Digital Future Text-to-Speech SDK Programmer's Guide for C++

Version 3.5.0

The software described in this manual is furnished under a valid license agreement or nondisclosure agreement with Digital Future and/or NeoSpeech Inc and/or Cepstral LLC. The software may be used only in accordance with the terms of the agreement.

Copyright Notice

Disclaimer

Trademark

Chapter 1: Overview

Digital Future Text-to-Speech SDK (DF TTS SDK) is the only **true OS native** (no COM/ActiveX, client-server, etc overheads) cross-platform provider-independent technology that provides standardized and unified API's for the implementation of the conversion of text data into speech.

The currently supported technologies are:

- Voiceware Co., Ltd. VoiceText™ for U.S. English, Chinese, Japanese, Korean
- Cepstral LLC for U.S. English, U.K. English, French, German, Spanish, Italian
- AT&T Natural Voices ™ (for internal use only and not covered by the license of this SDK. If you are interested in using this technology, you must obtain developer licensing from AT&T directly. Please contact us at sales@digitalfuturesoft.com for details.)
- Mac OS X Speech Manager
- Microsoft ® Speech API 5.x

This document contains descriptions and examples of DF TTS SDK API's, necessary for developing text-to-speech enabled software applications in C++.

The Appendix of the present manual includes explanations on the DF TTS SDK Tag Set, which controls inflection information (pitch, sound speed, volume, pause) as well as the SSML Tag Set, one of the VoiceXML 2.0 standards defined by W3C (World Wide Web Consortium).

Supported platforms: Microsoft ® Windows ™ (all versions), Windows Mobile (2003 and up), Mac OS X and Linux.

Supported compilers/C++ IDE's:

Microsoft ® Windows: Microsoft ® Visual C++ 6, Microsoft ® Visual C++ 2005, Microsoft ® Visual C++ 2008;
Microsoft ® Windows Mobile: Microsoft ® eMbedded Visual C++ 4.0, Microsoft ® Visual C++ 2005, Microsoft ® Visual C++ 2008;
Mac OS X: XCode;
Linux: gcc, cc;

Chapter 2: Getting Started

DF TTS SDK API's are aimed at facilitating the integration of text-to-speech technology in software applications.

Microsoft ® Windows ™ SDK setup notes:

1. The developer must include the following header file to be able to use the API's:

#include "dftts.h" (header file located in the C++\Include folder);

dfttsval.h *and* ***dfttslang.h*** *must be placed in the same directory as dftts.h.*

2. The following lib needs to be added to the project:

dftts.lib (located in the Windows\C++\Dll\[VC6, VC2005 or VC2008]\[Release... or Debug...] folder);

3. The following DLLs need to be distributed with the application (the best practice is to place them in the application directory):

***dftts.dll*;**
vt_eng.dll;
vt_chi.dll;
vt_jpn.dll;
vt_kor.dll;
***swift.dll*;**

Note: All other Dll's are located in the Windows\C++\Dll\[VC6, VC2005 or VC2008]\[Release... or Debug...] folder.

Mac OS X SDK setup notes:

1. The developer must include the following header file to be able to use the API's:

#include "dftts.h" (header file located in the MacOSX/C++ (Carbon)/Include or MacOSX/Objective-C++ (Cocoa)/Include folder);

dfttsval.h *and* ***dfttslang.h*** *must be placed in the same directory as dftts.h.*

2. The following frameworks must be linked to and placed in /Library/Frameworks then distributed with your application:

dfttssdk.framework (located in the ...Frameworks/Release... folder for your programming language);

swift.framework (located in the ...Frameworks/Release... folder for your programming language);

Linux x86 setup notes:

1. The developer must include the following header file to be able to use the API's:

#include "dftts.h" (header file located in the Linux/C++/Include folder);

dfttsval.h *and* ***dfttslang.h*** *must be placed in the same directory as dftts.h.*

2. The developer must link against the following libraries:

2.1 If static linking is preferred:

libdftts.a (located in the Linux/C++/Lib/Release... folder);

2.2 If shared linking is preferred:

libdftts.so (located in the Linux/C++/Lib/Release... folder);

*** ***libdftts.so must be placed in a directory and its path must be exported in the shell script (see 4.);***

3. At least 1 voice for Linux must be installed. See the download instructions or see READMEFIRST!!!.TXT for available voice downloads.

4. The easiest way to execute the developer TTS application, is to use and run a shell script file that exports all library paths and then runs the application.

***** Shell scripts are included with the sample for Linux (see SH files in Linux/C++/Sample/[Full or Demo].**

Chapter 3: DF TTS SDK API Reference

This chapter describes DF TTS SDK Function References, which can be divided into 5 categories.

- Basic API's (loading/unloading the synthesis engine and the User Dictionary, setting event callbacks)

InitDFTTSEngineEx()
InitDFTTSEngineEx2()
UninitDFTTSEngine()
LoadDFTTSUserDict()
UnloadDFTTSUserDict()
SetDFTTSOnWordCallBack()
SetDFTTSOnExportCallBack()

- Sound Card API's (Play/Stop/Pause/Resume of synthesized sound output via sound card)

DFTTSSpeak()
DFTTSStop()
DFTTSPause()
DFTTSResume()

- File API's (Synthesize and save to a voice file)

DFTTSExportToFile()
DFTTSExportToFileEx()

- Buffering API's (Synthesize to a voice buffer)

DFTTSExportToBuffer() (not currently supported by the Desktop SDK)

- Information API's (Get Engine Information)

GetDFTTSEngineInfo()
GetDFTTSNumVoices()
GetDFTTSVoice()
GetDFTTSInstalledVoices() *[deprectated as of v 3.5.0]*
CleanDFTTSInstalledVoicesRetrieval() *[deprectated as of v 3.5.0]*

InitDFTTSEngineEx

Loads the synthesizer's TTS databases.

Synopsis

```
#include "dftts.h"
InitDFTTSEngineEx( HWND hwndWinOwner,
char* szNeoSpeechDBFolderPath[NEOSPEECH_NUM_VOICES],
char* szNeoSpeechLicFilePath[NEOSPEECH_NUM_VOICES] );
```

Parameters

hwndWinOwner
Window handle (Win32) for speech event processing.

szNeoSpeechDBFolderPath
The paths (char* array) where the NeoSpeech synthesizer databases of all supported NeoSpeech Voicetext ™ voices are located.
NEOSPEECH_NUM_VOICES is currently 12.
Leave an array member empty or NULL for the default directory or if you do not plan on using the voice.

Below is how the char* array should be structured:
Index 0: The DB path for the voice Kate
Index 1: The DB path for the voice Paul
Index 2: The DB path for the voice Miyu
Index 3: The DB path for the voice Show
Index 4: The DB path for the voice Misaki
Index 5: The DB path for the voice Lily
Index 6: The DB path for the voice Wang
Index 7: The DB path for the voice Junwoo
Index 8: The DB path for the voice Sujin
Index 9: The DB path for the voice Yumi
Index 10: The DB path for the voice Gyuri
Index 11: The DB path for the voice Dayoung

szNeoSpeechLicFilePath
NeoSpeech VoiceText ™ License verification file path (char* array), filled in the order of szNeoSpeechDBFolderPath indices.
For NULL, Use the "verification.txt" file located in the default path for the voice.

Note: All other vendors' voices are automatically loaded and you do not need to specify DB paths for them.

Description

The function loads the synthesizer voice databases and it is used when the program starts.

Return Values

A pointer to a structure of type InitDFTTSEngineReturnValueEx:

```
struct InitDFTTSEngineReturnValueEx
{
	DFTTSVoiceEngineType vet[NUM_ENGINE_INITIALIZATIONS];

	InitDFTTSEngineReturnValue ert[NUM_ENGINE_INITIALIZATIONS];

};
```

Currently, NUM_ENGINE_INITIALIZATIONS is 16 (12 for NEOSPEECHVOICETEXT voices (i.e. for provider NeoSpeech), 1 for the whole CEPSTRAL engine (i.e. for provider Cepstral), 1 for the whole ATTNV engine (i.e. for provider AT&T - internal use only, use MSSAPI instead)), 1 for the whole MACOSXSPMAN engine (i.e. for provider Mac OS X Speech Manager) and 1 for the whole MSSAPI engine (i.e. for provider Microsoft Speech API 5.x).

The structure holds a load return value for each initialization.

The member array vet has values NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2), MACOSXSPMAN (4) and MSSAPI (3).

The first to twelfth member of ert (ert[0]-ert[11]) returns the load status of a voice of engine NEOSPEECHVOICETEXT in the order of the voice loading (see the Parameters section above).

The thirteenth member of ert (ert[12]) returns the load status of the whole engine CEPSTRAL.

The fourteenth member of ert (ert[13]) returns the load status of the whole engine ATTNV.

The fifteenth member of ert (ert[14]) returns the load status of the whole engine MACOSXSPMAN.

The sixteenth member of ert (ert[15]) returns the load status of the whole engine MSSAPI.

Each member can hold the following status codes:

When the database is successfully loaded, it returns INIT_DFTTS_ENGINE_SUCCESS. The following values are returned when error occurs: (Refer to dfttsval.h)

[INIT_DFTTS_ENGINE_ERROR_DB_PATH_DIFFERENT] Tried to load the synthesizer database with different values of szNeoSpeechDBFolderPath in
case of using multiple synthesizer databases
[INIT_DFTTS_ENGINE_ERROR_CHANNEL_MEM_FAIL] Failed to secure channel memory
[INIT_DFTTS_ENGINE_ERROR_DB_MORPHEME_ANALYSIS_FAIL] Failed to load DB for the Morpheme Analysis
[INIT_DFTTS_ENGINE_ERROR_DB_BREAK_INDEX_FAIL] Failed to load DB for the Break Index
[INIT_DFTTS_ENGINE_ERROR_DB_TEXT_PREP_FAIL] Failed to load DB for the Text Pre-Processing
[INIT_DFTTS_ENGINE_ERROR_DB_ACOU_MODEL_FAIL] Failed to load DB for the Acoustic Model
[INIT_DFTTS_ENGINE_ERROR_DB_UNIT_SEL_FAIL] Failed to load DB for Unit Selection
[INIT_DFTTS_ENGINE_ERROR_DB_PROS_MODEL_FAIL] Failed to load DB for Prosody Model
[INIT_DFTTS_ENGINE_ERROR_DB_SPEECH_DB_FAIL] Failed to load DB for Speech Database
[INIT_DFTTS_ENGINE_ERROR_DB_PITCH_LOC_INFO_FAIL] Failed to load DB for Pitch Location Information

Additional error codes:

INIT_DFTTS_ENGINE_ERROR_FAILED,
INIT_DFTTS_ENGINE_ERROR_INVALIDARG,
INIT_DFTTS_ENGINE_ERROR_OUTOFMEMORY,
INIT_DFTTS_ENGINE_ERROR_NOTIMPL,
INIT_DFTTS_ENGINE_ERROR_ABORT,
INIT_DFTTS_ENGINE_ERROR_UNKNOWN,
INIT_DFTTS_ENGINE_ERROR_BADHANDLE,
INIT_DFTTS_ENGINE_ERROR_EXCEPTION,
INIT_DFTTS_ENGINE_ERROR_EMPTY,
INIT_DFTTS_ENGINE_ERROR_FULL,
INIT_DFTTS_ENGINE_ERROR_INVALIDSTATE,
INIT_DFTTS_ENGINE_ERROR_BADVERSION,
INIT_DFTTS_ENGINE_ERROR_INSUFFICIENT_BUFFER,
INIT_DFTTS_ENGINE_ERROR_UNSUPPORTED,
INIT_DFTTS_ENGINE_ERROR_NOLICENSE,
INIT_DFTTS_ENGINE_ERROR_CREATECHILDPROCESS_FAILED,
INIT_DFTTS_ENGINE_ERROR_NOENVIRONMENTPATH,
INIT_DFTTS_ENGINE_ERROR_TIMEOUT,
INIT_DFTTS_ENGINE_ERROR_OUTOFRESOURCES,
INIT_DFTTS_ENGINE_ERROR_NOVOICES,
INIT_DFTTS_ENGINE_ERROR_CREATEFAIL,
INIT_DFTTS_ENGINE_ERROR_CONNECTFAIL,
INIT_DFTTS_ENGINE_ERROR_BINDFAIL,
INIT_DFTTS_ENGINE_ERROR_LISTENFAIL,
INIT_DFTTS_ENGINE_ERROR_CONNECTIONCLOSED,
INIT_DFTTS_ENGINE_ERROR_ACCEPTFAIL,
INIT_DFTTS_ENGINE_ERROR_SOCKETTIMEOUT,
INIT_DFTTS_ENGINE_ERROR_SOCKETERROR,
INIT_DFTTS_ENGINE_ERROR_NOMORESERVERS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EWOULDBLOCK,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EINPROGRESS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EALREADY,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTSOCK,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EDESTADDRREQ,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EMSGSIZE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROTOTYPE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOPROTOOPT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROTONOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ESOCKTNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EOPNOTSUPP,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPFNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EAFNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EADDRINUSE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EADDRNOTAVAIL,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETDOWN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETUNREACH,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETRESET,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNABORTED,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNRESET,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOBUFS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EISCONN,

INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTCONN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ESHUTDOWN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ETOOMANYREFS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNREFUSED,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ELOOP,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENAMETOOLONG,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EHOSTDOWN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EHOSTUNREACH,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTEMPTY,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROCLIM,
INIT_DFTTS_ENGINE_ERROR_THREADSTARTED,
INIT_DFTTS_ENGINE_ERROR_THREADNOTSTARTED,
INIT_DFTTS_ENGINE_ERROR_THREADCOULDNOTCREATE,
INIT_DFTTS_ENGINE_ERROR_BADNVFILE,
INIT_DFTTS_ENGINE_ERROR_NOAUDIODRIVER,
INIT_DFTTS_ENGINE_ERROR_DICTNOTFOUND,
[INIT_DFTTS_ENGINE_ERROR_OTHER] Other errors

