

# Timehouse Speech Server, THSpeechServer

Version 1.0  
24<sup>th</sup> of April 2003

<b>OVERVIEW .....</b>	<b>4</b>
GENERAL .....	4
FEATURE LIST .....	4
LICENSING .....	4
REQUIREMENTS.....	5
<i>SAPI4 requirements</i> .....	5
<i>SAPI5 requirements as stated in the SAPI5 documentation</i> .....	5
<b>INSTALLATION.....</b>	<b>6</b>
NORMAL INSTALLATION .....	6
MANUAL INSTALLATION OF THSPEECHSERVER.DLL .....	6
<b>USING THE SPEECH SYNTHESIZER.....</b>	<b>6</b>
SYNTHESIZER CONTROL TAGS .....	6
<i>SAPI 4 control tag and backslash format</i> .....	7
<i>SAPI 5 control tag format</i> .....	7
<i>List of known control tags</i> .....	8
<i>Example of the conversion</i> .....	9
THREADING AND EVENTS (DESKTOP APPLICATIONS ONLY).....	9
<i>Bookmarks</i> .....	10
<i>Visemes</i> .....	11
<b>MP3 FILES.....</b>	<b>11</b>
<b>ASP AND VISUAL BASIC QUICK START .....</b>	<b>12</b>
<b>METHOD REFERENCE IN VISUAL BASIC SYNTAX .....</b>	<b>13</b>
GENERAL .....	13
ERROR HANDLING .....	13
INITIALIZATION .....	14
<i>Init</i> .....	14
<i>SetTempPath</i> .....	15
VOICE SELECTION .....	16
<i>SelectVoice</i> .....	16
<i>GetDefaultVoice</i> .....	17
<i>GetVoiceCount</i> .....	17
<i>GetVoiceName</i> .....	18
<i>GetVoiceParam</i> .....	18
SPEAKING & CONVERTING TO MP3 .....	19
<i>SpeakToFile</i> .....	19
<i>WavToMp3</i> .....	21
SETTING VOICE PARAMETERS.....	22
<i>SetRate</i> .....	22
<i>GetRate</i> .....	23
<i>SetPitch (Only for SAPI4 synthesizers)</i> .....	24

<i>GetPitch (Only for SAPI4 synthesizers)</i> .....	24
<i>SetVolume</i> .....	25
<i>GetVolume</i> .....	25
<b>SPEAKING THROUGH THE SOUND CARD</b> .....	27
<i>Speak</i> .....	27
<i>GetStatus</i> .....	28
<i>Pause</i> .....	28
<i>Resume</i> .....	29
<i>Stop</i> .....	29
<b>HELPER METHODS</b> .....	30
<i>DeleteFile</i> .....	30
<i>ReadFile</i> .....	30
<b>REDISTRIBUTION AND GUIDELINES FOR DESKTOP APPLICATIONS</b> .....	31
<b>APPENDIX A, INSTALLATION ISSUES</b> .....	32
<i>SAPI4</i> .....	32
<i>SAPI5</i> .....	32
<b>APPENDIX B, LANGUAGE CODES</b> .....	33
<b>APPENDIX C, SAPI4 CONTROL TAGS</b> .....	36
<b>APPENDIX D, SAPI5 CONTROL TAGS</b> .....	48

## Overview

### **General**

This component provides an ActiveX interface for any Microsoft Windows Speech API (SAPI) compliant speech synthesizers. The component supports SAPI versions 4.x and 5.x and provides a similar and seamless interface for both versions.

There are SAPI compatible versions of speech synthesizers available for all of the world's main languages. An English speaking Microsoft Speech synthesizer can be downloaded free of charge from <http://www.microsoft.com/speech> or from [http://products.timehouse.fi/ss\\_download.asp](http://products.timehouse.fi/ss_download.asp).

To quickly find out how to use the THSpeechServer, please see section [ASP and Visual Basic quick start](#).

### **Feature list**

- Capable of handling multiple channels in multiple threads
- Save synthesized speech as WAV or MP3 files
- Control the quality of the MP3 file (bit rate etc.)
- Seamless interface to SAPI4 and SAPI5 synthesizers
- ActiveX-interface for Visual Basic and script (ASP, ASP.net, PHP etc.) developers
- C++ -interface
- Java interface
- Samples for ASP, PHP, Visual Basic, Java and C++, C# (.net). Ready sample projects for Visual Studio 6.0 and Visual Studio.net. Samples run with other compilers as well.
- Easy to use front end for Lame MP3 encoder

### **Licensing**

Timehouse SpeechServer is royalty free – you can distribute it freely with your product.

A single license gives you the right to distribute Timehouse SpeechServer with your products without any additional costs. We demand no royalty fees, you do not need to display our logo nor mention the use of the SpeechServer in the credits, all we ask is that you do your best to ensure there is no way to extract your registration key from the end-product.

One license allows using SpeechServer for one developer at a time. If there are more developers working with the SpeechServer, you need to purchase a separate license for each developer. Please see pricing list for details.

The download package contains neither the Microsoft SAPI technology, nor any speech synthesizers.

Information on redistribution of Microsoft speech technology can be found from <http://www.microsoft.com/speech/legal/>.

Please also check <http://www.microsoft.com/speech/> for the latest Microsoft speech products.

## **Requirements**

Timehouse Speech Server requires:

- Windows 95 with DCOM installed  
([http://www.microsoft.com/com/dcom/dcom95/dcom1\\_3.asp](http://www.microsoft.com/com/dcom/dcom95/dcom1_3.asp))
- Windows 98 / Me / NT 4.0 / 2000 / XP
- SAPI 4, or a newer version, installed in the computer
- At least one SAPI-compliant Speech synthesizer
- Microsoft IIS 4.x or newer if used with Web server, NT 4.0 or newer recommended

## **SAPI4 requirements**

All 32-bit Windows operating systems are supported.

Memory and processor requirements depend on the used synthesizer. We have successfully tested the Microsoft Speech Synthesis engine on a NT 4.0 system with a 200 MHz mmx Pentium processor and 128 MB of RAM (would probably work with lesser memory).

## **SAPI5 requirements as stated in the SAPI5 documentation**

Supported Operating Systems:

- Microsoft Windows(r) NT Workstation 4.0, service pack 6a
- Microsoft Windows 2000 Professional Workstation
- Microsoft Windows 98. However, Windows 95 is not supported.
- Microsoft Windows Millennium edition

Memory and processor requirements depend on the synthesizer in use. We have successfully tested Microsoft Speech Synthesis engine on an

NT 4.0 system with a 200 MHz mmx Pentium processor and 128 MB of RAM (would probably work with much less memory).

## Installation

### ***Normal installation***

Run the installer THSpeechServer.exe and follow the instructions.

### ***Manual installation of THSpeechServer.dll***

A few steps are required:

- Copy the THSpeechServer.dll and THSpeechServer.ini into Windows system directory (winnt\system32 for NT, 2000 and XP, windows\system for 95, 98 and Me)
- Open the command prompt and navigate to dll's directory.
- To install: Issue command regsvr32 THSpeechServer.dll
- To uninstall: Issue command regsvr32 THSpeechServer.dll /u

Installing:

```
C:\>cd winnt\system32
C:\winnt\system32>regsvr32 thspeechserver.dll
```

Uninstalling:

```
C:\>cd winnt\system32
C:\winnt\system32>regsvr32 thspeechserver.dll /u
C:\winnt\system32>del thspeechserver.dll
```

Windows NT comes with regsvr32.exe – for Windows 9x you need to obtain one from the Internet or an NT system, it will also work on Win 9x.

## Using the speech synthesizer

### ***Synthesizer control tags***

You have the ability to place special control tags into the text to be spoken by SAPI. These control tags change the attributes (speed, volume, anxiety etc.) of the text being spoken.

Unfortunately there are differences between the syntax of the control tags in SAPI 5 and SAPI 4. SAPI 4 uses backslashes as tag-markers, whereas SAPI 5 uses XML-like tags.

The following example starts speaking a sentence with the default speaking rate and then doubles the speaking rate.

SAPI 4:

`\rspd=100\Hello, \rspd=200\now I speak very fast.`

SAPI 5:

`<RATE SPEED="0"/>Hello, <RATE SPEED="6"/>now I speak very fast.`

With the Timehouse Speech Server you have a seamless interface for both SAPI 4 and SAPI 5 synthesizers, all you have to do is to decide which control tag format you want to use and let the component convert the tags to fit the needs of the synthesizer in use. The component will only convert tags, which are known to it. Unknown tags are removed to avoid speaking the control tags aloud. If conversion is not needed, the text is passed to the synthesizer as it is.

We recommend you to use the SAPI 5 tag format if you don't have previous experience of SAPI 4. This is because SAPI 5 will likely replace SAPI 4 as time passes and the SAPI 5 syntax is simply better than the SAPI 4 syntax.

You can study tag conversion with utility THSpeechTest.exe.

### **SAPI 4 control tag and backslash format**

All SAPI 4 control tags begin and end with a backslash (\).

A double backslash (\\) means, "This is one backslash". In Sapi 4, for example, the path "C:\Program files" would be "C:\\Program files".

All unknown control tags are skipped. For example text

"Hello\jargon=what\ there" would be synthesized as "Hello there".

Please note that you have to convert all backslashes from the source text into double backslashes before passing them to the synthesizer:

"C:\Program files" -> "C:\\Program \rspd=100\ files"

See appendix C for a complete list of SAPI4 control tags and their syntax.

### **SAPI 5 control tag format**

SAPI 5 uses XML-style tags. This means that all tags are enclosed like `<THIS>`. With SAPI 5 tags you can define the scope that tags will affect: `<RATE ABSSPEED="10">blaah, blaah</RATE>`. The end tag `</RATE>` restores the previous rate.

Global tags change the settings permanently. Global tags have a slash before tag's end mark like `<THIS/>`. For example: `<RATE`

`ABSSPEED="10"/>Speaking very fast until the end.`

See appendix D for complete list of SAPI5 control tags and their syntax.

## List of known control tags

SAPI4	SAPI5	Description
\mrk=number\	<BOOKMARK MARK="number"/>	Defines bookmark. Bookmark identifier must be a 32bit unsigned number even if the SAPI5 specification allows using strings.
\pau=number\	<SILENCE MSEC="number"/>	Produces silence for the specified number of milliseconds into the output audio stream.
\rpit=number\  The value of number is relative to the default pitch:  50 is 1/2 of the default pitch 100 is the default pitch 200 is 2 times the default pitch	<PITCH ABSMIDDLE="number"/>  The value of number is in range - 24...+24:  -24 is one octave below the default pitch 0 is the default pitch 24 is one octave above the default pitch	Sets the relative pitch adjustment at which speech is synthesized.  Note different ranges for SAPI4 and SAPI5.
\rspd=number\  The value of number is relative to the default speed:  50 is 1/2 of the default speed 100 is the default speed 200 is 2 times the default speed	<RATE ABSSPEED="number"/>  The value of number is in range - 10...+10:  -10 is 1/3 of the default speed 0 is the default speed 10 is 3 times the default speed	Sets the relative speed adjustment at which words are synthesized.  Note different ranges for SAPI4 and SAPI5.
\vol=number\  The value of is between 0...65535.	<VOLUME LEVEL="number"/>  The value of is between 0...100.	Sets the volume.

