Digital Future Text-to-Speech Programmer's Guide for Java (TM)

Version 3.5.0

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 Java.

Appendix 2 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 with C++ and .NET), Mac OS X and Linux.

Supported Java compilers/IDE's: Any Java compiler/IDE.

Sample built using: Eclipse Ganymede

Version of JRE/JDK used for sample: JRE/JDK 1.6.0_06

JNI layer library built with: Microsoft ® Visual C++ 2005 (on Windows), XCode (on Mac OS X), gcc (on Linux).

Chapter 2: Getting Started

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

I. Setup Notes.

Microsoft ® Windows ™ SDK setup notes:

1. The following DLL's must be distributed with the application:

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

Note: All the above Dll's are located in the Windows\Java\Dll\Release... folder.

Important Note: You must put the all the above Dll's in any java.library.path directory or just add a desired directory with all above Dll's in it to the SYSTEM PATH and restart the computer.

Mac OS X SDK setup notes:

1. The developer must place the following JNI library into any java.library.path directory (e.g. /Library/Java/Extensions):

libdfttsjni.jnilib

Note: The above JNI library is located in the MacOSX\Java\JNILib folder.

2. The following frameworks must be 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 following libraries must be distributed with the application:

libdfttsjni.so (JNI library)
libswift.so (Cepstral engine main library)
libceplang....so (Cepstral engine language library)
libceplex....so (Cepstral engine lexicon library)

Note 1: The above JNI library is located in the Linux\Java\Lib\Release... folder.

Note 2: The above Cepstral libraries will be located on the machine after a Cepstral voice installation (/opt/swift/lib directory by default). The "..." in the language and lexicon library names stands for the language part abbreviation of the library, where "en" will be English. If you use other languages, the corresponding language libraries must be also included.

Important Note: You must put the all the above libraries in any java.library.path directory or just add a desired directory with all above libraries in it to the SYSTEM PATH and restart the computer.

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

Important Note for all OS: The SDK is accessed through a native JNI layer library.
See http://java.sun.com/docs/books/jni/html/jniTOC.html for more information.

II. Java (TM) Layer Class

1. You must create a Java class with the name JNILayer with an SDK-required minimum declaration (see Appendix 1).

2. In your main code you must instantiate the class JNILayer and call its methods when necessary (see sample).

Chapter 3: DF TTS SDK Function References

This chapter describes DF TTS SDK Function References, which can be divided into 4 categories.

- Basic API's (loading/unloading the synthesis engine and the User Dictionary, language helper functions, capturing events)

InitDFTTSEngineEx3()
UninitDFTTSEngine()
MakeLanguage()
GetMainLanguage()
GetSubLanguage()
LoadDFTTSUserDict()
UnloadDFTTSUserDict()
onWord()
onExport()

- 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)

DFTTSExportToFileEx()

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

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

- Information API's (Get Voice Information)

GetDFTTSVoice()

InitDFTTSEngineEx3

Loads the synthesizer's TTS database.

Synopsis

```
public native void InitDFTTSEngineEx3(int hwndWinOwner,
                  String szNeoSpeechDBFolderPathKate,
                  String szNeoSpeechDBFolderPathPaul,
                  String szNeoSpeechDBFolderPathMiyu,
                  String szNeoSpeechDBFolderPathShow,
                  String szNeoSpeechDBFolderPathMisaki,
                  String szNeoSpeechDBFolderPathLily,
                  String szNeoSpeechDBFolderPathWang,
                  String szNeoSpeechDBFolderPathJunwoo,
                  String szNeoSpeechDBFolderPathSujin,
                  String szNeoSpeechDBFolderPathYumi,
                  String szNeoSpeechDBFolderPathGyuri,
                  String szNeoSpeechDBFolderPathDayoung,
                  String szNeoSpeechLicFilePathKate,
                  String szNeoSpeechLicFilePathPaul,
                  String szNeoSpeechLicFilePathMiyu,
                  String szNeoSpeechLicFilePathShow,
                  String szNeoSpeechLicFilePathMisaki,
                  String szNeoSpeechLicFilePathLily,
                  String szNeoSpeechLicFilePathWang,
                  String szNeoSpeechLicFilePathJunwoo,
                  String szNeoSpeechLicFilePathSujin,
                  String szNeoSpeechLicFilePathYumi,
                  String szNeoSpeechLicFilePathGyuri,
                  String szNeoSpeechLicFilePathDayoung,
                  int[] psiLoadedEngines,
                  int[] psiLoadedEnginesReturnValues
          );
```

Parameters

hwndWinOwner
Ignored for Java. Set to 0.

12 szNeoSpeechDBFolderPath[VoiceName] parameters
The paths where the NeoSpeech synthesizer databases are located for the supported NeoSpeech voices.

12 szNeoSpeechLicFilePath[VoiceName] parameters
NeoSpeech VoiceText ™ License verification files for the indicated voices

[IN] psiLoadedEngines
An array with currently used engine types.

[IN] psiLoadedEnginesReturnValues
An array with the load status return values for each engine.

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 database and it is used when the program starts.

Return Values

None

Notes

The return status values are held by 2 arrays.

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)), 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.

See the Java (TM) sample for further clarification.

Each member of psiLoadedEnginesReturnValues can hold the following status codes:

When the database is successfully loaded, it returns 0 (Success). The following **common** values are returned when error occurs:

[1] Tried to load the synthesizer database with different values of szNeoSpeechDBFolderPath in case of using multiple synthesizer databases
[2] Failed to secure channel memory
[3] Failed to load DB for the Morpheme Analysis
[4] Failed to load DB for the Break Index
[5] Failed to load DB for the Text Pre-Processing
[6] Failed to load DB for the Acoustic Model
[7] Failed to load DB for Unit Selection
[8] Failed to load DB for Prosody Model
[9] Failed to load DB for Speech Database
[10] Failed to load DB for Pitch Location Information

.
.
.
[78] Other errors

See Also
UninitDFTTSEngine()

