInstance
A handle to the synthesis engine instance. The value returned by **eciNew**.

dictHandle
A handle to the dictionary set. The value returned by **eciNewDict** or **eciGetDict**.

whichDictionary
An enumeration representing the particular volume of a dictionary set:

eciMainDict – main dictionary
eciRootDict – root dictionary
eciAbbvDict – abbreviation dictionary

key
The key of the entry in this dictionary. A pointer to a constant C-string containing the key.

translationValue
The translation for this entry in this dictionary. A pointer to a constant C-string containing the value.

Return Values

DictNoError
    Call executed properly.

DictFileNotFound
    The specified file could not be found.

DictOutOfMemory
    Ran out of heap space when creating internal data structures.

DictInternalError
    Error occurred in the internal text-to-speech engine.

DictAccessError
    Error occurred when claiming operating system specific resources for dictionary access.

Remarks

Updates the dictionary volume. The update action depends on the existence of the key and value of the translationValue parameter:

- The entry is added when the key does not exist and the translationValue is not NULL.
- The entry is updated if the **key** already exists and the translationValue is not NULL.
- The entry is deleted if the translationValue is NULL. An ECIDictError is returned which indicates any errors that occurred in the call.

Example

```
Not provided in this release.
```

See Also

eciDictLookup, eciGetDict

eciVersion

Returns the version of the API.

Syntax

```
void eciVersion(
  char* buffer
);
```

Parameters

buffer
The buffer where the version string is returned. Must be at least 10 bytes. Should not be NULL.

Return Values

None.

Remarks

Copies the ViaVoice Outloud version number as a null-terminated C-string to the specified buffer, which must have room for at least 10 characters, including the terminating null.

Example

```
Not provided in this release.
```

See Also

None.

Appendix A

Notices

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

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

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

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

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

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

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

Trademarks

Appendix B

Annotations

An annotation is a special code that you can place in the input text to customize the output. Most annotations have equivalent tags.

A little background information illuminates the duplication between annotations and tags: The tags used in this program are part of an industry-wide standard common to all text-to-speech programs. This industry standard is known as "**SAPI**" (Speech Application Programmers Interface).

However, the text-to-speech engine you are currently using also has its own version of tags, called "annotations." Annotations predate SAPI tags, and many users have already built systems that depend solely on annotations. For this reason, we cannot simply replace annotations with tags.

Annotations have a different form than tags, but accomplish the same things. The tags are really just a layer of commands over this underlying system of annotations. Each tag is converted to an annotation "behind the scenes."

When and How Do I Use Annotations?

- The only time you must use annotations is if you want to make use of their functions within an entry for the user's special words dictionary. For example, you might use the annotation equivalent of the emphasis tag to mark a phrase as having the stress of a compound word.
- If you wish to use an annotation elsewhere (outside the Special Words Dictionary), you must enclose the annotation within the Eng tag as follows:

```
\Eng;{F063EDA0-8C65-11CF-8FC8-0020AF14F271}: " annotation "\
```

Once you have issued the full engine identifier (the long string of numbers and letters in curly brackets), you may eliminate it from all following tags, leaving just the **Eng** tag and the annotation: \Eng: " *annotation* "\. For a complete listing of the annotations and their tag equivalents, see Tags.

Form

An annotation consists of a ` (backquote) followed immediately by a string of characters. For example:

`vs5	Use speaking rate 5.
`4	Put very heavy emphasis (level 4) on the following word.
`ts2	Pronounce all characters individually by name.

Punctuation marks can immediately follow any annotation. Otherwise, all annotations (except one--the phrase-final intonation annotation) must have at least one unit of white space on either side.

A phrase-final intonation annotation must be followed immediately by a punctuation mark.

Language and Dialect

The ViaVoice Outloud text-to-speech engine is available in a number of languages, and we are adding more all the time. Using the language and dialect tags, you can specify which language and which dialect you want the text-to-speech engine to produce while processing the text.

Language and Dialect Tags

There is one tag for specifying a language, \Vce=Language=*language*\, and another tag for specifying a dialect of that language, \Vce=Dialect=*dialect* \. You can combine the two as follows:

```
\Vce=Language=language, Dialect=dialect\
```

For example:

```
\Vce=Language=Spanish, Dialect=Castilian\
```

Default Language and Dialect

- If you do not specify a language (i.e., you do not insert a language tag in your input text), one of two things can happen: either the lowest numbered language available will be used, or the application that is using ViaVoice Outloud will determine the default language for you.
- If you do not specify a dialect, the default dialect will be used. Refer to the following table for the default dialect.
- If you specify a dialect that does not exist for the current language, the tag will be ignored. For example, if the current language is French (\Vce=Language=French) and you input the tag \Vce=Dialect=British\, it will be ignored because there is no text-to-speech engine for a British dialect of French.

	Tags	**Annotations**	**Dialect or Language**
1	\Vce=Language=English\	`l1	English
	\Vce=Dialect=American\	`l1.0	American English (default)
	\Vce=Dialect=British\	`l1.1	British English

2	\Vce=Language=Spanish\	`l2	Spanish
	\Vce=Dialect=Castilian\	`l2.0	Castilian Spanish (default)
3	\Vce=Language=French\	`l3	French
	\Vce=Dialect=Standard\	`l3.0	Standard French (default)
4	\Vce=Language=German\	`l4	German
	\Vce=Dialect=Standard\	`l4.0	Standard German (default)
5	\Vce=Language=Italian\	`l5	Italian
	\Vce=Dialect=Standard\	`l5.0	Standard Italian (default)
6	\Vce=Language=Chinese\	`l6	Chinese
	\Vce=Dialect=Standard\	`l6.0	Mandarin Chinese (default)

In order for the language and dialect tags to work properly, you must have installed the text-to-speech engine of the language you which to use. For example, you must install the Standard German text-to-speech engine in order for the `\Vce=Language=German\` tag to work.

- If you hear the text being pronounced using the accent of your default language, then the language of the text has not been installed.
- If you get an error message, then the language you want to hear has probably been installed incorrectly.

Emphasis

Each word in an utterance is pronounced with a level of emphasis relative to other words in the utterance.

