VoiceText™ Server 2007

VTAP (VoiceText Access Protocol) .Net API Programmer's Guide

Software Version 3.7

NeoSpeech Co., Ltd.

Revision History

Last revision date : Jan. 1, 2007

Date	Page No.	Revision / Added items
2007.01.01		New Manual added.

VoiceText™ VTAP .Net API

The software described in this manual is furnished under a valid license agreement or nondisclosure agreement with NeoSpeech, Inc. The software may be used only in accordance with the terms of the agreement

Copyright Notice

DISCLAIMER. THIS MANUAL IS DISTRIBUTED AS-IS. NEOSPEECH, INC. SHALL NOT BE LIABLE FOR TECHNICAL OR EDITORIAL ERRORS OR OMISSIONS CONTAINED HEREIN; NOR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL. THE INFORMATION IN THIS MANUAL IS SUBJECT TO CHANGE WITHOUT ANY PRIOR NOTICE..

Trademark

2007-01
Printed in Fremont, California.

Table of Contents

Chapter 1 : Overview

VoiceText™ text-to-speech product by NeoSpeech, Inc. is the leading text-to-speech solution, which synthesizes an arbitrary input text to produce synthetic speech that is natural, human-like and clear by accurately analyzing and processing linguistic information such as phonetic content, sentence structure, prosody and pauses.

As the proud leader of high quality, high performance voice engines in Korean, English, Japanese and Chinese that are able to synthesize variety of text and symbols in best quality voice, VoiceText™ is widely used in industries including telecommunications, finance service and public welfare. With the precise pronunciation, natural human-like synthesized voice, variety of supported platforms, high quality service, stability, fast synthesis speed and variety of voice types and language, VoiceWare proudly guarantees that VoiceText™ is the best speech synthesis product in the world.

This document contains the documentation and examples of VTAP (VoiceText™ Access Protocol) API in .Net programming language. This API is required for developing applications using VoiceText™ Server.

Chapter 2 : Installation

VTAP C API consists of libraries, which can be copied to the system for use.

VTAP .Net API Libraries

VTAP .Net API Library for .Net Framework 2.0 : vtap_api_dotnet20.dll

---

Note

The explanation inside this manual is based on C#. It could vary based on .Net language which is being used.

Chapter 3 : Getting Started

This chapter gives an overview of VoiceText™ Access Protocol and VTAP .Net API. This API is required for development using VoiceText™ Server.

VTAP(VoiceText Access Protocol)

VTAP is a protocol used in communication between the VoiceText™ Server and a client to synthesize (convert) text into voice. In addition to the basic methods for text synthesis, VTAP also includes the methods to monitor the system status and methods to configure time-out settings used to communication with the VoiceText™ Server.

VTAP .Net API

VTAP .Net API is a set of API's using VoiceText™ Access Protocol. It is used by a client to request service to VoiceText™ Server. VTAP API's can be classified as TTS API's and Etc API's by usage. They can also be classified as File API and Buffering API by synthesis method.

Chapter 4 : Structures

This chapter describes the structure of used by the VTAP .Net API.

TTSMARK – SSML Mark Structure

TTSMARK Frame Buffer is a structure that holds information of <mark> tag

```
#define     MAX_MARK_NAME     (512)

public class TTSMARK{
    int             nOffsetInStream ;
    int             nOffsetInBuffer ;
    int             nPosInText ;
    string          sMarkName;
} ;
```

Parameters

nOffsetInStream

Location within the entire voice buffer. In sample unit.

nOffsetInBuffer

Location within the currently returned voice buffer. In sample unit.

nPosInText

It is not used.

sMarkName

It shows the name attribute value of mark tag.

Description

Mark tag name value can be saved up to 512bytes including null characters and the rest will be lost.

Mark tag information is saved in the order of mark tag of requested text.

It is used along with TTSRequestBufferSSMLEx() API.
Please refer to TTSRequestBufferSSMLEx() description for more detailed usage of SSML formatted text synthesis.

Chapter 5 : VTAP .Net API References

This chapter lists VTAP API in .Net programming language with detailed definitions.

VTAP API's can be classified as below according to usage and method.

Classification by Usage

TTS API

- TTSRequestFile ()
- TTSRequestFileEx ()
- TTSRequestBuffer ()
- TTSRequestBufferEx ()
- TTSRequestBufferSSMLEx ()

Etc API

- TTSRequestStatus ()

Classification by Synthesis method

File API (synthesis output to file)

- TTSRequestFile ()
- TTSRequestFileEx ()

Buffering API (synthesis output to buffer)

- TTSRequestBuffer ()
- TTSRequestBufferEx ()
- TTSRequestBufferSSMLEx ()

TTSRequestFile , TTSRequestFileEx

TTSRequestFile synthesizes the input text and outputs the resulting voice to a file

int **TTSRequestFile** (

string	*szServer* ,
int	*nPort* ,
string	***pText*** ,
string	*szSaveDir* ,
string	*szSaveFile* ,
int	*nSpeakerID* ,
int	*nVoiceFormat*

)

int **TTSRequestFile** (

string	*szServer* ,
int	*nPort* ,
byte[]	***pText*** ,
int	***nTextLen*** ,
string	*szSaveDir* ,
string	*szSaveFile* ,
int	*nSpeakerID* ,
int	*nVoiceFormat*

