

Microsoft Speech SDK (SAP1.1)

File Edit View Search Favorites |

Type in the word(s) to search for:

voice mail

Selected topic: Found 3

Title	Location	Rank
Speech Telephone Ap...	Microsoft Spa...	1
Promoting Recognized	Microsoft Spa...	2

Search previous results  
 Match similar words  
 Speech titles only

Stop Refresh Home Font Print Desktop

Microsoft Speech SDK  
Speech Automation 5.1

Using Way File Input with Speech Recognition Engines

**Overview**

This document is intended to help developers of speech recognition (SR) applications use the Microsoft Speech API's speech recognition and audio APIs to connect a wav file with an SR engine. The topics covered include:

- Typical file input scenario
- In-process (InProc) versus shared engines
- Relevant to setting up and using wav files as input to the speech recognizer
- Sample source code (written in both C++ and Visual Basic 6.0) to help guide developers.

**Typical file input scenario**

There are many different types of audio input configurations used by SR applications, which include:

- A microphone shared by all desktop applications
- A telephony card communicating with one or more SR engines
- Sending audio from a persisted wav file to an SR engine

The shared desktop microphone scenario uses the default SR engine and the default audio input. The user selects each in Speech properties in Control Panel, and each is hosted in the shared speech server.

The telephony scenario can use either the SAP1's standard multimedia audio input object or a custom audio object combined with an InProc SR engine.

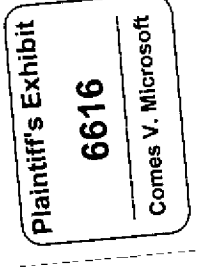
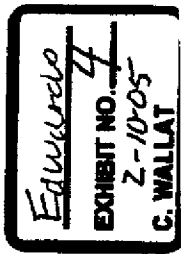
The wav file input scenario is special because it uses controlled, reproducible audio input and requires a dedicated SR engine, without interference from other applications (e.g., a shared desktop microphone). The file input scenario should use a generic SAP1 audio stream connected to the input wav file and an InProc SR engine.

**Typical scenarios that would use the wav file audio input configuration include:**

- Offline transcription applications (e.g., convert [redacted] to email)
- SR engine testing (e.g., measure and improve engine accuracy with reproducible audio input data)
- SR application testing (e.g., verify and improve application behavior when responding to reproducible voice commands)

Follow these basic steps to perform SR on a wav file:

1. Create and configure basic SAP1 audio stream object for wav file input
2. Create an InProc SR engine using the code samples in this document
3. Set the audio stream object from step 1 as the SR engine's input



4. Activate grammars and begin SR
5. Respond to recognition events until end of audio stream is reached

**Relevant wav audio file input APIs for COM/C/C++ Developers:**

- SpStream object, ISpStream interface: Basic SAPI audio stream
- ISpStream::BindToFile: Set up audio stream for wav file input
- SpBindToFile: Helper function to setup stream with a wav file
- SpInProcRecognizer, ISpRecognizer: InProc SR engine
- ISpRecognizer::SetInput: Set stream object as engine's input
- SPEL\_START\_SR\_STREAM, SPEL\_END\_SR\_STREAM: Event signaling engine has reached the beginning or the end of the wav file, respectively

**Relevant wav audio file input APIs for Automation/Visual Basic/Scripting Developers:**

- SpFileStream object: Basic file-based SAPI audio stream
- SpInProcRecognizer, ISpRecognizer: InProc SR engine
- SpInProcRecoContext, ISpRecoContext: InProc SR context
- ISpRecoContext::AudioInputStream property: Set file stream object as engine's input
- ISpRecoContext::EndStream/StartStream events

**Wav audio file input outcome specific to BAPI**

**Finite-length audio input stream**

Unlike microphone input which has no predetermined stream length, a finite-length audio input stream is a file which has a specific length that is known before recognition begins. Similarly, applications that use microphone input will toggle between actively listening and not listening states until the speech application is closed. However, transcription applications are typically designed to listen to one continuous audio stream, and then close when the stream ends. Consequently, the application must specifically acknowledge the end audio stream event (SPEL\_SR\_END\_STREAM for C/C++, ISpRecoContext::EndStream event for Automation). Transcription applications can potentially record multiple recognitions on a single audio stream, if the speaker pauses or breaths between sections of audio. If the transcription application exits after the first recognition event is received, it will miss any further recognizable audio that remains.