Example (taken from the Java sample)

```
    private JNILayer JNILayer;


        public Frame1() {

                JNILayer = new JNILayer();

                JNILayer.setFrame(this);

                //Initialize the SDK

                int[] iLoadedEngines;

                int[] iLoadedEnginesReturnValues;

                iLoadedEngines = new int[NUM_ENGINE_INITIALIZATIONS];

                iLoadedEnginesReturnValues = new int[NUM_ENGINE_INITIALIZATIONS];

                String[] sNeoSpeechDBFolderPaths;

                sNeoSpeechDBFolderPaths = new String[NEOSPEECH_NUM_VOICES];

                String[] szNeoSpeechLicFilePaths;

                szNeoSpeechLicFilePaths = new String[NEOSPEECH_NUM_VOICES];


                for( int i = 0; i<NUM_ENGINE_INITIALIZATIONS-1; i++ )
                {

                        iLoadedEngines[i] = 0;
                        iLoadedEnginesReturnValues[i] = 0;

                         if( i < NEOSPEECH_NUM_VOICES)
                         {
                                if( i == 1 ) //Paul (Windows only) - Change your paths accordingly!
                                {
                                        sNeoSpeechDBFolderPaths[i]="C:\\Program
Files\\VW\\VT\\Paul\\M16";

                                        szNeoSpeechLicFilePaths[i]="C:\\Program
Files\\VW\\VT\\Paul\\M16\\data-common\\verify\\verification.txt";

                                }
                                else
                                {
                                        sNeoSpeechDBFolderPaths[i]="";

                                        szNeoSpeechLicFilePaths[i]="";
                                }
                         }
                }

                JNILayer.InitDFTTSEngineEx3(0,
```

```
                        sNeoSpeechDBFolderPaths[0], //Kate
                        sNeoSpeechDBFolderPaths[1], //Paul
                        sNeoSpeechDBFolderPaths[2], //Miyu
                        sNeoSpeechDBFolderPaths[3], //Show
                        sNeoSpeechDBFolderPaths[4], //Misaki
                        sNeoSpeechDBFolderPaths[5], //Lily
                        sNeoSpeechDBFolderPaths[6], //Wang
                        sNeoSpeechDBFolderPaths[7],   //Junwoo
                        sNeoSpeechDBFolderPaths[8], //Sujin
                        sNeoSpeechDBFolderPaths[9], //Yumi
                        sNeoSpeechDBFolderPaths[10], //Gyuri
                        sNeoSpeechDBFolderPaths[11], //Dayoung
                        szNeoSpeechLicFilePaths[0], //Kate
                        szNeoSpeechLicFilePaths[1], //Paul
                        szNeoSpeechLicFilePaths[2], //Miyu
                        szNeoSpeechLicFilePaths[3], //Show
                        szNeoSpeechLicFilePaths[4], //Misaki
                        szNeoSpeechLicFilePaths[5], //Lily
                        szNeoSpeechLicFilePaths[6], //Wang
                        szNeoSpeechLicFilePaths[7],   //Junwoo
                        szNeoSpeechLicFilePaths[8], //Sujin
                        szNeoSpeechLicFilePaths[9], //Yumi
                        szNeoSpeechLicFilePaths[10], //Gyuri
                        szNeoSpeechLicFilePaths[11], //Dayoung
            iLoadedEngines,
            iLoadedEnginesReturnValues
    );


/*Catch load errors */

String sEngineName = "";
int result = 0;

String[] sNeospVoiceNames = {
    NEOSPEECH_KATE_ENG_NAMESTR,
    NEOSPEECH_PAUL_ENG_NAMESTR,
    NEOSPEECH_MIYU_JPN_NAMESTR,
    NEOSPEECH_SHOW_JPN_NAMESTR,
    NEOSPEECH_MISAKI_JPN_NAMESTR,
    NEOSPEECH_LILY_CHI_NAMESTR,
    NEOSPEECH_WANG_CHI_NAMESTR,
    NEOSPEECH_JUNWOO_KOR_NAMESTR,
    NEOSPEECH_SUJIN_KOR_NAMESTR,
    NEOSPEECH_YUMI_KOR_NAMESTR,
    NEOSPEECH_GYURI_KOR_NAMESTR,
    NEOSPEECH_DAYOUNG_KOR_NAMESTR
};

for (int intI = 0; intI <= iLoadedEngines.length- 1; intI++)
{



    if (iLoadedEngines[intI] == NEOSPEECHVOICETEXT)
    {

        sEngineName = "NEOPSEECHVOICETEXT Engine Load Result for voice " +
            sNeospVoiceNames[intI] + ": ";

    }
    else if (iLoadedEngines[intI] == CEPSTRAL)
    {

        sEngineName = "CEPSTRAL Engine Load Result: ";

    }
    else if (iLoadedEngines[intI] == ATTNV)
    {

        sEngineName = "ATTNV Engine (RESERVED INTERNAL USE) Load Result: ";
```

```
            }
            else if (iLoadedEngines[intI] == MACOSXSPMAN)
            {

                sEngineName = "MAC OS X SPMAN Load Result: ";

            }
            else if (iLoadedEngines[intI] == MSSAPI)
            {

                sEngineName = "MS SAPI Load Result: ";

            }

            result = iLoadedEnginesReturnValues[intI];

            System.out.println(sEngineName + result + '\n');

        }


}
```

UninitDFTTSEngine

Unloads the synthesizer's DB

Synopsis

```
public native int UninitDFTTSEngine();
```

Parameters

None.

Description

The function frees the assigned memory by unloading the synthesizer DB's and all internal objects. For Java this function also clears the global JNI object reference.

It **MUST** be called before the program exits otherwise memory corruption will occur!

Return Values

0 (Success)
1 (Failed)

See Also

InitDFTTSEngineEx3()

Notes

Mac OS X Java (TM) users: In order to make sure the function is always called on exit, you must capture the Mac OS X Quit event (System Menu>Mac Button + Q). You must implement an "OSXAdapter". See folder MacOSX/Java/Sample/OSXAdapter for the example from Apple.

Example (see sample)

```
private void thisWindowClosing(WindowEvent e) {


        /*********************************
        * IMPORTANT FOR MAC OS X USERS!
        *
        * JNILayer.UninitDFTTSEngine();
        *
        * MUST ALSO BE CALLED WHEN THE APP QUITS (WHEN THE USER PRESSES MAC BUTTON + Q)!
        *
        * SEE THE SAMPLE PACKAGE FROM APPLE IN FOLDER OSXAdapter ON HOW TO CAPTURE THE QUIT EVENT FOR MAC OS X
        *
        * THE SAMPLE IS MADE CROSS-PLATFORM AS IT CHECKS FOR THE OS NAME
        *
        */


        JNILayer.UninitDFTTSEngine();

        System.out.println("Speech SDK Memory Released!");

        System.exit(0);

}
```

MakeLanguage

Creates a language id from a main language id and a sub-language id.