See Also

InitDFTTSEngineEx2()
UninitDFTTSEngine()

Example

```
//Init voice engines (IMPORTANT: SET YOUR OWN PATHS HERE FOR THE LISTED VOICES.
//ALL OTHER VOICES ARE INITIALIZED AUTOMATICALLY)

char* szNeospDBPaths[NEOSPEECH_NUM_VOICES] =
{
	NULL, //Kate
	"C:\\Program Files\\VW\\VT\\Paul\\M16", //Paul
	NULL, //Miyu
	NULL, //Show
	NULL, //Misaki
	NULL, //Lily
	NULL, //Wang
	NULL, //Junwoo
	NULL, //Sujin
	NULL, //Yumi
	NULL, //Gyuri
	NULL  //Dayoung
};

char* szNeospLicPaths[NEOSPEECH_NUM_VOICES] =
{
	NULL, //Kate
"C:\\Program Files\\VW\\VT\\Paul\\M16\\data-common\\verify\\verification.txt",
//Paul
	NULL, //Miyu
	NULL, //Show
	NULL, //Misaki
	NULL, //Lily
```

```
	NULL, //Wang
	NULL, //Junwoo
	NULL, //Sujin
	NULL, //Yumi
	NULL, //Gyuri
	NULL  //Dayoung
};


InitDFTTSEngineReturnValueEx* initresult = InitDFTTSEngineEx( hDlg,
szNeospDBPaths, szNeospLicPaths );

/************************************************

	initresult is an pointer to InitDFTTSEngineReturnValueEx with 2 array
members.

	//NEOSPEECHVOICETEXT ENGLISH (NOT SUPPORTED BY MAC OS X and Linux)
	initresult.vet[0] = NEOSPEECHVOICETEXT; //KATE
	initresult.vet[1] = NEOSPEECHVOICETEXT; //PAUL

	//NEOSPEECHVOICETEXT JAPANESE
	initresult.vet[2] = NEOSPEECHVOICETEXT; //MIYU
	initresult.vet[3] = NEOSPEECHVOICETEXT; //SHOW
	initresult.vet[4] = NEOSPEECHVOICETEXT; //MISAKI

	//NEOSPEECHVOICETEXT CHINESE
	initresult.vet[5] = NEOSPEECHVOICETEXT; //LILY
	initresult.vet[6] = NEOSPEECHVOICETEXT; //WANG

	//NEOSPEECHVOICETEXT KOREAN
	initresult.vet[7] = NEOSPEECHVOICETEXT; //JUNWOO
	initresult.vet[8] = NEOSPEECHVOICETEXT; //SUJIN
	initresult.vet[9] = NEOSPEECHVOICETEXT; //YUMI
	initresult.vet[10] = NEOSPEECHVOICETEXT; //GYURI
	initresult.vet[11] = NEOSPEECHVOICETEXT; //DAYOUNG

	//CEPSTRAL
	initresult.vet[12] = CEPSTRAL;

	//AT&T NV (NOT SUPPORTED BY MAC OS X and Linux)
	initresult.vet[13] = ATTNV;

	//MAC OS X SPMAN (NOT SUPPORTED BY Windows and Linux)
	initresult.vet[14] = MACOSXSPMAN;

	//MICROSOFT SAPI (NOT SUPPORTED BY MAC OS X and Linux)
	initresult.vet[15] = MSSAPI;

	ert memebers are of type InitDFTTSEngineReturnValue

	(NOT SUPPORTED BY Linux and Mac OS X)
	initresult.ert[0] - error code for NeoSpeech Kate
	initresult.ert[1] - error code for NeoSpeech Paul
	initresult.ert[2] - error code for NeoSpeech Miyu
	initresult.ert[3] - error code for NeoSpeech Show
	initresult.ert[4] - error code for NeoSpeech Misaki
```

```
	initresult.ert[5] - error code for NeoSpeech Lily
	initresult.ert[6] - error code for NeoSpeech Wang
	initresult.ert[7] - error code for NeoSpeech Junwoo
	initresult.ert[8] - error code for NeoSpeech Sujin
	initresult.ert[9] - error code for NeoSpeech Yumi
	initresult.ert[10] - error code for NeoSpeech Gyuri
	initresult.ert[11] - error code for NeoSpeech Dayoung

	initresult.ert[12] - error code for the whole Cepstral engine

	(NOT SUPPORTED BY Linux and Mac OS X)
	initresult.ert[13] - error code for the whole AT&T NV engine

	(NOT SUPPORTED BY Linux and Windows)
	initresult.ert[14] - error code for the whole MAC OS X SPMAN

	(NOT SUPPORTED BY Linux and Mac OS X)

	initresult.ert[15] - error code for the whole MS SAPI

	**************************************************/

char* szEngineVoiceNames[NUM_ENGINE_INITIALIZATIONS] =
	{"NeoSpeech Kate","NeoSpeech Paul","NeoSpeech Miyu","NeoSpeech Show",
	"NeoSpeech Misaki","NeoSpeech Lily","NeoSpeech Wang","NeoSpeech Junwoo",
	"NeoSpeech Sujin","NeoSpeech Yumi","NeoSpeech Gyuri","NeoSpeech Dayoung",
	"Cepstral engine","AT&T NV engine","Mac OS X Speech Manager
engine","Microsoft SAPI engine"};

for( size_t y=0;y<NUM_ENGINE_INITIALIZATIONS;y++ )
{

	if( initresult->ert[y] != INIT_DFTTS_ENGINE_SUCCESS )
	{

		char szErrNo[2];

		sprintf( szErrNo, "%d", (short)initresult->ert[y] );

		printf(szEngineVoiceNames[y]);

		printf(" initialization error: ");

		printf( szErrNo );

		printf( ".\nThe voice or engine may not be installed " \
"or you specified wrong voice path (in the case of a NeoSpeech voice).\n" );


	}


}
```

InitDFTTSEngineEx2

Loads the synthesizer's TTS database (an alternative to InitDFTTSEngineEx).

Synopsis

```
#include "dftts.h"
void InitDFTTSEngineEx2( HWND hwndWinOwner,
      char* szNeoSpeechDBFolderPath[NEOSPEECH_NUM_VOICES],
      char* szNeoSpeechLicFilePath[NEOSPEECH_NUM_VOICES],
      DFTTSVoiceEngineType* psiLoadedEngines,
      InitDFTTSEngineReturnValue* psiLoadedEnginesReturnValues
      );
```

Parameters

hwndWinOwner
Window handle (Win32) for speech event processing.

szNeoSpeechDBFolderPath
The paths (char* array) where the NeoSpeech synthesizer databases of all supported NeoSpeech Voicetext ™ voices are located.
NEOSPEECH_NUM_VOICES is currently 12.
Leave an array member empty or NULL for the default directory or if you do not plan on using the voice.

Below is how the char* array should be structured:
Index 0: The DB path for the voice Kate
Index 1: The DB path for the voice Paul
Index 2: The DB path for the voice Miyu
Index 3: The DB path for the voice Show
Index 4: The DB path for the voice Misaki
Index 5: The DB path for the voice Lily
Index 6: The DB path for the voice Wang
Index 7: The DB path for the voice Junwoo
Index 8: The DB path for the voice Sujin
Index 9: The DB path for the voice Yumi
Index 10: The DB path for the voice Gyuri
Index 11: The DB path for the voice Dayoung

szNeoSpeechLicFilePath
NeoSpeech VoiceText ™ License verification file path (char* array), filled in the order of szNeoSpeechDBFolderPath indices.
For NULL, Use the "verification.txt" file located in the default path for the voice.

Note: All other vendor's voices are automatically loaded and you do not need to specify DB paths for them.

[OUT] psiLoadedEngines
Pointer to an array with currently used engine types.

[OUT] psiLoadedEnginesReturnValues
Pointer to an array with the load status return values for each engine (voice in the case of NEOSPEECHVOICETEXT).

Description
The function loads the synthesizer database and it is used when the program starts.

Return Values
None

Notes
Instead of using structure of type InitDFTTSEngineReturnValueEx, the return status values are held by 2 arrays.

This is an alternative to InitDFTTSEngineEx and for the C++ SDK developer makes no difference which of both functions should be used. *(This version is strictly used by the VB6, .NET or Java SDK developers since it provides more portable implementation.).*

The array psiLoadedEngines has member values NEOSPEECHVOICETEXT (0) (first 12 members to match with the 12 NeoSpeech voices supported), CEPSTRAL (1) for the whole Cepstral engine (member # 13), ATTNV (2) for the whole ATTNV engine (member # 14 (internal use only, use MSSAPI instead)), MACOSXSPMAN (4) for the whole MACOSXSPMAN engine (member # 15) and MSSAPI (3) for the whole MSSAPI engine (member # 16).

The first 12 members of psiLoadedEnginesReturnValues (psiLoadedEnginesReturnValues[0] through [11]) return the load status of each supported NEOSPEECHVOICETEXT voice in the load order (see Parameters section).

The thirteenth member of psiLoadedEnginesReturnValues (psiLoadedEnginesReturnValues [12]) returns the load status of the whole engine CEPSTRAL.

The fourteenth member (psiLoadedEnginesReturnValues [13]) returns the load status of the whole engine ATTNV.

The fifteenth member (psiLoadedEnginesReturnValues [14]) returns the load status of the whole engine MACOSXSPMAN.

The sixteenth member (psiLoadedEnginesReturnValues [15]) returns the load status of the whole engine MSSAPI.

Each member of psiLoadedEnginesReturnValues can hold the following status codes:

When the database is successfully loaded, it returns INIT_DFTTS_ENGINE_SUCCESS. The following values are returned when error occurs: (Refer to dfttsval.h)

[INIT_DFTTS_ENGINE_ERROR_DB_PATH_DIFFERENT] Tried to load the synthesizer database with different values of szNeoSpeechDBFolderPath in
case of using multiple synthesizer databases
[INIT_DFTTS_ENGINE_ERROR_CHANNEL_MEM_FAIL] Failed to secure channel memory
[INIT_DFTTS_ENGINE_ERROR_DB_MORPHEME_ANALYSIS_FAIL] Failed to load DB for the Morpheme Analysis
[INIT_DFTTS_ENGINE_ERROR_DB_BREAK_INDEX_FAIL] Failed to load DB for the Break Index
[INIT_DFTTS_ENGINE_ERROR_DB_TEXT_PREP_FAIL] Failed to load DB for the Text Pre-Processing
[INIT_DFTTS_ENGINE_ERROR_DB_ACOU_MODEL_FAIL] Failed to load DB for the Acoustic Model

[INIT_DFTTS_ENGINE_ERROR_DB_UNIT_SEL_FAIL] Failed to load DB for Unit Selection
[INIT_DFTTS_ENGINE_ERROR_DB_PROS_MODEL_FAIL] Failed to load DB for Prosody Model
[INIT_DFTTS_ENGINE_ERROR_DB_SPEECH_DB_FAIL] Failed to load DB for Speech Database
[INIT_DFTTS_ENGINE_ERROR_DB_PITCH_LOC_INFO_FAIL] Failed to load DB for Pitch Location Information

Additional error codes:

INIT_DFTTS_ENGINE_ERROR_FAILED,
INIT_DFTTS_ENGINE_ERROR_INVALIDARG,
INIT_DFTTS_ENGINE_ERROR_OUTOFMEMORY,
INIT_DFTTS_ENGINE_ERROR_NOTIMPL,
INIT_DFTTS_ENGINE_ERROR_ABORT,
INIT_DFTTS_ENGINE_ERROR_UNKNOWN,
INIT_DFTTS_ENGINE_ERROR_BADHANDLE,
INIT_DFTTS_ENGINE_ERROR_EXCEPTION,
INIT_DFTTS_ENGINE_ERROR_EMPTY,
INIT_DFTTS_ENGINE_ERROR_FULL,
INIT_DFTTS_ENGINE_ERROR_INVALIDSTATE,
INIT_DFTTS_ENGINE_ERROR_BADVERSION,
INIT_DFTTS_ENGINE_ERROR_INSUFFICIENT_BUFFER,
INIT_DFTTS_ENGINE_ERROR_UNSUPPORTED,
INIT_DFTTS_ENGINE_ERROR_NOLICENSE,
INIT_DFTTS_ENGINE_ERROR_CREATECHILDPROCESS_FAILED,
INIT_DFTTS_ENGINE_ERROR_NOENVIRONMENTPATH,
INIT_DFTTS_ENGINE_ERROR_TIMEOUT,
INIT_DFTTS_ENGINE_ERROR_OUTOFRESOURCES,
INIT_DFTTS_ENGINE_ERROR_NOVOICES,
INIT_DFTTS_ENGINE_ERROR_CREATEFAIL,
INIT_DFTTS_ENGINE_ERROR_CONNECTFAIL,
INIT_DFTTS_ENGINE_ERROR_BINDFAIL,
INIT_DFTTS_ENGINE_ERROR_LISTENFAIL,
INIT_DFTTS_ENGINE_ERROR_CONNECTIONCLOSED,
INIT_DFTTS_ENGINE_ERROR_ACCEPTFAIL,
INIT_DFTTS_ENGINE_ERROR_SOCKETTIMEOUT,
INIT_DFTTS_ENGINE_ERROR_SOCKETERROR,
INIT_DFTTS_ENGINE_ERROR_NOMORESERVERS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EWOULDBLOCK,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EINPROGRESS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EALREADY,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTSOCK,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EDESTADDRREQ,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EMSGSIZE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROTOTYPE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOPROTOOPT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROTONOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ESOCKTNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EOPNOTSUPP,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPFNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EAFNOSUPPORT,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EADDRINUSE,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EADDRNOTAVAIL,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETDOWN,

INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETUNREACH,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENETRESET,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNABORTED,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNRESET,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOBUFS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EISCONN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTCONN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ESHUTDOWN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ETOOMANYREFS,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ECONNREFUSED,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ELOOP,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENAMETOOLONG,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EHOSTDOWN,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EHOSTUNREACH,
INIT_DFTTS_ENGINE_ERROR_SOCKET_ENOTEMPTY,
INIT_DFTTS_ENGINE_ERROR_SOCKET_EPROCLIM,
INIT_DFTTS_ENGINE_ERROR_THREADSTARTED,
INIT_DFTTS_ENGINE_ERROR_THREADNOTSTARTED,
INIT_DFTTS_ENGINE_ERROR_THREADCOULDNOTCREATE,
INIT_DFTTS_ENGINE_ERROR_BADNVFILE,
INIT_DFTTS_ENGINE_ERROR_NOAUDIODRIVER,
INIT_DFTTS_ENGINE_ERROR_DICTNOTFOUND,
[INIT_DFTTS_ENGINE_ERROR_OTHER] Other errors

See Also
InitDFTTSEngineEx()
UninitDFTTSEngine()

Example (see example for InitDFTTSEngineEx)

UninitDFTTSEngine
Unloads the synthesizer's DB

Synopsis
#include "dftts.h"
```
UninitDFTTSEngineReturnValue UninitDFTTSEngine();
```

Parameters
None.