The component does not convert any other tags. Conversion does not support scoped tags – all tags are considered global. E.g. you cannot use this SAPI5 format:

<RATE ABSSPEED="+10">I am very fast,</RATE> and now I am slower.

Scoped SAPI 5-tags are considered global tags in conversion.

Following XML entities are also converted:

<b>SAPI4</b>	<b>SAPI5</b>	<b>Description</b>
<	&lt;	Character <
>	&gt;	Character >
&	&amp;	Character &
\\	\	Character \



## Example of the conversion

Here is a sample of SAPI 5 coded text:

```
1<2, 2>1 a&b
<bookmark mark="123"/>Just had a bookmark,
<silence msec="1000"/>and a pause,
<pitch absmiddle="10"/>high pitch,
<pitch absmiddle="0"/>normal pitch,
<rate absspeed="10"/>I speak very fast so you can't maybe understand
this,
<rate absspeed="0"/>normal,
<volume level="0"/>should be total silence,
<volume level="100"/>and volume at maximum.
```

And the same text converted to SAPI4 format:

```
1<2, 2>1 a&b
\mrk=123\Just had a bookmark,
\pau=1000\and a pause,
\rpit=300\high pitch,
\rpit=100\normal pitch,
\rspd=300\I speak very fast so you can't maybe understand this,
\rspd=100\normal,
\vol=0\should be total silence,
\vol=65535\and volume at maximum.
```

## ***Threading and events (desktop applications only)***

Timehouse SpeechServer supports apartment model threading. This means that methods may be called and objects may be destroyed only from the thread it was created. You may have many running threads each having their own SpeechServer object.

This rest of this section applies only to speaking through a sound card with Speak-method.

SpeechServer starts internally a new thread that runs the speech synthesizer, unless the parameter "nonewthread" is specified in Init-method. This allows the synthesizer to run in the background independent of your program.

The Speak()-method works in the background and sends events, whenever something interesting happens. Text buffers are queued and the synthesizer speaks them in the order they are sent through the Speak()-method.

Timehouse SpeechServer sends events to event handlers about synthesizer events.

These events are:

- Synthesizer starts speaking (OnAudioStart)
- Synthesizer stops speaking (OnAudioStop)

- Synthesizer starts speaking one text buffer sent by speak-method (OnTextDataStarted)
- Text buffer has been spoken (OnTextDataDone)
- Specially coded bookmark was encountered (OnBookMark)
- State of talking head has changed (OnViseme)

Example of events:

Operation	Event(s)	Description
Speak() called twice: Speak( "the cat is", 1 ) Speak( "in the moon", 2 )	OnAudioStart OnTextDataStarted(1)	Sound card started and speaking first buffer started
Pause() called	OnAudioStop	Sound card stopped
Resume() called	OnAudioStart	Sound card started
(time passes)	OnTextDataDone(1)	First buffer spoken
(time passes)	OnTextDataStarted(2)	Second buffer started
Stop() called	OnTextDataDone(2) OnAudioStop	Second buffer spoken and audio card stopped

OnBookMark events are sent just when the text in the bookmark's place is played through the audio card. The synthesizer decides the appropriate times to send OnViseme events.

These events are sent using the application's message queue. This means that to fire events, the application must dispatch messages by:

- calling DoEvents() in Visual Basic or Visual Basic for Applications
- calling GetMessage() & DispatchMessage() in C++
- running AWT-program in Java

In C++ or Java programs you may specify "firenopost" flag in Initmehtod. With this flag, events will be fired straight from the synthesizer thread, violating normal threading rules.

## Bookmarks

Bookmarks are 32-bit numbers that you can embed anywhere in the text. You can use them for example to mark each space in the text. A good choice for the bookmark number would be the index of the current character or word.

Bookmarks are coded differently in SAPI4 and SAPI5:

SAPI 4: This\mrk=5\ is\mrk=8\ text.

SAPI 5: This<BOOKMARK MARK="5"/> is<BOOKMARK MARK="8"/> text.

## Visemes

As the synthesizer speaks it sends information that can be used to animate a "talking head". This information contains:

- The phoneme which is currently synthesized
- The phoneme's duration
- The next phoneme
- Whether to stress or not to stress the current letter
- Whether to emphasize the current word or not

Actually phonemes that look similar when spoken are mapped to the same viseme. Visemes are numbered beginning from zero:

VISEME	Described SAPI Phonemes	Word examples
0	Silence	
1	ae, ax, ah	C <u>a</u> t, a <u>g</u> o, c <u>u</u> t
2	Aa	F <u>a</u> ther
3	Ao	D <u>o</u> g
4	ey, eh, uh	<u>A</u> te, p <u>e</u> t, b <u>o</u> ok
5	Er	F <u>u</u> r
6	y, iy, ih, ix	<u>Y</u> ard, f <u>ee</u> l, f <u>i</u> ll
7	W, uw	<u>W</u> ith, t <u>oo</u>
8	ow	G <u>o</u>
9	aw	F <u>o</u> ul
10	oy	T <u>o</u> y
11	ay	B <u>i</u> te
12	H	<u>H</u> elp
13	R	<u>R</u> ed
14	L	<u>L</u> id
15	s, z	<u>S</u> it, z <u>a</u> p
16	sh, ch, jh, zh	<u>S</u> he, <u>ch</u> in, j <u>oy</u> , ple <u>a</u> sure
17	Th, dh	<u>Th</u> in, <u>th</u> en
18	f, v	<u>F</u> ork, <u>v</u> at
19	D, t, n	<u>D</u> ig, t <u>a</u> lk, <u>n</u> o
20	k, g, ng	<u>c</u> ut, <u>g</u> ut, s <u>i</u> ng
21	P, b, m	<u>p</u> ut, b <u>i</u> g, <u>m</u> at

Timehouse Speech Server provides SAPI 5 style information also for SAPI 4 synthesizers. SAPI 4 synthesizers do not give similar information as SAPI 5 does; therefore information is reduced just to the current viseme.

Please see Init-method in the next section for further information.

## MP3 files

MP3 encoding uses the external lame MP3 encoder lame\_enc.dll. You can download it from

<http://www.mp3-tech.org/software/encoders/lamewin32.exe>.

After installation, copy lame\_enc.dll to the same directory where THSpeechServer.dll is installed (Windows\system or Winnt\system32).

For licensing information, see:

<http://www.mp3licensing.com>

## ASP and Visual Basic quick start

Here is the minimal code to use the component:

```
Dim Synt As Object
Dim bResult As Boolean
Dim sFilename As String
Dim sId As String

Set Synt = CreateObject("THSpeechServer.THSynthesizer")
sFilename = "c:\test.wav"

' Initialize the object
bResult = Synt.Init("sapi4,sapi5", "")
If bResult = False Then
    MsgBox "TH Speech Server initialization failed."
End If

' Select default voice or first if no default
sId = Synt.GetDefaultVoice()
If sId = "" Then
    sId = Synt.GetVoiceName(1)
    sId = Mid(sId, InStr(sId, Chr(9)) + 1)
End If

bResult = Synt.SelectVoice(sId)
If bResult = False Then
    MsgBox "Voice selection failed."
End If

' Speak to the file
bResult = Synt.SpeakToFile("wav,plaintext", "Hello world!", sFilename)
If bResult = False Then
    MsgBox "Error saving file."
Else
    MsgBox "File " & sFilename & " created."
End If

' Discard the object
Set Synt = Nothing
```

## Method reference in Visual Basic syntax

### **General**

The component is implemented as a COM/ActiveX object with a dual interface. This allows for using it from Visual Basic using the CreateObject() command.

To use multiple synthesizers, create several THSpeechServer objects instead of using a single object and changing the voice with the SelectVoice() -call.

### **Error handling**

Most of the methods return a Boolean value: 0 (false) means failure and -1 (true) means success. Because Visual Basic's true value (-1) is different from many other languages (1), test only for the false value since it is zero in all languages:

```
bResult = Synt.Init()  
If bResult = False Then  
    ' -- Handle error --  
End If
```

After a failed operation you can query the error message by using the GetError() -method.

Table of error origins:

-10	Initialize-call failed, likely CoInitialize failed
-11	Initialize-call failed, creating notify window
-100	General SAPI4 error
-101	SAPI4 was not installed (Init, GetVoiceCount)
-102	No SAPI4 synthesizer was installed (GetVoiceCount)
-103	Creating SAPI4 audio object failed
-104	Creating SAPI4-synthesizer failed
-105	SAPI4-speak call failed
-106	SAPI4-speak to file call failed
-107	SAPI4 does not have default voice
-200	General SAPI5 error
-201	SAPI5 was not installed
-202	No SAPI5 synthesizer was installed
-204	Creating SAPI5-synthesizer failed
-205	SAPI5-speak call failed
-206	SAPI5-speak to file call failed
-300	General error in SAPI4&SAPI5 controller
-301	Initialize-call failed because no wanted SAPI system files were installed (4 or 5 or both)

## Initialization

### Init

#### Syntax:

```
Synt.Init(sFlags as String, sKey as String)
```

#### Usage:

Initializes the component.

#### Arguments:

String sFlags

String may contain combination of these separated with comma:

- "sapi4" => initializes use of SAPI4 compliant synthesizers
- "sapi5" => initializes use of SAPI5 compliant synthesizers
- "tag4" => Use SAPI4 control tags
- "tag5" => Use SAPI5 control tags
- "onlyfile" => Use only SpeakToFile-method
- "nonewthread"=> Do not create worker thread for synthesizer (doevents-parameter does nothing if this is specified).
- "doevents" => Dispatch all Windows messages while waiting any Speech Server call to complete. Without this parameter calling program is blocked until a call completes.
- "fireall" => Fire all possible events in. Affects only Speak-call. Please see events.
- "firenoviseme"=> Fire all events except viseme-events. Please see events.
- "firenopost" => Fire events straight from synthesizer thread violating normal threading rules. *Use with caution and only in C++ or Java programs without a message queue!*

If you pass an empty string as the sFlags parameter, it will initialize SAPI4 and SAPI5 synthesizers without any tag conversion (equal to initialization string "sapi4,sapi5").

You can specify either one of the "tag4" or "tag5" strings but not both. If you do not specify a tag-parameter, all tags are passed to the synthesizer as they are and it is up to you to check that the tag format is appropriate for your current synthesizer. Use the GetVoiceParam-method to find out whether the current synthesizer is sapi4- or sapi5-compliant.

