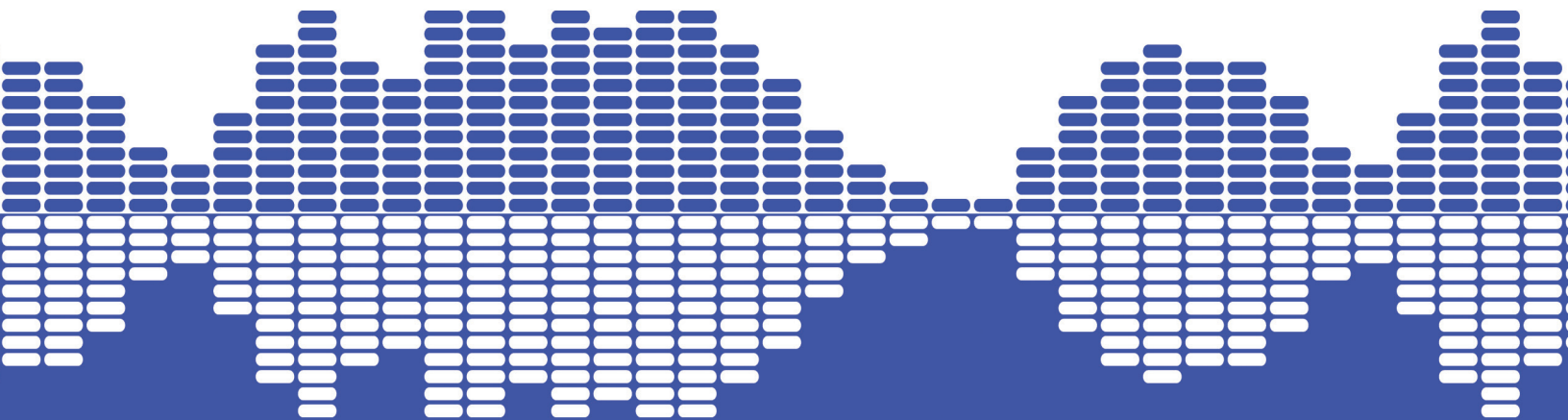




CHANT®

Integrate Speech Technology for Hands-free Operation



Copyright © 2011 Chant Inc. All rights reserved.

Chant, SpeechKit, Getting the World Talking with Technology, talking man, and headset are trademarks or registered trademarks of Chant Inc. Other marks are trademarks or registered trademarks of their respective holders.

Integrate Speech Technology for Hands-free Operation

You really don't have to sit in front of a computer with a mouse and keyboard to use information technology. Your applications can be enhanced to speak and listen to you from where ever you need them to.

Speech recognition is the process of converting an acoustic signal (i.e. audio data), captured by a microphone or a telephone, to a set of words. These words can be used for controlling computer functions, data entry, and application processing.

Speech synthesis is the process of converting words to phonetic and prosodic symbols and generating synthetic speech audio data. Synthesized speech can be used for answering questions, event notification, and reading documents aloud.

WHAT IS SPEECHKIT?

Chant SpeechKit is comprised of application ready software components that handle the complexities of speech recognition and speech synthesis to minimize the programming necessary to develop software that speaks and listens.

It simplifies the process of managing Nuance Dragon NaturallySpeaking, IBM ViaVoice, Microsoft SAPI 4, Microsoft SAPI 5, or Nuance Vocon 3200 recognizers, and managing Acapela, Cepstral, Nuance RealSpeak Solo, Nuance Vocalizer, Microsoft SAPI 4 or Microsoft SAPI 5 synthesizers.

SpeechKit includes ActiveX, C++, C++Builder, Delphi, Java, .NET Framework, Silverlight, and Web component library formats to support all your programming languages and provides sample projects for popular IDEs—such as the latest Visual Studio 2010 from Microsoft.

The component libraries can be integrated with 32-bit, 64-bit, and mobile applications.

