LumenVox's Speech Recognition Engine is a flexible API that performs speech recognition on audio data from any audio source.

The Speech Engine is speaker and hardware independent: it supports SRGS and SISR on both Windows and Linux platforms.

The Speech Engine provides speech application developers with an efficient development and runtime platform, allowing for dynamic language, grammar, audio format, and logging capabilities to customize every step of their applications. Grammars are entered as a simple list of words or pronunciations, or in the industry standard Speech Recognition Grammar Specification (SRGS), as defined by the W3C.

Just 13 lines of code (8 calls to the Speech Engine) will implement a simple "yes-no" speech recognition system. The system must provide the audio and the audio data length for SoundData and SoundDataLength.

### Sample Code

```c
void RecognizeSpeech (void* SoundData, int SoundDataLength)
{
    const char* GrammarString =
        "#ABNF 1.0
"        "language en-US;
"        "mode voice;
"        "tag-format <lumenvox/1.0>;
"        "$yes = (yes | yeah | okay):'true';
"        "$no = (nope | no):'false';
"    LVSpeechPort Port;
    Port.OpenPort ();Port.LoadGrammarFromBuffer (0, GrammarString);Port.LoadVoiceChannel (0, SoundData,SoundDataLength, ULAW_8KHZ);
    Port.Decode (0, 0, LV_DECODE_SEMANTIC_INTERPRETATION      |
                  LV_DECODE_BLOCK                                         );
    int NumInterpretations = Port.GetNumberOfInterpretations (0);
    for (int i = 0; i < NumInterpretations; ++i)
    {
        cout << Port.GetInterpretationString (0,i);
    }
    Port.ClosePort ();
}
```

Use the Speech Engine API in 5 Easy Steps!

1. Open a new LVSpeechPort representing your connection to the Speech Engine.
2. Create the grammar using SRGS ABNF/XML format and optionally, semantic interpretations. Each SpeechPort can have multiple grammars loaded at the same time.
3. Load your sound data into a voice channel using LoadVoiceChannel(); make sure to specify the encoding type. Each port also has multiple sound channels for multiple incoming sound sources.
4. Call Decode() with the new sound channel and grammar.
5. Get the recognized results.
Supporting Standards

LumenVox supports the W3C’s Speech Recognition Grammar Specification (SRGS), part of the VoiceXML 2.0 and SALT specifications. Companies that track these specifications are dedicated to the future of speech, and to integrating with other companies committed to promoting speech recognition. The LumenVox SRGS implementation is backward compatible with the existing LumenVox BNF grammar format; current deployments will leverage the power of the SRGS system immediately and transparently.

LumenVox recognizes that the speech community will need to work together to develop solutions for businesses, and as such, LumenVox applications complement the following technologies:

**VXML**
VoiceXML (VXML) is a mark-up language designed to code speech applications with many of the same architectural components as HTML. VoiceXML platforms connect to a combination of speech recognition engines, text-to-speech synthesis, telephony interfaces and VoiceXML interpreter software to process the call. In order to interface VXML with any speech engine, the engine must understand SRGS and SISR.

LumenVox’s Speech Engine is compliant with what VXML expects, and our engine powers the speech recognition portion of several VXML platforms.

**SALT**
Speech Application Language Tags (SALT) is similar to VoiceXML but also adds support for multi-modal systems. SALT extends existing mark-up languages such as XHTML, XML, and HTML. Similar to our work with VXML, the LumenVox Speech Recognition Engine conforms to SALT specifications.

**Semantic Interpretation**
LumenVox has implemented the W3C’s Semantic Interpretation for Speech Recognition (SISR) working draft, also part of the VoiceXML 2.0 specification. SISR allows grammar authors to embed snippets of JavaScript code into their SRGS grammars, to automatically transform what a speaker says into a format understandable to an application. With LumenVox’s Semantic Tags, callers can say, “September thirteenth two thousand four,” and your application will understand “2004-09-13.”

LumenVox is committed to supporting the W3C’s working draft. As the draft evolves, we will support both new and old drafts, so application developers can be confident that their grammars and tags will perform to specification.
Advanced Features

Noise Reduction Module

When noise is present, it will degrade the performance of any speech recognition system. Quality noise reduction improves the accuracy of Voice Activity Detection and Core Recognition, both essential parts of a speech recognition system.

To improve application robustness in noisy environments, LumenVox implemented a Noise Reduction Module (NRM) into our Speech Recognition Engine. The NRM automatically adapts to the acoustic environment, and dynamically updates its estimate of noise levels. The adaptive algorithm enables the NRM to reduce the effects of noise.

The waveforms below demonstrate the power of LumenVox’s Noise Reduction Module. In the original audio [Fig. 1], a truck driver speaks on a cell phone while driving. In addition to noise from the truck engine and blowing wind, another vehicle engine starts in the middle of the recording. Although traditional noise reduction implementations often fail to adapt to such dramatic changes, LumenVox’s NRM adjusts to the new noise characteristics rapidly and automatically. [Fig. 2]

Voice Activity Detection

Voice Activity Detection (VAD), also referred to as barge-in and/or End-Of-Speech (EOS) detection, identifies when a person begins speaking, finishes speaking, or pauses while speaking.

LumenVox’s VAD implementation delivers high performance despite challenging conditions: hisses, pops, abrupt changes in background noise, telephone line echo, and squawks from two-way radio communication.

The Voice Activity Detection module is highly configurable and can be adapted to work equally well within telephone, VoIP, or microphone-based applications.

Media Resource Control Protocol (MRCP)

Speech synthesizers...Audio recorders...DTMF recognizers...Speech recognizers...Speech verifiers...a fully functioning, media-rich application needs lots of components to work together. Until now, all of these components had to be provided by a single vendor, or required extensive custom programming to integrate them. MRCP changes all this. The Media Resource Control Protocol allows you to seamlessly manage diverse media resources. MRCP provides a common language to speak to all of these devices.

With MRCP, vendors can compete on the basis of their strengths, rather than attempting to create an all-inclusive, yet mediocre package. Customers can take the best product from each vendor, creating a speech application package that is tailored to their particular needs.


n-best Results

Instead of returning only the top scoring result, you can instruct the engine to return several of the highest scoring, most likely answers, often called n-best results. Returning n-best results is particularly effective when callers need to spell names, street addresses, or e-mail addresses. Without n-best results, if a caller spells a name beginning with “N,” but the engine returns a low confidence score, the caller would be asked to repeat the letter-and given how similar “N” is to “M,” it’s likely that the second answer would have a similarly low confidence score. With n-best results, the system can prompt the caller using several of the likely results, such as “Did you mean M, as in ‘Mary’?” When the caller responds, “No,” the system goes to its next option, “Perhaps you meant ‘N’ as in ‘Nancy’?”

Returning n-best results improves the caller’s experience: instead of asking the caller to simply repeat an answer that received a low confidence score, the system can confirm the caller’s intention using several likely choices.

Server-Side Grammar

LumenVox offers even more efficient support for large grammars, by allowing clients to pre-load grammars onto the server, allowing users to send the grammar prior to the decode requests.

Typically, the grammar itself accompanies each decode request, but in the case of large grammars, sending the grammar to the server prior to decoding is more efficient—reducing network traffic.

We are delighted that LumenVox is extending the capabilities of our platform.

John Hibel   Vocalocity’s Vice President of Marketing and Business Development