Synopsis

```
public native short MakeLanguage( short mainlang,
                                  short sublang
                                  );
```

Parameters

mainlang

The id of the main or primary language. See the Frame1.java in the sample for all language constants.

sublang

The id of the sub-language. See the Frame1.java in the sample for all language constants.

Description

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

Return Values
The language id needed for the SDK.

See Also
GetMainLanguage()
GetSubLanguage()

GetMainLanguage
A Helper function that returns the main language id from a language id.

Synopsis
```
public native short GetMainLanguage( short lang );
```

Parameters
lang
The id of the language.

Description
The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

Return Values
The id of the primary language used in the creation of the language id (lang parameter).

See Also
MakeLanguage()
GetSubLanguage()

GetSubLanguage
A Helper function that returns the sub-language id from a language id.

Synopsis
```
public native short  GetSubLanguage( short lang );
```

Parameters
lang
The id of the language.

Description
The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

Return Values
The id of the sub-language used in the creation of the language id (lang parameter).

See Also
MakeLanguage()
GetMainLanguage()

LoadDFTTSUserDict
Loads the User Dictionary.

Synopsis
```
public native int LoadDFTTSUserDict(int iDictIndex, String szDictName,
                  String szDictFileName, int vet, short lang,
                  String szVoiceName, String 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))

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

vet
Load for the specified engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) or MSSAPI (3)).

Use the following values (see sample):

```
public final int NEOSPEECHVOICETEXT = 0;

public final int CEPSTRAL = 1;

public final int ATTNV = 2;

public final int MSSAPI = 3;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

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 MSSAPI and ATTNV where instead of a dictionary file we supply the actual character contents).

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 has to be used after the completion of the loading of the synthesizer DB's. iDictIndex is used during synthesis by NEOSPEECHVOICETEXT.

MACOSXSPMAN does not support user dictionaries through the SDK.

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

Return Values

0 is returned when user dictionary is successfully loaded. The following common error codes are returned when errors occur:

[1] iDictIndex value is not within the valid range
[2] User dictionary file corresponding to iDictIndex is already loaded.
[3] Loading failed because there was no user dictionary files or valid entry.
[4] Invalid dictionary file.
[5] Invalid voice.
[6] Invalid phoneme set.
[7] Invalid phoneme.
[8] Other errors.

See Also

UnloadDFTTSUserDict()

Example (see sample)

```
int dictreturn = 0;

//User Dictionary Loading:
//(read dictionary documentation per engine located in folder DictionarySystems)

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

//JNILayer.MakeLanguage(LANG_ENGLISH, SUBLANG_ENGLISH_US) = 1033 - You can just use 1033

// *** YOU MUST CHANGE THE DICTIONARY PATH TO MATCH YOURS! *** //

dictreturn = JNILayer.LoadDFTTSUserDict(1, "",
"C:\\CPP\\Projects\\DFTTSSDK\\samples\\Java\\DFTTSSDKSampleJava\\neospeech_userdict_eng.csv",
                NEOSPEECHVOICETEXT, JNILayer.MakeLanguage(LANG_ENGLISH, SUBLANG_ENGLISH_US),
                "", "");

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

dictreturn = JNILayer.LoadDFTTSUserDict(-1, "",
 "C:\\CPP\\Projects\\DFTTSSDK\\samples\\Java\\DFTTSSDKSampleJava\\cepstral_dictionary.txt",
                CEPSTRAL, LANG_SUBLANG_NEUTRAL, "David", "");

//MACOSXSPMAN: No dictionary support.

if (dictreturn != 0 )
{
JOptionPane.showMessageDialog( this, "SDK Load Dictionary failed with error: " + dictreturn );
}
```

UnloadDFTTSUserDict
Unloads the User Dictionary.

Synopsis
```
public native int UnloadDFTTSUserDict(int iDictIndex, String szDictName,
                int vet, 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 engines MSSAPI and ATTNV (internal use only)).

vet
Unload for the specified engine type (NEOSPEECHVOICETEXT (0), ATTNV (2) or MSSAPI (3))

Use the following values (see sample):

```
public final int NEOSPEECHVOICETEXT = 0;

public final int ATTNV = 2;

public final int MSSAPI = 3;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

Description
The function unloads the user dictionary.
Any user dictionary that has not been unloaded is automatically unloaded during the process of unloading the synthesizer DB.

CEPSTRAL engine does not support user dictionary unloading. The dictionary data unloads with the unloading the voice DB.

Return Values
0 is returned when user dictionary is successfully loaded. The following common error codes are returned when errors occur:
[1] iDictIndex value is not within the valid range
[2] User dictionary file corresponding to iDictIndex is already unloaded.

See Also
LoadDFTTSUserDict()

onWord
Receives the speak event values of the current spoken word start and end characters (see sample).

Synopsis
```
public void onWord(int start, int end)
{

      //TODO: Enter your event handling code here

}
```

Parameters
[IN] start
The number of the start word character with regard to the whole text.
[IN] end
The number of the end word character with regard to the whole text.

Description
Receives the speak event values of the current spoken word start and end characters (see sample).

When the speech is finished, start is 0 (zero) and end is -1.

MACOSXSPMAN: onExport event data is sent here (onWord). MACOSXSPMAN treats exports as speech.

MSSAPI engine may not get the onWord even with Java (TM) due to Java (TM) message handling irregularities.

Return Values
None.

See Also
onExport ()

Example (see sample)

```
//MACOSXSPMAN: onExport event data is sent here (onWord). MACOSXSPMAN treats exports as speech.

            public void onWord(int start, int end)
            {

                    if(start==0 && end==-1)
                    {
                        frame.bIsSpeaking = false;
                        frame.bIsPaused=false;

                        if( frame.button2.isEnabled())
                                frame.button2.setEnabled(false);

                        if( frame.button3.isEnabled())
                                frame.button3.setEnabled(false);

                    }
                    else
                    {
                        if( !frame.button2.isEnabled())
                                frame.button2.setEnabled(true);

                        if( !frame.button3.isEnabled())
                                frame.button3.setEnabled(true);

                    }


             frame.label1.setText("onWord: Start Character: " + start  + ", End Character: " + end);


            }
```

onExport

Gets called by the JNI when the audio file synthesis is complete (see sample).