)

int **TTSRequestFileEx** (

string	*szServer* ,
int	*nPort* ,
string	***pText*** ,
string	*szSaveDir* ,
string	*szSaveFile*
int	*nSpeakerID* ,
int	*nVoiceFormat*
int	*nTextFormat ,*
int	*nVolume ,*
int	*nSpeed ,*
int	*nPitch ,*
int	*nDictNum*

)

int **TTSRequestFileEx** (

string	*szServer* ,
int	*nPort* ,
byte[]	***pText*** ,
int	***nTextLen*** ,
string	*szSaveDir* ,
string	*szSaveFile*
int	*nSpeakerID* ,
int	*nVoiceFormat*
int	*nTextFormat ,*
int	*nVolume ,*
int	*nSpeed ,*
int	*nPitch ,*
int	*nDictNum*

)

Parameters

szServer

IP address of VoiceText™ Server

nPort

Port number used for sending synthesis requests to and response from VoiceText™ Server, default is 7000

pText

Text string to be synthesized, string or byte[] types could be used

nTextLen

Length of *pText*

szSaveDir

Directory to which the synthesized voice file is to be saved.

The max length is 128bytes including null characters.

szSaveFile

File name under which the synthesized voice file is to be saved.

The max length is 128bytes including null characters.

nSpeakerID

ID of the speaker's voice (e.g. voice engine)

The available voice engines are listed below. *nSpeakerID* must be set to the ID of a voice engine that is installed on the system.

TTS_JUNWOO_DB	(3)	Junwoo (Korean, Male)
TTS_SUJIN_DB	(8)	Sujin (Korean, Female)
TTS_YUMI_DB	(10)	Yumi (Korean, Female)
TTS_GYURI_DB	(11)	Gyuri (Korean, Female)
TTS_DAYOUNG_DB	(12)	Dayoung (Korean, Female)
TTS_CHORONG_DB	(13)	Chorong (Korean, Female)
TTS_KATE_DB	(100)	Kate (English, Female)
TTS_PAUL_DB	(101)	Paul (English, Male)
TTS_LILY_DB	(200)	Lily (Chinese, Female)
TTS_WANG_DB	(201)	Wang (Chinese, Male)
TTS_MIYU_DB	(300)	Miyu (Japanese, Female)
TTS_SHOW_DB	(301)	Show (Japanese, Male)
TTS_MISAKI_DB	(302)	Misaki (Japanese, Female)

nVoiceFormat

Output file format of synthesis.

The available voice formats are as follows.

(For the list of supported file formats, refer to Appendix B.)

FORMAT_DEFAULT	(0)	Default
FORMAT_WAV	(1)	16bit linear PCM Wave
FORMAT_PCM	(2)	16bit linear PCM
FORMAT_MULAW	(3)	8bit Mu-law PCM
FORMAT_ALAW	(4)	8bit A-law PCM
FORMAT_ADPCM	(5)	4bit Dialogic ADPCM
FORMAT_ASF	(6)	Windows ASF
FORMAT_OGG	(10)	Ogg Vorbis
FORMAT_8BITWAV	(11)	8bit unsigned linear PCM Wave
FORMAT_AWAV	(12)	8bit A-law PCM Wave
FORMAT_MUWAV	(13)	8bit Mu-law PCM Wave

nTextFormat

Format of the text to be synthesized. The following format can be used as nTextFormat. Other values will be recognized as TEXT_NORMAL(0).

TEXT_NORMAL	(0)	Normal Text
TEXT_SSML	(1)	SSML Text
TEXT_HTML	(2)	HTML Text
TEXT_EMAIL	(3)	E-Mail Text

nVolume

It sets the volume to be used in synthesis. The effective range is from 0 to 500 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(500), below min value will be recognized as min value(0).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nSpeed

It sets the speed of the synthesized voice to be used in synthesis. The effective range is from 50 to 400 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(400), below min value will be recognized as min value(50).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nPitch

It sets the pitch to be used in synthesis. The effective range is from 50 to 200 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(200), below min value will be recognized as min value(50).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nDictNum

It sets the index numbers of the user-defined dictionary to be used in synthesis. The effective range is from 0 to 100 and the default value is 0. Other values out of the effective range are all recognized default value.
nTextFormat is TEXT_SSML(1) this value will be ignored.

Description

When *szSaveDir* parameter value is set to “test” and the VoiceText™ Server’s default output directory, NfsDir, is set to “c:\vw\vtsvc\result”, “c:\vw\vtsvc\result\test” directory will be created and the output file resulting from synthesis requests made via this API will be saved in this directory. The value of *szSaveDir* must be a relative path, relative to NfsDir. Please make sure that the value of *szSaveDir* is not set to an absolute path such as “c:\vw\vtsvc\result\test”. This parameter can be set to null. When it is set to null, the default output directory of VoiceText™ Server, NfsDir, will be used.

In addition, if the value of *szSaveFile* is set to “sample”, the value of *szSaveDir* to NULL, the default output directory of the VoiceText™ Server to “c:\vw\vtsvc\result” and *nVoiceFormat* to wave; then the output resulting from calling this API will be saved as “c:\vw\vtsvc\result \sample.wav” file. Note no file extension should be attached to *szSaveFile*, because it is automatically attached by the server according to the synthesize format.