Description
The function frees the assigned memory by unloading the synthesizer DB's and the internal speech objects.

It must be called before the program exits otherwise memory leaks and file holds will occur!

Return Values
```
UNINIT_DFTTS_ENGINE_SUCCESS,
UNINIT_DFTTS_ENGINE_FAIL
```

See Also
InitDFTTSEngine()

Example
UninitDFTTSEngine();

LoadDFTTSUserDict
Loads the User Dictionary.

Synopsis
#include "dftts.h"
```
LoadDFTTSUserDictReturnValue LoadDFTTSUserDict( int iDictIndex,
char* szDictName,
char* szDictFileName,
DFTTSVoiceEngineType vet,
unsigned short lang,
char* szVoiceName,
char* szDictContents );
```

Parameters
iDictIndex
Dictionary index in case of using more than one user dictionary (only valid for NEOSPEECHVOICETEXT).
Default user dictionary uses the value of 0 and it can take the values between 1~1023.

szDictName
Dictionary name (only valid with engines MSSAPI and ATTNV (internal use only, use MSSAPI instead)).

szDictFileName
The name (path) of user dictionary file (only valid for NEOSPEECHVOICETEXT and CEPSTRAL).

vet
Load for the specified engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV or MSSAPI).

lang
Language id (only valid for NEOSPEECHVOICETEXT and MSSAPI).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)

SUBLANGID(lgid)

U.S. English language id is 1033.

szVoiceName
Voice to load the dictionary for (only valid for CEPSTRAL, where the dictionary is loaded for a voice)

szDictContents
Phoneme contents for the dictionary (only ATTNV and MSSAPI where instead of a dictionary file we supply the actual character contents). See dictionary system references provided in folder DictionarySystems.

Description

This function loads the user dictionaries that are separate from and in addition to the default dictionary that is included in the TTS DB.

This function must be used after the loading of the synthesizer DB's. iDictIndex is used during synthesis by NEOSPEECHVOICETEXT, so do not forget to specify it.

The dictionaries are sent natively to the provider or in the case of MSSAPI the dictionary system has been emulated to be consistent with the user dictionary standard of the SDK.

The loaded SDK user dictionary is always unloaded on program exit.

User dictionaries are not supported with the MACOSXSPMAN engine type.

Note: For detailed instructions on how to build dictionaries for each engine provider, see instructions provided with the SDK in folder DictionarySystems.

Return Values

LOAD_USER_DICT_SUCCESS is returned when user dictionary is successfully loaded. The following error codes are returned when errors occur: (refer to dfttsval.h)

[LOAD_USER_DICT_ERROR_DICTIDX_NOT_VALID] iDictIndex value is not within the valid range
[LOAD_USER_DICT_ERROR_DICT_ALREADY_LOADED] User dictionary file corresponding to *dictidx* is already loaded.
[LOAD_USER_DICT_ERROR_NO_DICT_FILE_OR_ENTRY] Loading failed because there was no user dictionary files or valid entry

Other common return values (see dfttsval.h for all return values):

LOAD_USER_DICT_ERROR_INVALID_DICT_FILE,
LOAD_USER_DICT_ERROR_INVALID_VOICE,
LOAD_USER_DICT_ERROR_INVALID_PHONEME_SET,
LOAD_USER_DICT_ERROR_INVALID_PHONEME,
.
.
.
[LOAD_USER_DICT_ERROR_OTHER] Other errors

See Also
UnloadDFTTSUserDict()

Example

```
/*This is how to load a dictionary for the NeoSpeech provider (the dictionary is
loaded
by language (the dict name, voice name, dict content do NOT matter), the
dictionary number matters)*/

LoadDFTTSUserDictReturnValue dictreturn = LoadDFTTSUserDict( 1, NULL,
                  "neospeech_userdict_eng.csv",
                  NEOSPEECHVOICETEXT, 1033, NULL, NULL );


//This is how to load a dictionary for the Cepstral provider (the dictionary
is //loaded
//by voice, the dictionary number, dict name, dict content do NOT matter.)

LoadDFTTSUserDictReturnValue dictreturn = LoadDFTTSUserDict( -1, NULL,
                  "cepstral_dictionary.txt",
                  CEPSTRAL, 0, "David", NULL );
```

UnloadDFTTSUserDict
Unloads the User Dictionary.

Synopsis
#include "dftts.h"

```
UnloadDFTTSUserDictReturnValue UnloadDFTTSUserDict( int iDictIndex,
      const char* szDictName,
      DFTTSVoiceEngineType vet,
      unsigned short lang
      );
```

Parameters
iDictIndex
Dictionary index in case of using more than one user dictionary (only valid for NEOSPEECHVOICETEXT).
Default user dictionary uses the value of 0 and it can take the values between 1~1023.

szDictName
Dictionary name (only valid with engine MSSAPI and ATTNV (internal use only, use MSSAPI instead)).

vet
Unload for the specified engine type (NEOSPEECHVOICETEXT, ATTNV or MSSAPI. CEPSTRAL does not support dictionary unloading. MACOSXSPMAN does not support user dictionaries through the SDK.).

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

Description

The function unloads the user dictionary.

Any user dictionary that has not been unloaded is automatically unloaded during the process of unloading the SDK.

MACOSXSPMAN does not support user dictionaries through the SDK.

CEPSTRAL does not support dictionary unloading. The dictionary data unloads with unloading the SDK.

Return Values

UNLOAD_USER_DICT_SUCCESS is returned when user dictionary is successfully loaded. The following common error codes are returned when errors occur: (refer to dfttsval.h for all error codes)

[UNLOAD_USER_DICT_ERROR_DICTIDX_NOT_VALID] iDictIndex value is not within the valid range
[UNLOAD_USER_DICT_ERROR_DICT_UNLOADED] User dictionary file corresponding to iDictIndex is already unloaded.
.
.
.
[UNLOAD_USER_DICT_ERROR_OTHER] Other errors

See Also

LoadDFTTSUserDict()

Example

```
if (UnloadDFTTSUserDict(1,"",NEOSPEECHVOICETEXT,1033) != UNLOAD_USER_DICT_SUCCESS)
return -1;
```

SetDFTTSOnWordCallBack

Sets the callback function to receive the onWord event.

Synopsis

#include "dftts.h"

```
typedef void DFTTSOnWordCallBack( int start, int end );

void SetDFTTSOnWordCallBack( DFTTSOnWordCallBack* DFTTSOnWordCallBackFunc );
```

Parameters

DFTTSOnWordCallBackFunc

A pointer to the callback function of type `DFTTSOnWordCallBack` to receive the event data. The function declaration must comply with the typedef.

Description

Sets the callback function to receive the onWord event with the start and end character being spoken.

Notes

On Windows, *DFTTSSpeak() SDK function* sends the message WM_USER+12359 with the beginning character number of the word being spoken (zero-based and in regard to the whole text fed to the engine) in WPARAM and the end character number in LPARAM. The SDK sends WM_USER+12359 with WPARAM "0" and LPARAM "-1" when synthesizing is complete. To test this behavior use Spy++. You can capture the message directly in the window procedure when using Microsoft ® Windows.

These values are acquired on all platforms (Windows, Mac OS X and Linux) by using a callback function of type `DFTTSOnWordCallBack`. Therefore, the callback is the recommended way.

Return Values

None.

See Also

SetDFTTSOnExportCallBack ()

Example

```
SetDFTTSOnWordCallBack( callBackOnWord );

VOID callBackOnWord( int start, int end )
{

	char szMsg[256];

	sprintf( szMsg, "Start character: %d. End character: %d.\n", start, end );

	printf( szMsg );

}
```

SetDFTTSOnExportCallBack

Sets the callback function to receive the onExport event (when an audio file is being generated).

Synopsis

#include "dftts.h"

```
typedef void DFTTSOnExportCallBack( int start, int end );

void SetDFTTSOnExportCallBack(DFTTSOnExportCallBack* DFTTSOnExportCallBackFunc);
```

Parameters

DFTTSOnExportCallBackFunc

A pointer to the callback function of type `DFTTSOnExportCallBack` to receive the event data. The function declaration must comply with the typedef.

Description

Sets the callback function to receive the onExport event (when an audio file is being generated).

Notes

The only currently supported event behavior is that the callback function receives start as 0 (zero) and end as -1 (minus one) when the audio file generation has ended. No values of start and end are sent during the generation process at this time. In fact, the onExport event should be called onExportFinished at this time.

On Windows, *DFTTSExportToFile...() SDK function* sends the message WM_USER+12360 with WPARAM "0" and LPARAM "-1" when audio file synthesizing is complete. To test this behavior use Spy++.

On Mac OS X, when using provider MACOSXSPMAN (Mac OS X Speech Manager) the export event is considered a speak event and you must use a function of type `DFTTSOnWordCallBack` with `SetDFTTSOnWordCallBack` to capture the export events.

The values are acquired on all platforms (see above regarding MACOSXSPMAN) by using a callback function of type `DFTTSOnExportCallBack`. Therefore, the callback is the recommended way.

Return Values

None.

See Also

SetDFTTSOnWordCallBack ()

Example

```
SetDFTTSOnExportCallBack( callBackOnExport );

VOID callBackOnExport( int start, int end )
{
      printf( "Export successful!\n" );
}
```

DFTTSSpeak

It plays synthesized TTS output through a sound card.

Synopsis

```
#include "dftts.h"
DFTTSSpeakReturnValue DFTTSSpeak( HWND hwndWinOwner,
      DFTTSVoiceEngineType vet,
      const char* szVoiceName,
      nt iVoiceID,
      unsigned short lang,
      char* szText,
      int iPitch,
      int iSpeed,
      int iVolume,
      int iPause,
      int iDictID,
      DFTTSTextType ttTextType,
      short ofOutPutFormat
      );
```

Parameters

hwndWinOwner
Window handle (WIN32) where to send speech event messages

vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

szVoiceName
Internal voice name (matters for CEPSTRAL, ATTNV, MACOSXSPMAN and MSSAPI). Example: "David" or "Mike16".

iVoiceID
Internal voice id (matters for NEOSPEECHVOICETEXT).

Use the following constants (NOT specified in dfttsval.h):

```
#define NEOSPEECH_KATE_ENG 0

#define NEOSPEECH_PAUL_ENG 1

#define NEOSPEECH_MIYU_JPN 0

#define NEOSPEECH_SHOW_JPN 1

#define NEOSPEECH_MISAKI_JPN 2

#define NEOSPEECH_LILY_CHI 0

#define NEOSPEECH_WANG_CHI 1

#define NEOSPEECH_JUNWOO_KOR 3

#define NEOSPEECH_SUJIN_KOR 8
```

```
#define NEOSPEECH_YUMI_KOR 10

#define NEOSPEECH_GYURI_KOR 11

#define NEOSPEECH_DAYOUNG_KOR 12
```

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

szText
Text string to be synthesized - last character must be NULL. (No size limit)

iPitch
Defines the pitch of synthesized voice.
The default value is set to 100(%). The possible pitch range varies depending on the engine type (see below). ATTNV does not support pitch modifications by design.
For –1, use default value.

iSpeed
Defines the speed of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iVolume
Defines the volume of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iPause
Defines the length of pause between sentences of synthesized voice (NEOSPEECHVOICETEXT ONLY). The default value is set to 670(msec). The range is 0~20000(msec) and the lower value indicates shorter pause.
For –1, use default value