Synopsis

```
public void onExport(int start, int end)
{

      //TODO: Enter your event handling code here

}
```

Parameters

[IN] start
Unused. Reserved.
[IN] end
Unused. Reserved.

Description

Gets called by the JNI when the audio file synthesis is complete (see sample).

The onExport with MACOSXSPMAN is treated as and onWord event so do not use this function with MACOSXSPMAN.

Return Values

None.

See Also
onWord ()

Example (see sample)
```
//MACOSXSPMAN: Do not use. Event data is sent to onWord
public void onExport(int start, int end)
{
	//start and end are SDK-reserved

	frame.label1.setText("onExport: Export successful!");

}
```

DFTTSSpeak
It plays synthesized TTS output through a sound card.

Synopsis
```
public native int DFTTSSpeak(int hwndWinOwner, int vet,
	String szVoiceName, int iVoiceID, short lang, String szText, int iPitch,
	int iSpeed, int iVolume, int iPause, int iDictID, int ttTextType,
	short ofOutPutFormat);
```

Parameters
hwndWinOwner
Ignored for Java. Set to 0.

vet
Which engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) (internal use only), MSSAPI (3) or MACOSXSPMAN (4))

Use the following values (see sample):

```
	public final int NEOSPEECHVOICETEXT = 0;

	public final int CEPSTRAL = 1;

	public final int ATTNV = 2;

	public final int MSSAPI = 3;

	public final int MACOSXSPMAN = 4;
```

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

iVoiceID
Internal voice id (matters for NEOSPEECHVOICETEXT).

Use the following values (see sample):

```
public final int NEOSPEECH_KATE_ENG = 0;

public final int NEOSPEECH_PAUL_ENG = 1;

public final int NEOSPEECH_MIYU_JPN = 0;

public final int NEOSPEECH_SHOW_JPN = 1;

public final int NEOSPEECH_MISAKI_JPN = 2;

public final int NEOSPEECH_LILY_CHI = 0;

public final int NEOSPEECH_WANG_CHI = 1;

public final int NEOSPEECH_JUNWOO_KOR = 3;

public final int NEOSPEECH_SUJIN_KOR = 8;

public final int NEOSPEECH_YUMI_KOR = 10;

public final int NEOSPEECH_GYURI_KOR = 11;

public final int NEOSPEECH_DAYOUNG_KOR = 12;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

szText
Text string to be synthesized (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 declarations (see sample) outline the default values and supported ranges between engines:

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

```
public final int DF_TTS_MSSAPI_MAX_VOLUME = 100;
```

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.

Use the following values (see sample):

```
public final short DFTTS_TEXT_TYPE_PLAIN = 0;

public final short DFTTS_TEXT_TYPE_XML = 1;
```

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 sample).

Possible values are defined by the following constants (see sample):

```
public final short SPSF_Default      = -1;
public final short SPSF_NoAssignedFormat     = 0;
public final short SPSF_Text  = SPSF_NoAssignedFormat + 1;
public final short SPSF_NonStandardFormat    = SPSF_Text + 1;
public final short SPSF_ExtendedAudioFormat  = SPSF_NonStandardFormat + 1;
public final short SPSF_8kHz8BitMono  = SPSF_ExtendedAudioFormat + 1;
public final short SPSF_8kHz8BitStereo       = SPSF_8kHz8BitMono + 1;
public final short SPSF_8kHz16BitMono = SPSF_8kHz8BitStereo + 1;
public final short SPSF_8kHz16BitStereo      = SPSF_8kHz16BitMono + 1;
public final short SPSF_11kHz8BitMono = SPSF_8kHz16BitStereo + 1;
public final short SPSF_11kHz8BitStereo      = SPSF_11kHz8BitMono + 1;
public final short SPSF_11kHz16BitMono       = SPSF_11kHz8BitStereo + 1;
public final short SPSF_11kHz16BitStereo     = SPSF_11kHz16BitMono + 1;
public final short SPSF_12kHz8BitMono = SPSF_11kHz16BitStereo + 1;
public final short SPSF_12kHz8BitStereo      = SPSF_12kHz8BitMono + 1;
public final short SPSF_12kHz16BitMono       = SPSF_12kHz8BitStereo + 1;
public final short SPSF_12kHz16BitStereo     = SPSF_12kHz16BitMono + 1;
public final short SPSF_16kHz8BitMono = SPSF_12kHz16BitStereo + 1;
public final short SPSF_16kHz8BitStereo      = SPSF_16kHz8BitMono + 1;
public final short SPSF_16kHz16BitMono       = SPSF_16kHz8BitStereo + 1;
public final short SPSF_16kHz16BitStereo     = SPSF_16kHz16BitMono + 1;
public final short SPSF_22kHz8BitMono = SPSF_16kHz16BitStereo + 1;
public final short SPSF_22kHz8BitStereo      = SPSF_22kHz8BitMono + 1;
public final short SPSF_22kHz16BitMono       = SPSF_22kHz8BitStereo + 1;
public final short SPSF_22kHz16BitStereo     = SPSF_22kHz16BitMono + 1;
public final short SPSF_24kHz8BitMono = SPSF_22kHz16BitStereo + 1;
public final short SPSF_24kHz8BitStereo      = SPSF_24kHz8BitMono + 1;
public final short SPSF_24kHz16BitMono       = SPSF_24kHz8BitStereo + 1;
public final short SPSF_24kHz16BitStereo     = SPSF_24kHz16BitMono + 1;
public final short SPSF_32kHz8BitMono = SPSF_24kHz16BitStereo + 1;
public final short SPSF_32kHz8BitStereo      = SPSF_32kHz8BitMono + 1;
public final short SPSF_32kHz16BitMono       = SPSF_32kHz8BitStereo + 1;
public final short SPSF_32kHz16BitStereo     = SPSF_32kHz16BitMono + 1;
public final short SPSF_44kHz8BitMono = SPSF_32kHz16BitStereo + 1;
```