If you do not specify the "onlyfile" -parameter, a sound card is required when selecting SAPI4 synthesizer with SelectVoice-call. The sound card must also be compatible with the synthesizer (usually a 16 bit sound is required). SAPI5 does not have this limitation – it needs a sound card only when calling Speak-method. You should specify this parameter if you don't need speaking through sound card. Otherwise a compatible sound card is required.

The "nonewthread" -parameter is recommended if you are not using the Speak-method. If not specified, the component handles the synthesizer in its own thread and minimizes requirements for the calling application. With this parameter the synthesizer may stop speaking if the calling application is busy (not dispatching Windows messages). For SpeakToFile-only usage it is recommended to specify this parameter to avoid unnecessary overhead.

String sKey

The license key or empty string if running demonstration version.

*Return value (VARIANT\_BOOL):*

True(-1)    Succeeded  
False(0)    Failed, use GetLastError() to get English error message

*Example:*

```
Dim Synt As Object
Dim bResult As Boolean

Set Synt = CreateObject("THSpeechServer.THSynthesizer")
` Use sapi4&sapi5 synthesizers with sapi5 tag format.
bResult = Synt.Init("sapi4,sapi5,tag5", "")
If bResult = False Then
    MsgBox Synt.GetError()
End If
```

## **SetTempPath**

*Syntax:*

Synt.SetTempPath(sPath as String)

*Usage:*

Sets the path used for temporary files. SpeakToFile method writes always temporary files even if result is returned in memory buffer. If you do not specify the temporary path, it will be one in the following order:

- The path specified by the TMP environment variable.

- The path specified by the TEMP environment variable, if TMP is not defined.
- The Windows directory, if both TMP and TEMP are not defined.

*Arguments:*

String sPath

Full path to temporary folder with or without ending backslash (in other words, "c:\temp" and "c:\temp\" are both ok).

*Return value (VARIANT\_BOOL):*

True(-1) Succeeded

False(0) Failed, use GetLastError() to get English error message.

Current version never fails.

## **Voice selection**

### **SelectVoice**

*Syntax:*

```
Synt.SelectVoice(sVoiceID as String)
```

*Usage:*

Initializes the component to use a specific voice.

*Arguments:*

String sVoiceID

ID of the desired synthesizer. Use THSapiTest.exe to get the ID.

SAPI4 Id's are in format:

```
{C77C5170-2867-11D0-847B-444553540000}
```

SAPI5 Id's are in format:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSMary
```

*Return value (VARIANT\_BOOL):*

True: Success

False: Failed – call GetLastError for details

*Example:*

```
Dim bResult As Boolean

' Microsoft Speech Synthesis Engine: Mary
bResult = Synt.SelectVoice( ' Line continues...
"{C77C5170-2867-11D0-847B-444553540000}")
If bResult = False Then
    MsgBox (Synt.GetError())
End If
```



## GetDefaultVoice

### Syntax:

```
Synt.GetDefaultVoice()
```

### Usage:

Returns ID of the system's default synthesizer. Only SAPI 5 has a default synthesizer. The user can select and manipulate the default synthesizer from the control panel. If you are using SAPI 5, this is the preferred method to select a synthesizer.

### Return value (VARIANT/VT\_BSTR):

ID of the synthesizer for SelectVoice-function. "" Means failure (SAPI 5 not installed or no default voice).

### Example:

```
Dim sId As String
Dim sName As String

sId = Synt.GetDefaultVoice()
If sId = "" Then
    sName = Synt.GetVoiceName(1)
    sId = Mid(sName, InStr(sName, Chr(9)) + 1)
End If

Synt.SelectVoice (sId)
```

## GetVoiceCount

### Syntax:

```
Synt.GetVoiceCount()
```

### Usage:

Returns number of installed synthesizers.

### Return value (VARIANT/VT\_I4):

The number of installed synthesizers or actually the number of voices is saved in this parameter.

Zero means that there are no synthesizers installed or an error has occurred. Call GetError() for details.

### Example:

```
Dim nVoices As Integer

nVoices = Synt.GetVoiceCount()
If nVoices = 0 Then
    MsgBox (Synt.GetError())
End if
```

## GetVoiceName

### Syntax:

```
Synt.GetVoiceName( nIndex as Integer )
```

### Usage:

Returns the name and ID of one synthesizer.

### Arguments:

nIndex

Index between 1...number of voices installed.

### Return value (VARIANT/VT\_BSTR):

Empty string "" indicates failure.

Otherwise the name is returned in the format:  
Human readable name<tab>ID

### For example:

```
Microsoft Speech Synthesis Engine: Mary  
{C77C5170-2867-11D0-847B-444553540000}
```

In the above example the Tab-character chr(9) is replaced with line break.

### Example:

```
Dim sName, sId As String  
  
sName = Synt.GetVoiceName(1)  
If sName = "" Then  
    MsgBox (Synt.GetError())  
else  
    sId = Mid(sName, InStr(sName, Chr(9)) + 1)  
    sName = Left(sName, InStr(sName, Chr(9)) - 1)  
    Synt.SelectVoice( sId )  
End If
```

## GetVoiceParam

### Syntax:

```
Synt.GetVoiceName( sSelector as String )
```

### Usage:

Returns the specified synthesizer's parameter. Call this only after selecting a synthesizer by using the SelectVoice() -call.

### Arguments:

sSelector

Defines which parameter is returned. Must be one of following:

**Language**

Returns language id of the synthesizer. Please see appendix B for language ids.

**Version**

Returns "4" for SAPI4-synthesizer and "5" for SAPI5-synthesizer

**Hasviseme**

Returns "0" or "1" depending on whether the synthesizer supports the OnViseme event. Use this if you need viseme information and be aware that not all synthesizers support sending OnViseme-events. (This feature is not useful for Web server use)

*Return value (VARIANT/VT\_BSTR):*

An empty string indicates failure. Other values are returned parameters.

*Example:*

```
Dim sValue As String

' Assume here that voice has been selected before
sVer = Synt.GetVoiceParam("version")
If sVer = "" Then
    MsgBox (Synt.GetError())
Else
    MsgBox ("Version is " & sVer )
End If
```

***Speaking & converting to MP3*****SpeakToFile**

*Syntax:*

```
Synt.SpeakToFile(sFlags as String, sText as String, sFile as String)
```

*Usage:*

Speaks given text to file in WAV-format. The format of the WAV-file (sample rate and sample width) is set by the synthesizer and therefore varies between synthesizers. Method returns after speaking is completed (it may take a while to speak long text).

*Arguments:*

String sFlags

Empty string saves WAV-file in synthesizer's natural format. Please see method WavToMp3 for options controlling saving in MP3-format.

SFlags may contain also parameter:

- "plaintext"  
Given sText does not contain any tagging. SpeechServer will convert all tagging characters so that they will be spoken. For example SAPI5-tagging: "1 < 2" will be converted to "1 &lt; 2".

#### String sText

Text to speak. The text may contain control tags – remember to convert control characters before inserting control tags, or using built-in conversion. For example, if using:

SAPI4-tag format, convert:

"c:\>file.txt" -> "c:\\>file.txt"

SAPI5-tag format, convert:

"c:\>file.txt" -> "c:\&gt;file.txt"

#### String sFile

The synthesized speech is saved to this file. An empty string "" means speak to memory buffer and return memory buffer. The method needs to write a file even if the memory buffer is returned. In this case a temporary file is created to the folder described in method SetTempPath.

You need write permissions to the specified folder – if you are using the Internet Information Server and have problems using the memory buffer, please check the permissions of the temporary folder.

*Return value (VARIANT/VT\_BOOL or VARIANT/ VT\_ARRAY|VT\_UI1):*

If failed: Boolean value false.

If parameter sFile is empty: Memory buffer containing synthesized file.

If parameter sFile contained file name and successful: Boolean value true.

*Notes:*

It is easiest to detect a possible error condition after this call by calling the `GetError()` method. If it returns an empty string, the operation was successful.

*Example:*

```
Dim rBuf As Variant
Dim bResult As Boolean

' Speak to the file
bResult = Synt.SpeakToFile("wav", "Hello world!", sFileName)

If Synt.GetError() <> "" Then
    MsgBox (Synt.GetError())
Else
    MsgBox ("File " & sFileName & " created.")
End If

' Speak to memory buffer
rBuf = Synt.SpeakToFile("wav", "Hello world!", "")

If Synt.GetError() <> "" Then
    MsgBox (Synt.GetError())
Else
    MsgBox ("Buffer lenght = " & UBound(rBuf)-LBound(rBuf)+1)
End If
```

## WavToMp3

*Syntax:*

```
Synt.WavToMp3(sFlags as String, sSourceFile as String, sDestFile as String)
```

*Usage:*

Converts given WAV-file to MP3 format. Please see section [MP3 licensing & other information](#).

*Arguments:*

String sFlags

String may contain combination of these parameters separated with commas:

- "mp3"  
Save in MP3 format. In future versions there may be other formats also.
- "kilobits=x"  
Sets saving quality in kilobits per seconds (default is 128).  
Example:        kilobits=128.
- "samplerate=x"  
Sets the resampling rate in Herzes (default allows encoder to decide best sampling rate according to other parameters).  
Example:        samplerate=44100.
- "mono"  
Sets output format to mono.

- "stereo"  
Sets output format to stereo (default).

String sSourceFile

The source file in WAV-format, must be a mono or stereo file with 8 or 16 bits sample size. The sampling rate can be anything.

String sDestFile

The MP3 data is saved to this file. An empty string "" means, convert to memory buffer and return memory buffer.

*Return value (VARIANT/VT\_BOOL or VARIANT/ VT\_ARRAY|VT\_UI1):*

If failed: Boolean value false.

If parameter sFile is empty: Memory buffer containing synthesized file.

If parameter sFile contained file name and successful: Boolean value true.

*Notes:*

It is easiest to detect a possible error condition after this call by calling the GetError() method. If it returns empty string, the operation was successful.

*Example:*

```
Dim rBuf As Variant
Dim bResult As Boolean

' Speak to the file
bResult = Synt.SpeakToFile("wav", "Hello world!", sFileName)

If Synt.GetError() <> "" Then
    MsgBox (Synt.GetError())
Else
    MsgBox ("File " & sFileName & " created.")
End If

' Speak to memory buffer
rBuf = Synt.SpeakToFile("wav", "Hello world!", "")

If Synt.GetError() <> "" Then
    MsgBox (Synt.GetError())
Else
    MsgBox ("Buffer lenght = " & UBound(rBuf)-LBound(rBuf)+1)
End If
```

## **Setting voice parameters**

### **SetRate**

*Syntax:*