Return Values

For successfully synthesized requests, TTS_RESULT_SUCCESS(1) is returned. When an error occurs, a negative value is returned. For the error code values and descriptions, see the section on Errors.

Notes

The synthesis output file is saved to the hard disk of VoiceText™ server. In order to allow a remote client to use the synthesized voice file, it is necessary to share the output directory over the network using NFS (Network File System) or File Share function. As the synthesis output files are not managed separately by VoiceText™ server, these files need to be deleted or backed-up to a certain directory after use. Please note that no synthesis can occur if disk space is not enough on the server. Make sure to manage the output files properly.

Directory name(*szSaveDir*) and file name(*szSaveFile*) to save the synthesized voice files must be used with the authorized character strings in the system in which synthesis server is installed.

Errors

The descriptions of returned error codes are as follows : Please refer to Return Values in Appendix A for error code value.

[TTS_HOSTNAME_ERROR]

This error code is returned when an invalid IP address has been provided.

An IP address is invalid if the format is incorrect.

[TTS_SOCKET_ERROR]

This error code is returned when the socket() used by this API returns an error.

For possible causes of socket() error, please refer to the relevant system manual.

[TTS_CONNECT_ERROR]

This error code is returned if the connect() function used by this API returns an error.

When this error is returned, make sure that VoiceText™ Server is executed and confirm that the network environment is working normally.

For possible causes of connect() error, please refer to the relevant system manual.

[TTS_READWRITE_ERROR]

This error code is returned when there has been an error in sending/receiving data.

This error could occur if the connection to the server has timed out due to slow response time of the VoiceText™ Server.

For possible causes of read(), write() error, please refer to the relevant system manual.

[TTS_TEXT_ERROR]

This error code is returned when an invalid text length was provided.

Text length must be a positive integer.

It occurs when the designated texts are longer than the texts to be synthesized.

[TTS_VOICEFORMAT_ERROR]

This error code is returned when an invalid value has been assigned to *nVoiceFormat*.

Check whether a valid voice format was specified as *nVoiceFormat*.

Please refer to the appendix B for the supported voice formats.

[TTS_PARAM_ERROR]

This error code is returned when invalid values have been assigned to *szSaveFile* or *szSaveDir* parameters.

[TTS_DISK_ERROR]

This error code is returned when there are problems on server hard disk.

Check if there is enough disk space available of the server.

[TTS_SPEAKER_ERROR]

This error code is returned when an invalid value has been set to *nSpeakerID.*

Make sure that the voice engine corresponding to *nSpeakerID* is installed on the server.

[TTS_MAX_ERROR]

This error code is returned if the number of requested channels exceeds the maximum number of channels allowed by VoiceText™ Server.

Balance the traffic load between multiple servers after installing VoiceText™ Server on other machines.

[TTS_RESULT_ERROR]

This error code is returned if VoiceText™ Server has encountered a problem during synthesis.

For details on this error, see the service log file on the server.

Example

```
// In case the texts to be synthesized are string type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int nReturn = 0;
			string Text = "Hello~ This is VoiceText Demo!!";

			// tts file ex request test
			nReturn = a.TTSRequestFileEx("127.0.0.1", 7000, Text, "c#.net",
"fileex_test", libttsapi.TTS_KATE_DB, libttsapi.FORMAT_WAV,
libttsapi.TEXT_NORMAL, 100, 100, 100, 0);
			if (nReturn == libttsapi.TTS_RESULT_SUCCESS)
			{
				Console.WriteLine("RequestFileEx Success !!!");
			}
			else
			{
				Console.WriteLine("RequestFileEx Failed (" + nReturn.ToString() +
")!!!");
			}
		}
	}
}
```

```
// In case the texts to be synthesized are byte[] type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int nReturn = 0;
			string Text = "Hello~ This is VoiceText Demo!!";
			byte[] bText = null;
			int bText_size = 0;

			bText = Encoding.Default.GetBytes(Text.ToCharArray());
			bText_size = bText.Length;

			// tts file ex request test
			nReturn = a.TTSRequestFileEx("127.0.0.1", 7000, bText, bText_size,
"c#.net", "fileex_test2", libttsapi.TTS_KATE_DB, libttsapi.FORMAT_WAV,
libttsapi.TEXT_NORMAL, 100, 100, 100, 0);
			if (nReturn == libttsapi.TTS_RESULT_SUCCESS)
			{
				Console.WriteLine("RequestFileEx Success !!!");
			}
			else
			{
				Console.WriteLine("RequestFileEx Failed (" + nReturn.ToString() +
")!!!");
			}
		}
	}
}
```

TTSRequestBuffer , TTSRequestBufferEx

TTSRequestBuffer synthesizes the input text and outputs the resulting voice to buffer.

byte[] **TTSRequestBuffer** (

string	*szServer* ,
int	*nPort* ,
string	***pText*** ,
out int	*nVoiceLen* ,
int	*nSpeakerID* ,
int	*nVoiceFormat ,*
int	*bFirst ,*
int	*bAll ,*
out int	*nReturn*