SPEECHKIT FEATURES

With SpeechKit, you can enhance application modality by:

- controlling application functions without having to use a mouse or keyboard;
- prompting users for applicable data capture;
- capturing data by speaking rather than typing; and
- confirming data capture with audio acknowledgement.

You can expand current application capabilities with SpeechKit by:

- archiving recorded responses in addition to structured data;
- enforcing domain constraints for enhanced data accuracy;
- integrating interactive help for higher productivity; and
- liberating application access with wireless connection.

SpeechKit application ready components offer you the following advantages:

- Voice-enable any type of Win32, Win64, and WinCE application with SpeechKit components in ActiveX, C++Builder, C++, Delphi, Java, .NET Framework, Silverlight, and Web formats.
- Develop and deploy your applications independent of specific recognizers:
 - o Nuance Dragon NaturallySpeaking,
 - o IBM SAPI (ViaVoice),
 - o Microsoft SAPI 4, SAPI 5, and
 - o Nuance VoCon 3200.
- Develop and deploy your applications independent of specific synthesizers:
 - o Acapela BabTTS,
 - o Cepstral Swift,
 - o Microsoft SAPI 4 and SAPI 5,
 - o Nuance RealSpeak Solo, and
 - o Nuance Vocalizer.
- Select and adjust recognizer, synthesizer, and audio options and property settings dynamically.
- Leverage a common persistence framework across programming languages and component formats.
- Support Unicode and ANSI systems with single component library.

SpeechKit application ready components enable applications to speak:

- Synthesize speech from anywhere within your application.
- Automatically synthesize message box text.
- Easily enumerate and select a voice.
- Easily adjust the spoken output speed, volume, and pitch.
- Take advantage of built-in audio management and synthesize to the audio format needed by your application.
- Synthesize text from strings, buffers, streams, and files.
- Playback audio or write to buffers, streams, and files.
- Access detailed synthesis result attributes and properties.

- Persist property settings across executions.
- Process requests synchronously or asynchronously with built-in queue manager.

SpeechKit application ready components enable applications to listen:

- Capture spoken input as if it was typed using a keyboard.
- Select menus, list items, click buttons, and click hypertext links by speaking instead of using a mouse.
- Simulate keyboard input and mouse clicks.
- Recognize spoken languages supported by recognizers.
- Leverage common dictation text formatting across speech APIs.
- Access detailed recognition result attributes and properties.
- Correct recognition results.
- Playback recorded audio associated with recognition results.
- Manage context-based and context-free recognition by dynamically adding, removing, enabling, and disabling, command, grammar, and dictation vocabularies.
- Take advantage of built-in audio management and recognize from the audio format needed by your application.
- Recognize from buffers, files, microphone (live), and stream audio sources.
- Write audio to buffers, streams, and files.
- Persist property settings across executions.
- Process requests synchronously or asynchronously with built-in queue manager.
- Use built-in What Can I Say dialog to display what commands your application responds to and Where Can I Go dialog to display hypertext links that can be invoked by speaking the link text.

The SpeechKit component library includes classes for managing audio, speech recognition, and speech synthesis.

SPEECH RECOGNITION COMPONENT ARCHITECTURE

Speech recognition is the process of converting an acoustic signal (i.e. audio data), captured by a microphone or a telephone, to a set of words. These words can be used for controlling computer functions, data entry, and word processing. A speech recognition engine (i.e., recognizer) is a software program that converts audio data into recognized speech.

