



**voiXtreme**

**Especificaciones para la utilización de  
la librería de la voz**

voiXtreme\_dll\_es\_v100.doc

20080310 Versión 1.0 Es	
-------------------------	--

## Índice

<b>INTRODUCCIÓN</b> .....	<b>3</b>
RESUMEN .....	3
<b>PRESENTACIÓN DEL SISTEMA</b> .....	<b>3</b>
<b>EJEMPLOS DE UTILIZACIÓN</b> .....	<b>3</b>
<b>INICIALIZACIÓN, FUNCIONES DEFINIDAS Y SINTAXIS</b> .....	<b>4</b>
A.- TTS .....	5
B.- ASR.....	5
C.- ACTIVACIÓN DE LA PARTE DE LA GRAMÁTICA QUE CONTIENE SÓLO DÍGITOS .....	5
<b>SINTAXIS DE LAS FUNCIONES Y PROTOTIPOS</b> .....	<b>5</b>
A.- INICIALIZACIÓN, DESINICIALIZACIÓN Y CONFIGURACIÓN .....	5
B.- TTS.....	6
C.- ASR.....	6
<b>LICENCIAS DE USO</b> .....	<b>7</b>



## Introducción

### Resumen

Este documento presenta el modo de utilización del sistema de reconocimiento de la voz (*ASR – Automatic Speech Recognition*) y de la síntesis del habla (*TTS – Text to Speech*), a través de la librería `voixtreme.dll`.

El mismo está orientado al programador y es una versión preliminar, que presenta el conjunto mínimo de opciones necesario para su utilización inicial. El objetivo es incluirla dentro de la instalación de los componentes, con vistas a recibir feedback de los programadores y usuarios.

## Presentación del sistema

Denominaremos “*voiXtreme*”, “*SofToGo Voice Library*” o simplemente “*VoiceLib*” al ensamble de dos poderosos sistemas relacionados con el habla y los sistemas electrónicos: *ASR (Automatic Speech Recognition)* y *TTS (Text to Speech)*, junto a la posibilidad de acceder a sus métodos a través de una única dll desarrollada para Windows Mobile.

Básicamente, ASR permite escuchar una determinada alocución, procesarla sobre la base de una gramática predeterminada y devolver la cadena de texto -perteneciente a esa gramática- más cercana a la alocución pronunciada.

TTS realiza el proceso inverso. Hace las veces de “lector” de texto, de modo de poder escuchar un determinado texto que viene dado por una cadena de caracteres.

Se presenta aquí un manual del programador para acceder a las funcionalidades de `voixtreme.dll`. Cada uno de los componentes se presenta con el objeto de ser discutido, por lo que puede haber cambios entre las sucesivas versiones. Dichos cambios se traducirán en una actualización de este documento.

## Ejemplos de utilización

Junto a la instalación se presenta una aplicación (`voixtreme_cpp_test.exe`, con posibilidad de lanzarla desde “`Start/voiXtreme cpp Test`” o bien desde “`Start/Programs/voiXtreme cpp Test`”) desarrollada en C++, compilada con Microsoft Visual Studio 2005 (Professional Edition) Versión 8.0.50727.762 (SP.050727-7600).

Independientemente, se puede obtener el código fuente de esta aplicación y el de otra aplicación similar en Visual Basic desde el sitio <http://www.softogo.com>.

Se supone que el programador conoce la gramática instalada, que para este ejemplo es la siguiente:

```
!grammar Digit;
!start <Speech>;

<Speech>: !repeat(<Digit>,1,*) | <digitex> | <commande> ;

<Digit>: 0 !id(0) |
         1 !id(1) |
         2 !id(2) |
         3 !id(3) |
         4 !id(4) |
         5 !id(5) |
         6 !id(6) |
         7 !id(7) |
         8 !id(8) |
         9 !id(9) ;

<digitex>: repetir |
          anular;

<commande> : aceptar |
            comenzar |
            terminar ;
```

## Inicialización, funciones definidas y sintaxis

Después de cargar la dll con la función `LoadLibrary`, se debe inicializar la librería con la función `StgVoiceInit`.