```
Synt.SetRate(nRate As Long)
```

**Usage:**

Sets the rate of the synthesizer. SAPI4 uses words per minute as scale and SAPI5 uses range -10...10 where zero is natural speed, -10 minimum and 10 is maximum. To make this voice parameter adjustable, query for minimum and maximum values and let the user set value between these.

Changing the rate of speech during a task does not affect the current task instantly. The change will take effect once the speech buffer is depleted (a few seconds). If you want to be able to change the rate instantly, you have to stop the current task, change the rate, and restart the task, or you need to use the SAPI control tags mentioned earlier.

**Arguments:**

Long nRate

Speech rate in synthesizer's scale. Pseudo value -9999 sets rate to minimum and 9999 to maximum.

**Return value (VARIANT\_BOOL):**

True(-1)      Succeeded

False(0)      Failed, use GetLastError() to get verbose error message

**Example:**

```
' Find min&max values for rate
' First save current value
nVal = Synt.GetRate()

' Set to minimum and ask for actual used value
Synt.SetRate (-9999)
nMin = Synt.GetRate()

' Set to maximum and ask for actual used value
Synt.SetRate (9999)
nMax = Synt.GetRate()

' Restore default value
Synt.SetRate (nVal)
MsgBox( "Rate: Min=" & Str(nMin) & " Max=" & Str(nMax) )
```

## **GetRate**

**Syntax:**

```
nRate = Synt.GetRate()
```

**Usage:**

Returns the current speech rate.

**Return value (VARIANT/VT\_LONG):**

-9999        Failed  
Any other    Current rate in synthesizer's internal scale.

*Example – please see SetRate example.*

## **SetPitch (Only for SAPI4 synthesizers)**

### *Syntax:*

```
Synt.SetPitch(nPitch As Long)
```

### *Usage:*

Sets the pitch of the synthesizer. SAPI4 uses Hertz as scale. SAPI5 does not support this method. To make this voice parameter adjustable, query for minimum and maximum values and let user set value between these.

Changing the pitch of speech during a task does not affect the current task instantly. The change will take effect once the speech buffer is depleted (a few seconds). If you want to be able to change the pitch instantly, you have to stop the current task, change the pitch, and restart the task, or you need to use the SAPI control tags mentioned earlier.

### *Arguments:*

Long nPitch

Speech pitch in synthesizer's scale. Pseudo value -9999 sets pitch to minimum and 9999 to maximum.

### *Return value (VARIANT\_BOOL):*

True(-1)    Succeeded

False(0)    Failed, use GetLastError() to get English error message

### *Example:*

```
' Find min&max values for pitch
' First save current value
nVal = Synt.GetPitch()

' Set to minimum and ask for actual used value
Synt.SetPitch(-9999)
nMin = Synt.GetPitch()

' Set to maximum and ask for actual used value
Synt.SetPitch (9999)
nMax = Synt.GetPitch ()

' Restore default value
Synt.SetPitch (nVal)
MsgBox( "Pitch: Min=" & Str(nMin) & " Max=" & Str(nMax) )
```

## **GetPitch (Only for SAPI4 synthesizers)**

### *Syntax:*



```
nPitch = Synt.GetPitch()
```

**Usage:**

Returns current speech pitch. SAPI5 does not support this method.

**Return value (VARIANT/VT\_LONG):**

-9999        Failed  
Any other    Current pitch in synthesizer's internal scale.

*Example – please see SetPitch example.*

## **SetVolume**

**Syntax:**

```
Synt.SetVolume(nVolumePercent As Long)
```

**Usage:**

Depending on the synthesizer and Windows version used, setting the synthesizer's output volume may affect the overall wave volume in Windows. Therefore it is recommended to store the original wave volume and set it back when quitting the program.

Microsoft's synthesizers do not change global mixer settings; instead they calculate volume in digital signal. Setting 100 is maximum, 50 is -6 dB, 25 is -12 dB, 12 is -24 dB etc.

**Arguments:**

Long nVolumePercent  
Volume percent from maximum.

**Return value (VARIANT\_BOOL):**

True(-1)    Succeeded  
False(0)    Failed, use GetLastError() to get English error message

**Example:**

```
' Set volume to maximum (never clips the signal)  
Synt.SetVolume(100)
```

## **GetVolume**

**Syntax:**

```
nVolumePercent = Synt.GetVolume()
```

**Usage:**

Returns current speech volume.

*Return value (VARIANT/VT\_LONG):*

-9999      Failed

Any other    Current volume percent.

## ***Speaking through the sound card***

### **Speak**

#### ***Syntax:***

`Synt.Speak (sFlags as String, sText as String, nData as Long)`

#### ***Usage:***

Speaks the given text via the soundcard. Puts the given text at the end of the speech queue and returns immediately (asynchronous call). If you gave parameter "nonewthread" to Init-call, the synthesizer may stop speaking if the calling application is busy (not dispatching Windows messages).

#### ***Function arguments:***

String sFlags

Parameter may contain parameter:

- "plaintext"  
Given sText does not contain any tagging. SpeechServer will convert all tagging characters so that they will be spoken. For example SAPI5-tagging: "1 < 2" will be converted to "1 &lt; 2".

String sText

Text to speak. The text may contain control tags – remember to convert control characters before inserting control tags and passing text to the synthesizer. For example, if using:

SAPI4-tag format, convert:

"c:\>file.txt" -> "c:\\>file.txt"

SAPI5-tag format, convert:

"c:\>file.txt" -> "c:\&gt;file.txt"

Long nData

32-bit number attached to this text. This number is passed to all events fired for this text.

#### ***Return value (VARIANT\_BOOL):***

True(-1) Succeeded

False(0) Failed, use GetLastError() to get English error message

#### ***Example:***

' Speak the text

```

bResult = Synt.Speak ("", "Hello world!")

If bResult = False Then
    MsgBox (Synt.GetError())
End If

```

## GetStatus

### Syntax:

```
nResult = Synt.GetStatus ()
```

### Usage:

Returns the status of the synthesizer.

### Return value (VARIANT/VT\_I4):

0	Synthesizer idle, not speaking
1	Synthesizer speaking
2	Synthesizer paused

### Example:

```

' Speak the text
bResult = Synt.Speak ("", "Hello world!")

If bResult = False Then
    MsgBox (Synt.GetError())
Else
    ' Wait until text is spoken
    ' NOT RECOMMENDED WAY - USE EVENTS INSTEAD
    While Synt.GetStatus() <> 0
        Wend
End If

```

## Pause

### Syntax:

```
bResult = Synt.Pause(bExecute As Boolean)
```

### Usage:

Pauses speaking or makes a query if pausing is possible.

### Function arguments:

Boolean bExecute	
True	Execute the command
False	Just query (for updating pause button or similar)

### Return value (VARIANT\_BOOL):

True(-1)	Succeeded
False(0)	Failed, use GetError() to get English error message

### Example:

```

' Pause
Synt.Pause(True)

```

## Resume

### Syntax:

```
bResult = Synt.Resume(bExecute As Boolean)
```

### Usage:

Resumes paused speaking or makes a query if resuming is possible.

### Function arguments:

Boolean bExecute

True=Execute the command

False=Just query (for updating resume button or similar)

### Return value (VARIANT\_BOOL):

True(-1) Succeeded

False(0) Failed, use GetLastError() to get English error message

### Example:

```
' Resume  
Synt.Resume(True)
```

## Stop

### Syntax:

```
bResult = Synt.Stop()
```

### Usage:

Stops speaking immediately and flushes all queued texts.  
Returns after operation is completed (synchronous call).

### Return value (VARIANT\_BOOL):

True(-1) Succeeded

False(0) Failed, use GetLastError() to get English error message

### Example:

```
' Speak the text  
bResult = Synt.Speak ("", "Hello world!")  
If bResult = False Then  
    MsgBox (Synt.GetError())  
End If  
  
' Stop speaking  
bResult = Synt.Stop()  
If bResult = False Then  
    MsgBox (Synt.GetError())  
End If
```

## ***Helper methods***

### **DeleteFile**

#### ***Syntax:***

```
Synt.DeleteFile(sFile as String)
```

#### ***Usage:***

Deletes the given file. Use this method to delete any temporary files you have created.

#### ***Arguments:***

String sFile

Full path name of the file to delete. Name may contain wild characters \* and ?.

#### ***Return value (VARIANT\_BOOL):***

True(-1)    Succeeded

False(0)    Failed, use GetLastError() to get English error message

### **ReadFile**

#### ***Syntax:***

```
Synt.ReadFile(sFile as String)
```

#### ***Usage:***

Reads given file into memory.

#### ***Arguments:***

String sFile

Full path name of the file.

#### ***Return value (VARIANT/VT\_BOOL or VARIANT/ VT\_ARRAY|VT\_UI1):***

If failed: Boolean value false.

If successful: Memory buffer containing the file.

#### ***Example:***