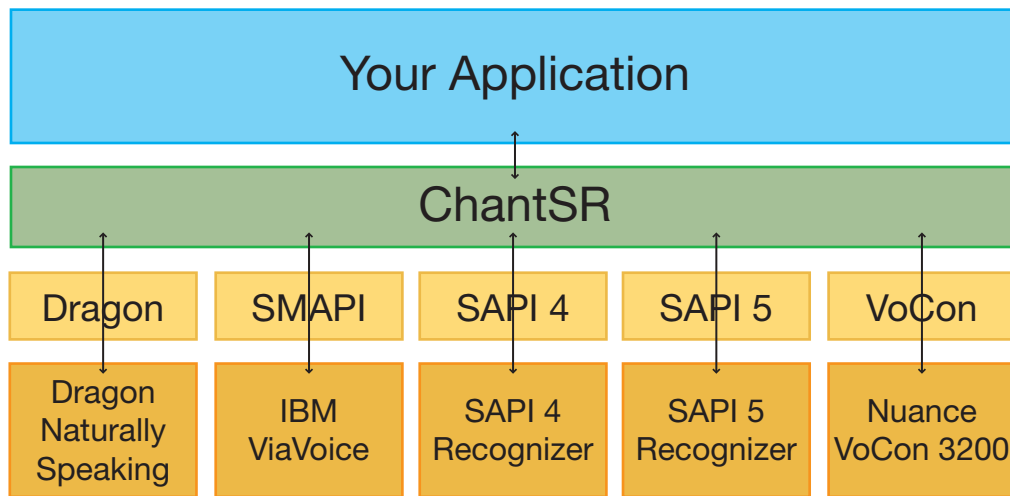
The SpeechKit component library includes a speech recognition management class that provides you a productive way to develop software that listens. Your application sets properties and invokes methods through the speech recognition management class. This class handles the low-level functions with speech recognition engines (i.e., recognizers).

The speech recognition management class, ChantSR, enables you to establish a session with a recognizer, through which spoken language captured live via a microphone or from recorded audio can

be processed and converted to text. Your application uses the ChantSR class to manage the activities for speech recognition on behalf of your application. The ChantSR class manages the resources and interacts directly with a recognizer application program interface (API). The ChantSR class supports the following speech APIs:

- Nuance Dragon NaturallySpeaking COM objects,
- IBM SAPI (ViaVoice),
- Microsoft SAPI 4, SAPI 5, and
- Nuance VoCon 3200.

Your application receives recognized speech as text and notification of other processing states through event callbacks.



The ChantSR object simplifies the process of developing applications that recognize speech by handling the low-level activities directly with a recognizer.

You instantiate a ChantSR class object before you want the engine to start recognizing speech for your application. You destroy the ChantSR object and release its resources when you no longer want to recognize speech for your application.

The ChantSR class encapsulates all of the technologies necessary to make the process of recognizing speech simple and efficient for your application. Optionally, it can save the session properties for your application to ensure they persist across application invocations.

SELECTING A RECOGNIZER

The ChantSR object defaults the recognizer selection first by looking for SAPI 5 default (i.e., speech control panel default) recognizer, followed by IBM ViaVoice, Dragon NaturallySpeaking, Nuance VoCon 3200, and SAPI 4 recognizer. You can specify the default selection criteria by editing the order and

list of speech recognition API entries in an exported Application Framework.

Since there may be more than one recognizer available on the system in which your application runs, you may want to integrate selection into your application.

Alternatively, your application may obtain an enumerated list of available engines and then select the engine you want by setting the Engine property.

SPEECH RECOGNITION APPLICATION FRAMEWORK

Since object persistence varies among component hosts and doesn't exist for some component formats, the Chant Application Framework provides an optional way to persist property settings across application invocations.

You may preset and save ChantSR Recognizer properties. For example, if you want to have the Microphone setup wizard launch on initialization, your application may set the Microphone properties.

Your application may get and set the property values of an Application Framework at any time during execution. This is done automatically when you get and set ChantSR string and number property values. To persist ChantSR property values, your application saves and loads an Application Framework. An Application Framework may be maintained in an XML file.

VOCABULARIES

Vocabularies define the listening context from which to recognize speech. Speech recognition engines typically support four types of vocabularies:

- command,
- grammar,
- dictation, and
- dictation topic.

You can add, enable, and disable vocabularies as needed to manage the listening context in your application.

COMMAND VOCABULARY

A command vocabulary consists of words and phrases that are spoken as commands. It is a simple list of the possible spoken words and phrases. The speech recognition engine matches recognized speech against the list and returns the best match. Command vocabularies are typically very small. For example, a command vocabulary may contain a 100 or fewer entries.