The following enumeration outlines the default values and supported ranges between different engine providers (dfttsval.h):

```
typedef enum DFTTSPredefinedSpeechParams
{
	DF_TTS_DEFAULT_PITCH = 100, /*100%*/
	DF_TTS_DEFAULT_SPEED = 100, /*100%*/
	DF_TTS_DEFAULT_VOLUME = 100, /*100%*/
	DF_TTS_DEFAULT_PAUSE = 670, /*670 msec*/
	DF_TTS_NEOSPEECH_MIN_PITCH = 50,
	DF_TTS_NEOSPEECH_MAX_PITCH = 200,
	DF_TTS_NEOSPEECH_MIN_SPEED = 50,
	DF_TTS_NEOSPEECH_MAX_SPEED = 400,
	DF_TTS_NEOSPEECH_MIN_VOLUME = 0,
	DF_TTS_NEOSPEECH_MAX_VOLUME = 500,
	DF_TTS_NEOSPEECH_MIN_PAUSE = 0,
	DF_TTS_NEOSPEECH_MAX_PAUSE = 20000,
	DF_TTS_CEPSTRAL_MIN_PITCH = 100,
	DF_TTS_CEPSTRAL_MAX_PITCH = 500,
	DF_TTS_CEPSTRAL_MIN_SPEED = 0,
	DF_TTS_CEPSTRAL_MAX_SPEED = 400,
	DF_TTS_CEPSTRAL_MIN_VOLUME = 0,
	DF_TTS_CEPSTRAL_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_SPEED = 13,
	DF_TTS_ATTNV_MAX_SPEED = 800,
	DF_TTS_ATTNV_MIN_VOLUME = 0,
	DF_TTS_ATTNV_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_ATTNV_MAX_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_MACOSXSPMAN_MIN_PITCH = 1,
	DF_TTS_MACOSXSPMAN_MAX_PITCH = 1000,
	DF_TTS_MACOSXSPMAN_MIN_SPEED = 1,
	DF_TTS_MACOSXSPMAN_MAX_SPEED = 1000,
	DF_TTS_MACOSXSPMAN_MIN_VOLUME = 100,
	DF_TTS_MACOSXSPMAN_MAX_VOLUME = 500,
	DF_TTS_MSSAPI_MIN_PITCH = 30,
	DF_TTS_MSSAPI_MAX_PITCH = 350,
	DF_TTS_MSSAPI_MIN_SPEED = 30,
	DF_TTS_MSSAPI_MAX_SPEED = 350,
	DF_TTS_MSSAPI_MIN_VOLUME = 0,
	DF_TTS_MSSAPI_MAX_VOLUME = 100

} DFTTSPredefinedSpeechParams;
```

iDictID
ID of the user dictionary when multiple user dictionaries are in use. The default dictionary value is 0 and the range is between 1~1023 (ONLY applicable with NEOSPEECHVOICETEXT).
For –1, use default value.

ttTextType
Defines the text type to be synthesized.
For regular text, use DFTTS_TEXT_TYPE_PLAIN, and for VoiceXML/SSML text, use DFTTS_TEXT_TYPE_XML. The value of –1 is regarded as regular text.

ofOutPutFormat
Speech output format (MSSAPI only). If -1 uses the value of the previously set format even through the export function.

Therefore, with MSSAPI it is always recommended to set this value (see Windows samples).

Possible values are defined by the following enumeration (not included into dfttsval.h):

```
typedef enum SPSTREAMFORMAT
    {
      SPSF_Default      = -1,
      SPSF_NoAssignedFormat   = 0,
      SPSF_Text   = SPSF_NoAssignedFormat + 1,
      SPSF_NonStandardFormat  = SPSF_Text + 1,
      SPSF_ExtendedAudioFormat      = SPSF_NonStandardFormat + 1,
      SPSF_8kHz8BitMono = SPSF_ExtendedAudioFormat + 1,
      SPSF_8kHz8BitStereo     = SPSF_8kHz8BitMono + 1,
      SPSF_8kHz16BitMono      = SPSF_8kHz8BitStereo + 1,
      SPSF_8kHz16BitStereo    = SPSF_8kHz16BitMono + 1,
      SPSF_11kHz8BitMono      = SPSF_8kHz16BitStereo + 1,
      SPSF_11kHz8BitStereo    = SPSF_11kHz8BitMono + 1,
      SPSF_11kHz16BitMono     = SPSF_11kHz8BitStereo + 1,
      SPSF_11kHz16BitStereo   = SPSF_11kHz16BitMono + 1,
      SPSF_12kHz8BitMono      = SPSF_11kHz16BitStereo + 1,
      SPSF_12kHz8BitStereo    = SPSF_12kHz8BitMono + 1,
      SPSF_12kHz16BitMono     = SPSF_12kHz8BitStereo + 1,
      SPSF_12kHz16BitStereo   = SPSF_12kHz16BitMono + 1,
      SPSF_16kHz8BitMono      = SPSF_12kHz16BitStereo + 1,
      SPSF_16kHz8BitStereo    = SPSF_16kHz8BitMono + 1,
      SPSF_16kHz16BitMono     = SPSF_16kHz8BitStereo + 1,
      SPSF_16kHz16BitStereo   = SPSF_16kHz16BitMono + 1,
      SPSF_22kHz8BitMono      = SPSF_16kHz16BitStereo + 1,
      SPSF_22kHz8BitStereo    = SPSF_22kHz8BitMono + 1,
      SPSF_22kHz16BitMono     = SPSF_22kHz8BitStereo + 1,
      SPSF_22kHz16BitStereo   = SPSF_22kHz16BitMono + 1,
      SPSF_24kHz8BitMono      = SPSF_22kHz16BitStereo + 1,
      SPSF_24kHz8BitStereo    = SPSF_24kHz8BitMono + 1,
      SPSF_24kHz16BitMono     = SPSF_24kHz8BitStereo + 1,
      SPSF_24kHz16BitStereo   = SPSF_24kHz16BitMono + 1,
      SPSF_32kHz8BitMono      = SPSF_24kHz16BitStereo + 1,
      SPSF_32kHz8BitStereo    = SPSF_32kHz8BitMono + 1,
      SPSF_32kHz16BitMono     = SPSF_32kHz8BitStereo + 1,
      SPSF_32kHz16BitStereo   = SPSF_32kHz16BitMono + 1,
      SPSF_44kHz8BitMono      = SPSF_32kHz16BitStereo + 1,
      SPSF_44kHz8BitStereo    = SPSF_44kHz8BitMono + 1,
      SPSF_44kHz16BitMono     = SPSF_44kHz8BitStereo + 1,
      SPSF_44kHz16BitStereo   = SPSF_44kHz16BitMono + 1,
      SPSF_48kHz8BitMono      = SPSF_44kHz16BitStereo + 1,
      SPSF_48kHz8BitStereo    = SPSF_48kHz8BitMono + 1,
      SPSF_48kHz16BitMono     = SPSF_48kHz8BitStereo + 1,
      SPSF_48kHz16BitStereo   = SPSF_48kHz16BitMono + 1,
      SPSF_TrueSpeech_8kHz1BitMono  = SPSF_48kHz16BitStereo + 1,
      SPSF_CCITT_ALaw_8kHzMono      = SPSF_TrueSpeech_8kHz1BitMono + 1,
      SPSF_CCITT_ALaw_8kHzStereo    = SPSF_CCITT_ALaw_8kHzMono + 1,
      SPSF_CCITT_ALaw_11kHzMono     = SPSF_CCITT_ALaw_8kHzStereo + 1,
```

```
        SPSF_CCITT_ALaw_11kHzStereo   = SPSF_CCITT_ALaw_11kHzMono + 1,
        SPSF_CCITT_ALaw_22kHzMono     = SPSF_CCITT_ALaw_11kHzStereo + 1,
        SPSF_CCITT_ALaw_22kHzStereo   = SPSF_CCITT_ALaw_22kHzMono + 1,
        SPSF_CCITT_ALaw_44kHzMono     = SPSF_CCITT_ALaw_22kHzStereo + 1,
        SPSF_CCITT_ALaw_44kHzStereo   = SPSF_CCITT_ALaw_44kHzMono + 1,
        SPSF_CCITT_uLaw_8kHzMono      = SPSF_CCITT_ALaw_44kHzStereo + 1,
        SPSF_CCITT_uLaw_8kHzStereo    = SPSF_CCITT_uLaw_8kHzMono + 1,
        SPSF_CCITT_uLaw_11kHzMono     = SPSF_CCITT_uLaw_8kHzStereo + 1,
        SPSF_CCITT_uLaw_11kHzStereo   = SPSF_CCITT_uLaw_11kHzMono + 1,
        SPSF_CCITT_uLaw_22kHzMono     = SPSF_CCITT_uLaw_11kHzStereo + 1,
        SPSF_CCITT_uLaw_22kHzStereo   = SPSF_CCITT_uLaw_22kHzMono + 1,
        SPSF_CCITT_uLaw_44kHzMono     = SPSF_CCITT_uLaw_22kHzStereo + 1,
        SPSF_CCITT_uLaw_44kHzStereo   = SPSF_CCITT_uLaw_44kHzMono + 1,
        SPSF_ADPCM_8kHzMono     = SPSF_CCITT_uLaw_44kHzStereo + 1,
        SPSF_ADPCM_8kHzStereo   = SPSF_ADPCM_8kHzMono + 1,
        SPSF_ADPCM_11kHzMono    = SPSF_ADPCM_8kHzStereo + 1,
        SPSF_ADPCM_11kHzStereo  = SPSF_ADPCM_11kHzMono + 1,
        SPSF_ADPCM_22kHzMono    = SPSF_ADPCM_11kHzStereo + 1,
        SPSF_ADPCM_22kHzStereo  = SPSF_ADPCM_22kHzMono + 1,
        SPSF_ADPCM_44kHzMono    = SPSF_ADPCM_22kHzStereo + 1,
        SPSF_ADPCM_44kHzStereo  = SPSF_ADPCM_44kHzMono + 1,
        SPSF_GSM610_8kHzMono    = SPSF_ADPCM_44kHzStereo + 1,
        SPSF_GSM610_11kHzMono   = SPSF_GSM610_8kHzMono + 1,
        SPSF_GSM610_22kHzMono   = SPSF_GSM610_11kHzMono + 1,
        SPSF_GSM610_44kHzMono   = SPSF_GSM610_22kHzMono + 1,
        SPSF_NUM_FORMATS  = SPSF_GSM610_44kHzMono + 1
}       SPSTREAMFORMAT;
```

Notes

The function synthesizes text entries and produces the TTS output through a sound card. **On Windows, it sends the message WM_USER+12359 with the beginning character number of the word being spoken (zero-based and in regard to the whole text fed to the engine) in WPARAM and the end character number in LPARAM. The SDK sends WM_USER+12359 with WPARAM "0" and LPARAM "-1" when synthesizing is complete. To test this behavior use Spy++.** Check the sample code to see how these values are acquired by using the callback function onWord (see also `SetDFTTSOnWordCallBack`). You can also capture the message directly in the window procedure.

If DFTTSSpeak() is called again during playback, it stops what was being played for the specified engine provider and plays the new TTS output that was requested. This behavior may not be supported with all engines.

Return Values

DFTTS_SPEAK_SUCCESS is returned when it was executed successfully. The following error codes are returned when errors occur: (refer to dfttsval.h)

[DFTTS_SPEAK_ERROR_CHANNEL_MEM_FAIL] Failed to secure channel memory
[DFTTS_SPEAK_ERROR_TEXT_NULL] The text string is a NULL pointer
[DFTTS_SPEAK_ERROR_TEXT_ZERO_LEN] The length of text string is 0
[DFTTS_SPEAK_ERROR_DB_NOT_LOADED] The TTS DB of the voice requested is not loaded
[DFTTS_SPEAK_ERROR_SET_SOUND_CARD_FAIL] Failed to set the sound card

Additional return values:

DFTTS_SPEAK_ERROR_UNIMPLEMENTED,
DFTTS_SPEAK_ERROR_INTERNAL,
DFTTS_SPEAK_ERROR_INVALID_PARAM,
DFTTS_SPEAK_ERROR_INVALID_POINTER,
DFTTS_SPEAK_ERROR_OBJECT_NOT_FOUND,
DFTTS_SPEAK_ERROR_UNKNOWN_ENCODING,
DFTTS_SPEAK_ERROR_INTERRUPTED,
DFTTS_SPEAK_ERROR_INVALID_VOICE,
DFTTS_SPEAK_ERROR_WRONG_EVENT,
DFTTS_SPEAK_ERROR_ENGINE_INUSE,
DFTTS_SPEAK_ERROR_NETWORK_ERROR,
DFTTS_SPEAK_ERROR_INVALID_KEY,
DFTTS_SPEAK_ERROR_QUEUE_FULL,
DFTTS_SPEAK_ERROR_TOKEN_TIMEOUT,
DFTTS_SPEAK_ERROR_FAILED,
DFTTS_SPEAK_ERROR_INVALIDARG,
DFTTS_SPEAK_ERROR_OUTOFMEMORY,
DFTTS_SPEAK_ERROR_NOTIMPL,
DFTTS_SPEAK_ERROR_ABORT,
DFTTS_SPEAK_ERROR_UNKNOWN,
DFTTS_SPEAK_ERROR_BADHANDLE,
DFTTS_SPEAK_ERROR_EXCEPTION,
DFTTS_SPEAK_ERROR_EMPTY,
DFTTS_SPEAK_ERROR_FULL,
DFTTS_SPEAK_ERROR_INVALIDSTATE,
DFTTS_SPEAK_ERROR_BADVERSION,
DFTTS_SPEAK_ERROR_INSUFFICIENT_BUFFER,
DFTTS_SPEAK_ERROR_UNSUPPORTED,
DFTTS_SPEAK_ERROR_NOLICENSE,
DFTTS_SPEAK_ERROR_CREATECHILDPROCESS_FAILED,
DFTTS_SPEAK_ERROR_NOENVIRONMENTPATH,
DFTTS_SPEAK_ERROR_TIMEOUT,
DFTTS_SPEAK_ERROR_OUTOFRESOURCES,
DFTTS_SPEAK_ERROR_NOVOICES,
DFTTS_SPEAK_ERROR_CREATEFAIL,
DFTTS_SPEAK_ERROR_CONNECTFAIL,
DFTTS_SPEAK_ERROR_BINDFAIL,
DFTTS_SPEAK_ERROR_LISTENFAIL,
DFTTS_SPEAK_ERROR_CONNECTIONCLOSED,
DFTTS_SPEAK_ERROR_ACCEPTFAIL,
DFTTS_SPEAK_ERROR_SOCKETTIMEOUT,
DFTTS_SPEAK_ERROR_SOCKETERROR,
DFTTS_SPEAK_ERROR_NOMORESERVERS,
DFTTS_SPEAK_ERROR_SOCKET_EWOULDBLOCK,
DFTTS_SPEAK_ERROR_SOCKET_EINPROGRESS,
DFTTS_SPEAK_ERROR_SOCKET_EALREADY,
DFTTS_SPEAK_ERROR_SOCKET_ENOTSOCK,
DFTTS_SPEAK_ERROR_SOCKET_EDESTADDRREQ,
DFTTS_SPEAK_ERROR_SOCKET_EMSGSIZE,
DFTTS_SPEAK_ERROR_SOCKET_EPROTOTYPE,
DFTTS_SPEAK_ERROR_SOCKET_ENOPROTOOPT,