```
' Read the file
sErr = oSynt.ReadFile(sFileName, rFile)
If sErr <> "" Then
    MsgBox (sErr)
Else
    MsgBox ("File " & sFileName & " read, size=" & (UBound(rFile) -
LBound(rFile) + 1))
End If
```

## Redistribution and guidelines for desktop applications

Redistribution is similar to [manual installation](#). The only files you need to install for the target system are THSpeechServer.dll and THSpeechServer.ini. Place THSpeechServer.dll to the self-registering file group targeted for the system-folder and THSpeechServer.ini to the normal file group, also targeting the system-folder. Timehouse SpeechServer uses the already installed speech synthesizers.

If you are using C++ or Java interface, registering the DLL is not required since interface is native unlike Visual Basic / scripting interface using COM.

To make your application work with future versions of the intended synthesizer you should have the speech setup dialog showing all installed voices on the user system (methods GetVoiceCount() and GetVoiceName()) and allow the user to select the voice from that list. Save the voice ID from the list and use it in the SelectVoice()-call when your program starts. If SelectVoice fails, show error message "Initializing speech synthesizer failed. Opening speech settings dialog. [And detailed English error message here to be able to give better support (GetError()-method)].".

Add requirement for using the application: "Application requires an already installed SAPI4 or SAPI5 compatible speech synthesizer. For example Microsoft Speech Synthesis Engine."

If there are not any synthesizers installed in the system, the Init-call may return failed code (false) or GetVoiceCount returns zero. For the user, the situation in these two cases is the same, but the actual reason is slightly different:

1. If Init fails: There are no required Microsoft's system files (SAPI4 or SAPI5) installed. (No SAPI version installed on the system).
2. If Init succeeds but GetVoiceCount returns zero: All speech synthesizers have been uninstalled but the SAPI system files still exist (since they cannot be uninstalled).

## **Appendix A, installation issues**

SAPI4 and SAPI5 can coexist. In other words it is possible to install and use both synthesizers in the same system.

Please read page "Licensing Microsoft Speech Technology" from the web address:

<http://www.microsoft.com/speech/legal/>

### **SAPI4**

With SAPI4 Microsoft provides two executables for installations:

- SAPI4 system files in file spchapi.exe
- Microsoft Text-to-Speech Engine 4.0 in file msttsa22l.exe

To install the synthesizer, first execute spchapi.exe and then msttsa22l.exe.

### **SAPI5**

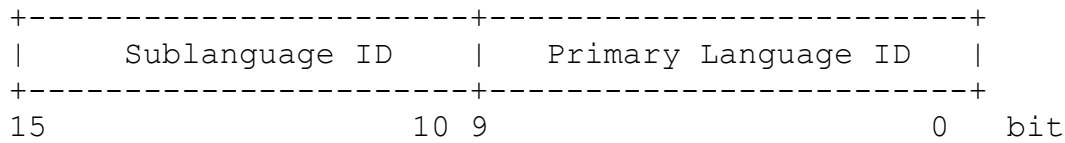
Unfortunately Microsoft does not provide any ready installation executable for SAPI5. Therefore we have created an installation program with InstallShield for Windows Installer (version 1.52). This installation program is a single executable (Sapi51AndMSTts51eng.exe) that installs SAPI5 system files, SAPI5 speech control panel and Microsoft Text to Speech Engine English. You are free to use this installation program. You can download installation program from [http://products.timehouse.fi/ss\\_download.asp](http://products.timehouse.fi/ss_download.asp).

For more information download SAPI5 SDK from <http://www.microsoft.com/speech/>.



## Appendix B, language codes

A language ID is a 16 bit value which is the combination of a primary language ID and a secondary language ID. The bits are allocated as follows:



List of primary language Ids. Please note that to get primary language from language Id, you must and the Id with number 0x3FF (1023 in decimal).

0x00	LANG_NEUTRAL	Neutral
0x01	LANG_ARABIC	Arabic
0x02	LANG_BULGARIAN	Bulgarian
0x03	LANG_CATALAN	Catalan
0x04	LANG_CHINESE	Chinese
0x05	LANG_CZECH	Czech
0x06	LANG_DANISH	Danish
0x07	LANG_GERMAN	German
0x08	LANG_GREEK	Greek
0x09	LANG_ENGLISH	English
0x0a	LANG_SPANISH	Spanish
0x0b	LANG_FINNISH	Finnish
0x0c	LANG_FRENCH	French
0x0d	LANG_HEBREW	Hebrew
0x0e	LANG_HUNGARIAN	Hungarian
0x0f	LANG_ICELANDIC	Icelandic
0x10	LANG_ITALIAN	Italian
0x11	LANG_JAPANESE	Japanese
0x12	LANG_KOREAN	Korean
0x13	LANG_DUTCH	Dutch
0x14	LANG_NORWEGIAN	Norwegian
0x15	LANG_POLISH	Polish
0x16	LANG_PORTUGUESE	Portuguese
0x18	LANG_ROMANIAN	Romanian

0x19	LANG_RUSSIAN	Russian
0x1a	LANG_CROATIAN	Croatian
0x1a	LANG_SERBIAN	Serbian
0x1b	LANG_SLOVAK	Slovak
0x1c	LANG_ALBANIAN	Albanian
0x1d	LANG_SWEDISH	Swedish
0x1e	LANG_THAI	Thai
0x1f	LANG_TURKISH	Turkish
0x20	LANG_URDU	Urdu
0x21	LANG_INDONESIAN	Indonesian
0x22	LANG_UKRANIAN	Ukrainian
0x23	LANG_BELARUSIAN	Belarusian
0x24	LANG_SLOVENIAN	Slovenian
0x25	LANG_ESTONIAN	Estonian
0x26	LANG_LATVIAN	Latvian
0x27	LANG_LITHUANIAN	Lithuanian
0x29	LANG_FARSI	Farsi
0x2a	LANG_VIETNAMESE	Vietnamese
0x2b	LANG_ARMENIAN	Armenian
0x2c	LANG_AZERI	Azeri
0x2d	LANG_BASQUE	Basque
0x2f	LANG_MACEDONIAN	FYRO Macedonian
0x36	LANG_AFRIKAANS	Afrikaans
0x37	LANG_GEORGIAN	Georgian
0x38	LANG_FAEROESE	Faeroese
0x39	LANG_HINDI	Hindi
0x3e	LANG_MALAY	Malay
0x3f	LANG_KAZAK	Kazak
0x40	LANG_KYRGYZ	Kyrgyz
0x41	LANG_SWAHILI	Swahili
0x43	LANG_UZBEK	Uzbek
0x44	LANG_TATAR	Tatar
0x45	LANG_BENGALI	Bengali
0x46	LANG_PUNJABI	Punjabi
0x47	LANG_GUJARATI	Gujarati

0x48	LANG_ORIYA	Oriya
0x49	LANG_TAMIL	Tamil
0x4a	LANG_TELUGU	Telugu
0x4b	LANG_KANNADA	Kannada
0x4c	LANG_MALAYALAM	Malayalam
0x4d	LANG_ASSAMESE	Assamese
0x4e	LANG_MARATHI	Marathi
0x4f	LANG_SANSKRIT	Sanskrit
0x50	LANG_MONGOLIAN	Mongolian
0x56	LANG_GALICIAN	Galician
0x57	LANG_KONKANI	Konkani
0x58	LANG_MANIPURI	Manipuri
0x59	LANG_SINDHI	Sindhi
0x5a	LANG_SYRIAC	Syriac
0x60	LANG_KASHMIRI	Kashmiri
0x61	LANG_NEPALI	Nepali
0x65	LANG_DIVEHI	Divehi
0x7f	LANG_INVARIANT	

## Appendix C, SAPI4 control tags

This is a copy of Microsoft SAPI 4.0a documentation. You can download full SDK from

<http://www.microsoft.com/speech/speechsdk/sdk40a.asp>.

### ***Control Tags***

This section describes the text-to-speech control tags that can be embedded in the source text to improve the prosody of text-to-speech translation:

- [Syntax](#)
- [Conventions](#)
- [Chr](#)
- [Com](#)
- [Ctx](#) (some speech engines may not support this tag)
- [Dem](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [Emp](#) (some speech engines may not support this tag)
- [Eng](#)
- [Mrk](#)
- [Pau](#)
- [Pit](#)
- [Pra](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [Prn](#) (some speech engines may not support this tag)
- [Pro](#) (some speech engines may not support this tag)
- [Prt](#) (some speech engines may not support this tag)
- [RmS](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [RmW](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [RPit](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [RPrn](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [RSpd](#) (new for SAPI 4.0 -- some speech engines may not support this tag)
- [Rst](#) (new for SAPI 4.0)
- [Spd](#)
- [Vce](#) (some speech engines may not support this tag)
- [Vol](#)

A text-to-speech engine can usually translate individual words to speech successfully. However, as soon as the engine speaks a sentence, the perceived quality of its translation decreases because the engine cannot correctly synthesize human prosody -- the inflection, accent, and timing of human speech.

The prosody of translated speech can be improved by using text-to-speech control tags to better simulate human speech. Control tags can be embedded in the source text passed to an engine with the [ITTSentral::TextData](#) member function, or they can be inserted into the text that is currently playing by calling the [ITTSentral::Inject](#) member function. This allows an application to alter prosody of text as it is spoken, without having to reconstruct the text-to-speech buffers.

This section describes control tags that you can use to alter the prosody of text translated into speech. All tags are optional except the bookmark (Mrk) tag, which must be supported.

## Syntax

Text-to-speech control tags follow these general rules of syntax:

- All tags begin and end with a backslash character (\).
- The backslash character is not allowed within a tag.
- An odd number of backslash characters in tagged text produce undefined behavior in the engine.
- Tags are case-insensitive. For example, `\vce\` is the same as `\VCE\`.
- Tags are white-space -- dependent. For example, `\Rst\` is not the same as `\ Rst \`.

To include a backslash character in tagged text, but outside a tag, use a double backslash (`\\`).

If the application has the tagged text bit on and wishes to speak a filename, such as `"c:\windows\system\test.txt"`, then it should double up the backslashes (e.g., `"c:\\windows\\system\\test.txt"`).

Samples:

1. `\ ctx="e-mail" \` - Should be parsed as an "e-mail" tag.
2. `\ctx="e-mail"` - Should be ignored since it's an unclosed tag at the end of a document.
3. `\\ctx="e-mail"\\` - Speaks "back-slash c t x equals e-mail back-slash"

4. \\\\\\\ctx="e-mail"\\\\\\ - Speaks "backslash backslash backslash c t x equals e-mail backslash backslash backslash backslash"
5. \ctm="e-mail" - Ignored because it's an unknown tag.

When the text-to-speech engine encounters a tag it does not understand, the tag is ignored.

Tags are persistent from one call to the [ITTSentral::TextData](#) member function to another. For example, if the `\ctx="e-mail"` tag is passed to an engine that supports that tag, the engine stays in the "e-mail" context until another tag changes the context.

## Conventions

This section uses the following typographic conventions.

Example	Description
<b>Chr</b>	Bold type indicates speech-inflection keywords.
<i>string</i>	Italic indicates placeholders for information you supply, such as a character or context string.
[[option]]	Double square brackets indicate items that are optional.
[[option...]]	Three dots (an ellipsis) following an item indicate that more items having the same form may appear.
"C"	Quotation marks are required to delimit strings.

## Chr

`\Chr=string[,string...]\`

Sets the character of the voice.

Example: `\Chr="Angry"`

## Parameter Description

<i>string</i>	String that specifies the characteristics of the voice.
---------------	---

The default characteristic is "Normal," which produces a normal tone of voice. Although the Chr tag is less specific than setting the inflection, stress, attack, and whispering qualities individually, it is easier to use and allows the engine more flexibility and intelligence in its response.

Several characteristics can be specified in the same tag, separated by commas. Depending on its capabilities, an engine may not support all of the characteristics listed here, or it may support additional characteristics. Some common characteristics are the following:

"Angry"  
"Business"  
"Calm"  
"Depressed"  
"Excited"  
"Falsetto"  
"Happy"  
"Loud"  
"Monotone"  
"Perky"  
"Quiet"  
"Sarcastic"  
"Scared"  
"Shout"  
"Tense"  
"Whisper"

## **Com**

`\Com=string\`

Embeds a comment in the text. Comments are not translated into speech.

Example: `\com="This is a comment." \`

Parameter	Description
<i>String</i>	Text of the comment.

## **Ctx**

`\Ctx=string\`

**Note:** Some speech engines may not support this tag.

Sets the context for the text that follows, which determines how symbols are spoken.

Example: `\ctx="Unknown" \`

Parameter	Description
<i>string</i>	String that specifies the context.

This parameter can be one of these strings:

Context string	Description
-------------------	-------------

"Address"	Addresses and/or phone numbers
"C"	Code in the C or C++ programming language
"Document"	Text document
"E-Mail"	Electronic mail
"Numbers"	Numbers, dates, times, and so on
"Spreadsheet"	Spreadsheet document
"Unknown"	Context is unknown (default)
"Normal"	Normal/default speech mode.

## ***Dem***

`\Dem\`

**Note:** Some speech engines may not support this tag.

De-emphasizes the next word.

## ***Emp***

`\Emp\`

**Note:** Some speech engines may not support this tag.

Emphasizes the next word to be spoken.

Example: the `\Emp\`truth, the `\Emp\`whole truth, and nothing `\Emp\`but the truth.

## ***Eng***

`\Eng; GUID: command\`

`\Eng:command\`

Embeds an engine-specific command that affects only the engine with the specified globally unique identifier (GUID). Subsequent engine-specific tags for that engine can omit the GUID until an engine-specific tag for a different engine is used.

## **Parameter      Description**

*GUID*              Globally unique identifier of the engine.

*command*          Engine-specific command.

## ***Mrk***

`\Mrk=number\`

Indicates a bookmark in the text.



Example: \Mrk=75000\

Parameter	Description
<i>number</i>	Number of the bookmark.

When the text-to-speech engine encounters this tag, it notifies the application by calling the [ITTSBufNotifySink::BookMark](#) member function.

Bookmarks have a DWORD range (specified in the dwMarkNum parameter of the [ITTSBufNotifySink::BookMark](#) member function), and the Mrk tag accepts a decimal representation. Bookmark number zero (\Mrk=0\) is reserved; a [ITTSBufNotifySink::BookMark](#) member function is not sent for bookmark number zero.

Bookmark tags are inserted directly into the text sent to the engine when calling the [ITTSCentral::TextData](#) member function.

### ***Pau***

\Pau=*number*\

Pauses speech for the specified number of milliseconds.

Example: \Pau=1000\

Parameter	Description
<i>number</i>	Number of milliseconds to pause.

### ***Pit***

\Pit=*number*\

Sets the baseline pitch of the text-to-speech mode to the specified value in hertz.

Example: \Pit=90\

Parameter	Description
<i>number</i>	Pitch, in hertz.

The actual pitch fluctuates above and below this baseline. Embedding a Pit tag in text is the same as calling the [ITTSAttributes::PitchSet](#) member function.

### ***Pra***

\Pra=*value*\

**Note:** Some speech engines may not support this tag.

Sets the pitch range.

Parameter	Description
<i>value</i>	Pitch range in Hz.

### ***Prn***

`\Prn=text=pronunciation\`

**Note:** Some speech engines may not support this tag.

Indicates how to pronounce text by passing the phonetic equivalent to the engine.

Example: `\Prn=tomato=tomaato\`

Parameter	Description
-----------	-------------

<i>text</i>	Text to pronounce.
-------------	--------------------

<i>pronunciation</i>	Phonetic equivalent for pronouncing the text.
----------------------	---

Using the **Prn** tag without specifying the word pronunciation (`\Prn=text\`) will undo the changes to the pronunciation; that is, it will return the word to the original pronunciation.

This tag is valid only for engines that support the International Phonetic Alphabet. The engine should continue to use this pronunciation for the current text-to-speech mode and should store the pronunciation in its lexicon for later use. If a lexicon entry already exists for a particular word, the **Prn** tag should be ignored for that word.

### ***Pro***

`\Pro=number\`

**Note:** Some speech engines may not support this tag.

Activates and deactivates prosodic rules, which affect pitch, speaking rate, and volume of words independently of control tags embedded in the text. Prosodic rules are applied by the engine.

Example: `\Pro=0\`

Parameter	Description
-----------	-------------

<i>number</i>	Number that indicates whether to activate or deactivate prosodic rules. Setting number to 1, the default, activates prosodic rules. Setting number to 0 deactivates prosodic rules.
---------------	---

Prosody does not have a corresponding [ITTSAttributes](#) interface as speed and pitch do. If the engine supports control of prosody, use the Pro tag.

## ***Prt***

`\Prt=string\`

**Note:** Some speech engines may not support this tag.

Indicates the part of speech of the next word.

Example: `\prt="Abbr"\`

<b>Parameter</b>	<b>Description</b>
------------------	--------------------

<i>string</i>	String that indicates the part of speech.
---------------	---

This parameter can be one of these strings:

<b>String</b>	<b>Description</b>
"Abbr"	Abbreviation
"Adj"	Adjective
"Adv"	Adverb
"Card"	Cardinal number
"Conj"	Conjunction
"Cont"	Contraction
"Det"	Determiner
"Interj"	Interjection
"N"	Noun
"Ord"	Ordinal number
"Prep"	Preposition
"Pron"	Pronoun
"Prop"	Proper noun
"Punct"	Punctuation
"Quant"	Quantifier
"V"	Verb

**For more information** about the parts of speech, see [VOICEPARTOFSPEECH](#).

## ***RmS***

`\RmS=number\`

**Note:** Some speech engines may not support this tag.

Sets reading mode to spelling out each letter of each word, or turns it off. Not all engines support this tag, so an application can get more consistent results by normalizing the text itself and putting spaces between letters.

### Parameter Description

*number*      Number that indicates whether to spell out each letter of each word. Setting number to 1 sets reading mode to spelling out each letter of each word. Setting number to 0, the default, turns it off.

### ***RmW***

`\RmW=number\`

**Note:** Some speech engines may not support this tag.

Sets reading mode to leaving audible pauses between each word, or turns it off. Not all engines support this tag, so an application can get more consistent results by normalizing the text itself and putting periods between words.

### Parameter Description

*number*      Number that indicates whether to leave audible pauses between each word. Setting number to 1 sets reading mode to leave audible pauses between each word. Setting number to 0, the default, turns it off.

### ***RPit***

`\RPit=value\`

**Note:** Some speech engines may not support this tag.

Sets the relative pitch.

### Parameter Description

*value*      Relative pitch. 100 is the default/original for the voice.

### ***RPrn***

`\RPrn=value\`

**Note:** Some speech engines may not support this tag.

Set the relative pitch range.

## Parameter Description

*value*            Relative pitch range. 100 is the default/original for the voice.

### ***RSpd***

`\RSpd=value\`

**Note:** Some speech engines may not support this tag.

Sets the relative speed. 100 is the default/original for the voice.

## Parameter Description

*value*            Relative speed. 100 is the default/original for the voice.

### ***Rst***

`\Rst\`

Resets the engine to the default settings for the current mode, as though the mode had just been re-created.

### ***Spd***

`\Spd=number\`

Sets the baseline average talking speed of the text-to-speech mode to the specified number of words per minute.

Example: `\Spd=120\`

## Parameter Description

*number*           Baseline average talking speed, in words per minute.

Embedding a Spd tag in text is the same as calling the [ITTSAttributes::SpeedSet](#) member function.

### ***Vce***

`\Vce=character=value[ [character=value] ]\`

**Note:** Some speech engines may not support this tag.

Instructs the engine to change its speaking voice to one that has the specified characteristics. The voice characteristics change as though a new mode object were created (using that voice) and used. The pitch, speed, volume, etc. revert to the defaults for the new voice. If [ITTSCentral::ModeGet](#) is called, it will reflect the new mode.

## Parameter Description

*charact* One of the characteristics listed in the table below.

*value* String that specifies the characteristic.

Characteristics are specified in the order of importance. The engine selects a voice that most closely matches the characteristics specified at the beginning of the list.

The *charact=value* argument can be any of the following:

- **Language**=*language*. Requests the engine speak in the specified language
- **Accent**=*accent*. Requests the engine speak in the specified accent. For example, if language="English" and Accent="French" the engine will speak English with a French accent.
- **Dialect**=*dialect*. Requests the engine speak in the specified dialect
- **Gender**=*gender*. Specifies the gender of the voice: "Male", "Female", or "Neutral".
- **Speaker**=*speakername*. Specifies the name of the voice, or NULL if the name is unimportant
- **Age**=*age*. Specifies the age of the voice, which can be one of the values shown below
- **style**=*style*. The personality of the voice -- for example: "Business", "Casual", "Computer", "Excited", or "Singsong".

### Age string

### Description

"Baby"	About 1 year old
"Toddler"	About 3 years old
"Child"	About 6 years old
"Adolescent"	About 14 years old
"Adult"	Between 20 and 60 years old
"Elderly"	Over 60 years old

## Vol

`\Vol=number\`

Sets the baseline speaking volume for the text-to-speech mode.

Example: \vol=32768\

<b>Parameter</b>	<b>Description</b>
<i>number</i>	Baseline speaking volume.

The volume level is a linear range from 0 for absolute silence to 65535 for the maximum monaural volume. The default is 65535.

If you specify a value greater than 65535, the engine assumes that you want to set the left and right channels separately and converts the value to a double word, using the low word for the left channel and the high word for the right channel. For example, a value of 65536 sets the left channel to the maximum baseline speaking volume and the right channel to the minimum.

Embedding a Vol tag in text is similar to calling the [ITTSAttributes::VolumeSet](#) member function.

## Appendix D, SAPI5 control tags

Please download SAPI 5.1 SDK from address

<http://www.microsoft.com/speech/sapi51/SpeechSDK/sdk51.asp>.

### SAPI Elements

#### <BOOKMARK>

Inserts a bookmark into the input stream using the bookmark element. If an application specifies interest in bookmark events, it will receive an event when synthesis has passed this element in an input stream. If the audio output destination supports handling of events, then an application will receive this event once the synthesized speech up to this bookmark has been output. Otherwise, an application receives a bookmark event when the voice implementation has synthesized speech up to this bookmark.

<BOOKMARK

MARK = int

syntax: />

content: empty

order: many (default)

parents: SAPI

children: (*none*)

attributes: MARK

model: closed