You can override the default emphasis patterns by placing a Word Accent (\xWac\) tag before the word you want to modify.

Tags	**Annotation**	**Description**
\xWac=00\	`00	reduced emphasis
\xWac=0\	`0	no emphasis
\xWac=1\	`1	normal emphasis
\xWac=2\	`2	added emphasis
\xWac=3\ or \Emp\	`3	heavy emphasis
\xWac=4\	`4	very heavy emphasis

Emphasis level 1, or "normal emphasis," is the default level of emphasis for a content word, and emphasis level 00 is the default emphasis for a function word. The last content word in an intonation phrase (the nuclear accent) will receive emphasis level 2, unless you annotate the utterance to change the default pattern. For example, the default emphasis pattern for the phrase "run through fields of barley" is:

Run	through	fields	of	barley
1	1	1	00	2

Uses of Emphasis Tags

Reduced Emphasis (\xWac=00\)

The reduced emphasis tag can be used to reduce a word to a function word.

No Emphasis (\xWac=0\)

When two words form a single compound word (as in "wet suit" in example (b) below), the second word receives less emphasis than the first. The no emphasis tag is used to achieve this effect.

Normal Emphasis (\xWac=1\)

The normal emphasis tag can be used to mark a word like "can" (in sentence (b)) as a content word rather than a function word.

While this tag is used to assign normal emphasis to a word that would otherwise receive no emphasis, the word will still receive the nuclear accent in appropriate contexts. You can use the added emphasis tag to shift the nuclear accent of the phrase to another word.

Added Emphasis (\xWac=2\)

Typically, the last content word in an intonational phrase receives emphasis level 2 automatically. Sometimes, however, it is more appropriate for this emphasis to fall earlier in the phrase. The *added emphasis* tag can be used to mark words in this way. Note that this causes all subsequent words to be de-emphasized.

Heavy Emphasis and Very Heavy Emphasis (\xWac=3\ and \xWac=4\)

To give added emphasis to a word, you can increase the emphasis level. Note that setting the emphasis level to 4 also causes all preceding words to be de-emphasized. The higher levels of emphasis are also useful in contradicting a previous statement or expressing incredulity.

Pause Length

Pause Length Tag

You can use the pause tag to:

Increase or reduce the pause length created by punctuation symbols

Insert pauses between words where there is no punctuation

Tag	**Annotation**	**Effect**
\Pau=N\	`pN	Creates a pause N milliseconds long.

Punctuation Pauses

By default, certain punctuation marks correspond to a number of milliseconds:

Punctuation	**Duration in Milliseconds**
periods, colons, question marks	400 ms
commas, semicolons, dashes	150 ms

You can add to a punctuation pause or replace it with a shorter pause. To <u>add to</u> a punctuation pause, follow the punctuation with a single space and then the tag. To <u>replace</u> a punctuation pause, place the punctuation immediately after the tag.

Inserting Pauses

Inserting pauses can be useful for synthesizing the hesitations that occur in natural speech.

Built-in Voices

ViaVoice Outloud provides at least five pre-defined voices for each language, and each one has a corresponding voice tag that can be inserted into the text. You may use any pre-defined voice with any language.

See also the following tags:

Gender and Age

Style

Speakers

Voice Tags

Tag	Annotation	Voice
\Vce=StoredVoice="Adult Male 1"\	`v1	Adult Male voice
\Vce=StoredVoice="Adult Male 2"\	- - -	A male with a higher-pitched voice
\Vce=StoredVoice="Adult Female 1"\	`v2	Adult Female voice
\Vce=StoredVoice="Child 1"\	`v3	Child voice
\Vce=StoredVoice="Elderly Female 1"\	`v7	Elderly Female voice
\Vce=StoredVoice="Elderly Male 1"\	`v8	Elderly Male voice

The voice tag will stay in effect until a new voice tag is entered.

- Although the tag labels are not case sensitive, the voice names are, and must be typed in exactly as indicated, including quotation marks.
- Voice names are sensitive to the currently selected User Interface Language (UIL). If you have changed your UIL, you must use the voice names that correspond to the current UIL and are listed in the General dialog box. For example, if you have changed your UIL to Deutsch (German), and want to use the tag for "Adult Male 1", you must type in: `\Vce=StoredVoice="Erwachsener Mann 1"\.`

Voice Characteristics

Individual voices derive their uniqueness from a number of physical factors. In addition, an individual's voice can take on different qualities at different times, depending on such things as mood and circumstance. You can modify these attributes with a set of voice characteristics tags.

See also the following tags:

Gender and Age
Style
Speakers

Voice Characteristics Tags

Tag	**Annotation**	**Attribute**	**Effect**
\xVct=male\	`vg0	Vocal tract (male)	Set **vocal tract** configuration to male
\xVct=female\	`vg1	Vocal tract female)	Set **vocal tract** configuration to female.
\Pit=N\	`vbN	Pitch baseline	Set **pitch baseline** to N. Tag range is 40-422 Hz.. Annotation range is 0-100.
\xHsz=N\	`vhN	Head size	Set **head-size** to N. Range is 0 (very small head) to 100 (very large head).
\xRgh=N\	`vrN	Roughness	Set **roughness** to N Range is 0 (smooth) to 100 (rough).
\xBth=N\	`vyN	Breathiness	Set **breathiness** to N Range is 0 (not breathy) to 99 (very breathy). Note: Setting breathiness to 100 yields a whisper.

\xPfl=N\	`vfN	Pitch fluctuation	Set **pitch fluctuation** to N Range is 0 (narrow=monotonic) to 100 (wide).
\Spd=N\	`vsN	Speed	Set **speed** of the utterance to N. Tag range is 70-350 words per minute. Annotation range is 0-100.
\Vol=N\	`vvN	Volume	Set speech **volume** to N. Tag range is 1 to 65535 (linear). Annotation range is 0-100.
\Rst\	(no annotation)		**Reset** the voice to the original characteristics for the selected speaker.

Voice characteristics tags affect the currently selected voice and remain in effect until a new voice or speaker is specified with the \Vce\ tag or until the same tag is used again with a different value. Restarting the program also resets all of the characteristics to their default values.