Antes de terminar de utilizar la *Voice*, normalmente al salir de la aplicación se debe desinicializar la librería a través de la función `StgVoiceDeInit`.

```
// Inicialización.
m_hStgVoice = LoadLibrary( _T("stgvoice.dll") ) ;
m_StgVoiceSetWindow = (StgVoiceSetWindowFn)GetProcAddress(
m_hStgVoice, _T("StgVoiceSetWindow"));
m_StgVoiceSetWindow( m_hWnd ) ;
m_StgVoiceInit = (StgVoiceInitFn)GetProcAddress( m_hStgVoice,
_T("StgVoiceInit") );
m_StgVoiceInit( ) ;
m_StgVoiceSay = (StgVoiceSayFn)GetProcAddress( m_hStgVoice,
_T("StgVoiceSay") );
m_StgVoiceListen = (StgVoiceListenFn)GetProcAddress( m_hStgVoice,
_T("StgVoiceListen") );
m_StgVoiceActivateRule = (StgVoiceActivateRuleFn)GetProcAddress(
m_hStgVoice, _T("StgVoiceActivateRule") );
m_StgVoiceDeactivateRule = (StgVoiceDeactivateRuleFn)GetProcAddress(
m_hStgVoice, _T("StgVoiceDeactivateRule") );

// Deinicialización
m_StgVoiceDeInit = (StgVoiceDeInitFn)GetProcAddress( m_hStgVoice,
_T("StgVoiceDeInit") );
m_StgVoiceDeInit( ) ;
```



Ejemplos de llamados

### a.- TTS

```
m_StgVoiceSay( _T("Hola, bienvenido a esta aplicación.") ) ;
```

### b.- ASR

Cuando se llama a:

```
m_StgVoiceListen( ) ;
```

Se espera el mensaje WMU\_RESULT, que se envía a la ventana (configurada a través de un llamado a StgVoiceSetWindow) para recibir la string resultado del ASR.

```
#define WMU_RESULT (WM_USER + 4)
```

Especificación de la función llamada cuando llega el mensaje WMU\_RESULT  
ON\_MESSAGE( WMU\_RESULT, &CdlltestDlg::OnResultGiven )

Cuando llega el mensaje, se llama a:

```
LRESULT CdlltestDlg::OnResultGiven(WPARAM wParam, LPARAM lParam)  
{  
    wchar_t * strResult = (wchar_t *) lParam ;  
    SetDlgItemText( IDC_EDIT1, strResult ) ;  
    return 0 ;  
}
```

que setea el texto en el campo IDC\_EDIT1.

### c.- Activación de la parte de la gramática que contiene sólo dígitos

```
m_StgVoiceActivateRule( _T("<Digit>") ) ;  
m_StgVoiceDeactivateRule( _T("<commande>") ) ;  
m_StgVoiceDeactivateRule( _T("<digitex>") ) ;
```

## Sintaxis de las funciones y prototipos

### a.- Inicialización, desinicialización y configuración

```
STDAPI StgVoiceInit( void ) ;
```

Inicializa la librería de la Voice (ASR+TTS). Es necesario llamar a esta función antes de una llamada a cualquier otra función.

```
STDAPI StgVoiceDeInit( void ) ;
```

Contraparte de StgVoiceInit. Se debe llamar a esta función cuando no se va a utilizar más FlexBorwserVoice.

```
STDAPI StgVoiceSetWindow( HWND p_hWnd ) ;
```

Setea la ventana que recibirá el mensaje WMU\_RESULT con el resultado del reconocimiento.



STDAPI `StgVoiceSetDebuggingMode( bool p_bShowMessageBoxes ) ;`  
Permite la activación del modo de debugging. En el modo debugging, cada error de ejecución lanza un mensaje de texto informando del error. Por defecto está desactivado.

STDAPI `StgVoiceActivateRule( wchar_t * p_strRule ) ;`  
Activa una regla de la gramática. Esto permite que la función `Listen` pueda devolver una cadena de caracteres, una de cuyas partes pudo haber sido generada por esta regla.