```
<ElementType name="BOOKMARK" content="empty" model="closed">
  <description>Inserts a bookmark into the input stream
  using the bookmark element. If an application specifies
  interest in bookmark events, it will receive an event when
  synthesis has passed this element in an input stream. If the
  audio output destination supports handling of events, then an
  application will receive this event once the synthesized speech
  up to this bookmark has been output. Otherwise, an application
  receives a bookmark event when the voice implementation has
  synthesized speech up to this bookmark. </description>
  <attribute type="MARK"/>
source: </ElementType>
```

#### <CONTEXT>

The context can specify the type of normalization rules which should be applied to the scoped text. SAPI does not guarantee any predefined contexts.

<CONTEXT

ID = string

>

mixed content

syntax: </CONTEXT>

content: mixed

order: many (default)

parents: SAPI

children: (*none*)

attributes: ID



model: closed

```
<ElementType name="CONTEXT" content="mixed" model="closed">
  <description>The context can specify the type of
normalization rules which should be applied to the scoped text.
SAPI does not guarantee any predefined contexts. </description>
  <attribute type="ID"/>
source: </ElementType>
```

## <EMPH>

Places emphasis on the words contained by this element.

syntax: <EMPH />

content: empty

order: many (default)

parents: SAPI

children: (none)

attributes: (none)

model: closed

```
<ElementType name="EMPH" content="empty" model="closed">
  <description>Places emphasis on the words contained by
this element. </description>
source: </ElementType>
```

## <LANG>

Changes the LANGID of the scoped text. When the LANGID is changed, SAPI will try to detect if the current voice can handle the new language. If voice does not speak the specified language, then an engine must choose another language it speaks as a best attempt. Using the VOICE tag and REQUIRED attribute, this fall back path can be prevented if not desirable.