DFTTS_SPEAK_ERROR_SOCKET_EPROTONOSUPPORT,
DFTTS_SPEAK_ERROR_SOCKET_ESOCKTNOSUPPORT,
DFTTS_SPEAK_ERROR_SOCKET_EOPNOTSUPP,
DFTTS_SPEAK_ERROR_SOCKET_EPFNOSUPPORT,
DFTTS_SPEAK_ERROR_SOCKET_EAFNOSUPPORT,
DFTTS_SPEAK_ERROR_SOCKET_EADDRINUSE,
DFTTS_SPEAK_ERROR_SOCKET_EADDRNOTAVAIL,
DFTTS_SPEAK_ERROR_SOCKET_ENETDOWN,
DFTTS_SPEAK_ERROR_SOCKET_ENETUNREACH,
DFTTS_SPEAK_ERROR_SOCKET_ENETRESET,
DFTTS_SPEAK_ERROR_SOCKET_ECONNABORTED,
DFTTS_SPEAK_ERROR_SOCKET_ECONNRESET,
DFTTS_SPEAK_ERROR_SOCKET_ENOBUFS,
DFTTS_SPEAK_ERROR_SOCKET_EISCONN,
DFTTS_SPEAK_ERROR_SOCKET_ENOTCONN,
DFTTS_SPEAK_ERROR_SOCKET_ESHUTDOWN,
DFTTS_SPEAK_ERROR_SOCKET_ETOOMANYREFS,
DFTTS_SPEAK_ERROR_SOCKET_ECONNREFUSED,
DFTTS_SPEAK_ERROR_SOCKET_ELOOP,
DFTTS_SPEAK_ERROR_SOCKET_ENAMETOOLONG,
DFTTS_SPEAK_ERROR_SOCKET_EHOSTDOWN,
DFTTS_SPEAK_ERROR_SOCKET_EHOSTUNREACH,
DFTTS_SPEAK_ERROR_SOCKET_ENOTEMPTY,
DFTTS_SPEAK_ERROR_SOCKET_EPROCLIM,
DFTTS_SPEAK_ERROR_THREADSTARTED,
DFTTS_SPEAK_ERROR_THREADNOTSTARTED,
DFTTS_SPEAK_ERROR_THREADCOULDNOTCREATE,
DFTTS_SPEAK_ERROR_BADNVFILE,
DFTTS_SPEAK_ERROR_NOAUDIODRIVER,
DFTTS_SPEAK_ERROR_DICTNOTFOUND,
DFTTS_SPEAK_ERROR_ALREADYPLAYING,
DFTTS_SPEAK_ERROR_AUDIOFORMATNOTSUPPORTED,
DFTTS_SPEAK_ERROR_XML_INVALID,
DFTTS_SPEAK_ERROR_WL_INVALID,
DFTTS_SPEAK_ERROR_INVALIDPHONESET,
DFTTS_SPEAK_ERROR_INVALIDPHONEME,
DFTTS_SPEAK_ERROR_MSGQ_CREATEFAILED,
DFTTS_SPEAK_ERROR_MSGQ_ALREADYEXISTS,
DFTTS_SPEAK_ERROR_MSGQ_NOTFOUND,
DFTTS_SPEAK_ERROR_MSGQ_INVALIDOP,
DFTTS_SPEAK_ERROR_MSGQ_NOTOPEN,
DFTTS_SPEAK_ERROR_MSGQ_LOCKFAILED,
DFTTS_SPEAK_ERROR_MSGQ_ABANDONED,
DFTTS_SPEAK_ERROR_MSGQ_OPENFAILED,

[DFTTS_SPEAK_ERROR_OTHER] Other errors

DFTTSStop
It stops the playback of synthesized voices from a sound card.

Synopsis
```
#include "dftts.h"
DFTTSStopReturnValue DFTTSStop( DFTTSVoiceEngineType vet,
                        unsigned short lang = LANG_SUBLANG_NEUTRAL
                              );
```

Parameters
vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

Notes
Using DFTTSSpeak(), stops the playback of synthesized voice before synthesizing for the indicated engine provider.

Return Values
DFTTS_STOP_SUCCESS,
DFTTS_STOP_ERROR_INVALID_POINTER,
DFTTS_STOP_ERROR_INVALID_ARG,
DFTTS_STOP_ERROR_OUT_OF_MEMORY,
DFTTS_STOP_ERROR_INVALID_FLAGS,
DFTTS_STOP_ERROR_ENGINE_INUSE,
DFTTS_STOP_ERROR_OTHER

DFTTSPause
Pauses the playback of synthesized voices from a sound card.

Synopsis
```
#include "dftts.h"
DFTTSPauseReturnValue DFTTSPause( DFTTSVoiceEngineType vet,
                                  unsigned short lang = LANG_SUBLANG_NEUTRAL
                                  );
```

Parameters
vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

Description
Pauses the playback of synthesized voice that is carried out using DFTTSSpeak().

Return Values
DFTTS_PAUSE_SUCCESS,
DFTTS_PAUSE_ERROR_UNIMPLEMENTED,
DFTTS_PAUSE_ERROR_INTERNAL,
DFTTS_PAUSE_ERROR_INVALID_PARAM,
DFTTS_PAUSE_ERROR_INVALID_POINTER,
DFTTS_PAUSE_ERROR_OBJECT_NOT_FOUND,
DFTTS_PAUSE_ERROR_UNKNOWN_ENCODING,
DFTTS_PAUSE_ERROR_INTERRUPTED,
DFTTS_PAUSE_ERROR_INVALID_VOICE,
DFTTS_PAUSE_ERROR_WRONG_EVENT,
DFTTS_PAUSE_ERROR_ENGINE_INUSE,
DFTTS_PAUSE_ERROR_NETWORK_ERROR,
DFTTS_PAUSE_ERROR_INVALID_KEY,

DFTTS_PAUSE_ERROR_QUEUE_FULL,
DFTTS_PAUSE_ERROR_TOKEN_TIMEOUT,
DFTTS_PAUSE_ERROR_OTHER

DFTTSResume

Resumes the playback of the synthesized voice from a sound card.

Synopsis

#include "dftts.h"

```
DFTTSResumeReturnValue DFTTSResume( DFTTSVoiceEngineType vet,
                                    unsigned short lang = LANG_SUBLANG_NEUTRAL
                                    );
```

Parameters

vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

Description

Resumes the playback of synthesized voice that was paused using DFTTSPause().

Return Values

DFTTS_RESUME_SUCCESS if succeeded (check `enum DFTTSResumeReturnValue` in dfttsval.h for the list of all error codes)

DFTTSExportToFile

It saves the synthesized output as a file.

Synopsis

```
#include "dftts.h"
DFTTSExportReturnValue DFTTSExportToFile( DFTTSVoiceEngineType vet,
      const char* szVoiceName,
      int iVoiceID,
      unsigned short lang,
      char* szText,
      int iPitch,
      int iSpeed,
      int iVolume,
      int iPause,
      int iDictID,
      DFTTSTextType ttTextType,
      const char* szFilePath,
      void* FileFormat
      );
```

Parameters

vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

szVoiceName
Internal voice name (matters for CEPSTRAL, ATTNV, MACOSXSPMAN and MSSAPI). Example: "David" or "Mike16".

iVoiceID
Internal voice id (matters for NEOSPEECHVOICETEXT).

Use the following constants (NOT specified in dfttsval.h):

```
#define NEOSPEECH_KATE_ENG 0

#define NEOSPEECH_PAUL_ENG 1

#define NEOSPEECH_MIYU_JPN 0

#define NEOSPEECH_SHOW_JPN 1

#define NEOSPEECH_MISAKI_JPN 2

#define NEOSPEECH_LILY_CHI 0

#define NEOSPEECH_WANG_CHI 1

#define NEOSPEECH_JUNWOO_KOR 3

#define NEOSPEECH_SUJIN_KOR 8

#define NEOSPEECH_YUMI_KOR 10
```

```
#define NEOSPEECH_GYURI_KOR 11

#define NEOSPEECH_DAYOUNG_KOR 12
```

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

szText
Text string to be synthesized - last character must be NULL (no size limit).

iPitch
Defines the pitch of synthesized voice.
The default value is set to 100(%). The possible pitch range varies depending on the engine type (see below). ATTNV does not support pitch modifications by design.
For –1, use default value.

iSpeed
Defines the speed of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iVolume
Defines the volume of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iPause
Defines the length of pause between sentences of synthesized voice (NEOSPEECHVOICETEXT ONLY). The default value is set to 670(msec). The range is 0~20000(msec) and the lower value indicates shorter pause.
For –1, use default value

The following enumeration outlines the default values and supported ranges between different engine providers (dfttsval.h):

```
typedef enum DFTTSPredefinedSpeechParams
{
	DF_TTS_DEFAULT_PITCH = 100, /*100%*/
	DF_TTS_DEFAULT_SPEED = 100, /*100%*/
	DF_TTS_DEFAULT_VOLUME = 100, /*100%*/
	DF_TTS_DEFAULT_PAUSE = 670, /*670 msec*/
	DF_TTS_NEOSPEECH_MIN_PITCH = 50,
	DF_TTS_NEOSPEECH_MAX_PITCH = 200,
	DF_TTS_NEOSPEECH_MIN_SPEED = 50,
	DF_TTS_NEOSPEECH_MAX_SPEED = 400,
	DF_TTS_NEOSPEECH_MIN_VOLUME = 0,
	DF_TTS_NEOSPEECH_MAX_VOLUME = 500,
	DF_TTS_NEOSPEECH_MIN_PAUSE = 0,
	DF_TTS_NEOSPEECH_MAX_PAUSE = 20000,
	DF_TTS_CEPSTRAL_MIN_PITCH = 100,
	DF_TTS_CEPSTRAL_MAX_PITCH = 500,
	DF_TTS_CEPSTRAL_MIN_SPEED = 0,
	DF_TTS_CEPSTRAL_MAX_SPEED = 400,
	DF_TTS_CEPSTRAL_MIN_VOLUME = 0,
	DF_TTS_CEPSTRAL_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_SPEED = 13,
	DF_TTS_ATTNV_MAX_SPEED = 800,
	DF_TTS_ATTNV_MIN_VOLUME = 0,
	DF_TTS_ATTNV_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_ATTNV_MAX_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_MACOSXSPMAN_MIN_PITCH = 1,
	DF_TTS_MACOSXSPMAN_MAX_PITCH = 1000,
	DF_TTS_MACOSXSPMAN_MIN_SPEED = 1,
	DF_TTS_MACOSXSPMAN_MAX_SPEED = 1000,
	DF_TTS_MACOSXSPMAN_MIN_VOLUME = 100,
	DF_TTS_MACOSXSPMAN_MAX_VOLUME = 500,
	DF_TTS_MSSAPI_MIN_PITCH = 30,
	DF_TTS_MSSAPI_MAX_PITCH = 350,
	DF_TTS_MSSAPI_MIN_SPEED = 30,
	DF_TTS_MSSAPI_MAX_SPEED = 350,
	DF_TTS_MSSAPI_MIN_VOLUME = 0,
	DF_TTS_MSSAPI_MAX_VOLUME = 100

} DFTTSPredefinedSpeechParams;
```

iDictID
ID of the user dictionary when multiple user dictionaries are in use. The default user's uses value 0 and the range is between 1~1023 (ONLY applicable with NEOSPEECHVOICETEXT). For −1, use default value.

ttTextType
Defines the text type to be synthesized.
For regular text, use DFTTS_TEXT_TYPE_PLAIN, and for VoiceXML/SSML text, use DFTTS_TEXT_TYPE_XML. The value of −1 is regarded as regular text.

szFilePath
File path to save the synthesized voice output under.

FileFormat
Defines the types of synthesized output formats. Void pointer.

NEOSPEECHVOICETEXT: The following are the types of synthesized output file format that the DF TTS SDK supports for NeoSpeech VoiceText™:

VT_FILE_API_FMT_S16PCM 16bits Linear PCM
VT_FILE_API_FMT_ALAW 8bits A-law PCM
VT_FILE_API_FMT_MULAW 8bits Mu-law PCM
VT_FILE_API_FMT_DADPCM 4bits Dialogic ADPCM
VT_FILE_API_FMT_S16PCM_WAVE 16bits Linear PCM WAVE
VT_FILE_API_FMT_U08PCM_WAVE 8bits Unsigned Linear PCM WAVE
VT_FILE_API_FMT_IMA_WAVE 4bits IMA ADPCM WAVE
VT_FILE_API_FMT_ALAW_WAVE 8bits A-law PCM WAVE
VT_FILE_API_FMT_MULAW_WAVE 8bits Mu-law PCM WAVE
VT_FILE_API_FMT_MULAW_AU 8bits Mu-law PCM SUN AU

CEPSTRAL: You MUST pass a pointer to DFTTSComplexExportData structure:
Cepstral voice engine MUST use this struct for the export (SDK v 2.0.0 and up)

```
struct DFTTSComplexExportData
{

	char* AudioEncoding;

	int AudioSamplingRate;

	int AudioChannels;

};
```

Possible member values:

szAudioEncoding:

"pcm16", "pcm8" PCM 16 bit/8 bit WAV
"ulaw" - μ-Law (8-bit), "alaw" - A-Law (8-bit)
"riff": Microsoft RIFF (WAV) file
"snd": Sun/NeXT .au (SND) format.
"raw": unheadered audio data, native byte order
"le": unheadered audio data, little-endian (LSB first)
"be": unheadered audio data, big-endian (MSB first)

iAudioSamplingRate:

8000 (8 KHz), 16000 (16 KHz), 11025 (11.025 kHz), etc.

iAudioChannels:

1 (mono), 2 (stereo)

ATTNV: Only PCM WAV encoding supported. All format settings will be ignored.