)

byte[] **TTSRequestBuffer** (

string	*szServer* ,
int	*nPort* ,
byte[]	***pText*** ,
int	***nTextLen*** ,
out int	*nVoiceLen* ,
int	*nSpeakerID* ,
int	*nVoiceFormat ,*
int	*bFirst ,*
int	*bAll ,*
out int	*nReturn*

)

byte[] **TTSRequestBufferEx** (

string	*szServer* ,
int	*nPort* ,
string	***pText*** ,
out int	*nVoiceLen* ,
int	*nSpeakerID* ,
int	*nVoiceFormat ,*
int	*nTextFormat ,*
int	*nVolume ,*
int	*nSpeed ,*
int	*nPitch ,*
int	*nDictNum ,*
int	*bFirst ,*
int	*bAll ,*
out int	*nReturn*

)

byte[] **TTSRequestBufferEx** (

string	*szServer* ,
int	*nPort* ,
byte[]	***pText*** ,
int	***nTextLen*** ,
out int	*nVoiceLen* ,
int	*nSpeakerID* ,
int	*nVoiceFormat ,*
int	*nTextFormat ,*
int	*nVolume ,*
int	*nSpeed ,*
int	*nPitch ,*
int	*nDictNum ,*
int	*bFirst ,*
int	*bAll ,*
out int	*nReturn*

)

Parameters

szServer

IP address of VoiceText™ Server

nPort

Port number used for sending synthesis requests to and response from VoiceText™ Server, default is 7000

pText

Text string to be synthesized, string or byte[] types could be used.

nTextLen

Length of *pText*

nVoiceLen

A pointer to an integer variable that holds the length of the buffer frame returned from this function. Note that the buffer length varies by synthesis output format.

nSpeakerID

ID of the speaker's voice (e.g. voice engine)

The available voice engines are listed below. *nSpeakerID* must be set to the ID of a voice engine that is installed on the system.

TTS_JUNWOO_DB	(3)	Junwoo (Korean, Male)
TTS_SUJIN_DB	(8)	Sujin (Korean, Female)
TTS_YUMI_DB	(10)	Yumi (Korean, Female)
TTS_GYURI_DB	(11)	Gyuri (Korean, Female)
TTS_DAYOUNG_DB	(12)	Dayoung (Korean, Female)
TTS_CHORONG_DB	(13)	Chorong (Korean, Female)
TTS_KATE_DB	(100)	Kate (English, Female)
TTS_PAUL_DB	(101)	Paul (English, Male)
TTS_LILY_DB	(200)	Lily (Chinese, Female)
TTS_WANG_DB	(201)	Wang (Chinese, Male)
TTS_MIYU_DB	(300)	Miyu (Japanese, Female)
TTS_SHOW_DB	(301)	Show (Japanese, Male)
TTS_MISAKI_DB	(302)	Misaki (Japanese, Female)

nVoiceFormat

Output file format of synthesis.

The available voice formats are as follows.

(For the list of supported file formats, refer to Appendix B.)

FORMAT_DEFAULT	(0)	Default
FORMAT_WAV	(1)	16bit linear PCM Wave
FORMAT_PCM	(2)	16bit linear PCM

FORMAT_MULAW	(3)	8bit Mu-law PCM
FORMAT_ALAW	(4)	8bit A-law PCM
FORMAT_ADPCM	(5)	4bit Dialogic ADPCM
FORMAT_ASF	(6)	Windows ASF
FORMAT_OGG	(10)	Ogg Vorbis
FORMAT_8BITWAV	(11)	8bit unsigned linear PCM Wave
FORMAT_AWAV	(12)	8bit A-law PCM Wave
FORMAT_MUWAV	(13)	8bit Mu-law PCM Wave

nTextFormat

Format of the text to be synthesized. The following format can be used as nTextFormat. Other values will be recognized as TEXT_NORMAL(0)

TEXT_NORMAL	(0)	Normal Text
TEXT_SSML	(1)	SSML Text
TEXT_HTML	(2)	HTML Text
TEXT_EMAIL	(3)	E-Mail Text

nVolume

It sets the volume to be used in synthesis. The effective range is from 0 to 500 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(500), below min value will be recognized as min value(0).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nSpeed

It sets the speed of the synthesized voice to be used in synthesis. The effective range is from 50 to 400 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(400), below min value will be recognized as min value(50).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nPitch

It sets the pitch to be used in synthesis. The effective range is from 50 to 200 and the default value is 100. You can manually regulate the value around 100. Over the max value will be recognized as max value(200), below min value will be recognized as min value(50).

nTextFormat is TEXT_SSML(1) this value will be ignored.

nDictNum

It sets the index numbers of the user-defined dictionary to be used in synthesis. The effective range is from 0 to 100 and the default value is 0. Other values out of the effective range are all recognized default value.

nTextFormat is TEXT_SSML(1) this value will be ignored.

bFirst

Whether or not the first frame of the voice output is being requested

TRUE	(1)	Request the first frame
FALSE	(0)	Request after the first frame

bAll

Whether the resulting voice buffer is to be sent in one frame (vs. multiple frames)

TRUE	(1)	To receive synthesis results in one frame buffer
FALSE	(0)	To receive synthesis results in multiple frame buffers

nReturn

Return value denoting the result status

Description

When using the multi-frame buffer method to receive the synthesized voice, read data from the frame buffer for *nVoiceLen* length. The size of the frame buffer being sent is specified by *nVoiceLen* parameter.