Vocal tract

Male and female vocal tracts have physical differences that affect the voice, some of which are reflected in the vocal tract setting. Other differences between male and female voices, namely pitch and head size, are controlled independently.

Pitch Baseline

Changing the pitch baseline will affect the overall pitch of the voice, where the highest pitch is associated with children, a high pitch with women, and a low pitch with men.

Head Size

This attribute controls one aspect of the deepness of the voice. Since a truly deep voice also has a low pitch baseline, ViaVoice Outloud allows you to control head size independently of pitch.

Roughness

This attribute controls the roughness or "creakiness" of the voice.

Breathiness

The maximum breathiness is a whisper.

Pitch Fluctuation

This attribute controls the degree of fluctuation in the speaker's voice. A large pitch fluctuation is characteristic of an excited speaker, and a small pitch fluctuation creates a monotone.

Speed

Speed controls the number of words spoken per minute.

Volume

The volume for built-in voices is set as loud as possible without causing distortion. (Louder settings may cause distortion when combined with other attribute changes.)

Voices for the Telephone

In order to use ViaVoice speakers over the telephone, it is best to change the sampling rate from 11kHz to 8kHz. There are two ways to do this, both of which can be found in the General Dialog box:

- Choose the "Features" tab and set audio to "Phone (8kHz)".
- In the speaker select bar, choose:
  "Reed-Tel (Adult Male for Telephone)" or
  "Shelley-Tel (Adult Female for Telephone)"

Appendix C

User Dictionaries

ViaVoice Outloud allows you to specify explicit pronunciations for words, abbreviations, and acronyms, preventing the normal letter-to-sound rules from applying. One way you can do this is to enter a Symbolic Phonetic Representation (SPR) annotation directly into the input text. A more permanent way is to enter the word (the input string or "key") and the pronunciation you want (the output or “translation” value) in one of the user dictionaries:

Main Dictionary

Abbreviations Dictionary

Roots Dictionary

How to Modify the Dictionaries

Each dictionary entry has two parts: a key and a translation. An invalid key or translation will cause the dictionary look-up to fail, and the pronunciation of the word will be generated by the normal pronunciation rules. Valid entries for each dictionary are discussed under each dictionary's topic.

To add, modify, or delete an entry in any of the dictionaries, use the eciUpdateDict function of the API.

See Also

Dictionary Processing of Abbreviations

Main Dictionary (eciMainDict)

The important feature of the main dictionary is that a valid translation can consist of any valid input string. So You can use the main dictionary for the following:

- Strings that translate into more than one word.
- Abbreviations and acronyms (as long as the key doesn't contain periods).
- Strings containing digits or other non-letter symbols (these aren't allowed in the other dictionaries).
- Strings that require translations with annotations or SPRs. (Note that you must always use annotations rather than tags in dictionary entries.)

The main dictionary is case-sensitive. So for example, if you entered the key WHO to translate to "World Health Organization," lower case "who" would still be pronounced as expected (`[hu]).

Allowable Main Dictionary Entries

Key	**Translation**
Letters, in upper or lower case Strings containing digits or non-letter symbols like @, #, $, %, &, *, + Apostrophes, question marks, parentheses, brackets	With the exception of tags, anything that is legal input to the text-to-speech engine, including white space, punctuation, SPRs, and annotations
NO: punctuation (except those listed above), or white space	**NO:** tags

Main Dictionary Examples

Key	**Translation**
AWSA	American Woman Suffrage `0 Association
ECSU	`[1i] `[1si] `[1Es] `[1yu]
UConn	`[1yu2kan]

wheelpower	wheel `0 power
safekeeping	safe keeping
WYSIWYG	`[1wI0zi0wIg]
Win32	win thirty two
486DX	4 86 dee ecks

See Also

Abbreviations Dictionary (eciAbbvDict), Roots Dictionary (eciRootDict).

You can temporarily override the use of both internal and user-defined abbreviations with a tag; see Dictionary Processing of Abbreviations.

Roots Dictionary (eciRootDict)

The roots dictionary is used for ordinary words, like nouns, verbs, or adjectives, and for proper names. The distinctive property of the roots dictionary is that you only have to enter the root form of a word; all other forms of the word will automatically get pronounced in the same way.

For example, the spelling-to-sound rules normally pronounce “roof” as [ruf] (which has the vowel of "boot"). You can use the roots dictionary to specify the alternate pronunciation [rUf] (which has the vowel of "book"). Then, all words with this root, such as "roofer" and "roofing," will also be pronounced this way; there is no need to list the other words separately in the dictionary.

- The roots dictionary is not case-sensitive. So, for example, if you enter a root in lowercase, it will still be found and pronounced as specified even when it begins with an uppercase (capital) letter as the first word in a sentence.
- The roots dictionary is designed to provide alternate pronunciations of *existing roots*, and may not work properly in the case of unknown roots. For example, the entry "prego" occurring in the hypothetical word "pregoness" will not be accessed from the user roots dictionary because the linguistic analysis rules assume that the word contains the root "go" rather than the root "prego".

Allowable Roots Dictionary Entries

Key	**Translation**
a single word in ordinary spelling	a single word in ordinary spelling
letters, in upper and lower case	a valid SPR
NO: digits, non-letter symbols, punctuation, or white space.	NO: digits, non-letter symbols, punctuation, or white space, tags, or annotations.

Roots Dictionary Examples

Key	Translation	Would apply to:
roof	`[.lrUf]	roof, roofs, roofer, roofing
bonny	`[.lba.0ni]	Bonny
figure	`[.1fI.0gR]	figures, figuring, figured, refigure
tomato	`[.0tx.1ma.0to	tomatoes, tomato’s
almond	`[a.0mXnd]	almonds, almond’s
wash	`[.1warS}	wash, washing, washed, washes
Wilhelmina	Wilma	Wilhelmina, Wilhelmina’s
Miyuki	`[.0mI.1yu.0ki]	Miyuki, Miyuki’s
Baltimore	`[.1bcl.mR]	Baltimore, Baltimore’s

See Also

Main Dictionary (eciMainDict), Abbreviations Dictionary (eciAbbvDict)

You can temporarily override the use of both internal and user-defined abbreviations with a tag; see Dictionary Processing of Abbreviations.