MACOSXSPMAN: Only AIFF encoding supported. All format settings will be ignored.

MSSAPI: Use the format settings specified by the following enumeration:

```
typedef enum SPSTREAMFORMAT
    {
      SPSF_Default      = -1,
      SPSF_NoAssignedFormat   = 0,
      SPSF_Text   = SPSF_NoAssignedFormat + 1,
      SPSF_NonStandardFormat  = SPSF_Text + 1,
      SPSF_ExtendedAudioFormat      = SPSF_NonStandardFormat + 1,
      SPSF_8kHz8BitMono = SPSF_ExtendedAudioFormat + 1,
      SPSF_8kHz8BitStereo     = SPSF_8kHz8BitMono + 1,
      SPSF_8kHz16BitMono      = SPSF_8kHz8BitStereo + 1,
      SPSF_8kHz16BitStereo    = SPSF_8kHz16BitMono + 1,
      SPSF_11kHz8BitMono      = SPSF_8kHz16BitStereo + 1,
      SPSF_11kHz8BitStereo    = SPSF_11kHz8BitMono + 1,
      SPSF_11kHz16BitMono     = SPSF_11kHz8BitStereo + 1,
      SPSF_11kHz16BitStereo   = SPSF_11kHz16BitMono + 1,
      SPSF_12kHz8BitMono      = SPSF_11kHz16BitStereo + 1,
      SPSF_12kHz8BitStereo    = SPSF_12kHz8BitMono + 1,
      SPSF_12kHz16BitMono     = SPSF_12kHz8BitStereo + 1,
      SPSF_12kHz16BitStereo   = SPSF_12kHz16BitMono + 1,
      SPSF_16kHz8BitMono      = SPSF_12kHz16BitStereo + 1,
      SPSF_16kHz8BitStereo    = SPSF_16kHz8BitMono + 1,
      SPSF_16kHz16BitMono     = SPSF_16kHz8BitStereo + 1,
      SPSF_16kHz16BitStereo   = SPSF_16kHz16BitMono + 1,
      SPSF_22kHz8BitMono      = SPSF_16kHz16BitStereo + 1,
      SPSF_22kHz8BitStereo    = SPSF_22kHz8BitMono + 1,
      SPSF_22kHz16BitMono     = SPSF_22kHz8BitStereo + 1,
      SPSF_22kHz16BitStereo   = SPSF_22kHz16BitMono + 1,
      SPSF_24kHz8BitMono      = SPSF_22kHz16BitStereo + 1,
      SPSF_24kHz8BitStereo    = SPSF_24kHz8BitMono + 1,
      SPSF_24kHz16BitMono     = SPSF_24kHz8BitStereo + 1,
      SPSF_24kHz16BitStereo   = SPSF_24kHz16BitMono + 1,
      SPSF_32kHz8BitMono      = SPSF_24kHz16BitStereo + 1,
      SPSF_32kHz8BitStereo    = SPSF_32kHz8BitMono + 1,
      SPSF_32kHz16BitMono     = SPSF_32kHz8BitStereo + 1,
      SPSF_32kHz16BitStereo   = SPSF_32kHz16BitMono + 1,
      SPSF_44kHz8BitMono      = SPSF_32kHz16BitStereo + 1,
      SPSF_44kHz8BitStereo    = SPSF_44kHz8BitMono + 1,
      SPSF_44kHz16BitMono     = SPSF_44kHz8BitStereo + 1,
      SPSF_44kHz16BitStereo   = SPSF_44kHz16BitMono + 1,
      SPSF_48kHz8BitMono      = SPSF_44kHz16BitStereo + 1,
      SPSF_48kHz8BitStereo    = SPSF_48kHz8BitMono + 1,
      SPSF_48kHz16BitMono     = SPSF_48kHz8BitStereo + 1,
      SPSF_48kHz16BitStereo   = SPSF_48kHz16BitMono + 1,
      SPSF_TrueSpeech_8kHz1BitMono  = SPSF_48kHz16BitStereo + 1,
      SPSF_CCITT_ALaw_8kHzMono      = SPSF_TrueSpeech_8kHz1BitMono + 1,
      SPSF_CCITT_ALaw_8kHzStereo    = SPSF_CCITT_ALaw_8kHzMono + 1,
      SPSF_CCITT_ALaw_11kHzMono     = SPSF_CCITT_ALaw_8kHzStereo + 1,
      SPSF_CCITT_ALaw_11kHzStereo   = SPSF_CCITT_ALaw_11kHzMono + 1,
      SPSF_CCITT_ALaw_22kHzMono     = SPSF_CCITT_ALaw_11kHzStereo + 1,
      SPSF_CCITT_ALaw_22kHzStereo   = SPSF_CCITT_ALaw_22kHzMono + 1,
      SPSF_CCITT_ALaw_44kHzMono     = SPSF_CCITT_ALaw_22kHzStereo + 1,
      SPSF_CCITT_ALaw_44kHzStereo   = SPSF_CCITT_ALaw_44kHzMono + 1,
      SPSF_CCITT_uLaw_8kHzMono      = SPSF_CCITT_ALaw_44kHzStereo + 1,
      SPSF_CCITT_uLaw_8kHzStereo    = SPSF_CCITT_uLaw_8kHzMono + 1,
```

```
        SPSF_CCITT_uLaw_11kHzMono     = SPSF_CCITT_uLaw_8kHzStereo + 1,
        SPSF_CCITT_uLaw_11kHzStereo   = SPSF_CCITT_uLaw_11kHzMono + 1,
        SPSF_CCITT_uLaw_22kHzMono     = SPSF_CCITT_uLaw_11kHzStereo + 1,
        SPSF_CCITT_uLaw_22kHzStereo   = SPSF_CCITT_uLaw_22kHzMono + 1,
        SPSF_CCITT_uLaw_44kHzMono     = SPSF_CCITT_uLaw_22kHzStereo + 1,
        SPSF_CCITT_uLaw_44kHzStereo   = SPSF_CCITT_uLaw_44kHzMono + 1,
        SPSF_ADPCM_8kHzMono     = SPSF_CCITT_uLaw_44kHzStereo + 1,
        SPSF_ADPCM_8kHzStereo   = SPSF_ADPCM_8kHzMono + 1,
        SPSF_ADPCM_11kHzMono    = SPSF_ADPCM_8kHzStereo + 1,
        SPSF_ADPCM_11kHzStereo  = SPSF_ADPCM_11kHzMono + 1,
        SPSF_ADPCM_22kHzMono    = SPSF_ADPCM_11kHzStereo + 1,
        SPSF_ADPCM_22kHzStereo  = SPSF_ADPCM_22kHzMono + 1,
        SPSF_ADPCM_44kHzMono    = SPSF_ADPCM_22kHzStereo + 1,
        SPSF_ADPCM_44kHzStereo  = SPSF_ADPCM_44kHzMono + 1,
        SPSF_GSM610_8kHzMono    = SPSF_ADPCM_44kHzStereo + 1,
        SPSF_GSM610_11kHzMono   = SPSF_GSM610_8kHzMono + 1,
        SPSF_GSM610_22kHzMono   = SPSF_GSM610_11kHzMono + 1,
        SPSF_GSM610_44kHzMono   = SPSF_GSM610_22kHzMono + 1,
        SPSF_NUM_FORMATS  = SPSF_GSM610_44kHzMono + 1
}       SPSTREAMFORMAT;
```

Return Values

DFTTS_EXPORT_SUCCESS is returned when successfully synthesized and the following error codes are returned when errors occur: (refer to dfttsval.h)

[DFTTS_EXPORT_ERROR_FORMAT_NOT_SUPPORTED] Used format that is not supported.
[DFTTS_EXPORT_ERROR_CHANNEL_MEM_FAIL] Failed to secure channel memory
[DFTTS_EXPORT_ERROR_TEXT_NULL] Text string is a NULL pointer
[DFTTS_EXPORT_ERROR_TEXT_ZERO_LEN] The length of text string is 0.
[DFTTS_EXPORT_ERROR_DB_NOT_LOADED] The TTS DB of the voice requested is not loaded
[DFTTS_EXPORT_ERROR_GEN_FILE_FAIL] Failed to generate the synthesized voice file

Additional return values:

DFTTS_EXPORT_ERROR_BUFFER_NULL,
DFTTS_EXPORT_ERROR_THREAD_IN_USE,
DFTTS_EXPORT_ERROR_UNIMPLEMENTED,
DFTTS_EXPORT_ERROR_INTERNAL,
DFTTS_EXPORT_ERROR_INVALID_PARAM,
DFTTS_EXPORT_ERROR_INVALID_POINTER,
DFTTS_EXPORT_ERROR_OBJECT_NOT_FOUND,
DFTTS_EXPORT_ERROR_UNKNOWN_ENCODING,
DFTTS_EXPORT_ERROR_INTERRUPTED,
DFTTS_EXPORT_ERROR_INVALID_VOICE,
DFTTS_EXPORT_ERROR_WRONG_EVENT,
DFTTS_EXPORT_ERROR_ENGINE_INUSE,
DFTTS_EXPORT_ERROR_NETWORK_ERROR,
DFTTS_EXPORT_ERROR_INVALID_KEY,
DFTTS_EXPORT_ERROR_QUEUE_FULL,
DFTTS_EXPORT_ERROR_TOKEN_TIMEOUT,
DFTTS_EXPORT_ERROR_FAILED,
DFTTS_EXPORT_ERROR_INVALIDARG,
DFTTS_EXPORT_ERROR_OUTOFMEMORY,
DFTTS_EXPORT_ERROR_NOTIMPL,

DFTTS_EXPORT_ERROR_ABORT,
DFTTS_EXPORT_ERROR_UNKNOWN,
DFTTS_EXPORT_ERROR_BADHANDLE,
DFTTS_EXPORT_ERROR_EXCEPTION,
DFTTS_EXPORT_ERROR_EMPTY,
DFTTS_EXPORT_ERROR_FULL,
DFTTS_EXPORT_ERROR_INVALIDSTATE,
DFTTS_EXPORT_ERROR_BADVERSION,
DFTTS_EXPORT_ERROR_INSUFFICIENT_BUFFER,
DFTTS_EXPORT_ERROR_UNSUPPORTED,
DFTTS_EXPORT_ERROR_NOLICENSE,
DFTTS_EXPORT_ERROR_CREATECHILDPROCESS_FAILED,
DFTTS_EXPORT_ERROR_NOENVIRONMENTPATH,
DFTTS_EXPORT_ERROR_TIMEOUT,
DFTTS_EXPORT_ERROR_OUTOFRESOURCES,
DFTTS_EXPORT_ERROR_NOVOICES,
DFTTS_EXPORT_ERROR_CREATEFAIL,
DFTTS_EXPORT_ERROR_CONNECTFAIL,
DFTTS_EXPORT_ERROR_BINDFAIL,
DFTTS_EXPORT_ERROR_LISTENFAIL,
DFTTS_EXPORT_ERROR_CONNECTIONCLOSED,
DFTTS_EXPORT_ERROR_ACCEPTFAIL,
DFTTS_EXPORT_ERROR_SOCKETTIMEOUT,
DFTTS_EXPORT_ERROR_SOCKETERROR,
DFTTS_EXPORT_ERROR_NOMORESERVERS,
DFTTS_EXPORT_ERROR_SOCKET_EWOULDBLOCK,
DFTTS_EXPORT_ERROR_SOCKET_EINPROGRESS,
DFTTS_EXPORT_ERROR_SOCKET_EALREADY,
DFTTS_EXPORT_ERROR_SOCKET_ENOTSOCK,
DFTTS_EXPORT_ERROR_SOCKET_EDESTADDRREQ,
DFTTS_EXPORT_ERROR_SOCKET_EMSGSIZE,
DFTTS_EXPORT_ERROR_SOCKET_EPROTOTYPE,
DFTTS_EXPORT_ERROR_SOCKET_ENOPROTOOPT,
DFTTS_EXPORT_ERROR_SOCKET_EPROTONOSUPPORT,
DFTTS_EXPORT_ERROR_SOCKET_ESOCKTNOSUPPORT,
DFTTS_EXPORT_ERROR_SOCKET_EOPNOTSUPP,
DFTTS_EXPORT_ERROR_SOCKET_EPFNOSUPPORT,
DFTTS_EXPORT_ERROR_SOCKET_EAFNOSUPPORT,
DFTTS_EXPORT_ERROR_SOCKET_EADDRINUSE,
DFTTS_EXPORT_ERROR_SOCKET_EADDRNOTAVAIL,
DFTTS_EXPORT_ERROR_SOCKET_ENETDOWN,
DFTTS_EXPORT_ERROR_SOCKET_ENETUNREACH,
DFTTS_EXPORT_ERROR_SOCKET_ENETRESET,
DFTTS_EXPORT_ERROR_SOCKET_ECONNABORTED,
DFTTS_EXPORT_ERROR_SOCKET_ECONNRESET,
DFTTS_EXPORT_ERROR_SOCKET_ENOBUFS,
DFTTS_EXPORT_ERROR_SOCKET_EISCONN,
DFTTS_EXPORT_ERROR_SOCKET_ENOTCONN,
DFTTS_EXPORT_ERROR_SOCKET_ESHUTDOWN,
DFTTS_EXPORT_ERROR_SOCKET_ETOOMANYREFS,
DFTTS_EXPORT_ERROR_SOCKET_ECONNREFUSED,
DFTTS_EXPORT_ERROR_SOCKET_ELOOP,
DFTTS_EXPORT_ERROR_SOCKET_ENAMETOOLONG,