To receive the synthesized voice in one frame buffer, set *bAll* and *bFirst* to 1. To receive the synthesized voice in multiple frame buffers, set *bAll* to 0 and set *bFirst* to 1 in the first function call and 0 in the subsequent calls.

In case of using separated buffer synthesis, there is the limit in supported voice format.
(For more information please refer to appendix B).

When the synthesis finishes successfully, *nReturn* will point to the value of either TTS_RESULT_CONTINUE(0) or TTS_RESULT_SUCCESS(1). If the last frame buffer is being sent, it will be set to TTS_RESULT_SUCCESS(1). Otherwise, it will be set to TTS_RESULT_CONTINUE(0).

When an error occurs, a negative value will be assigned to *nReturn*. For details on error codes, see the section on Errors.

Return Values

After a successful synthesis, a pointer to the synthesized voice buffer frame is returned. In case of error, NULL pointer is returned. The length of buffer frame is assigned to *nVoiceLen* parameter. For details on error codes, refer to Errors.

Notes

In order to play the voice buffer frames without interruption, do not play each frame right after receiving it but play the frames after buffering the frames again. (Ex., double buffering)

When you set *nTextFormat* as TEXT_SSML, you can not obtain the SSML Mark information.
In order to obtain Mark information you must use TTSRequestBufferSSMLEx ()

Errors

The descriptions of returned error codes are as follows : Please refer to Return Values in Appendix A for error code value.

[TTS_HOSTNAME_ERROR]

This error code is returned when an invalid IP address has been provided.

An IP address is invalid if the format is incorrect.

[TTS_SOCKET_ERROR]

This error code is returned when the socket() used by this API returns an error.

For possible causes of socket() error, please refer to the relevant system manual.

[TTS_CONNECT_ERROR]

This error code is returned if the connect() function used by this API returns an error.

When this error is returned, make sure that VoiceText™ Server is executed and confirm that the network environment is working normally.

For possible causes of connect() error, please refer to the relevant system manual.

[TTS_READWRITE_ERROR]

This error code is returned when there has been an error in sending/receiving data.

This error could occur if the connection to the server has timed out due to slow response time of the VoiceText™ Server.

For possible causes of read(), write() error, please refer to the relevant system manual.

[TTS_TEXT_ERROR]

This error code is returned when an invalid text length was provided.

Text length must be a positive integer.

It occurs when the designated texts are longer than the texts to be synthesized.

[TTS_VOICEFORMAT_ERROR]

This error code is returned when an invalid value has been assigned to *nVoiceFormat*.

Check whether a valid voice format was specified as *nVoiceFormat*.

Please refer to the appendix B for the supported voice formats.

[TTS_PARAM_ERROR]

This error code is returned when invalid values have been assigned to *szSaveFile* or

szSaveDir parameters.

[TTS_DISK_ERROR]

This error code is returned when there are problems on server hard disk.

Check if there is enough disk space available of the server.

[TTS_SPEAKER_ERROR]

This error code is returned when an invalid value has been set to *nSpeakerID.*

Make sure that the voice engine corresponding to *nSpeakerID* is installed on the server.

[TTS_MAX_ERROR]

This error code is returned if the number of requested channels exceeds the maximum number of channels allowed by VoiceText™ Server.

Balance the traffic load between multiple servers after installing VoiceText™ Server on other machines.

[TTS_RESULT_ERROR]

This error code is returned if VoiceText™ Server has encountered a problem during synthesis.

For details on this error, see the service log file on the server.

Example

```
// In case the texts to be synthesized are string type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int nReturn = 0;
			byte[] result;
			int nVoiceLen = 0;
			string Text = "Hello~ This is VoiceText Demo!!";

			// tts buffer ex request test
			result = a.TTSRequestBufferEx("127.0.0.1", 7000, Text, out nVoiceLen,
libttsapi.TTS_KATE_DB, libttsapi.FORMAT_WAV, libttsapi.TEXT_NORMAL, 100, 100,
100, 0, 1, 1, out nReturn);
			if (nReturn == libttsapi.TTS_RESULT_SUCCESS)
			{
				Console.WriteLine("RequestBufferEx Success (length=" +
nVoiceLen.ToString() + ")!!!");
				libttsapi.WriteByteToFile("bufferex.wav", result, nVoiceLen);
			}
			else
			{
				Console.WriteLine("RequestBufferEx Failed (" + nReturn.ToString()
+ ")!!!");
			}
		}
	}
}
```

```
// In case the texts to be synthesized are byte[] type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int nReturn = 0;
			byte[] result;
			int nVoiceLen = 0;
			string Text = "Hello~ This is VoiceText Demo!!";
			byte[] bText = null;
			int bText_size = 0;

			bText = Encoding.Default.GetBytes(Text.ToCharArray());
			bText_size = bText.Length;

			// tts buffer ex request test
			result = a.TTSRequestBufferEx("127.0.0.1", 7000, bText, bText_size, out
nVoiceLen, libttsapi.TTS_KATE_DB, libttsapi.FORMAT_WAV,
libttsapi.TEXT_NORMAL, 100, 100, 100, 0, 1, 1, out nReturn);
			if (nReturn == libttsapi.TTS_RESULT_SUCCESS)
			{
				Console.WriteLine("RequestBufferEx Success (length=" +
nVoiceLen.ToString() + ")!!!");
				libttsapi.WriteByteToFile("bufferex2.wav", result, nVoiceLen);
			}
			else
			{
				Console.WriteLine("RequestBufferEx Failed (" + nReturn.ToString()
+ ")!!!");
			}
		}
	}
}
```