STDAPI `StgVoiceDeactivateRule( wchar_t * p_strRule ) ;`  
Desactiva una regla de la gramática. Esto impide que la función `Listen` pueda devolver una cadena de caracteres en la que una de sus partes haya sido generada por esta regla.

## **b.- TTS**

STDAPI `StgVoiceSay( wchar_t * p_strToSay ) ;`  
Lee la cadena de caracteres pasada por parámetro y devuelve un código de error, 0 si ok.

STDAPI `StgVoiceSpell( wchar_t * p_strToSpell ) ;`  
Deletrea la cadena de caracteres numérica pasada por parámetro y devuelve un código de error, 0 si ok.

STDAPI `StgVoiceIsSpeaking( void ) ;`  
Devuelve un valor booleano indicando si el TTS está activo pronunciando o deletreando un texto.

STDAPI `StgVoiceStopSpeaking( void ) ;`  
Finaliza abruptamente la pronunciación o el deletreo de una frase.

STDAPI `StgVoiceSaySync( wchar_t * p_strToSay ) ;`  
Lee la cadena de caracteres pasada por parámetro y devuelve un código de error, 0 si ok. No devuelve el control hasta que se haya pronunciado toda la cadena de texto pasada como parámetro.

STDAPI `StgVoiceSpellSync( wchar_t * p_strToSpell ) ;`  
Deletrea la cadena de caracteres numérica pasada por parámetro y devuelve un código de error, 0 si ok. No devuelve el control hasta que se haya pronunciado toda la cadena de texto pasada como parámetro.

## **c.- ASR**

STDAPI `StgVoiceListen( void ) ;`  
Lanza el reconocedor de la voz del usuario. La función es asincrónica, en el sentido de que el sistema lanza el reconocedor y devuelve inmediatamente el control, permitiendo seguir con la ejecución de las sentencias sucesivas. El programador debe programar el handler para el tratamiento del mensaje `WMU_RESULT`, momento en el que el ASR devuelve la cadena de texto reconocida. Opcionalmente es posible esperar el final del reconocimiento testeando la función `StgVoiceIsListening`, hasta que devuelva `false`, momento en el que la cadena resultante se puede obtener mediante la función `StgVoiceGetString`.



STDAPI `StgVoiceListenSync( wchar_t * p_strResult ) ;`  
Devuelve una cadena de caracteres, producto del reconocimiento de la voz del usuario. La función es sincrónica, en el sentido de que el sistema queda esperando el resultado de esta función para seguir con la ejecución de las sentencias sucesivas.

STDAPI `StgVoiceIsListening( void )`  
Devuelve un valor booleano indicando si el ASR está activo recibiendo input del sistema de audio.

STDAPI `StgVoiceGetResultString( wchar_t * p_strResult ) ;`  
Devuelve la última cadena de texto reconocida por el ASR.

STDAPI `StgVoiceStopListening( void ) ;`  
Finaliza abruptamente el reconocedor de la voz sin devolver un resultado.

Los parámetros del sistema vocal están preconfigurados con opciones por defecto. Los métodos y propiedades que permiten una configuración más detallada se irán agregando en sucesivas versiones.

## Licencias de Uso

Mientras la librería no esté registrada, al lanzar la función `Init` se muestra un cartel indicando la ausencia de registración. En este estado tanto el ASR cuanto el TTS modifica todas las apariciones de los caracteres A y los dígitos 0 (cero), devolviéndolos o leyéndolos como E y como 5 (cinco) respectivamente.

La librería se registra a partir del acceso directo `voiXtremeLic` (`STGVoiceLic.exe`) directamente desde la terminal Windows Mobile, o a través del acceso directo, en la PC: `Start/Programs/WireLess Utilities/voiXtreme Libraries/Es/Licenciar (las PDA)`.

Para más información sobre cómo licenciar las librerías, refiérase a la documentación "Licencias Leame (pdf)", en el mismo directorio.