DFTTS_EXPORT_ERROR_SOCKET_EHOSTDOWN,
DFTTS_EXPORT_ERROR_SOCKET_EHOSTUNREACH,
DFTTS_EXPORT_ERROR_SOCKET_ENOTEMPTY,
DFTTS_EXPORT_ERROR_SOCKET_EPROCLIM,
DFTTS_EXPORT_ERROR_THREADSTARTED,
DFTTS_EXPORT_ERROR_THREADNOTSTARTED,
DFTTS_EXPORT_ERROR_THREADCOULDNOTCREATE,
DFTTS_EXPORT_ERROR_BADNVFILE,
DFTTS_EXPORT_ERROR_NOAUDIODRIVER,
DFTTS_EXPORT_ERROR_DICTNOTFOUND,
DFTTS_EXPORT_ERROR_ALREADYPLAYING,
DFTTS_EXPORT_ERROR_AUDIOFORMATNOTSUPPORTED,
DFTTS_EXPORT_ERROR_XML_INVALID,
DFTTS_EXPORT_ERROR_WL_INVALID,
DFTTS_EXPORT_ERROR_INVALIDPHONESET,
DFTTS_EXPORT_ERROR_INVALIDPHONEME,
DFTTS_EXPORT_ERROR_MSGQ_CREATEFAILED,
DFTTS_EXPORT_ERROR_MSGQ_ALREADYEXISTS,
DFTTS_EXPORT_ERROR_MSGQ_NOTFOUND,
DFTTS_EXPORT_ERROR_MSGQ_INVALIDOP,
DFTTS_EXPORT_ERROR_MSGQ_NOTOPEN,
DFTTS_EXPORT_ERROR_MSGQ_LOCKFAILED,
DFTTS_EXPORT_ERROR_MSGQ_ABANDONED,
DFTTS_EXPORT_ERROR_MSGQ_OPENFAILED,
[DFTTS_EXPORT_ERROR_OTHER] Other errors

See Also

DFTTSExportToFileEx()
InitDFTTSEngineEx()
GetDFTTSEngineInfo()

Example

```
int iFileFormat = VT_FILE_API_FMT_S16PCM_WAVE;

DFTTSComplexExportData ceFileFormat;

ceFileFormat.AudioEncoding = "pcm16";
ceFileFormat.AudioSamplingRate = 16000;
ceFileFormat.AudioChannels = 1;

int sapiFileFormat = SPSF_16kHz16BitStereo;

void* FileFormat = 0;

if( vet  == NEOSPEECHVOICETEXT )
{
      FileFormat = &iFileFormat;

}
else if( vet == CEPSTRAL )
{
      FileFormat = &ceFileFormat;
```

```
}
else if( vet == MSSAPI )
{
	FileFormat = &sapiFileFormat;

}

short result2 = DFTTSExportToFile( …
, FileFormat);
```

DFTTSExportToFileEx

It saves the synthesized output as a file. Alternative to DFTTSExportToFile.

Synopsis

```
#include "dftts.h"
DFTTSExportReturnValue DFTTSExportToFileEx( DFTTSVoiceEngineType vet,
	const char* szVoiceName,
	int iVoiceID,
	unsigned short lang,
	char* szText,
	int iPitch,
	int iSpeed,
	int iVolume,
	int iPause,
	int iDictID,
	DFTTSTextType ttTextType,
	const char* szFilePath,
	short ffFileFormat,
	const char* szAudioEncoding,
	int iAudioSamplingRate,
	int iAudioChannels
	);
```

Parameters

vet
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

szVoiceName
Internal voice name (matters for CEPSTRAL, ATTNV, MACOSXSPMAN and MSSAPI). Example: "David" or "Mike16".

iVoiceID
Internal voice id (matters for NEOSPEECHVOICETEXT).

Use the following constants (NOT specified in dfttsval.h):

```
#define NEOSPEECH_KATE_ENG 0

#define NEOSPEECH_PAUL_ENG 1

#define NEOSPEECH_MIYU_JPN 0

#define NEOSPEECH_SHOW_JPN 1
```

```
#define NEOSPEECH_MISAKI_JPN 2

#define NEOSPEECH_LILY_CHI 0

#define NEOSPEECH_WANG_CHI 1

#define NEOSPEECH_JUNWOO_KOR 3

#define NEOSPEECH_SUJIN_KOR 8

#define NEOSPEECH_YUMI_KOR 10

#define NEOSPEECH_GYURI_KOR 11

#define NEOSPEECH_DAYOUNG_KOR 12
```

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

szText
Text string to be synthesized - last character must be NULL (no size limit).

iPitch
Defines the pitch of synthesized voice.
The default value is set to 100(%). The possible pitch range varies depending on the engine type (see below). ATTNV does not support pitch modifications by design.
For –1, use default value.

iSpeed
Defines the speed of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iVolume

Defines the volume of synthesized voice. The default value is set to 100%. The range varies depending on the engine type (see below).
For –1, use default value.

iPause
Defines the length of pause between sentences of synthesized voice (NEOSPEECHVOICETEXT ONLY). The default value is set to 670(msec). The range is 0~20000(msec) and the lower value indicates shorter pause.
For –1, use default value

The following enumeration outlines the default values and supported ranges between different engine providers (dfttsval.h):

```
typedef enum DFTTSPredefinedSpeechParams
{
	DF_TTS_DEFAULT_PITCH = 100, /*100%*/
	DF_TTS_DEFAULT_SPEED = 100, /*100%*/
	DF_TTS_DEFAULT_VOLUME = 100, /*100%*/
	DF_TTS_DEFAULT_PAUSE = 670, /*670 msec*/
	DF_TTS_NEOSPEECH_MIN_PITCH = 50,
	DF_TTS_NEOSPEECH_MAX_PITCH = 200,
	DF_TTS_NEOSPEECH_MIN_SPEED = 50,
	DF_TTS_NEOSPEECH_MAX_SPEED = 400,
	DF_TTS_NEOSPEECH_MIN_VOLUME = 0,
	DF_TTS_NEOSPEECH_MAX_VOLUME = 500,
	DF_TTS_NEOSPEECH_MIN_PAUSE = 0,
	DF_TTS_NEOSPEECH_MAX_PAUSE = 20000,
	DF_TTS_CEPSTRAL_MIN_PITCH = 100,
	DF_TTS_CEPSTRAL_MAX_PITCH = 500,
	DF_TTS_CEPSTRAL_MIN_SPEED = 0,
	DF_TTS_CEPSTRAL_MAX_SPEED = 400,
	DF_TTS_CEPSTRAL_MIN_VOLUME = 0,
	DF_TTS_CEPSTRAL_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_SPEED = 13,
	DF_TTS_ATTNV_MAX_SPEED = 800,
	DF_TTS_ATTNV_MIN_VOLUME = 0,
	DF_TTS_ATTNV_MAX_VOLUME = 500,
	DF_TTS_ATTNV_MIN_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_ATTNV_MAX_PITCH = 0, /*AT&T NV do not support this*/
	DF_TTS_MACOSXSPMAN_MIN_PITCH = 1,
	DF_TTS_MACOSXSPMAN_MAX_PITCH = 1000,
	DF_TTS_MACOSXSPMAN_MIN_SPEED = 1,
	DF_TTS_MACOSXSPMAN_MAX_SPEED = 1000,
	DF_TTS_MACOSXSPMAN_MIN_VOLUME = 100,
	DF_TTS_MACOSXSPMAN_MAX_VOLUME = 500,
	DF_TTS_MSSAPI_MIN_PITCH = 30,
	DF_TTS_MSSAPI_MAX_PITCH = 350,
	DF_TTS_MSSAPI_MIN_SPEED = 30,
	DF_TTS_MSSAPI_MAX_SPEED = 350,
	DF_TTS_MSSAPI_MIN_VOLUME = 0,
	DF_TTS_MSSAPI_MAX_VOLUME = 100

} DFTTSPredefinedSpeechParams;
```

iDictID
ID of the user dictionary when multiple user dictionaries are in use. The default user's uses value 0 and the range is between 1~1023 (ONLY applicable with NEOSPEECHVOICETEXT). For –1, use default value.

ttTextType
Defines the text type to be synthesized.
For regular text, use DFTTS_TEXT_TYPE_PLAIN, and for VoiceXML/SSML text, use DFTTS_TEXT_TYPE_XML. The value of –1 is regarded as regular text.

szFilePath
File path to save the synthesized voice output under.

ffFileFormat
Defines the types of synthesized output formats.
CEPSTRAL: Ignored. See below.
ATTNV: Ignored. Only PCM WAV support.
MACOSXSPMAN: Ignored. Only AIFF support.
NEOSPEECHVOICETEXT: The following are the types of synthesized output file format that the DF TTS SDK supports for NeoSpeech VoiceText™:

VT_FILE_API_FMT_S16PCM 16bits Linear PCM
VT_FILE_API_FMT_ALAW 8bits A-law PCM
VT_FILE_API_FMT_MULAW 8bits Mu-law PCM
VT_FILE_API_FMT_DADPCM 4bits Dialogic ADPCM
VT_FILE_API_FMT_S16PCM_WAVE 16bits Linear PCM WAVE
VT_FILE_API_FMT_U08PCM_WAVE 8bits Unsigned Linear PCM WAVE
VT_FILE_API_FMT_IMA_WAVE 4bits IMA ADPCM WAVE
VT_FILE_API_FMT_ALAW_WAVE 8bits A-law PCM WAVE
VT_FILE_API_FMT_MULAW_WAVE 8bits Mu-law PCM WAVE
VT_FILE_API_FMT_MULAW_AU 8bits Mu-law PCM SUN AU

MSSAPI: Use the format settings specified by the following enumeration:

```
typedef enum SPSTREAMFORMAT
    {
      SPSF_Default      = -1,
      SPSF_NoAssignedFormat   = 0,
      SPSF_Text   = SPSF_NoAssignedFormat + 1,
      SPSF_NonStandardFormat  = SPSF_Text + 1,
      SPSF_ExtendedAudioFormat      = SPSF_NonStandardFormat + 1,
      SPSF_8kHz8BitMono = SPSF_ExtendedAudioFormat + 1,
      SPSF_8kHz8BitStereo     = SPSF_8kHz8BitMono + 1,
      SPSF_8kHz16BitMono      = SPSF_8kHz8BitStereo + 1,
      SPSF_8kHz16BitStereo    = SPSF_8kHz16BitMono + 1,
      SPSF_11kHz8BitMono      = SPSF_8kHz16BitStereo + 1,
      SPSF_11kHz8BitStereo    = SPSF_11kHz8BitMono + 1,
      SPSF_11kHz16BitMono     = SPSF_11kHz8BitStereo + 1,
      SPSF_11kHz16BitStereo   = SPSF_11kHz16BitMono + 1,
      SPSF_12kHz8BitMono      = SPSF_11kHz16BitStereo + 1,
      SPSF_12kHz8BitStereo    = SPSF_12kHz8BitMono + 1,
      SPSF_12kHz16BitMono     = SPSF_12kHz8BitStereo + 1,
      SPSF_12kHz16BitStereo   = SPSF_12kHz16BitMono + 1,
      SPSF_16kHz8BitMono      = SPSF_12kHz16BitStereo + 1,
```

```
        SPSF_16kHz8BitStereo    = SPSF_16kHz8BitMono + 1,
        SPSF_16kHz16BitMono     = SPSF_16kHz8BitStereo + 1,
        SPSF_16kHz16BitStereo   = SPSF_16kHz16BitMono + 1,
        SPSF_22kHz8BitMono      = SPSF_16kHz16BitStereo + 1,
        SPSF_22kHz8BitStereo    = SPSF_22kHz8BitMono + 1,
        SPSF_22kHz16BitMono     = SPSF_22kHz8BitStereo + 1,
        SPSF_22kHz16BitStereo   = SPSF_22kHz16BitMono + 1,
        SPSF_24kHz8BitMono      = SPSF_22kHz16BitStereo + 1,
        SPSF_24kHz8BitStereo    = SPSF_24kHz8BitMono + 1,
        SPSF_24kHz16BitMono     = SPSF_24kHz8BitStereo + 1,
        SPSF_24kHz16BitStereo   = SPSF_24kHz16BitMono + 1,
        SPSF_32kHz8BitMono      = SPSF_24kHz16BitStereo + 1,
        SPSF_32kHz8BitStereo    = SPSF_32kHz8BitMono + 1,
        SPSF_32kHz16BitMono     = SPSF_32kHz8BitStereo + 1,
        SPSF_32kHz16BitStereo   = SPSF_32kHz16BitMono + 1,
        SPSF_44kHz8BitMono      = SPSF_32kHz16BitStereo + 1,
        SPSF_44kHz8BitStereo    = SPSF_44kHz8BitMono + 1,
        SPSF_44kHz16BitMono     = SPSF_44kHz8BitStereo + 1,
        SPSF_44kHz16BitStereo   = SPSF_44kHz16BitMono + 1,
        SPSF_48kHz8BitMono      = SPSF_44kHz16BitStereo + 1,
        SPSF_48kHz8BitStereo    = SPSF_48kHz8BitMono + 1,
        SPSF_48kHz16BitMono     = SPSF_48kHz8BitStereo + 1,
        SPSF_48kHz16BitStereo   = SPSF_48kHz16BitMono + 1,
        SPSF_TrueSpeech_8kHz1BitMono  = SPSF_48kHz16BitStereo + 1,
        SPSF_CCITT_ALaw_8kHzMono      = SPSF_TrueSpeech_8kHz1BitMono + 1,
        SPSF_CCITT_ALaw_8kHzStereo    = SPSF_CCITT_ALaw_8kHzMono + 1,
        SPSF_CCITT_ALaw_11kHzMono     = SPSF_CCITT_ALaw_8kHzStereo + 1,
        SPSF_CCITT_ALaw_11kHzStereo   = SPSF_CCITT_ALaw_11kHzMono + 1,
        SPSF_CCITT_ALaw_22kHzMono     = SPSF_CCITT_ALaw_11kHzStereo + 1,
        SPSF_CCITT_ALaw_22kHzStereo   = SPSF_CCITT_ALaw_22kHzMono + 1,
        SPSF_CCITT_ALaw_44kHzMono     = SPSF_CCITT_ALaw_22kHzStereo + 1,
        SPSF_CCITT_ALaw_44kHzStereo   = SPSF_CCITT_ALaw_44kHzMono + 1,
        SPSF_CCITT_uLaw_8kHzMono      = SPSF_CCITT_ALaw_44kHzStereo + 1,
        SPSF_CCITT_uLaw_8kHzStereo    = SPSF_CCITT_uLaw_8kHzMono + 1,
        SPSF_CCITT_uLaw_11kHzMono     = SPSF_CCITT_uLaw_8kHzStereo + 1,
        SPSF_CCITT_uLaw_11kHzStereo   = SPSF_CCITT_uLaw_11kHzMono + 1,
        SPSF_CCITT_uLaw_22kHzMono     = SPSF_CCITT_uLaw_11kHzStereo + 1,
        SPSF_CCITT_uLaw_22kHzStereo   = SPSF_CCITT_uLaw_22kHzMono + 1,
        SPSF_CCITT_uLaw_44kHzMono     = SPSF_CCITT_uLaw_22kHzStereo + 1,
        SPSF_CCITT_uLaw_44kHzStereo   = SPSF_CCITT_uLaw_44kHzMono + 1,
        SPSF_ADPCM_8kHzMono     = SPSF_CCITT_uLaw_44kHzStereo + 1,
        SPSF_ADPCM_8kHzStereo   = SPSF_ADPCM_8kHzMono + 1,
        SPSF_ADPCM_11kHzMono    = SPSF_ADPCM_8kHzStereo + 1,
        SPSF_ADPCM_11kHzStereo  = SPSF_ADPCM_11kHzMono + 1,
        SPSF_ADPCM_22kHzMono    = SPSF_ADPCM_11kHzStereo + 1,
        SPSF_ADPCM_22kHzStereo  = SPSF_ADPCM_22kHzMono + 1,
        SPSF_ADPCM_44kHzMono    = SPSF_ADPCM_22kHzStereo + 1,
        SPSF_ADPCM_44kHzStereo  = SPSF_ADPCM_44kHzMono + 1,
        SPSF_GSM610_8kHzMono    = SPSF_ADPCM_44kHzStereo + 1,
        SPSF_GSM610_11kHzMono   = SPSF_GSM610_8kHzMono + 1,
        SPSF_GSM610_22kHzMono   = SPSF_GSM610_11kHzMono + 1,
        SPSF_GSM610_44kHzMono   = SPSF_GSM610_22kHzMono + 1,
        SPSF_NUM_FORMATS  = SPSF_GSM610_44kHzMono + 1
}       SPSTREAMFORMAT;
```