TTSRequestBufferSSML

Synthesizing SSML formatted texts into voice buffer

```
byte[] TTSRequestBufferSSML (
    string          szServer ,
    int             nPort ,
    string          pText ,
    out int         nVoiceLen ,
    int             nSpeakerID ,
    int             nVoiceFormat ,
    out int         pMarkSize ,
    out TTSMARK[]   ppTTSMark ,
    int             bFirst ,
    out int         nReturn
)
```

```
byte[] TTSRequestBufferSSML (
    string          szServer ,
    int             nPort ,
    byte[]          pText ,
    int             nTextLen ,
    out int         nVoiceLen ,
    int             nSpeakerID ,
    int             nVoiceFormat ,
    out int         pMarkSize ,
    out TTSMARK[]   ppTTSMark ,
    int             bFirst ,
    out int         nReturn
)
```

Parameters

szServer

IP address of VoiceText™ Server

nPort

Port number used for sending synthesis requests to and response from VoiceText™ Server, default is 7000

pText

Text string to be synthesized, string or byte[] types could be used.

nTextLen

Length of *pText*

nVoiceLen

A pointer to an integer variable that holds the length of the buffer frame returned from this function. Note that the buffer length varies by synthesis output format.

nSpeakerID

ID of the speaker's voice (e.g. voice engine)

The available voice engines are listed below. *nSpeakerID* must be set to the ID of a voice engine that is installed on the system.

TTS_JUNWOO_DB	(3)	Junwoo (Korean, Male)
TTS_SUJIN_DB	(8)	Sujin (Korean, Female)
TTS_YUMI_DB	(10)	Yumi (Korean, Female)
TTS_GYURI_DB	(11)	Gyuri (Korean, Female)
TTS_DAYOUNG_DB	(12)	Dayoung (Korean, Female)
TTS_CHORONG_DB	(13)	Chorong (Korean, Female)
TTS_KATE_DB	(100)	Kate (English, Female)
TTS_PAUL_DB	(101)	Paul (English, Male)
TTS_LILY_DB	(200)	Lily (Chinese, Female)
TTS_WANG_DB	(201)	Wang (Chinese, Male)
TTS_MIYU_DB	(300)	Miyu (Japanese, Female)
TTS_SHOW_DB	(301)	Show (Japanese, Male)
TTS_MISAKI_DB	(302)	Misaki (Japanese, Female)

nVoiceFormat

Output file format of synthesis.

The available voice formats are as follows.

FORMAT_DEFAULT	(0)	Default
FORMAT_PCM	(2)	16bit linear PCM
FORMAT_MULAW	(3)	8bit Mu-law PCM
FORMAT_ALAW	(4)	8bit A-law PCM

pMarkSize

ppTTSMark parameter's effective number is received.

ppTTSMark

mark information in the current voice Frame buffer is received.

bFirst

Whether or not the first frame of the voice output is being requested

TRUE	(1)	Request of the first frame
FALSE	(0)	Request after the first frame

nReturn

Return value denoting the result status

Description

It requests the synthesis of the SSML formatted texts. For the detailed information on the SSML grammar supported by VoiceText please refer to the enclosed SSML manual.

When there is <mark> tag in requested texts <mark> information can not be obtained. The returned <mark> information's sequence is as same as the requested texts' <mark> tag.

When pMarkSize is above one, you can approach ppTTSMark and obtain <mark> information. When pMarkSize is below zero, it means there is no <mark> information in the current voice frame buffer.

Only the separated buffer request is possible in TTSRequestBufferSSML function.

When using the multi-frame buffer method to receive the synthesized voice, read data from the frame buffer for *nVoiceLen* length. The size of the frame buffer being sent is specified by *nVoiceLen* parameter.

bFirst sets TRUE(1) at the first frame request and FALSE(0) at the next frame.

When the synthesis finishes successfully, *nReturn* will point to the value of either TTS_RESULT_CONTINUE(0) or TTS_RESULT_SUCCESS(1). If the last frame buffer is being sent, it will be set to TTS_RESULT_SUCCESS(1). Otherwise, it will be set to TTS_RESULT_CONTINUE(0).

When an error occurs, a negative value will be assigned to *nReturn*. For details on error codes, see the section on Errors.

In SSML's grammatical errors, the error code TTS_SSML_ERROR(-13) is returned to *nReturn* Parameter and the information on the first generated grammatical errors in the synthesis server log is saved.

Return Values

After a successful synthesis, a pointer to the synthesized voice buffer frame is returned. In case of error, NULL pointer is returned. The length of buffer frame is assigned to *nVoiceLen* parameter.

Errors

The descriptions of returned error codes are as follows : Please refer to Return Values in Appendix A for error code value.