GRAMMAR VOCABULARY

A grammar vocabulary consists of words and phrases and combinations of words and phrases. Grammars support substitution of words within phrases for words defined in the grammars or set at run-time. They are typically small. For example, a grammar vocabulary may contain 500 or fewer word and phrase combinations. You define a text file to contain the words and phrases. Speech APIs require different grammar formats as follows:

Speech API	Grammar Format
Microsoft SAPI 4 Direct Speech Recognition	Text file format is similar to .inf or .ini files.
Microsoft SAPI 5	Text file format is in an XML format prescribed by SAPI5 or W3C SRGS XML.
IBM SAPI	Text file format is in syntax known as Backus-Naur Form (BNF).
Nuance VoCon 3200	Text file format is in syntax known as Backus-Naur Form (BNF).

DICTATION VOCABULARY

A dictation vocabulary represents a dictionary of all possible words from which speech is recognized. These vocabularies are integrated with the speech recognition engine. They are typically very large. For example, a dictionary may contain 30,000 and significantly more words.

DICTATION TOPIC

A dictation topic is a dictionary of words for a specific subject area. Topics improve recognition accuracy by increasing the probability the spoken words are in the domain of the topic dictionary.

SLEEP VOCABULARIES

Some recognizers support sleep and awake states to optimize resource consumption.

Some applications need to associate commands with “waking up” the recognizer from a “sleep” or suspended state. Vocabularies containing these wake up commands are referred to as sleep vocabularies.

ANNOTATIONS

Some grammar syntax support annotations that allow one or more words to be spoken and alternate symbols to be returned to the application. This can reduce the need to edit recognized speech before displaying and simplify parsing. Annotations can be strings or numbers.

DYNAMIC GRAMMARS

Some applications need the ability to create grammars at runtime to accommodate end-user provided information, database sources, and application processing states.

For tailoring and customizing grammars on end user systems, GrammarKit provides a grammar management component for compiling grammars within your application. For additional information about grammars, see the document: [Design Grammars for High-performance Speech Recognition](#).

For less complex dynamic processing needs, your application can set word lists within predefined grammars.

WORD LISTS

Word lists enable you to substitute a list of words or phrases at runtime.

VOICE COMMANDS

Your application may define voice commands and associate them with application functions:

- Append recognized speech to existing text or replaces the text in edit control when the control has focus.

- Simulate a mouse click for a button control.
- Simulate a mouse click to invoke a hypertext link. Specify the descriptor parameter as the URL of the web page to parse for hypertext links or null to voice enable the hypertext links of the page hosting the object declaration.
- Drop down (i.e., pop up) a menubar menu to display its contents.
- Simulate a mouse button click. You may optionally provide X and Y location coordinates otherwise the click simulation occurs at the current mouse location.
- Pop up a menu to display.
- Simulate a mouse click on a horizontal scroll bar.
- Simulate a mouse click on a vertical scroll bar.
- Select an item from a combobox control when the control has focus and the recognized speech matches and entry in the list.
- Select an item from a listbox control when the control has focus and the recognized speech matches and entry in the list. Option Description
- Invoke a menu item.
- Invoke a system menu item.
- Set focus to the control and optionally enable a vocabulary. Disable the vocabulary when the control loses focus.

SPEECH RECOGNITION PROFILES

A speaker profile is a collection of information used by a speech recognition engine to increase its recognition accuracy for a specific individual's voice and environment.

A speech recognition engine saves training and background noise information to use in recognizing speech. Some speech recognition engines, such as Microsoft's Recognizer, capture and save information over time to adapt the profile for optimum speech recognition accuracy.

For additional information about profiles, see the document: [Administer Speaker Profiles for Accurate Speech Recognition](#).

SPEECH RECOGNITION LEXICONS

A lexicon is a collection of words and information about these words used by a speech recognition engine to increase its recognition accuracy.

Recognizers can use lexicons as a reference for the pronunciation information about words specific to your application to improve recognition quality. You may deploy lexicons with your application.

For additional information about lexicons, see the document: [Tailor Pronunciations for Maximum Clarity](#).