```
public final short SPSF_44kHz8BitStereo      = SPSF_44kHz8BitMono + 1;
public final short SPSF_44kHz16BitMono       = SPSF_44kHz8BitStereo + 1;
public final short SPSF_44kHz16BitStereo     = SPSF_44kHz16BitMono + 1;
public final short SPSF_48kHz8BitMono = SPSF_44kHz16BitStereo + 1;
public final short SPSF_48kHz8BitStereo      = SPSF_48kHz8BitMono + 1;
public final short SPSF_48kHz16BitMono       = SPSF_48kHz8BitStereo + 1;
public final short SPSF_48kHz16BitStereo     = SPSF_48kHz16BitMono + 1;
public final short SPSF_TrueSpeech_8kHz1BitMono      = SPSF_48kHz16BitStereo + 1;
public final short SPSF_CCITT_ALaw_8kHzMono  = SPSF_TrueSpeech_8kHz1BitMono + 1;
public final short SPSF_CCITT_ALaw_8kHzStereo        = SPSF_CCITT_ALaw_8kHzMono + 1;
public final short SPSF_CCITT_ALaw_11kHzMono = SPSF_CCITT_ALaw_8kHzStereo + 1;
public final short SPSF_CCITT_ALaw_11kHzStereo       = SPSF_CCITT_ALaw_11kHzMono + 1;
public final short SPSF_CCITT_ALaw_22kHzMono = SPSF_CCITT_ALaw_11kHzStereo + 1;
public final short SPSF_CCITT_ALaw_22kHzStereo       = SPSF_CCITT_ALaw_22kHzMono + 1;
public final short SPSF_CCITT_ALaw_44kHzMono = SPSF_CCITT_ALaw_22kHzStereo + 1;
public final short SPSF_CCITT_ALaw_44kHzStereo       = SPSF_CCITT_ALaw_44kHzMono + 1;
public final short SPSF_CCITT_uLaw_8kHzMono  = SPSF_CCITT_ALaw_44kHzStereo + 1;
public final short SPSF_CCITT_uLaw_8kHzStereo        = SPSF_CCITT_uLaw_8kHzMono + 1;
public final short SPSF_CCITT_uLaw_11kHzMono = SPSF_CCITT_uLaw_8kHzStereo + 1;
public final short SPSF_CCITT_uLaw_11kHzStereo       = SPSF_CCITT_uLaw_11kHzMono + 1;
public final short SPSF_CCITT_uLaw_22kHzMono = SPSF_CCITT_uLaw_11kHzStereo + 1;
public final short SPSF_CCITT_uLaw_22kHzStereo       = SPSF_CCITT_uLaw_22kHzMono + 1;
public final short SPSF_CCITT_uLaw_44kHzMono = SPSF_CCITT_uLaw_22kHzStereo + 1;
public final short SPSF_CCITT_uLaw_44kHzStereo       = SPSF_CCITT_uLaw_44kHzMono + 1;
public final short SPSF_ADPCM_8kHzMono       = SPSF_CCITT_uLaw_44kHzStereo + 1;
public final short SPSF_ADPCM_8kHzStereo     = SPSF_ADPCM_8kHzMono + 1;
public final short SPSF_ADPCM_11kHzMono      = SPSF_ADPCM_8kHzStereo + 1;
public final short SPSF_ADPCM_11kHzStereo    = SPSF_ADPCM_11kHzMono + 1;
public final short SPSF_ADPCM_22kHzMono      = SPSF_ADPCM_11kHzStereo + 1;
public final short SPSF_ADPCM_22kHzStereo    = SPSF_ADPCM_22kHzMono + 1;
public final short SPSF_ADPCM_44kHzMono      = SPSF_ADPCM_22kHzStereo + 1;
public final short SPSF_ADPCM_44kHzStereo    = SPSF_ADPCM_44kHzMono + 1;
public final short SPSF_GSM610_8kHzMono      = SPSF_ADPCM_44kHzStereo + 1;
public final short SPSF_GSM610_11kHzMono     = SPSF_GSM610_8kHzMono + 1;
public final short SPSF_GSM610_22kHzMono     = SPSF_GSM610_11kHzMono + 1;
public final short SPSF_GSM610_44kHzMono     = SPSF_GSM610_22kHzMono + 1;
public final short SPSF_NUM_FORMATS   = SPSF_GSM610_44kHzMono + 1;
```

Notes

The Java (TM) implementation has a problem with NEOSPEECHVOICETEXT when the voice is speaking because of irregular Java (TM) window message handling and thread issues. Therefore, you should not use the NEOSPEECHVOICETEXT engine with `DFTTSSpeak`. The synthesis will be cut off and the engine processing thread will be terminated by the virtual machine after a few seconds.

MSSAPI engine may not get the onWord even with Java (TM) due to Java (TM) message handling irregularities.

Return Values

0 is returned when it was executed successfully. The following **common** error codes are returned when errors occur:

[1] Failed to secure channel memory
[2] The text string is a NULL pointer
[3] The length of text string is 0
[4] The TTS DB of the voice requested is not loaded
[5] Failed to set the sound card
.
.
.
[101] Other errors

Example (see sample)

```
Object obj = comboBox1.getItemAt( comboBox1.getSelectedIndex() );

VoiceData vo = (VoiceData)obj;

int result = JNILayer.DFTTSSpeak(0,
                vo.m_vet,
                vo.m_sVoiceName,
                vo.m_VoiceID,
                vo.m_lang,
                textArea1.getText(), -1, -1, -1, -1,
                1, DFTTS_TEXT_TYPE_XML,(short)-1);


if( result != 0 )
{
      JOptionPane.showMessageDialog(this, "SDK Speak error: " + result);

}
```

DFTTSStop

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

Synopsis

```
public native int  DFTTSStop( int vet, short lang );
```

Parameters

vet
Which engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) (internal use only), MSSAPI (3) or MACOSXSPMAN (4))

Use the following values (see sample):

```
      public final int NEOSPEECHVOICETEXT = 0;

      public final int CEPSTRAL = 1;

      public final int ATTNV = 2;

      public final int MSSAPI = 3;

      public final int MACOSXSPMAN = 4;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

Notes
Using DFTTSSpeak(), stops the playback of synthesized voice before synthesizing.