Abbreviations Dictionary (eciAbbvDict)

The abbreviations dictionary is used for abbreviations (both with and without periods) which do not require the use of annotations in their translation.

The abbreviations dictionary is case-sensitive. So for example, if you entered the key Mar to translate to "march," lower-case "mar" would still be pronounced as expected (`[mar]).

When you enter a key in the abbreviations dictionary, it is not necessary to follow it with a "trailing" period (such as "etc."). The dictionary look-up routine will still find and pronounce the abbreviation as you have specified, whether or not it has a trailing period. However, if you want an abbreviation to be pronounced as such <u>only</u> when it followed by a period in the text, then you must enter the trailing period in the key. The following table summarizes the use of trailing periods:

Key entry:	**Will match:**
`inv`	inv. inv
`sid.`	sid. (not sid)

An abbreviation dictionary entry invokes different assumptions about how to interpret a following period in the text than does a main dictionary entry. Since the period cannot be part of a main entry, it is automatically interpreted as end-of-sentence punctuation. A period following an abbreviations dictionary entry, on the other hand, is ambiguous. It will be interpreted as end-of-sentence punctuation only if other appropriate conditions pertain (such as when it is followed by two spaces and an upper-case letter).

For example, the first period in the following entry will not be interpreted as indicating the end of the sentence: It rained 1 in. on Monday.

However, it will be regarded as such in the following entry: It rained 1 in. On Monday, it was sunny.

Allowable Abbreviations Dictionary Entries

Keys	Translation
• Sequences of one or more letters separated by periods (x.x.x. or xx.xx.xx) • Sequences of letters, with or without the trailing period that may be considered part of the abbreviation (xxx. or xxx) • Upper or lower case letters • Internal apostrophes (not the first or last character in the sequence) One or more valid words in ordinary spelling	• One or more valid words in ordinary spelling
NO: digits, non-letter symbols, white space, or punctuation (except periods)	**NO:** digits, punctuation, SPRs, or annotations

Abbreviations Dictionary Examples

Key	Translation
Is.D.	eye ess dee
Punct	Punctuation
Para	Paragraph
Ltjg	Lieutenant junior grade
int'l	International

See Also

Main Dictionary (eciMainDict), Roots Dictionary (eciRootDict).

You can temporarily override the use of both internal and user-defined abbreviations with an annotation; see Dictionary Processing of Abbreviations.

Appendix D

SPR Symbols

Symbolic Phonetic Representation

A Symbolic Phonetic Representation (SPR) is the phonetic spelling of a single word. It represents the sounds of the word, how these sounds are divided into syllables, and which syllables receive stress. For example, in ViaVoice Outloud's phonetic spelling,

"though" = `` `[.1Do] ``

"shocking" = `` `[.1Sa.0kIG] ``

The periods signal the start of a new syllable (they are optional in the notation), and the numbers `1` and `0` signal primary stress and zero stress. The letters, `D, o, S, a, k, I, and G` are part of the phonetic alphabet used by ViaVoice Outloud. The [ ] brackets and ` backquote indicate that the string is a phonetic representation rather than normal English spelling.

You may want to use SPRs when the normal letter-to-sound rules wouldn't produce the correct pronunciation. You can enter SPRs for particular words in the user dictionaries to have the pronunciation apply whenever a certain string is encountered. In the dictionary, the SPR must constitute the entire word. Alternatively, you can enter the SPR in the input text itself by enclosing it in the \xSPR\ tag.

Form

Each word entered as an SPR must be a string of allowable SPR symbols enclosed in square [ ] brackets, and preceded by a ` backquote character. Within the brackets there are restrictions on where to place syllable stress markers. An invalid SPR is read out character by character.

If you are entering an SPR directly into the input text, the SPR must be enclosed in the \xSPR\ tag. For example: ("What `` \`[.1kYn.dx] `` tools?", 1)

If you are entering an SPR into one of the user dictionaries, you do <u>not</u> need to enclose the SPR in the \xSPR\ tag.

SPR Tables

The symbols allowed in Symbolic Phonetic Representations (SPRs) for most vowels and consonants are listed in the tables below.

Regular Vowels

Symbol	Example Words
a	rod, father
A	back, had
e	cake, pain
E	hedge, let
i	see, speak, believe
I	pick, ill
o	both, oak
c	law, cough
u	zoo, truth
U	took, put
H	but, mug, son
R	butter, hurt

Reduced (Unstressed Vowels)

Symbol	Example Words
x	sofa, alone, suppose, tedious, America
X	roses, connect, melody, symphony, hinted

Diphthongs

Symbol	Example Words
W	out, cow
O	toil, boy
Y	life, fine

Consonants

	Symbol	Example Words
	b	bad, sob
	p	pit, rip
	d	dip, had
	t	tip, pet
Flap	F	writer, fiddle
	g	good, bug
	k	kill, make, back
	D	this, breathe
	T	thing, Beth
	v	vase, save
	f	field, if, graph
	Z	zip, phase
	s	seal, miss, ceiling
	Z	treasure, garage
	S	ship, wish
	J	Jane, huge
	C	chip, witch
	h	hot, hero

	m	man, hum, summer
	M	hmmm
	n	never, sun, winner
Syllabic Nasal	N	button, satin, eaten, burden
	G	sing, finger
	r	borrow, rake
	l	low, hall
	y	yes, Virginia
	w	wear, quick
Glottal Stop	?	kitten, Latin

Tips for SPR Symbols

Use the following tips for entering and interpreting SPR symbols for sounds with limited distribution, dialects, and sound-changing rules:

- If you are entering an SPR directly into the input text, the SPR must be enclosed in the \xSPR\ tag. For example: \xSPR=`[.1kYn.dx]\
- If you are entering an SPR into one of the user dictionaries, you do not need to enclose the SPR in the \xSPR\ tag.
- In the above table, there are words illustrating each sound symbol, with the underlined portions of the example words indicating a typical English spelling of the sound. Since English, like other languages, exhibits dialectal variation in pronunciation, your own pronunciation may not match every example.
- Note that the symbols are case-sensitive, so `[a] and `[A] represent two different sounds.
- The sounds of every language have specific distributional patterns. For example, in all dialects of English the sound [G], as in "sing" (`[sIG]), does not occur at the beginning of a word. Other sounds in English that have particularly narrow distribution are the glottal stop [?], the flap [F], and the syllabic nasal [N].   If you enter a sound symbol in a context where it does not normally occur, the resulting speech may sound unnatural.
- ViaVoice Outloud applies a sophisticated set of linguistic rules to its input to reflect the processes by which sounds change in specific contexts in natural language. For example, in American English, the sound [t] of "write" [.1rYt] is pronounced as a flap in "writer" [.1rY.0FR].   SPR input will undergo these modifications in the same way that ordinary text input does. In this example, whether you enter `[1rY.0tR] or `[1rY.0FR], the output of the program will be the same.