SPEECH SYNTHESIS COMPONENT ARCHITECTURE

Speech synthesis is the process of converting words to phonetic and prosodic symbols and generating synthetic speech audio data. Synthesized speech can be used for asking questions, event notification, and reading documents aloud.

A speech synthesis or text-to-speech engine is a software program that converts words into a digital audio data. It does this by converting words to phonetic and prosodic symbols. Prosodic symbols are codes that the text-to-speech engine uses to control the speed and emphasis in delivery of the synthesized speech. Most text-to-speech engines support varying degrees of control over speech synthesis by allowing you to imbed prosodic symbols in your text messages. This provides some programmatic control over the inflection used when synthesizing speech from text. Using the phonetic and prosodic symbols, the text-to-speech engine generates a digital audio stream from the synthesized speech.

The SpeechKit component library also includes a speech synthesis (i.e., text-to-speech) management class that provides you a productive way to develop software that speaks. Your application sets properties and invokes methods through the speech synthesis management class. This class handles the low-level functions with text-to-speech engines (i.e., synthesizers or voices).

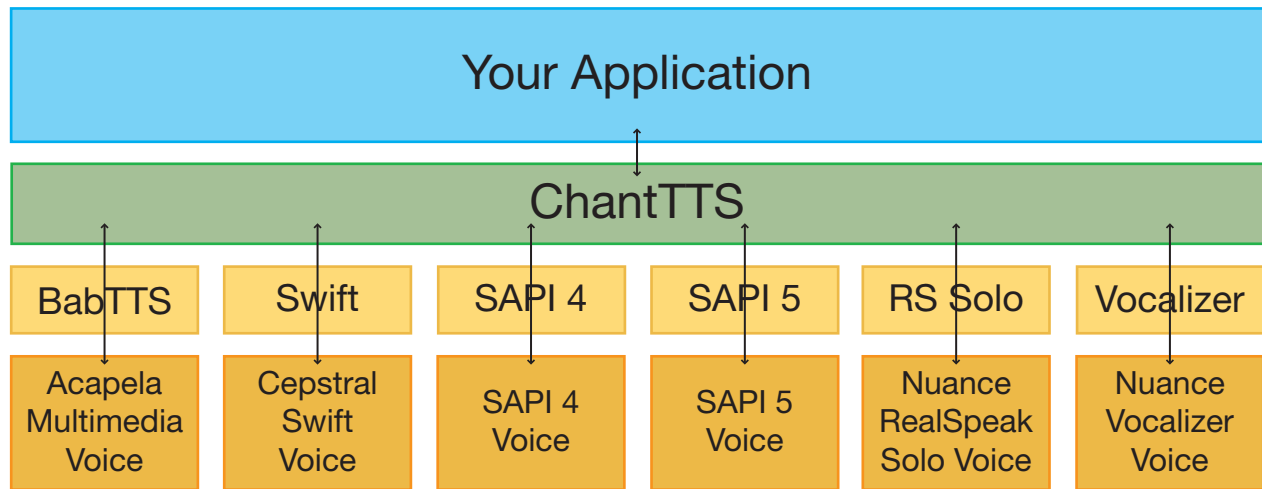
The speech synthesis management class, `ChantTTS`, enables you to establish a session with a synthesizer, through which speech is synthesized from text. Your application uses the `ChantTTS` class to manage the synthesizer resources on behalf of the application. The `ChantTTS` class manages the resources and interacts directly with a synthesizer application program interface (API). The `ChantTTS` class supports the following speech APIs:

- Acapela BabTTS,
- Cepstral Swift,
- Microsoft SAPI 4, and SAPI 5, and
- Nuance RealSpeak Solo.
- Nuance Vocalizer.

Your application receives notification of other processing states through event callbacks.

The `ChantTTS` class encapsulates all of the technologies necessary to make the process of synthesizing speech simple for your application. Optionally, it can save the session properties for your application to ensure they persist across application invocations.

The ChantTTS class simplifies the process of synthesizing speech by handling the low-level activities directly with a synthesizer.