Return Values
```
0 (Success),
1 (Invalid pointer),
2 (Other error)
```

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

Synopsis
```
public native int  DFTTSPause( int vet, short lang );
```

Parameters
vet
Which engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) (internal use only), MSSAPI (3) or MACOSXSPMAN (4))

Use the following values (see sample):

```
public final int NEOSPEECHVOICETEXT = 0;

public final int CEPSTRAL = 1;

public final int ATTNV = 2;

public final int MSSAPI = 3;

public final int MACOSXSPMAN = 4;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

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

Return Values
```
0 (Success),
1 (Unimplemeted),
2 (Internal error),
3 (Invalid parameter),
4 (Invalid pointer),
.
.
.
15 (Other error)
```

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

Synopsis
```
public native int  DFTTSResume( int vet, short lang );
```

Parameters
vet
Which engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) (internal use only), MSSAPI (3) or MACOSXSPMAN (4))

Use the following values (see sample):

```
public final int NEOSPEECHVOICETEXT = 0;

public final int CEPSTRAL = 1;

public final int ATTNV = 2;

public final int MSSAPI = 3;

public final int MACOSXSPMAN = 4;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

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

Return Values
0 if succeeded.

DFTTSExportToFileEx
It saves the synthesized output as a file.

Synopsis
```
public native int DFTTSExportToFileEx(int vet, String szVoiceName,
      int iVoiceID, short lang, String szText, int iPitch, int iSpeed,
      int iVolume, int iPause, int iDictID, int ttTextType, String szFilePath,
      short ffFileFormat, String szAudioEncoding, int iAudioSamplingRate,
      int iAudioChannels);
```

Parameters
vet
Which engine type (NEOSPEECHVOICETEXT (0), CEPSTRAL (1), ATTNV (2) (internal use only), MSSAPI (3) or MACOSXSPMAN (4))

Use the following values (see sample):

```
      public final int NEOSPEECHVOICETEXT = 0;

      public final int CEPSTRAL = 1;

      public final int ATTNV = 2;

      public final int MSSAPI = 3;

      public final int MACOSXSPMAN = 4;
```

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

iVoiceID
Internal voice id (matters for NEOSPEECHVOICETEXT).

Use the following values (see sample):

```
    public final int NEOSPEECH_KATE_ENG = 0;
```

```
public final int NEOSPEECH_PAUL_ENG = 1;

public final int NEOSPEECH_MIYU_JPN = 0;

public final int NEOSPEECH_SHOW_JPN = 1;

public final int NEOSPEECH_MISAKI_JPN = 2;

public final int NEOSPEECH_LILY_CHI = 0;

public final int NEOSPEECH_WANG_CHI = 1;

public final int NEOSPEECH_JUNWOO_KOR = 3;

public final int NEOSPEECH_SUJIN_KOR = 8;

public final int NEOSPEECH_YUMI_KOR = 10;

public final int NEOSPEECH_GYURI_KOR = 11;

public final int NEOSPEECH_DAYOUNG_KOR = 12;
```

lang
Language id (only NEOSPEECHVOICETEXT, use 0 (zero) for the other engines)

The new language system of the SDK is designed after the Windows language id implementation.

See the Frame1.java in the sample for all language constants. A language id is derived from a primary language id and a sub-language id.

To create a language id you must use MakeLanguage with the desired main and sub- language ids.

U.S. English language id is 1033.

To obtain a primary language id from a created with MakeLanguage id, use GetMainLanguage.

To obtain a sub-language id from a created with MakeLanguage id, use GetSubLanguage.

szText
Text string to be synthesized (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 declarations (see sample) outline the default values and supported ranges between engines:

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

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.

Use the following values (see sample):

```
public final short DFTTS_TEXT_TYPE_PLAIN = 0;

public final short DFTTS_TEXT_TYPE_XML = 1;
```

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

Use the following constants (see sample):

```
public final short VT_FILE_API_FMT_S16PCM = 0;
public final short VT_FILE_API_FMT_ALAW = 1;
public final short VT_FILE_API_FMT_MULAW = 2;
public final short VT_FILE_API_FMT_DADPCM = 3;
public final short VT_FILE_API_FMT_S16PCM_WAVE = 4;
public final short VT_FILE_API_FMT_U08PCM_WAVE = 5;
public final short VT_FILE_API_FMT_IMA_WAVE = 6;
public final short VT_FILE_API_FMT_ALAW_WAVE = 7;
public final short VT_FILE_API_FMT_MULAW_WAVE = 8;
public final short VT_FILE_API_FMT_MULAW_AU = 9;
```