```
<LANG
  LANGID = int
>
```

mixed content

syntax: </LANG>

content: mixed

order: many (default)

parents: SAPI

children: (none)

attributes: LANGID

model: closed

```
<ElementType name="LANG" content="mixed" model="closed">
  <description>Changes the LANGID of the scoped text.
When the LANGID is changed, SAPI will try to detect if the
current voice can handle the new language. If voice does not
speak the specified language, then an engine must choose
another language it speaks as a best attempt. Using the VOICE
tag and REQUIRED attribute, this fall back path can be
prevented if not desirable.
</description>
  <attribute type="LANGID"/>
source: </ElementType>
```

## <PARTOFSP>

The part of speech of contained word(s). The PARTOFSP tag is used to force a particular pronunciation of a word (for example, the word record as a noun versus the word record as a verb).

<PARTOFSP

PART = enumeration: noun|verb|modifier|function|interjection|unknown

>

mixed content

syntax: </PARTOFSP>

content: mixed

order: many (default)

parents: SAPI

children: (*none*)

attributes: PART

model: closed

<ElementType name="PARTOFSP" content="mixed" model="closed">

<description>The part of speech of contained word(s).  
The PARTOFSP tag is used to force a particular pronunciation of  
a word (for example, the word record as a noun versus the word  
record as a verb). </description>

<attribute type="PART"/>

source: </ElementType>

## <PITCH>

The scoped/global element PITCH modifies the underlying numerical values of a speech block. Relative attribute values, those preceded by a dash (-) or a plus sign (+), increment the underlying numerical value by the specified amount. SAPI compliant engines have the option of supporting only the guaranteed range of values and behaving as -10 for adjustments below -10 and behaving as +10 for values above +10.

<PITCH

[ ABSMIDDLE = int ]

MIDDLE = int

>

mixed content

syntax: </PITCH>

content: mixed

order: many (default)

parents: SAPI

children: (*none*)

attributes: ABSMIDDLE, MIDDLE

model: closed

<ElementType name="PITCH" content="mixed" model="closed">

<description>The scoped/global element PITCH modifies  
the underlying numerical values of a speech block. Relative  
attribute values, those preceded by a dash (-) or a plus sign  
(+), increment the underlying numerical value by the specified  
amount. SAPI compliant engines have the option of supporting  
only the guaranteed range of values and behaving as -10 for

source: adjustments below -10 and behaving as +10 for values above

```
+10.</description>
    <attribute type="MIDDLE"/>
    <attribute type="ABSMIDDLE"/>
</ElementType>
```

## <PRON>

Pronounces the contained text (possibly empty) according to the provided Unicode string.

```
<PRON
  SYM = char
>
```

mixed content

syntax: </PRON>

content: mixed

order: many (default)

parents: SAPI

children: (none)

attributes: SYM

model: open

```
<ElementType name="PRON" content="mixed" model="open">
  <description>Pronounces the contained text (possibly
empty) according to the provided Unicode string.
  </description>
  <attribute type="SYM"/>
source: </ElementType>
```

## <RATE>

Set the relative speed adjustment at which words are synthesized.

```
<RATE
  [ ABSSPEED = int ]
  [ SPEED = int ]
>
```

mixed content

syntax: </RATE>

content: mixed

order: many (default)

parents: SAPI

children: (none)

attributes: ABSSPEED, SPEED

model: closed

```
<ElementType name="RATE" content="mixed" model="closed">
  <description>Set the relative speed adjustment at which
words are synthesized.</description>
  <attribute type="SPEED"/>
  <attribute type="ABSSPEED"/>
source: </ElementType>
```

## <SAPI>

At the beginning of the SAPI tag, the state of the voice is the same state as the insertion point of the SAPI tag. At the close of the SAPI tag, the voice returns to the same state as

that of the insertion point. SAPI tags may be nested. When a nested SAPI tag is closed, the voice state returns to what it was at the insertion point of the nested tag.

```
<SAPI >
  |
  | <BOOKMARK>
  | <SILENCE>
  | <EMPH>
  | <SPELL>
  | <PARTOFSP>
  | <PRON>
  | <LANG>
  | <VOICE>
  | <RATE>
  | <VOLUME>
  | <PITCH>
  | <CONTEXT>
```

mixed content

syntax: **</SAPI>**

content: *mixed*

order: *many* (default)

parents: No parents found. This is probably the document element.

children: [BOOKMARK](#), [CONTEXT](#), [EMPH](#), [LANG](#), [PARTOFSP](#), [PITCH](#), [PRON](#), [RATE](#),  
[SILENCE](#), [SPELL](#), [VOICE](#), [VOLUME](#)

attributes: *(none)*

model: *open*

```
<ElementType name="SAPI" content="mixed" model="open">
  <description>At the beginning of the SAPI tag, the
state of the voice is the same state as the insertion point of
the SAPI tag. At the close of the SAPI tag, the voice returns
to the same state as that of the insertion point. SAPI tags may
be nested. When a nested SAPI tag is closed, the voice state
returns to what it was at the insertion point of the nested
tag. </description>
  <element type="BOOKMARK"/>
  <element type="SILENCE"/>
  <element type="EMPH">
    <description> Place emphasis on the words
contained by this element. It is up to the engine
implementation to design what emphasis is for the engine.
</description>
  </element>
  <element type="SPELL">
    <description>Spell out words letter by letter
contained by this element. NOTE: The engine should not
normalize the text scoped in the SPELL tag. This includes
numbers, words, etc. Words which contain punctuation, such as
"U.S.A" should spell out the letters as well as the punctuation
scoped within the tag. </description>
  </element>
  <element type="PARTOFSP"/>
```

source: `<element type="PARTOFSP"/>`

```

        <element type="PRON">
            <description>String representing a phoneme for
a language supported by the voice implementing synthesized
speech. </description>
        </element>
        <element type="LANG"/>
        <element type="VOICE"/>
        <element type="RATE"/>
        <element type="VOLUME">
            <description>0 to 100 (no overflow
allowed)</description>
        </element>
        <element type="PITCH">
            <description>Set the relative pitch adjustment
of synthesized speech.</description>
        </element>
        <element type="CONTEXT"/>
    </ElementType>

```

## <SILENCE>

**Produces silence for a specified number of milliseconds to the output audio stream.**

**<SILENCE**

**MSEC** = *int*

syntax: **/>**

content: *empty*

order: *many* (default)

parents: [SAPI](#)

children: (*none*)

attributes: **MSEC**

model: *closed*

```

<ElementType name="SILENCE" content="empty" model="closed">
    <description>Produces silence for a specified number of
milliseconds to the output audio stream. </description>
    <attribute type="MSEC"/>

```

source: </ElementType>

## <SPELL>

**Spells out words letter by letter contained by this element. Note: The engine should not normalize the text scoped in the SPELL tag. This includes numbers, words, etc. Words that contain punctuation, such as "U.S.A." should spell out the letters as well as the punctuation scoped within the tag.**

syntax: **<SPELL />**

content: *empty*

order: *many* (default)

parents: [SAPI](#)

children: (*none*)

attributes: (*none*)

model: *closed*

```

<ElementType name="SPELL" content="empty" model="closed">
    <description>Spells out words letter by letter

```

source: contained by this element.

Note: The engine should not normalize the text scoped in the SPELL tag. This includes numbers, words, etc. Words that contain punctuation, such as "U.S.A." should spell out the letters as well as the punctuation scoped within the tag.

</description>

</ElementType>

## <VOICE>

Sets which voice implementation is used for synthesis of associated input stream text. The best voice implementation given the required and optional attributes will be selected by SAPI.

<VOICE

[ OPTIONAL = *string* ]

[ REQUIRED = *string* ]

>

mixed content

syntax: </VOICE>

content: *mixed*

order: *many* (default)

parents: [SAPI](#)

children: (*none*)

attributes: OPTIONAL, REQUIRED

model: *closed*

```
<ElementType name="VOICE" content="mixed" model="closed">
  <description>Sets which voice implementation is used
  for synthesis of associated input stream text. The best voice
  implementation given the required and optional attributes will
  be selected by SAPI. </description>
```

```
  <attribute type="REQUIRED"/>
```

```
  <attribute type="OPTIONAL"/>
```

source: </ElementType>

## <VOLUME>

The scoped/global elements VOLUME modify the underlying numerical values of a speech block. The underlying value can never be below zero or exceed 100. All negative value entries will result in zero and all values above 100 will result in 100. VOLUME may also receive an absolute value (no '-' or '+' character) of an integer between zero and 100.

<VOLUME

LEVEL = *int*

>

mixed content

syntax: </VOLUME>

content: *mixed*

order: *many* (default)

parents: [SAPI](#)

children: (*none*)

attributes: LEVEL

model: *closed*

```
<ElementType name="VOLUME" content="mixed" model="closed">
```

source: <description>The scoped/global elements VOLUME modify

the underlying numerical values of a speech block. The underlying value can never be below zero or exceed 100. All negative value entries will result in zero and all values above 100 will result in 100. VOLUME may also receive an absolute value (no '-' or '+' character) of an integer between zero and 100. </description>  
 <attribute type="LEVEL"/>  
 </ElementType>

## SAPI Attributes

### <... ABSMIDDLE="">

The value can range from -10 to +10. A value of 0 sets a voice to speak at its default pitch. A value of -10 sets a voice to speak at three-fourths (or  $\frac{3}{4}$ ) of its default pitch. A value of +10 sets a voice to speak at four-thirds (or  $\frac{4}{3}$ ) of its default pitch. Each increment between -10 and +10 is logarithmically distributed such that incrementing/decrementing by 1 is multiplying/dividing the pitch by the 24th root of 2 (about 1.03). Values more extreme than -10 and 10 will be passed to an engine but SAPI 5compliant engines may not support such extremes and instead may clip the pitch to the maximum or minimum pitch it supports. Values of -24 and +24 must lower and raise pitch by 1 octave respectively. All incrementing/decrementing by 1 must multiply/divide the pitch by the 24th root of 2. When scoped, this attribute is absolute.

syntax: [ ABSMIDDLE = *int* ]

required: no (default)

datatype: *int*

elements: PITCH