Syllable Stress and Boundaries

If a word has more than one syllable, at least one of these syllables must be marked for primary stress. Other syllables can also be marked with their stress level, either secondary or no stress. Syllables that are not marked for stress are assumed to have no stress.

Syllable Stress Markers

Use the following markers to indicate stress on a syllable:

1	primary stress (most prominent stress in the word)
2	secondary stress
0	no stress

Words with only one syllable do not have to be marked for stress. They are given stress level 1. Example: `spice` can be `` `[.1spYs] `` or `` `[.spYs] ``

The syllable stress marker (1, 2, or 0) should be within the syllable's bourndaries but always before the syllable's vowel. Put another way, a syllable stress marker always applies to the following vowel and its syllable. Thus, if you don't know where the syllable boundary is in a word like "restrict" or "construction," any of the following SPRs will correctly place the primary stress on the vowel in **bold** type:

"restrict"	**"construction"**
`[rX1str**I**kt]	`[kXn1str**H**kSXn]
`[rXs1tr**I**kt]	`[kXns1tr**H**kSXn]
`[rXst1r**I**kt]	`[kXnst1r**H**kSXn]
`[rXstr1**I**kt]	`[kXnstr1**H**kSXn]

Syllable Boundaries

Within the brackets of an SPR, a period signals the beginning of each syllable. This syllable boundary marker is optional in the SPR and is ignored by the pronunciation rules. It is there only as a visual aid for spotting syllable breaks (which is useful when entering syllable stress markers). You cannot override ViaVoice Outloud's internal syllabification rules by moving the syllable boundary marker.

Examples of Syllable Stress and Syllable Boundaries

The following table shows examples single and multiple syllable stress and bounrdaries.

Syllables	**Example**	**SPR**
Single Syllable	Spice	`` `[.1spYs] ``
	Cloves	`` `[.1klovz] ``
	Knead	`` `[.1nid] ``
Two Syllables	Honey	`` `[.1hH.0ni] ``
	Chocolate	`` `[.1Cc.0klxt] ``
	Cookbook	`` `[.1kUk.2bUk] ``
	Complete	`` `[.0kxm.1plit] ``
Three Syllables	Cinnamon	`` `[.1sI.0nx.0mXn] ``
	Celebrate	`` `[.1sEl.0X.2bret] ``
	Vanilla	`` `[.0vx.1nIl.0x] ``
	Raspberry	`` `[.1raz.2bE.0ri] ``
Four Syllables	Ingredients	`` `[.0IG.1gri.0di.0Xnts] ``
	Thermometer	`` `[.0TR.1ma.0mX.0tR] ``
Five Syllables	Refrigerator	`` `[.0rX.1frI.0JR.2e.0tR] ``
	Horizontally	`` `[.2hc.0rX.1zan.0tx.0li] ``

SPR Examples

The best way to become proficient at entering and reading SPRs is by actually doing it. The table below lists a variety of examples using the SPR symbols described in this appendix to help get you started.

Spelling	SPR
interesting	`` `[.1In.0trX.0stIG] ``
contemplate	`` `[.1kan.0txm.2plet] ``
tangled	`` `[.1tAG.0gxld] ``
clothed	`` `[.1kloDd] ``
plants	`` `[.1plAnts] ``
birds	`` `[.1bRdz] ``
singing	`` `[.1sI.0GIG] ``
bushes	`` `[.1bU.0SXz] ``
various	`` `[.1ver.0i.0xs] ``
insects	`` `[.1In.2sEkts] ``
flitting	`` `[.1flI.0FIG] ``
worms	`` `[.1wRmz] ``
crawling	`` `[.1krcl.0IG] ``
damp	`` `[.1dAmp] ``
earth	`` `[.1RT] ``
reflect	`` `[.0rX.1flEkt] ``
these	`` `[.1Diz] ``
different	`` `[.1dI.0frXnt] ``
each	`` `[.1iC] ``
complex	`` `[.1kam.2plEks] ``
manner	`` `[.1mA.0nR] ``

Appendix E Multi-Language SPRs

American English SPRs

Use the following guidelines for creating and understanding English SPRs:

- Only the following letters, numbers, and symbols are allowed in an American English SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal English spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[a] and `[A] represent two different sounds.
- A . (period) appears at the beginning of every syllable in an SPR generated by the text-to-speech program. However, syllable boundaries in SPR entries will have no effect on the program's internal syllabification rules.
- A syllable may be marked for stress level with a 0, 1, or 2 to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Regular Vowels

The following table includes the symbols for regular vowels.

American English Symbol	Example Words
a	rod, father
A	back, had
c	law, cough
e	cake, pain
E	hedge, let
H	but, mug, son
i	see, speak, believe

American English Symbol	Example Words
I	pick, ill
o	both, oak
R	butter, hurt
u	zoo, truth
U	took, put

Diphthongs

The following table includes the symbols for diphthongs.

American English Symbol	Example Words
O	toil, boy
W	out, cow
Y	life, fine

Reduced Vowels

The following table includes the symbols for reduced vowels.

American English Symbol	Example Words
x	sofa, alone, suppose, tedious, America
X	roses, connect, melody, symphony, hinted

Consonants

The following table includes the symbols for consonants.