MSSAPI: Use the format settings specified by the following constants (see sample):

```
public final short SPSF_Default       = -1;
public final short SPSF_NoAssignedFormat     = 0;
public final short SPSF_Text  = SPSF_NoAssignedFormat + 1;
public final short SPSF_NonStandardFormat    = SPSF_Text + 1;
public final short SPSF_ExtendedAudioFormat  = SPSF_NonStandardFormat + 1;
public final short SPSF_8kHz8BitMono  = SPSF_ExtendedAudioFormat + 1;
public final short SPSF_8kHz8BitStereo       = SPSF_8kHz8BitMono + 1;
public final short SPSF_8kHz16BitMono = SPSF_8kHz8BitStereo + 1;
public final short SPSF_8kHz16BitStereo      = SPSF_8kHz16BitMono + 1;
public final short SPSF_11kHz8BitMono = SPSF_8kHz16BitStereo + 1;
public final short SPSF_11kHz8BitStereo      = SPSF_11kHz8BitMono + 1;
public final short SPSF_11kHz16BitMono       = SPSF_11kHz8BitStereo + 1;
public final short SPSF_11kHz16BitStereo     = SPSF_11kHz16BitMono + 1;
public final short SPSF_12kHz8BitMono = SPSF_11kHz16BitStereo + 1;
public final short SPSF_12kHz8BitStereo      = SPSF_12kHz8BitMono + 1;
public final short SPSF_12kHz16BitMono       = SPSF_12kHz8BitStereo + 1;
public final short SPSF_12kHz16BitStereo     = SPSF_12kHz16BitMono + 1;
public final short SPSF_16kHz8BitMono = SPSF_12kHz16BitStereo + 1;
public final short SPSF_16kHz8BitStereo      = SPSF_16kHz8BitMono + 1;
public final short SPSF_16kHz16BitMono       = SPSF_16kHz8BitStereo + 1;
public final short SPSF_16kHz16BitStereo     = SPSF_16kHz16BitMono + 1;
public final short SPSF_22kHz8BitMono = SPSF_16kHz16BitStereo + 1;
public final short SPSF_22kHz8BitStereo      = SPSF_22kHz8BitMono + 1;
public final short SPSF_22kHz16BitMono       = SPSF_22kHz8BitStereo + 1;
public final short SPSF_22kHz16BitStereo     = SPSF_22kHz16BitMono + 1;
public final short SPSF_24kHz8BitMono = SPSF_22kHz16BitStereo + 1;
public final short SPSF_24kHz8BitStereo      = SPSF_24kHz8BitMono + 1;
public final short SPSF_24kHz16BitMono       = SPSF_24kHz8BitStereo + 1;
public final short SPSF_24kHz16BitStereo     = SPSF_24kHz16BitMono + 1;
public final short SPSF_32kHz8BitMono = SPSF_24kHz16BitStereo + 1;
public final short SPSF_32kHz8BitStereo      = SPSF_32kHz8BitMono + 1;
public final short SPSF_32kHz16BitMono       = SPSF_32kHz8BitStereo + 1;
public final short SPSF_32kHz16BitStereo     = SPSF_32kHz16BitMono + 1;
public final short SPSF_44kHz8BitMono = SPSF_32kHz16BitStereo + 1;
public final short SPSF_44kHz8BitStereo      = SPSF_44kHz8BitMono + 1;
public final short SPSF_44kHz16BitMono       = SPSF_44kHz8BitStereo + 1;
public final short SPSF_44kHz16BitStereo     = SPSF_44kHz16BitMono + 1;
public final short SPSF_48kHz8BitMono = SPSF_44kHz16BitStereo + 1;
public final short SPSF_48kHz8BitStereo      = SPSF_48kHz8BitMono + 1;
public final short SPSF_48kHz16BitMono       = SPSF_48kHz8BitStereo + 1;
public final short SPSF_48kHz16BitStereo     = SPSF_48kHz16BitMono + 1;
public final short SPSF_TrueSpeech_8kHz1BitMono      = SPSF_48kHz16BitStereo + 1;
public final short SPSF_CCITT_ALaw_8kHzMono  = SPSF_TrueSpeech_8kHz1BitMono + 1;
public final short SPSF_CCITT_ALaw_8kHzStereo        = SPSF_CCITT_ALaw_8kHzMono + 1;
public final short SPSF_CCITT_ALaw_11kHzMono = SPSF_CCITT_ALaw_8kHzStereo + 1;
public final short SPSF_CCITT_ALaw_11kHzStereo       = SPSF_CCITT_ALaw_11kHzMono + 1;
public final short SPSF_CCITT_ALaw_22kHzMono = SPSF_CCITT_ALaw_11kHzStereo + 1;
public final short SPSF_CCITT_ALaw_22kHzStereo       = SPSF_CCITT_ALaw_22kHzMono + 1;
public final short SPSF_CCITT_ALaw_44kHzMono = SPSF_CCITT_ALaw_22kHzStereo + 1;
public final short SPSF_CCITT_ALaw_44kHzStereo       = SPSF_CCITT_ALaw_44kHzMono + 1;
public final short SPSF_CCITT_uLaw_8kHzMono  = SPSF_CCITT_ALaw_44kHzStereo + 1;
public final short SPSF_CCITT_uLaw_8kHzStereo        = SPSF_CCITT_uLaw_8kHzMono + 1;
public final short SPSF_CCITT_uLaw_11kHzMono = SPSF_CCITT_uLaw_8kHzStereo + 1;
public final short SPSF_CCITT_uLaw_11kHzStereo       = SPSF_CCITT_uLaw_11kHzMono + 1;
public final short SPSF_CCITT_uLaw_22kHzMono = SPSF_CCITT_uLaw_11kHzStereo + 1;
public final short SPSF_CCITT_uLaw_22kHzStereo       = SPSF_CCITT_uLaw_22kHzMono + 1;
public final short SPSF_CCITT_uLaw_44kHzMono = SPSF_CCITT_uLaw_22kHzStereo + 1;
public final short SPSF_CCITT_uLaw_44kHzStereo       = SPSF_CCITT_uLaw_44kHzMono + 1;
public final short SPSF_ADPCM_8kHzMono       = SPSF_CCITT_uLaw_44kHzStereo + 1;
public final short SPSF_ADPCM_8kHzStereo     = SPSF_ADPCM_8kHzMono + 1;
public final short SPSF_ADPCM_11kHzMono      = SPSF_ADPCM_8kHzStereo + 1;
public final short SPSF_ADPCM_11kHzStereo    = SPSF_ADPCM_11kHzMono + 1;
public final short SPSF_ADPCM_22kHzMono      = SPSF_ADPCM_11kHzStereo + 1;
public final short SPSF_ADPCM_22kHzStereo    = SPSF_ADPCM_22kHzMono + 1;
public final short SPSF_ADPCM_44kHzMono      = SPSF_ADPCM_22kHzStereo + 1;
public final short SPSF_ADPCM_44kHzStereo    = SPSF_ADPCM_44kHzMono + 1;
public final short SPSF_GSM610_8kHzMono      = SPSF_ADPCM_44kHzStereo + 1;
```

```
public final short SPSF_GSM610_11kHzMono     = SPSF_GSM610_8kHzMono + 1;
public final short SPSF_GSM610_22kHzMono     = SPSF_GSM610_11kHzMono + 1;
public final short SPSF_GSM610_44kHzMono     = SPSF_GSM610_22kHzMono + 1;
public final short SPSF_NUM_FORMATS   = SPSF_GSM610_44kHzMono + 1;
```

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

ATTNV and MACOSXSPMAN ignore all format parameters. ATTNV currently only exports in PCM WAV format. MACOSXSPMAN currently only exports in AIFF format.