```
<AttributeType name="ABSMIDDLE" dt:type="int">
  <description> The value can range from -10 to +10. A
  value of 0 sets a voice to speak at its default pitch. A value
  of -10 sets a voice to speak at three-fourths (or  $\frac{3}{4}$ ) of its
  default pitch. A value of +10 sets a voice to speak at four-
  thirds (or  $\frac{4}{3}$ ) of its default pitch. Each increment between -
  10 and +10 is logarithmically distributed such that
  incrementing/decrementing by 1 is multiplying/dividing the
  pitch by the 24th root of 2 (about 1.03). Values more extreme
  than -10 and 10 will be passed to an engine but SAPI 5compliant
  engines may not support such extremes and instead may clip the
  pitch to the maximum or minimum pitch it supports. Values of -
  24 and +24 must lower and raise pitch by 1 octave respectively.
  All incrementing/decrementing by 1 must multiply/divide the
  pitch by the 24th root of 2. When scoped, this attribute is
  absolute.</description>
```

source: </AttributeType>

### <... ABSSPEED="">

The value can range from -10 to +10. A value of 0 sets a voice to speak at its default rate. A value of -10 sets a voice to speak at one-third (or  $\frac{1}{3}$ ) of its default rate. A value of +10 sets a voice to speak at 3 times its default rate. Each increment between -10 and +10 is logarithmically distributed such that incrementing/decrementing by 1 is multiplying/dividing the rate by the 10th root of 3 (about 1.12). Values more extreme than -10 and +10 will be passed to an engine, but SAPI 5compliant engines may not support such extremes and instead may clip the rate to the maximum or minimum rate it supports. When scoped, this attribute is absolute.

syntax: [ ABSSPEED = *int* ]

required: no (default)

datatype: *int*

elements: **RATE**

```
<AttributeType name="ABSSPEED" dt:type="int">
    <description>The value can range from -10 to +10. A
    value of 0 sets a voice to speak at its default rate. A value
    of -10 sets a voice to speak at one-third (or 1/3) of its
    default rate. A value of +10 sets a voice to speak at 3 times
    its default rate. Each increment between -10 and +10 is
    logarithmically distributed such that incrementing/decrementing
    by 1 is multiplying/dividing the rate by the 10th root of 3
    (about 1.12). Values more extreme than -10 and +10 will be
    passed to an engine, but SAPI 5compliant engines may not
    support such extremes and instead may clip the rate to the
    maximum or minimum rate it supports. When scoped, this
    attribute is absolute.</description>
```

source: </AttributeType>

**<... ID="">**

**This specifies the type of context. Refer to the SAPI documentation for the vairous context ids.**

syntax: ID = *string*

required: yes

datatype: *string*

elements: **CONTEXT**

```
<AttributeType name="ID" dt:type="string" required="yes">
    <description>This specifies the type of context. Refer
    to the SAPI documentation for the vairous context
    ids.</description>
```

source: </AttributeType>

**<... LANGID="">**

**Language identifier. The language identifier is specified as a hexadecimal value. For example, the LANGID for English (US) expressed in the hexadecimal form is 409.**

syntax: LANGID = *int*

required: yes

datatype: *int*

elements: **LANG**

```
<AttributeType name="LANGID" dt:type="int" required="yes">
    <description>Language identifier. The language
    identifier is specified as a hexadecimal value. For example,
    the LANGID for English (US) expressed in the hexadecimal form
    is 409. </description>
```

source: </AttributeType>

**<... LEVEL="">**

**This specifies the volume as percent of the maximum volume of the current voice. Each voice implementation has it's own maximum volume. This value must between 0 and 100 inclusive. Values above 100 or below 0 are clipped to 100 and 0 respectively.**

syntax: LEVEL = *int*

required: yes



datatype: *int*

elements: **VOLUME**

```
<AttributeType name="LEVEL" dt:type="int" required="yes">
  <description> This specifies the volume as percent of
the maximum volume of the current voice. Each voice
implementation has it's own maximum volume. This value must
between 0 and 100 inclusive. Values above 100 or below 0 are
clipped to 100 and 0 respectively.</description>
```

source: </AttributeType>

### <... MARK="">

**The value of a bookmark may be any string or integer.**

syntax: MARK = *int*

required: yes

datatype: *int*

elements: **BOOKMARK**

```
<AttributeType name="MARK" dt:type="int" required="yes">
  <description>The value of a bookmark may be any string
or integer. </description>
```

source: </AttributeType>

### <... MIDDLE="">

**The value can range from -10 to +10. A value of 0 sets a voice to speak at its default pitch. A value of -10 sets a voice to speak at three-fourths (or  $\frac{3}{4}$ ) of its default pitch. A value of +10 sets a voice to speak at four-thirds (or  $\frac{4}{3}$ ) of its default pitch. Each increment between -10 and +10 is logarithmically distributed such that incrementing/decrementing by 1 is multiplying/dividing the pitch by the 24th root of 2 (about 1.03). Values more extreme than -10 and 10 will be passed to an engine but SAPI 5compliant engines may not support such extremes and instead may clip the pitch to the maximum or minimum pitch it supports. Values of -24 and +24 must lower and raise pitch by 1 octave respectively. All incrementing/decrementing by 1 must multiply/divide the pitch by the 24th root of 2. When scoped, this attribute is relative.**

syntax: MIDDLE = *int*

required: yes

datatype: *int*

elements: **PITCH**

```
<AttributeType name="MIDDLE" dt:type="int" required="yes">
  <description>The value can range from -10 to +10. A
value of 0 sets a voice to speak at its default pitch. A value
of -10 sets a voice to speak at three-fourths (or  $\frac{3}{4}$ ) of its
default pitch. A value of +10 sets a voice to speak at four-
thirds (or  $\frac{4}{3}$ ) of its default pitch. Each increment between -
10 and +10 is logarithmically distributed such that
incrementing/decrementing by 1 is multiplying/dividing the
pitch by the 24th root of 2 (about 1.03). Values more extreme
than -10 and 10 will be passed to an engine but SAPI 5compliant
engines may not support such extremes and instead may clip the
pitch to the maximum or minimum pitch it supports. Values of -
24 and +24 must lower and raise pitch by 1 octave respectively.
All incrementing/decrementing by 1 must multiply/divide the
pitch by the 24th root of 2. When scoped, this attribute is
```

source: relative.</description>

</AttributeType>

### <... MSEC="">

**Number of milliseconds, from zero to 65535, of silence. Value entries that exceed this range should be limited to 65535. Value entries that are below this range (negative values) should be set to zero.**

syntax: MSEC = *int*

required: yes

datatype: *int*

elements: SILENCE

```
<AttributeType name="MSEC" dt:type="int" required="yes">
  <description>Number of milliseconds, from zero to
  65535, of silence. Value entries that exceed this range should
  be limited to 65535. Value entries that are below this range
  (negative values) should be set to zero. </description>
```

source: </AttributeType>

### <... OPTIONAL="">

**The XML parser selects the first voice registered containing all of the specified attributes. A string that contains semicolon-delimited sub-strings is used to specify the attributes. The speak call will fail if the parser cannot find the required tags.**

syntax: [ OPTIONAL = *string* ]

required: no (default)

datatype: *string*

elements: VOICE

```
<AttributeType name="OPTIONAL" dt:type="string">
  <description>The XML parser selects the first voice
  registered containing all of the specified attributes. A string
  that contains semicolon-delimited sub-strings is used to
  specify the attributes. The speak call will fail if the parser
  cannot find the required tags.
</description>
```

source: </AttributeType>

### <... PART="">

**String name of part of speech. Valid SAPI parts of speech are noun, verb, modifier, function, interjection and unknown.**

syntax: PART = *enumeration*: noun|verb|modifier|function|interjection|unknown

required: yes

datatype: *enumeration*

values: noun|verb|modifier|function|interjection|unknown

elements: PARTOFSPEECH

```
<AttributeType name="PART" dt:type="enumeration"
dt:values="noun|verb|modifier|function|interjection|unknown"
required="yes">
  <description> String name of part of speech. Valid SAPI
  parts of speech are noun, verb, modifier, function, interjection
  and unknown. </description>
```

source: </AttributeType>

### <... REQUIRED="">

The XML parser selects the first voice registered containing all of the specified attributes. A string that contains semicolon-delimited sub-strings is used to specify the attributes. The speak call will fail if the parser cannot find the required tags.

syntax: [ REQUIRED = *string* ]

required: no (default)

datatype: *string*

elements: VOICE

```
<AttributeType name="REQUIRED" dt:type="string">
    <description>The XML parser selects the first voice
    registered containing all of the specified attributes. A string
    that contains semicolon-delimited sub-strings is used to
    specify the attributes. The speak call will fail if the parser
    cannot find the required tags.
</description>
```

source: </AttributeType>

### <... SPEED="">

The value can range from -10 to +10. A value of 0 sets a voice to speak at its default rate. A value of -10 sets a voice to speak at one-third (or 1/3) of its default rate. A value of +10 sets a voice to speak at 3 times its default rate. Each increment between -10 and +10 is logarithmically distributed such that incrementing/decrementing by 1 is multiplying/dividing the rate by the 10th root of 3 (about 1.12). Values more extreme than -10 and +10 will be passed to an engine, but SAPI 5 compliant engines may not support such extremes and instead may clip the rate to the maximum or minimum rate it supports. When scoped, this attribute is relative.

syntax: [ SPEED = *int* ]

required: no (default)

datatype: *int*

elements: RATE

```
<AttributeType name="SPEED" dt:type="int">
    <description>The value can range from -10 to +10. A
    value of 0 sets a voice to speak at its default rate. A value
    of -10 sets a voice to speak at one-third (or 1/3) of its
    default rate. A value of +10 sets a voice to speak at 3 times
    its default rate. Each increment between -10 and +10 is
    logarithmically distributed such that incrementing/decrementing
    by 1 is multiplying/dividing the rate by the 10th root of 3
    (about 1.12). Values more extreme than -10 and +10 will be
    passed to an engine, but SAPI 5 compliant engines may not
    support such extremes and instead may clip the rate to the
    maximum or minimum rate it supports. When scoped, this
    attribute is relative.</description>
```

source: </AttributeType>

### <... SYM="">

String representing a phoneme for a language supported by the voice implementing synthesizing speech. Refer to SAPI Phoneme Spec.

syntax: SYM = *char*

required: yes

datatype: *char*

elements: PRON

```
<AttributeType name="SYM" dt:type="char" required="yes">  
    <description>String representing a phoneme for a  
    language supported by the voice implementing synthesizing  
    speech. Refer to SAPI Phoneme Spec.</description>
```

**source:** </AttributeType>