American English Symbol	Example Words
? ("glottal stop")	kitten, Latin
b	bad, sob
C	chip, witch
d	dip, had

American English Symbol	Example Words
D	this, breathe
f	field, if, graph
F ("flap")	writer, fiddle
g	good, bug
G	sing, finger
h	hot, hero
J	Jane, huge
k	kill, make, back
l	low, hall
m	man, hum, summer
n	never, sun, winner
N ("syllabic nasal")	button, satin, eaten, burden
r	borrow, rake
s	seal, miss, ceiling
S	ship, wish
t	tip, pet
T	thing, Beth
v	vase, save
w	wear, quick
y	yes, Virginia
z	zip, phase
Z	treasure, garage

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
2	secondary stress
0	no stress

British English SPRs

Use the following guidelines for creating and understanding British English SPRs:

- Only the following letters, numbers, and symbols are allowed in an British English SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal English spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[a] and `[A] represent two different sounds.
- A . (period) appears at the beginning of every syllable in an SPR generated by the text-to-speech program. However, syllable boundaries in SPR entries will have no effect on the program's internal syllabification rules.
- A syllable may be marked for stress level with a 0, 1, or 2 to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Regular Vowels

The following table includes the symbols for regular vowels.

British English Symbol	Example Words
c	law, court, hall, water
@	rod, cough
u	zoo, truth
U	took, put
H	but, mug, son
R	butter, hurt
c	law, court, hall, water
c	law, court, hall, water
@	rod, cough

British English Symbol	Example Words
u	zoo, truth
U	took, put
H	but, mug, son
R	butter, hurt

Diphthongs

The following table includes the symbols for diphthongs.

British English Symbol	Example Words
O	toil, boy
W	out, cow
Y	life, fine

Reduced Vowels

The following table includes the symbols for reduced vowels.

British English Symbol	Example Words
x	sofa, alone, suppose, America
X	roses, connect, melody, symphony, hinted, tedious

Consonants

The following table includes the symbols for consonants.

British English Symbol	Example Words
b	bad, sob
d	dip, had
g	good, bug
p	pit, rip
t	tip, pet

British English Symbol	Example Words
v	vase, save
m	man, hum, summer
C	chip, witch
D	this, breathe
f	field, if, graph
F ("flap")	woody, paddle
G	sing, finger
h	hot, hero
J	Jane, huge
k	kill, make, back
l	low, hall
L ("syllabic l")	cattle, bottle, curdle
n	never, sun, winner
r	borrow, rake
s	seal, miss, ceiling
S	ship, wish
T	thing, Beth
w	wear, quick
y	yes, Virginia
z	zip, phase
Z	treasure, garage

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
2	secondary stress
0	no stress

German SPRs

Use the following guidelines for creating and understanding German SPRs:

- Only the following letters, numbers, and symbols are allowed in an German SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal German spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[e] and `[E] represent two different sounds.
- Two-character symbols must be contained in single quotes. For example: wägen `[.1v'E:'.0g@n]
- A . (period) appears at the beginning of every syllable.
- A syllable may be marked for stress level with a 0, 1, or 2 immediately to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Vowels

The following table includes the symbols for vowels.

German Symbol	Example Words
o	ober, ohne, Boot
a	Haar, haben, fahren
A	lassen, matt, Apfel
e	geben, Ehre, See
E	treffen, Geld, kämmen
'E:'	Käse, Mädchen, wägen
i	lieben, Titel, tief
I	bitte, Tisch, Licht
u	gut, Uhr, Uwe
U	Hund, Fluß, Mutter

German Symbol	Example Words
@	bitte, Kamera, Boden
O	Kopf, Stopp
'oe'	Löwe, hören, Söhne
'OE'	können, hölzern, östlich
y	Bücher, Tür, kühn
Y	fünf, füllen, Künstler

Diphthongs

The following table includes the symbols for diphthongs.

German Symbol	Example Words
'aj'	heim, Waise, Mai
'aw'	Haus, Maul, Frau
'oj'	heute, Gebäude, Häuser

Nasalized Vowels (occur in borrowings only):

The following table includes the symbols for nasalized vowels.

German Symbol	Example Words
'a~'	Chance
'E~'	Teint
'o~'	Pardon
'oe~'	Parfum

Consonants

The following table includes the symbols for consonants.

German Symbol	Example Words
b	Boden, Bett, oben
C	deutsch, Chile, Cello
d	dunkel, kindisch, bunde, Helden
f	fast, hoffen, Vater
g	geben, grau, Tage
G	Finger, längs, Anfang
h	hoch, Hand, Ahorn
j	Junge, ja, Jahr, Ministerium
J	Job, Dschungel
k	Katze, Ecke, Skulptur, lag, quitt
l	lesen, fallen, Pult
m	Mann, kommen, Atem
n	Nacht, können, Kind
p	Papier, Lippe, Grab
P	Pflanze, Stumphen
r	Rad, führen, mehr, Kerl
R	Wieder, über
s	Fuß, lassen, Last, Haus
S	schon, spielen, Stil, wäscht
t	Tag, bitte, Rad
T	Zauber, Polizei, Glanz
v	Wagen, viskös, Volum, oval
w	Eduard, aktuell, Januar
x	Buch, Bach, Wochen
X	ich, Chemie, Kelch, mancher

German Symbol	Example Words
z	See, Satz, lesen
Z	Jalousie, Genie

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
2	secondary stress
0	no stress

French SPRs

Use the following guidelines for creating and understanding French SPRs:

- Only the following letters, numbers, and symbols are allowed in an French SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal French spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[e] and `[E] represent two different sounds.
- Two-character symbols must be contained in single quotes. For example: jeune `[.Z1'oe'n]
- A . (period) appears at the beginning of every syllable in an SPR generated by the text-to-speech program. However, syllable boundaries in SPR entries will have no effect on the program's internal syllabification rules.
- A syllable may be marked for stress level with a 0, 1, or 2 immediately to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- In French, and underscore character (_) can be added to the end of an SPR (but within the [ ] brackets) to indicate liaison.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Vowels

The following table includes the symbols for vowels.