szAudioEncoding, iAudioSamplingRate, iAudioChannels
NEOSPEECHVOICETEXT: Ignored.
ATTNV: Ignored.
MACOSXSPMAN: Ignored.
MSSAPI: Ignored.
CEPSTRAL: Possible values:

szAudioEncoding:

"pcm16", "pcm8" PCM 16 bit/8 bit WAV
"ulaw" - μ-Law (8-bit), "alaw" - A-Law (8-bit)
"riff": Microsoft RIFF (WAV) file
"snd": Sun/NeXT .au (SND) format.
"raw": unheadered audio data, native byte order
"le": unheadered audio data, little-endian (LSB first)
"be": unheadered audio data, big-endian (MSB first)

iAudioSamplingRate:

8000 (8 KHz), 16000 (16 KHz), 11025 (11.025 kHz), etc.

iAudioChannels:

1 (mono), 2 (stereo)

Notes
This is an alternative to DFTTSExportToFile and for the C++ SDK developer makes no difference which of both functions they should use. *(This version is strictly used by the .NET, VB6, Java SDK developers since it provides more portable implementation).*

Return Values
DFTTS_EXPORT_SUCCESS is returned when successfully synthesized and the following error codes are returned when errors occur: (refer to dfttsval.h)

[DFTTS_EXPORT_ERROR_FORMAT_NOT_SUPPORTED] Used format that is not supported.
[DFTTS_EXPORT_ERROR_CHANNEL_MEM_FAIL] Failed to secure channel memory
[DFTTS_EXPORT_ERROR_TEXT_NULL] Text string is a NULL pointer
[DFTTS_EXPORT_ERROR_TEXT_ZERO_LEN] The length of text string is 0.
[DFTTS_EXPORT_ERROR_DB_NOT_LOADED] The TTS DB of the voice requested is not loaded
[DFTTS_EXPORT_ERROR_GEN_FILE_FAIL] Failed to generate the synthesized voice file
[DFTTS_EXPORT_ERROR_OTHER] Other errors

See DFTTSExportToFile for additional error messages.

See Also
DFTTSExportToFile()
InitDFTTSEngineEx()
GetDFTTSEngineInfo()

Example

```
/* Example for NeoSpeech Paul */
DFTTSExportReturnValue result = DFTTSExportToFileEx(
                                        NEOSPEECHVOICETEXT,
                                        "Paul", 1,
                                        1033, pszText, -1,
                                        -1, -1, -1,
                                        1, DFTTS_TEXT_TYPE_XML,
                                        szFileName, VT_FILE_API_FMT_ALAW_WAVE,
                                        "", -1,
                                        -1);

/* Example for Cepstral David */
DFTTSExportReturnValue result = DFTTSExportToFileEx(
                                        CEPSTRAL,
                                        "David", 1,
                                        0, pszText, -1,
                                        -1, -1, -1,
                                        -1, DFTTS_TEXT_TYPE_XML,
                                        szFileName, 0,
                                        "riff", 16000,
                                        2);
```

GetDFTTSEngineInfo

It gets DF TTS SDK engine-related information.

Synopsis

```
#include "dftts.h"
DFTTSParamInfoReturnValue GetDFTTSEngineInfo( DFTTSVoiceEngineType vet,
            unsigned short lang,
            short ttspParam,
            void* Value,
            int iValueSize,
            const char* szNeoSpeechLicFilePath
            );
```

Parameters

vetVType
Which engine type (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV (internal use only, use MSSAPI instead), MACOSXSPMAN or MSSAPI).

lang
Language id (only valid for NEOSPEECHVOICETEXT).

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):
MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

ttspParam
Specifies the DF TTS SDK Engine (specified by *vetVType*) information to be extracted
DFTTS_BUILD_DATE Date when the Engine was built
DFTTS_VERIFY_CODE Verification Result
DFTTS_MAX_CHANNEL Number of maximum channels specified on the license verification file
DFTTS_DB_DIRECTORY Assigned NeoSpeech VoiceText™ DB directory path when the synthesizer DB was installed
DFTTS_LOAD_SUCCESS_CODE Return code for successful synthesizer DB loading
DFTTS_MAX_SPEAKER Maximum number of voices supported. NeoSpeech VoiceText™ supports 2 U.S. English voices but they cannot co-exist on 1 device.
DFTTS_DEF_SPEAKER Default speaker voice ID (1=Paul, 0=Kate)
DFTTS_CODEPAGE code page (Win32) that the engine supports
DFTTS_DB_ACCESS_MODE Database access mode: File(0), RAM(1)
DFTTS_FIXED_POINT_SUPPORT Engine's integer simulation Yes(1)/No(0)
DFTTS_SAMPLING_FREQUENCY Sampling frequency of voice in Hz (8000, 11025, 16000)
DFTTS_MAX_PITCH_RATE Maximum pitch value
DFTTS_DEF_PITCH_RATE Default pitch value
DFTTS_MIN_PITCH_RATE Minimum pitch value
DFTTS_MAX_SPEED_RATE Maximum speed value
DFTTS_DEF_SPEED_RATE Default speed value
DFTTS_MIN_SPEED_RATE Minimum speed value
DFTTS_MAX_VOLUME Maximum volume
DFTTS_DEF_VOLUME Default volume
DFTTS_MIN_VOLUME Minimum volume
DFTTS_MAX_SENT_PAUSE Maximum sentence pause
DFTTS_DEF_SENT_PAUSE Default sentence pause
DFTTS_MIN_SENT_PAUSE Minimum sentence pause

Value
The variable pointer that saves the result of *request*
Use (char *) for *request* values of DFTTS_BUILD_DATE, DFTTS_DB_DIRECTORY and use (int *) for other cases.

iValueSize
Size of obtained parameter *value* in bytes.
If V*alue* is (char *), use the value of text string buffer length; if *value* is (int *), set it to 4 (sizeof(int)).

szNeoSpeechLicFilePath
License Verification File path for the NeoSpeech VoiceText™ Engine.
In case of NULL, use "verification.txt" under the database directory

Notes
Not all ttspParam are supported by CEPSTRAL, ATTNV, MACOSXSPMAN, MSSAPI engines.
It can be used before loading the TTS DB.

It can be used to check errors when developing TTS-enabled applications by getting various information on what the engine supports.

Return Values
It returns DFTTS_PARAM_INFO_SUCCESS when the information is successfully drawn. It returns DFTTS_PARAM_INFO_ERROR_ENGINE_NOT_SUPPORTED if the specified by vetVType engine is not supported.

GetDFTTSNumVoices
Returns the number of the currently installed SDK-supported voices.

Synopsis
```
#include "dftts.h"
size_t GetDFTTSNumVoices();
```

Parameters
None.

Return Values
The number (size_t type) of the installed SDK-supported voices on the current system.

GetDFTTSVoice
Returns information about an SDK-supported voice installed on the current system (voice name, engine, voice id, language id).

Synopsis
```
#include "dftts.h"
DFTTSVoiceInfoReturnValue GetDFTTSVoice( int iVoiceInfoIdIn,
      DFTTSVoiceEngineType* pvetOut,
      char* szVoiceNameOut,
      int* piVoiceNameLenOut,
      int* piVoiceIDOut,
      unsigned short* plangOut
      );
```

Parameters
[IN] iVoiceInfoIdIn
Zero-based number of the voice info retrieval.

[OUT] pvetOut
Voice engine type of the retrieved voice (NEOSPEECHVOICETEXT, CEPSTRAL, ATTNV, MACOSXSPMAN or MSSAPI).

[IN] [OUT] szVoiceNameOut
ASCII c-string name of the voice. Must have sufficient size to hold the retrieved voice name.

[OUT] piVoiceNameLenOut
The length of szVoiceNameOut. Does not count the null-terminating character ('\0').

[OUT] piVoiceIDOut
Voice id of the retrieved voice. Only matters for NEOSPEECHVOICETEXT. Other voices will have an id of -1.

[OUT] plangOut
Language id of the retrieved voice. Only makes a difference right now for NEOSPEECHVOICETEXT.

The new language system of the SDK is designed after the Windows language id implementation.
See dfttslang.h for all language constants. A language id is derived from a primary language value and a sub-language value.

To create a language id you must use the macro (see dfttslang.h):

MAKELANGID(p, s)

Where p is the main or primary language and s is the sub-language value.

See also (from dfttslang.h) the following macros:

PRIMARYLANGID(lgid)
SUBLANGID(lgid)

U.S. English language id is 1033.

Return Values

GET_DFTTS_VI_SUCCESS when the information is successfully retrieved.
GET_DFTTS_VI_NO_MORE_ITEMS when there is no voice information for the provided retrieval id (see sample).
GET_DFTTS_VI_ERROR_BUFFER_TOO_SMALL when the buffer szVoiceNameOut is too small to hold the retrieved voice name.
GET_DFTTS_VI_ERROR_OTHER other errors.

Example

```
//This is how to list all installed supported voices

char szVoiceName[1024];

int iVoiceNameLen = 0;

DFTTSVoiceEngineType vet = NOENGINE;

unsigned short lang = LANG_SUBLANG_NEUTRAL;

int iVoiceID = -1;

int z=0;

for( ; ; z++ )
{
      DFTTSVoiceInfoReturnValue result = GetDFTTSVoice(
                  z, &vet, szVoiceName, &iVoiceNameLen,
                  &iVoiceID, &lang );


      if( result == GET_DFTTS_VI_SUCCESS )
```

```
            {


                  printf( szVoiceName );
                  printf( "\n" );


            }
            else
            {
                  break;
            }

      }

      if( z == 0 )
      {
            printf( "No voices can be found on your system! Please download and
install at least 1 voice!\n" );
            exit(1);
      }
```

GetDFTTSInstalledVoices

Returns a list of installed voice name/engine pairs on the device.

Deprecated. Use GetDFTTSVoice instead.

CleanDFTTSInstalledVoicesRetrieval

Deallocates memory after the use of GetDFTTSInstalledVoices.

Deprecated. Use GetDFTTSVoice instead.

Appendix

VTML for NeoSpeech VoiceText ™ (See vtml.pdf or vtml.ps)

SSML (See http://www.w3.org/TR/speech-synthesis/)

Note: Not all tags are supported by all engines.