**Non-real-time audio input**

Microphone input and networked audio streams are typically real-time audio objects. This means that the audio object is designed to support audio buffering and dynamic state manipulation (e.g. stop->play->pause->play->stop) to handle delays and latency in the audio source and/or the SR engine's processing.

**Sample wav audio file input source code**

COM/C++ Developers

voice.midi

Search previous results  
 Match similar words  
 Search files only

Select topic: Found 3

Title	Location	Rank
Speech Technology Ac...	Microsoft Spa...	1
Perceiving Recognized	Microsoft Spa...	2
Using Wave File Input ...	Microsoft Spa...	3

Type in the word(s) to search for:

Voice mail | List Topics | Found 3 | Display

Title	Location	Rank
Speech Technology Ap...	Microsoft Spa...	1
Pending Recognized ...	Microsoft Spa...	2
Using Wave File Input ...	Microsoft Spa...	3

Search previous results  
 Match similar words  
 Search files only

C-style is very similar to C++ and COM

```

CCmptr<ISpStream> cpInputStream;
CCmptr<ISpRecognizer> cpRecognizer;
CCmptr<ISpRecoContext> cpRecoContext;
CCmptr<ISpRecoGrammar> cpRecoGrammar;

// Create basic SAPI stream object
// NOTE: The helper SpInFile can be used to perform the following operations
hr = cpInputStream.CoCreateInstance(CLSID_SpStream);
// Check hr

CSpStreamFormat sInputFormat;
// generate WaveFormatEx structure, assuming the wav format is 22kHz, 16-bit, Stereo
hr = sInputFormat.AssignFormat(SPF_22kHz16BitStereo);
// Check hr

// setup stream object with wav file NY_WAVE_AUDIO_FILENAME
// for read-only access, since it will only be access by the SR engine
hr = cpInputStream->BindToFile(NY_WAVE_AUDIO_FILENAME,
    SPF_OPEN_READONLY,
    sInputFormat.FormatId(),
    sInputFormat.WaveFormatExPtr(),
    SPPEI_ALL_EVENTS);
// Check hr

// Create 16-process speech recognition engine
hr = cpRecognizer.CoCreateInstance(CLSID_SpInProcRecognizer);
// Check hr

// connect wav input to recognizer
// SAPI will negotiate mismatched engine/input audio formats using system audio codec
hr = cpRecognizer->SetInput(cpInputStream, TRUE);
// Check hr

// Create recognition context to receive events
hr = cpRecognizer->CreateRecoContext(cpRecoContext);
// Check hr

// Create grammar, and load dictation
// ignore grammar ID for simplicity's sake
// ...
    
```

Speech Topics: Found 3

Title	Location	Rank
Speech Technology Ap...	Microsoft Spk...	1
Preparing Recognized	Microsoft Spk...	2
Using Wave File Input...	Microsoft Spk...	3

Search previous results  
 Match similar words  
 Search titles only

```

// NOT: Voice command apps would load CFG here
hr = cpRecoRecognizer->CreateGrammar(NULL, &cpRecoGrammar);
// Check hr
hr = cpRecoGrammar->LoadDictation(NULL, SPO_STATIC);
// Check hr

// check for recognitions and end of stream event
hr = cpRecoContext->SetInterest(SPEI_SPEI_RECOGNITION) | SPEI_SPEI_SR_END_STREAM);

// use Win32 events for command-line style application
hr = cpRecoContext->SetNotifyWin32Event();
// Check hr

// activate dictation, and begin recognition
hr = cpRecoGrammar->SetDictationState(SPRS_ACTIVE);
// Check hr

// while events occur, continue processing
// timeout should be greater than the audio stream length, or a reasonable amount of t
BOOL fEndStreamReached = FALSE;
while (!fEndStreamReached && S_OK == cpRecoContext->WaitForNotifyEvent(MV_REASONABLE_)
{
    CSpEvent spEvent;
    // pull all queued events from the reco context's event queue
    while (!fEndStreamReached && S_OK == spEvent.GetFrom(cpRecoContext))
    {
        // Check event type
        switch (spEvent.evtId)
        {
            // speech recognition engine recognized some audio
            case SPEI_RECOGNITION:
                // TODO: log/report recognized text
                break;

            // end of the wav file was reached by the speech recognition engine
            case SPEI_SR_END_STREAM:
                fEndStreamReached = TRUE;
                break;
        }
    }
    // clear any event data/object references
    spEvent.Clear();
}
    
```