French Symbol	Example Words
a	pattes, lac, cave
'a~'	banc, en, temps
c	paul, note, échalotte
e	café, déformer, été
E	père, annuaire, mer
'E~'	fin, plein, faim
'eu'	peu, jeûner, émeute

French Symbol	Example Words
i	film, type, rythmique
o	paule, tôt, eaux
'o~'	bon, pont, mon
'oe'	peur, jeune, déjeuner
'oe~'	un, aucun, brun
u	roue, où, aôut, tour
x	litres, marbre
y	utile, pure, Bruno

Consonants

The following table includes the symbols for consonants.

French Symbol	Example Words
b	bébé, balle, robe
d	dort, dolmen, addition
f	chef, faim, phare
g	guerre, bague, garer
H	lui, nuit, nuée
j	hiérarchie, paille, yéyé
k	kilo, caler, quai
l	litre, illisible, pâle
m	maman, femme, mettre
n	Anne, ni, anonyme
'ng'	parking, camping
'nj'	agneau, campagne
p	porte, prêt, guêpe
r	parer, rare, carreau
s	sans, ambition, façon

French Symbol	Example Words
S	cheval, lâche, schéma
t	ton, patte, théâtre
v	laver, wagon, visiter
w	oui, moi, voilà
z	jaser, réseau, zigzaguer
Z	rage, gîte, jouer

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
2	secondary stress
0	no stress

Liaison

_	(underscore character) allow liaison if the following word begins with a vowel. For example:	
	`[p0'oe't1it_]	The [t] will not be pronounced unless the following word begins with a vowel.
	`[p0'oe't1it]	The [t] will always be pronounced.

Standard Italian SPRs

Use the following guidelines for creating and understanding standard Italian SPRs:

- Only the following letters, numbers, and symbols are allowed in a Standard Italian SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal Italian spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[n] and `[N] represent two different sounds.
- A . (period) appears at the beginning of every syllable in an SPR generated by the text-to-speech program. However, syllable boundaries in SPR entries will have no effect on the program's internal syllabification rules.
- A syllable may be marked for stress level with a 0 or 1 to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Vowels

The following table includes the symbols for vowels.

Italian Symbol	Example Words
a	lasagna, allegro
c	posta, forte
e	nero, duetto
E	ecco, liceo
i	isola, formica
o	padrone, attore
u	luna, ufficio

Consonants

The following table includes the symbols for consonants.

Italian Symbol	Example Words
b	bocca
C	cece, ciao
d	data
f	fare
g	grande
J	Giovanni, congelare
k	casa, vecchio
p	partire
s	stare, noioso
S	scegliere, lasciare
t	toccare
v	vano, vivere
z	paese, sbaglio

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
0	no stress

Castilian Spanish SPRs

Use the following guidelines for creating and understanding Castilian Spanish SPRs:

- Only the following letters, numbers, and symbols are allowed in a Castilian Spanish SPR (Symbolic Phonetic Representation, or phonetic spelling of the word). In the following table, underlined portions of example words indicate the normal Spanish spelling of the sound. (Due to dialectal differences in pronunciation, there may be examples that don't match your pronunciation.)
- Letters are case sensitive, so `[t] and `[T] represent two different sounds.
- A . (period) appears at the beginning of every syllable in an SPR generated by the text-to-speech program. However, syllable boundaries in SPR entries will have no effect on the program's internal syllabification rules.
- A syllable may be marked for stress level with a 0 or 1 to the left of the vowel.
- If a word has more than one syllable, one of these syllables must be marked for stress level 1.
- [ ] (brackets) surround each SPR.
- A ` (backquote) precedes each opening bracket: `[
- If the SPR contains invalid characters, has no vowel, or does not have a required stress indicator, it will be read out as individual characters rather than pronounced as a word.

Vowels

The following table includes the symbols for vowels.

Spanish Symbol	**Example Words**
a	agua
e	este
i	igual
o	oso
u	uve

Consonants

The following table includes the symbols for consonants.

Spanish Symbol	Example Words
b	basta, hubo
C	coche, chico
d	dar, nada
f	flaco, afuera
g	goma, haga
j	jota, gente
k	coger, irak
l	loco, algo
m	mano, amor
n	no, mano
N	España
p	parte, apagar
r	arena, pero
R	ropa, perro
s	si, casa
t	toma, atar
T	ciudad, manzana
w	fuera, deuda
y	playa, mayor, ya, oigo, tiesto
z	mismo, desde

Syllable Stress

The following table includes the symbols for syllable stress.

1	primary stress (most prominent stress in the word)
0	no stress

Appendix F

Glossary

Alphanumeric

Alphabetic (a, b, c) and numeric (1, 2, 3) symbols.

Compound word

A word created from two other words. Here are some examples of compound words:

> hardwood, moon dancer, widespread, hand carry, tree trunk, overpower, outgrown, cow punch

A compound word can be combined with another word (which can be a compound), so there is no theoretical limit to the length of a compound:

> firewood bin, graham cracker pie crust, greenhouse gas

English spelling does not indicate whether or not something is a compound. The component words can be separated with a space or a hyphen, or not separated at all:

> free-fall, freeway, free will, highland, high-rise, high school

A compound word has a different stress pattern than a noun phrase consisting of the same words. For example, compare the pronunciation of the following:

> Ouch! That's hard wood.
>
> It's not a pine tree; it's a hardwood.
>
> Please paint that black board yellow.
>
> Please erase the blackboard this afternoon.

Content word

The type of word that constitutes most of the vocabulary, such as:

- Nouns (story, happiness, sun, mile)
- Verbs (ride, chew, listen, bring, believe, remain)
- Adjectives (brilliant, awful, three, new, darkest)
- Adverbs (often, far, much, calmly, happily)

Content words are distinguished from function words.

Emphasis

Emphasis is the prominence given to a word relative to other words in an utterance.