[TTS_HOSTNAME_ERROR]

This error code is returned when an invalid IP address has been provided.

An IP address is invalid if the format is incorrect.

[TTS_SOCKET_ERROR]

This error code is returned when the socket() used by this API returns an error.

For possible causes of socket() error, please refer to the relevant system manual.

[TTS_CONNECT_ERROR]

This error code is returned if the connect() function used by this API returns an error.

When this error is returned, make sure that VoiceText™ Server is executed and confirm that the network environment is working normally.

For possible causes of connect() error, please refer to the relevant system manual.

[TTS_READWRITE_ERROR]

This error code is returned when there has been an error in sending/receiving data.

This error could occur if the connection to the server has timed out due to slow response time of the VoiceText™ Server.

For possible causes of read(), write() error, please refer to the relevant system manual.

[TTS_TEXT_ERROR]

This error code is returned when an invalid text length was provided.

Text length must be a positive integer.

It occurs when the designated texts are longer than the texts to be synthesized.

[TTS_VOICEFORMAT_ERROR]

This error code is returned when an invalid value has been assigned to *nVoiceFormat*.

Check whether a valid voice format was specified as *nVoiceFormat*.

For the supported voice formats please refer to the appendix B.

[TTS_PARAM_ERROR]

This error code is returned when invalid values have been assigned to *szSaveFile* or *szSaveDir* parameters.

[TTS_DISK_ERROR]

This error code is returned when there are problems on server hard disk.

Check if there is enough disk space available of the server.

[TTS_SPEAKER_ERROR]

This error code is returned when an invalid value has been set to *nSpeakerID.*

Make sure that the voice engine corresponding to *nSpeakerID* is installed on the server.

[TTS_MAX_ERROR]

This error code is returned if the number of requested channels exceeds the maximum number of channels allowed by VoiceText™ Server.

Balance the traffic load between multiple servers after installing VoiceText™ Server on other machines.

[TTS_RESULT_ERROR]

This error code is returned if VoiceText™ Server has encountered a problem during synthesis.

For details on this error, see the service log file on the server.

[TTS_SSML_ERROR]

There are grammatical errors in requested SSML texts.

Please refer to the server log for SSML grammatical errors information

Example

```
// In case the texts to be synthesized are string type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int pMarkSize = 0;
			TTSMARK[] ppMark;
			int nReturn = libttsapi.TTS_RESULT_CONTINUE;
			byte[] result;
			int nVoiceLen = 0;
		       string Text = "<mark name=\"a\"/>Hello~ <mark name=\"b\"/This is
VoiceText Demo!!";
			int bFirst = 1;
			int count = 0;

			while (nReturn == libttsapi.TTS_RESULT_CONTINUE)
			{
				result = a.TTSRequestBufferSSML("127.0.0.1", 7000, Text, out
nVoiceLen, libttsapi.TTS_KATE_DB, libttsapi.FORMAT_PCM, out pMarkSize, out
ppMark, bFirst, out nReturn);
				bFirst = 0;
				for (int i = 0; i < pMarkSize; i++)
				{
					Console.WriteLine("[" + ppMark[i].nOffsetInBuffer.ToString() +
"][" + ppMark[i].nOffsetInStream.ToString() + "]" + ppMark[i].sMarkName);
					Console.WriteLine(ppMark[i].sMarkName.Length.ToString());
				}
				if (nReturn == libttsapi.TTS_RESULT_CONTINUE || nReturn ==
libttsapi.TTS_RESULT_SUCCESS)
				{
					Console.WriteLine("RequestBufferSSML Success (length=" +
nVoiceLen.ToString() + ")!!!");
					libttsapi.WriteByteToFile("bufferssml" + count.ToString() +
".pcm", result, nVoiceLen);
				}
				else
				{
					Console.WriteLine("RequestBufferSSML Failed (" +
nReturn.ToString() + ")!!!");
				}
				count++;
			}
		}
	}
}
```

```
// In case the texts to be synthesized are byte[] type
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int pMarkSize = 0;		TTSMARK[] ppMark;
			int nReturn = libttsapi.TTS_RESULT_CONTINUE;
			byte[] result;			int nVoiceLen = 0;
		       string Text = "<mark name=\"a\"/>Hello~ <mark name=\"b\"/This is
VoiceText Demo!!";

			int bFirst = 1;			int count = 0;
			byte[] bText = null;		int bText_size = 0;

			bText = Encoding.UTF8.GetBytes(Text.ToCharArray());
			bText_size = bText.Length;

			while (nReturn == libttsapi.TTS_RESULT_CONTINUE)
			{
				result = a.TTSRequestBufferSSML ("127.0.0.1", 7000,
bText,bText_size, out nVoiceLen, libttsapi.TTS_KATE_DB, libttsapi.FORMAT_PCM,
out pMarkSize, out ppMark, bFirst, out nReturn);
				bFirst = 0;

				for (int i = 0; i < pMarkSize; i++)
				{
					Console.WriteLine("[" + ppMark[i].nOffsetInBuffer.ToString() +
"][" + ppMark[i].nOffsetInStream.ToString() + "]" + ppMark[i].sMarkName);
					Console.WriteLine(ppMark[i].sMarkName.Length.ToString());
				}

				if (nReturn == libttsapi.TTS_RESULT_CONTINUE || nReturn ==
libttsapi.TTS_RESULT_SUCCESS)
				{
					Console.WriteLine("RequestBufferSSML Success (length=" +
nVoiceLen.ToString() + ")!!!");
					libttsapi.WriteByteToFile("bufferssml" + count.ToString() +
".pcm", result, nVoiceLen);
				}
				else
				{
					Console.WriteLine("RequestBufferSSML Failed (" +
nReturn.ToString() + ")!!!");
					break;
				}
				count++;
			}
		}
	}
}
```