Contents | Update | Search | Favorites  
Type in the search() to search for:  
"voice mail"

Selected topic: Found 3

Title	Location	Rank
Speech Telephony Ap...	Microsoft Spa...	1
Perceiving Recognized...	Microsoft Spa...	2
Using Wave File Input...	Microsoft Spa...	3

Search previous results  
Match entire words  
Search files only

```
// clear any event data/object references
spEvent.Clear();
/// END event polling loop - break on empty event queue OR end stream
/// END event polling loop - break on event timeout OR end stream

// deactivate dictation
hr = spRecoGrammar->SetDictationState (SPRS_INACTIVE);
// Check hr

// unload dictation topic
hr = spRecoGrammar->UnloadDictation();
// Check hr

// close the input stream, since we're done with it
// NOTE: smart pointer will call spStream's destructor, and consequently :Close, but
hr = spInputStream->Close();
// Check hr
```

### Automation/Visual Basic 6.0 Developers

Scripting code is similar to Visual Basic.

```
Option Explicit
Dim WithEvents RecoContext as ISpeechRecoContext ' context for receiving SR events
Dim Grammar as ISpeechRecoGrammar ' grammar
Dim InputFile as SpeechLib.SFileStream ' wav audio input file stream

' Setup InProc recg context and wav audio input file
Private Sub MyForm_Load()
    ' Create new recognizer
    Dim Recognizer as New SpinproRecoRecognizer
    ' create input file stream
    Set InputFile as New SpFileStream
    ' Defaults to open for read-only, and DoEvents false
    InputFile.Open MY_WAVE_AUDIO_FILENAME

    ' connect wav audio input to speech recognition engine
    Set Recognizer.AudioInputStream = InputFile

    ' create recognition context
    Set RecoContext = Recognizer.CreateRecoContext
```

Title	Location	Rank
Speech Desktop Ap...	Microsoft Spa...	1
Penning Recognize...	Microsoft Spa...	2
User Wave File Inp...	Microsoft Spa...	3

```

' Create new recognizer
Dim Recognizer as New SpInProcRecognizer

' create input file stream
Set InputFile as New SpFileStream
' Defaults to open for read-only, and noEvents false
InputFile.Open MY_WAVE_AUDIO_FILENAME

' connect wav audio input to speech recognition engine
Set Recognizer.AudioInputStream = InputFile

' create recognition context
Set RecoContext = Recognizer.CreateRecoContext

' create grammar
Set Grammar = RecoContext.CreateGrammar
' ... and load dictation
Grammar.DictationLoad

' start dictating
Grammar.DictationSetState SOPSActive
End Sub

' Event fired on app shutdown
Private Sub MyForm_Unload(Cancel as Boolean)
InputFile.Close ' close audio input file
End Sub

' Event fired when speech recognition engine recognizes audio
Private Sub RecoContext_Recognition(StreamNumber as Long, ScreenPosition as Variant, Recr
' Log/Report recognized phrase/information
End Sub

' End of wav Input Stream reached by speech recognition engine
Private Sub RecoContext_EndStream(StreamNumber as Long, ScreenPosition as Variant)
' Disable dictation and unload grammars on app close
Grammar.DictationSetState SOPSInactive
Grammar.DictationUnload

Unload Me ' shutdown app on end of stream
End Sub
    
```