You instantiate a ChantTTS class object before you want to synthesize speech within your application. You destroy the ChantTTS class object and release its resources when you no longer want to synthesize speech within your application.

When evaluating text-to-speech engines, you want to listen to the voice. But more importantly, you want to listen for effectiveness of the synthesis without imbedding any prosody in your messages as well as the effectiveness to the extent you can include prosody in your messages.

The ChantTTS object defaults the synthesizer selection first by looking for SAPI 5 default (i.e., speech control panel default) voice, followed by RealSpeak Solo default voice, Cepstral default voice, Acapela default voice, SAPI 4 (Microsoft) Mary or the first in SAPI 4 enumeration list, and Vocalizer voice. You can specify the default selection criteria by editing the order and list of text-to-speech API entries in an exported Application Framework.

Since there may be more than one synthesizer available on the system in which your application runs, you may want to integrate selection into your application.

Alternatively, your application may obtain an enumerated list of available engines and then select the engine you want by setting the Engine property.

SPEECH SYNTHESIS APPLICATION FRAMEWORK

Since object persistence varies among component hosts and doesn't exist for some component formats, the Chant Application Framework provides an optional way to persist property settings across

application invocations.

If your application wanted to use different voices, you could pre-define these voice selections in the Application Framework. At run time, your application could switch among them simply by setting the ChantTTS Synthesizer property value to the desired Synthesizer entry name instead of having to set all of the property values. For example, if you wanted a fast-speaking Male voice and a slow-speaking Male voice, you could create two Synthesizer entries with the applicable session settings: FastMale and SlowMale. Then your application could switch between them by setting the ChantTTS Synthesizer property to “FastMale” or “SlowMale” when appropriate.

An Application Framework may be saved and loaded from an XML file.

SPEECH SYNTHESIS LEXICONS

A lexicon is a collection of words and information about these words used by a text-to-speech engine to enhance the quality of its pronunciation.

Text-to-speech voices can use application lexicons as a reference for the pronunciation information about words specific to an application. You may deploy lexicons with your application.

For additional information about lexicons, see the document: Tailor Pronunciations for Maximum Clarity.

TEXT MARKUP

Text-to-speech (TTS) markup is text with imbedded indicators that control speech synthesis from the text. Speaking qualities such as the speed, pitch, emphasis, and word pronunciation may be tailored in reproducing speech from text.

Synthesizers (i.e., speech APIs) support unique markup syntax:

Synthesizer	Speech API	Markup Syntax
Cepstral (all languages)	Cepstral Swift	W3C SSML
Microsoft SAPI 4 (all languages)	SAPI 4	SAPI 4 Control Tags
Microsoft SAPI 5 (all languages)	SAPI 5	SAPI 5 XML Markup, W3C SSML
Nuance RealSpeak Solo (all languages)	Nuance RealSpeak Solo	L&H Native Control Sequence, SAPI 5 XML Markup
Nuance Vocalizer Automotive (all languages)	Nuance Vocalizer Automotive	L&H Native Control Sequence, SAPI 5 XML Markup
Nuance Vocalizer Mobile (all languages)	Nuance Vocalizer Mobile	L&H Native Control Sequence
Nuance Vocalizer Network (all languages)	Nuance Vocalizer Network	L&H Native Control Sequence, SAPI 5 XML Markup, W3C SSML

For additional information about TTS markup, see the document: [Fine-tune Speech Synthesis Using Text-to-Speech Markup](#).

MORE INFORMATION

To learn more about developing software that speaks and listens, explore how easily you can manage grammars, profiles, lexicons, recognizers, synthesizers, and text-to-speech markup directly within application software you develop in the following documents:

- [Develop Software That Speaks and Listens](#),
- [Design Grammars for High-performance Speech Recognition](#),
- [Tailor Pronunciations for Maximum Clarity](#),
- [Administer Speaker Profiles for Accurate Speech Recognition](#), and
- [Fine-tune Speech Synthesis Using Text-to-Speech Markup](#).