TTSRequestStatus

TTSRequestStatus requests the status of VoiceText™ Server.

int **TTSRequestStatus** (
    string *szServer* ,
    int *nPort* ,
)

Parameters

szServer

IP address of VoiceText™ Server

nPort

Port number used for sending status requests to and response from VoiceText™ Server. By default 7777 is used.

Description

TTSRequestStatus() is used to monitor and to request the status of VoiceText™ Server. When the value 1 is returned, the status of VoiceText™ Server is normal. If any other value is returned, the VoiceText™ Service is in pause state or the service has been stopped. Refer to below section for descriptions on return values.

Return Values

When VoiceText™ Service is running, TTS_SERVICE_ON(1) is returned. When the Service is paused, TTS_SERVICE_PAUSED(2) is returned. When the Service has been stopped TTS_SERVICE_OFF(0) is returned. If the connection to VoiceText™ Server fails or for any other failures, a negative value is returned. For explanation on detailed error codes, see the section on Errors.

Notes

The port number used to request VoiceText™ Server should be the management port number specified by the VoiceText™ Server Administration tool. (default 7777)

Errors

The descriptions of returned error codes are as follows : Please refer to Return Values in Appendix A for error code value.

[TTS_HOSTNAME_ERROR]

This error code is returned when an invalid IP address has been provided.

An IP address is invalid if the format is incorrect.

[TTS_SOCKET_ERROR]

This error code is returned when the socket() used by this API returns an error.

For possible causes of socket() error, please refer to the relevant system manual.

[TTS_CONNECT_ERROR]

This error code is returned if the connect() function used by this API returns an error.

When this error is returned, make sure that VoiceText™ Server is executed and confirm that the network environment is working normally.

For possible causes of connect() error, please refer to the relevant system manual.

[TTS_READWRITE_ERROR]

This error code is returned when there has been an error in sending/receiving data.

This error could occur if the connection to the server has timed out due to slow response time of the VoiceText™ Server.

For possible causes of read(), write() error, please refer to the relevant system manual.

[TTS_MAX_ERROR]

This error code is returned if the number of requested channels exceeds the maximum number of channels allowed by VoiceText™ Server.

Balance the traffic load between multiple servers after installing VoiceText™ Server on other machines.

Example

```
using System;
using System.Text;
using Voiceware;

namespace cs_using_vtap_api_dotnet20
{
	class Program
	{
		static void Main(string[] args)
		{
			libttsapi a = new libttsapi();
			int nReturn = 0;

			// tts status request test
			nReturn = a.TTSRequestStatus("127.0.0.1", 7777);
			if (nReturn == libttsapi.TTS_SERVICE_ON) {
				Console.WriteLine("Server is ON!!!");
			}
			else if (nReturn == libttsapi.TTS_SERVICE_PAUSED)   {
				Console.WriteLine("Server is PAUSED!!!");
			}
			else if (nReturn == libttsapi.TTS_SERVICE_OFF)   {
				Console.WriteLine("Server is OFF!!!");
			}
			else {
				Console.WriteLine("Server is ERROR(" + nReturn.ToString() + ")!!!");
			}
		}
	}
}
```

Appendix A : Return Value Table

Return value table

Value	Name
1	TTS_RESULT_SUCCESS
-9	TTS_RESULT_ERROR
-1	TTS_HOSTNAME_ERROR
-2	TTS_SOCKET_ERROR
-3	TTS_CONNECT_ERROR
-4	TTS_READWRITE_ERROR
-5	TTS_MEMORY_ERROR
-6	TTS_TEXT_ERROR
-7	TTS_VOICEFORMAT_ERROR
-8	TTS_PARAM_ERROR
-10	TTS_SPEAKER_ERROR
-11	TTS_DISK_ERROR
-13	TTS_SSML_ERROR
-100	TTS_MAX_ERROR

Appendix B : Supported Voice Format

Supported Voice Format

	Format	TTS Engine	TTS Server			
			File	One Frame Buffer	Multiple Frame Buffer	Multi-Frame Buffer SSML
1	16bit linear PCM	O	O	O	O	O
2	8bit A-law PCM	O	O	O	O	O
3	8bit Mu-law PCM	O	O	O	O	O
4	4bit Dialogic ADPCM	O	O	O	O	X
5	16bit linear PCM Wave	O	O	O	X	X
6	8bit unsigned linear PCM Wave	O	O	O	X	X
7	8bit A-law PCM Wave	O	O	O	X	X
8	8bit Mu-law PCM Wave	O	O	O	X	X
9	ASF	X	O	O	X	X
10	OGG	X	O	O	X	X

1. Both 16KHz, 8KHz all show same result
2. ASF format is not supported under UNIX environment