Return Values

0 is returned when successfully synthesized and the following **common** error codes are returned when errors occur:

[1] Used format that is not supported.
[2] Failed to secure channel memory
[3] Text string is a NULL pointer
[4] The length of text string is 0.
[5] The TTS DB of the voice requested is not loaded
[6] Failed to generate the synthesized voice file
.
.
.
[104] Other errors

Example (see sample)

```
// *** YOU MUST CHANGE THE WAV PATH TO MATCH YOURS! *** //

/*MAC OS X note: MACOSXSPMAN only AIFF format supported. All format args will be ignored*/

//JNILayer.MakeLanguage(LANG_ENGLISH, SUBLANG_ENGLISH_US) = 1033 - You can just use 1033

Object obj = comboBox1.getItemAt( comboBox1.getSelectedIndex() );

VoiceData vo = (VoiceData)obj;

String filepath = "test.wav";

if( vo.m_vet == MACOSXSPMAN )
      filepath = "test.aiff";

short format = 0;

if( vo.m_vet == NEOSPEECHVOICETEXT )
      format = VT_FILE_API_FMT_ALAW_WAVE;
else if( vo.m_vet == MSSAPI )
      format = SPSF_16kHz16BitStereo;

int result = JNILayer.DFTTSExportToFileEx(vo.m_vet,
  vo.m_sVoiceName, vo.m_VoiceID,
  (short)1033, textArea1.getText(), -1,
  -1, -1, -1,
  1, DFTTS_TEXT_TYPE_XML,
  filepath,
  format,
  "pcm16", 16000,
  2);

if (result != 0)
{

      JOptionPane.showMessageDialog(this, "SDK Export error: " + result);

}
```

GetDFTTSVoice

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

Synopsis

```
public native String GetDFTTSVoice( int iVoiceInfoIdIn,
                    int[] pvetOut,
                    int[] piVoiceNameLenOut,
                    int[] piVoiceIDOut,
                    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).

[OUT] piVoiceNameLenOut
The length of the returned voice name string.

[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.

Return Values

The voice name string (Java (TM) format).

Example (see sample)

```
//List installed voices

int z=0;

for( ;;z++ )
{

        int[] pvetOut = new int[] { -1 };

        int[] piVoiceNameLenOut = new int[] { -1 };

        int[] piVoiceIDOut = new int[] { -1 };

        short[] plangOut = new short[] { -1 };

        String szVoiceNameOut = JNILayer.GetDFTTSVoice( z,
                            pvetOut,
                            piVoiceNameLenOut,
                                piVoiceIDOut,
                                plangOut
                                );
```

```
        if( szVoiceNameOut.length() == 0 )
                  break;

        System.out.println("Voice Engine: " + pvetOut[0] + '\n');

        System.out.println("Voice Name: " + szVoiceNameOut + '\n');

        System.out.println("Voice Name Length: " + piVoiceNameLenOut[0] + '\n');

        System.out.println("Voice ID: " + piVoiceIDOut[0] + '\n');

        System.out.println("Language ID: " + plangOut[0] + '\n');

        //See class VoiceData

        comboBox1.addItem( new VoiceData(szVoiceNameOut,
                                pvetOut[0],
                                piVoiceIDOut[0],
                                plangOut[0]) );

}

if (z == 0)
{
JOptionPane.showMessageDialog(this, "No voices can be found on your system! Please download and
install at least 1 voice!");
}
```

Appendix 1

Java (TM) Layer Class

```
public class JNILayer
{
        public native String GetDFTTSVoice( int iVoiceInfoIdIn,
                  int[] pvetOut,
                  int[] piVoiceNameLenOut,
                      int[] piVoiceIDOut,
                      short[] plangOut
                      );


        public native void InitDFTTSEngineEx3(int hwndWinOwner,
                   String szNeoSpeechDBFolderPathKate,
                   String szNeoSpeechDBFolderPathPaul,
                   String szNeoSpeechDBFolderPathMiyu,
                   String szNeoSpeechDBFolderPathShow,
                   String szNeoSpeechDBFolderPathMisaki,
                   String szNeoSpeechDBFolderPathLily,
                   String szNeoSpeechDBFolderPathWang,
                   String szNeoSpeechDBFolderPathJunwoo,
                   String szNeoSpeechDBFolderPathSujin,
                   String szNeoSpeechDBFolderPathYumi,
                   String szNeoSpeechDBFolderPathGyuri,
                   String szNeoSpeechDBFolderPathDayoung,
                   String szNeoSpeechLicFilePathKate,
                   String szNeoSpeechLicFilePathPaul,
                   String szNeoSpeechLicFilePathMiyu,
                   String szNeoSpeechLicFilePathShow,
                   String szNeoSpeechLicFilePathMisaki,
                   String szNeoSpeechLicFilePathLily,
                   String szNeoSpeechLicFilePathWang,
                   String szNeoSpeechLicFilePathJunwoo,
                   String szNeoSpeechLicFilePathSujin,
                   String szNeoSpeechLicFilePathYumi,
                   String szNeoSpeechLicFilePathGyuri,
                   String szNeoSpeechLicFilePathDayoung,
                   int[] psiLoadedEngines,
                   int[] psiLoadedEnginesReturnValues
           );

       public native int UninitDFTTSEngine();

       public native short MakeLanguage( short mainlang,
                                  short sublang
                                  );

       public native short GetMainLanguage( short lang );

       public native short  GetSubLanguage( short lang );


        public native int LoadDFTTSUserDict(int iDictIndex, String szDictName,
                      String szDictFileName, int vet, short lang,
                      String szVoiceName, String szDictContents);

        public native int UnloadDFTTSUserDict(int iDictIndex, String szDictName,
                int vet, short lang);

       public native int DFTTSSpeak(int hwndWinOwner, int vet,
                   String szVoiceName, int iVoiceID, short lang, String szText, int iPitch,
                   int iSpeed, int iVolume, int iPause, int iDictID, int ttTextType,
```

```
                short ofOutPutFormat);

        public native int  DFTTSPause( int vet, short lang );

        public native int  DFTTSResume( int vet, short lang );

        public native int  DFTTSStop( int vet, short lang );

        public native int DFTTSExportToFileEx(int vet, String szVoiceName,
                int iVoiceID, short lang, String szText, int iPitch, int iSpeed,
                int iVolume, int iPause, int iDictID, int ttTextType, String szFilePath,
                short ffFileFormat, String szAudioEncoding, int iAudioSamplingRate,
                int iAudioChannels);

        static
        {


                System.loadLibrary("dfttsjni");



        }


//MACOSXSPMAN: onExport event data is sent here (onWord). MACOSXSPMAN treats exports as speech.

        public void onWord(int start, int end)
        {




        }

        //MACOSXSPMAN: Do not use. Event data is sent to onWord
        public void onExport(int start, int end)
        {
                //start and end are SDK-reserved


        }

}
```

Appendix 2

Controlling Speech with a Standard XML Tag Set

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.