Flap

The flap, [F], sounds like a quick [d]. It occurs only after a vowel and before an unstressed vowel or other syllabic sound like [R]. The following examples are from General American English:

sooty `[.1sU.0Fi]	woody `[.1wU.0Fi]
pretty `[.1prI.0Fi]	buddy `[.1bH.0Fi]
pity `[.1pI.0Fi]	tidy `[.1tay.0Fi]
data `[.1de.0Fx]	soda `[.1so.0Fx]
unity `[.1yu.0nX.0Fi]	comedy `[.1ka.0mX.0Fi]
motto `[.1ma.0Fo]	voodoo `[.1vu.0Fu]
rattle `[.1rA.0Fxl]	paddle `[.1pA.0Fxl]
cottage `[.1ka.0FXJ]	adage `[.1A.0FXJ]
British `[.1brI.0FXS]	Swedish `[.1swi.0FXS]
visitor `[.1vI.0zX.0FR]	odor `[.1o.0FR]
creator `[.2kri.1e.0FR]	cider `[.1say.0FR]
writer `[.1ray.0FR]	rider `[.1ray.0FR]
latter `[.1lA.0FR]	ladder `[.1lA.0FR]

As you can see from the above examples, the flap corresponds to the alphabet letters "t" or "d." You may enter the flap symbol in an SPR, but it is not necessary to do so. Wherever required, the text-to-speech engine will modify user SPRs to convert "t" or "d" to a flap. For example, if the dialect is set to American English and you enter `[.1pH.0ti] for "putty," the SPR will be converted to `[.1pH.0Fi].

Function word

Grammatical words such as:

- Conjunctions (and, or, but)
- Articles and determiners (a, an, the, this, those...)
- Auxiliaries (can, may, will, must, should...)
- Prepositions (to, from, over...)
- Pronouns (she, her, we, they, it...)

Function words are distinguished from content words and are normally pronounced with reduced emphasis.

Glottal stop	The glottal stop, [?], has a limited distribution in SPRs. It can only occur before the [N] sound, which is heard in "kitten" `[.1kI.0?N] and "Latin" `[.1lA.0?N].
Intonation	Changes in pitch across an utterance which are not related to the meaning of individual words. Intonation conveys, for example: • The difference between questions and statements • Contrastive emphasis, used in statements that contradict or parallel a previous statement (e.g., Terry has a cold but JANET has pneumonia.) • Statement completion or closure
Intonational phrase	In ViaVoice Outloud, an intonational phrase is usually marked off by punctuation, such as a comma, period, or question mark. One phrase: He's a child? Two phrases: He's a child, though growing quickly. Three phrases: He's a child, an old child, but a child. One exception to this rule is a sentence containing a quotation, as in the following example. In this case, the sentence is treated as a single intonational phrase (rather than two), and the nuclear accent falls on the last content word in the quotation rather than the last content word in the sentence: Because I'm a lousy \`2 singer I said."
Key	A key is the first half of a user dictionary entry. The key is the string of characters that will be searched for by the dictionary routine.
Nuclear accent	The last emphasized word in an intonational phrase that has a degree of emphasis of 2 or higher.
Phonetic spelling	A phonetic spelling uses special symbols like those found in the pronunciation guide of a dictionary. It has one symbol for each sound and indicates which syllables receive stress.
Pitch	How high or low a voice sounds.

Question intonation	In questions, intonation is marked by a low tone on the nuclear accented word of the intonation phrase, followed by a pitch rise.
Reduced emphasis	A word with reduced emphasis is shorter than normal and has no pitch accent (tone). A word that simply has *no emphasis* rather than *reduced emphasis* has no pitch accent, but it is not shortened.
Reduced vowel	The reduced vowels are `[x] (as in 'sof<u>a</u>') and `[X] (as in 'shad<u>e</u>d'). They are shortened and unstressed.
Root	The base form of a word, without prefixes (like un-) or suffixes (like plural -s or past tense -ed).
Speaker	A speaker, or mode, is a collection of settings that determine the characteristics of the speech to be produced. These settings include the language and dialect, the voice characteristics, the speaking style, and the sampling rate to be used. A speaker is not synonymous with a voice. The same voice settings can be used with different speakers across languages. For example, you could define one speaker named "Gramps" who uses the voice "Elderly Male 1." Then you could define another speaker named "Opa" who uses the same voice, but speaks German. Since voice settings can be shared among different speakers, they are labelled independently of the speaker.
Stress	Stress is the prominence given to a syllable, relative to other syllables in the word. For example, in sentence (a) the word desert has the greatest stress on the first syllable, and in sentence (b) the word desert has the greatest stress on the second syllable. (a) I've been through the desert in a car with no air \`0 conditioner." (b) "Let's desert this old car and walk from here."

Syllabic nasal	The syllabic nasal, [N], has a limited distribution in SPRs. The sound [N] only occurs after [d] and [t] (or [?] -- the glottal stop -- since [t] is converted to [?] in this context). For example: butt<u>on</u> `[.1bH.0?N] sat<u>in</u> `[.1sA.0?N] eat<u>en</u> `[.1i.0?N] burd<u>en</u> `[.1bR.0dN]
Syllable	A syllable is a unit of speech containing, at a minimum, a sonorant nucleus such as a vowel or diphthong. The syllable may also contain one or more consonants surrounding the vowel.
Translation	A translation is the second half of a user dictionary entry. The translation is the pronunciation or output specified by the user.
User Interface Language (UIL)	The User Interface Language (UIL) is the language used in the dialog boxes.
Vocal tract	Male and female vocal tracts have physical differences which affect the voice, some of which are reflected in the vocal tract setting. Other differences between male and female voices, namely pitch and head size, are controlled independently. The vocal tract attribute should not be confused with the "gender" label that some applications use to classify a speaker, or with the \Vce=Gender=*gender*\ tag, which resets the program to a built-in voice of the specified gender.
Voice	A voice is a set of characteristics that affect how the speaker sounds. These characteristics include features like head size, pitch, and breathiness. A voice is just one property of a speaker.
Voice box	The common term "voice box" refers to the larynx, a roughly cylindrical arrangement of cartilage and muscle located at the top of the wind pipe (trachea), and containing the "vocal cords" (which are really more like "folds" than cords).

Waveform	A waveform is an acoustic representation of speech. It plots variations in air pressure (vertical axis) across time (horizontal axis). For example, this is a picture of the waveform for the word "eloquence."
White space	One or more spaces made with the spacebar or Tab key.

Index