Voice Interoperability Initiative Architecture Series Whitepapers

Voice Input Data Handling on Multi-Agent Devices

July 21, 2021

Contents

Abstract

An end-user interacts with a voice agent through conversation. An agent is like a digital "person" that has its own voice and personality (brand), method of invocation, such as a custom wake word, and one or more capabilities. These capabilities are supported through functions on the device and cloud-based voice services. Cloud-based voice services provide information such as weather and facts, entertainment through podcasts, gaming, streaming music and videos, capabilities for shopping and purchasing, setting alarms and reminders, and much more. A multi-agent product is one that offers multiple agents with multiple voice services on a single device. Multi-agent products benefit users by providing simultaneous access to multiple complementary voice services. Users have the flexibility to choose their preferred agent to perform certain tasks.

A user may speak a wake word to begin a dialog with the corresponding agent. They may also speak about an agent and refer to its wake word when conversing with another agent. This latter use case may require special handling of potential wake word engine triggers to ensure that voice input data continue streaming to the originally invoked agent's cloud services. Additionally, agents may have specific voice processing requirements for their wake word engine and cloud services.

This whitepaper discusses wake words, the role of the audio front end, wake word engines, and methods for directing appropriately processed and formatted voice input to an agent's cloud-side voice services. It is part of a series that provides information for consideration when building products that support simultaneously available voice agents consistent with the Voice Interoperability Initiative (VII) Multi-Agent Design Guide. It is intended for technical architects, device maker engineers, and voice agent developers. You will benefit by already having familiarity with VII and the Multi-Agent Design Guide for many of the terms and concepts in this whitepaper.

Related topics that are not covered in this whitepaper include: on-device natural language processing, coordination of multi-agent audio output, voice data in human-to-human voice or video calls, multi-agent systems comprised of multiple synchronized devices, build costs, and latency concerns.

Overview

The Voice Interoperability Initiative (VII) is committed to providing customers choice and flexibility to interact with multiple simultaneously available voice services. To support you in creating multi-agent experiences that align with the Multi-Agent Design Guide, this whitepaper gives suggestions for directing appropriately processed and formatted voice input data to the invoked agent's cloud services.

Multi-agent products come in a variety of physical configurations, but share common, industry-wide architectures and components that are also found in single-agent devices. These components include an audio front end, a wake word engine, and client software for managing agent behavior and communicating with agent cloud-side voice services. Each of these components and related software topologies for directing voice input data are discussed and illustrated in the following sections.

Terminology

These terms are defined in the context of voice-enabled multi-agent devices.

Also refer to terms in the Multi-Agent Design Guide.

Active Agent
The agent that is currently in the listening, thinking or speaking state.

Available Agent
An agent that has been enabled and can be invoked on the device.

Acoustic Echo Cancellation (AEC)
A process implemented on audio front ends to remove echo, reverberation and sounds introduced into the environment such as those produced by speakers in smart devices.

Acoustic Signal Processing
The application of audio algorithms to extract certain information such as the user's voice from acoustic signals in the presence of noise and background conversations.

Applications Processor (AP)
The main processor on the device that runs application programs such as agent client software. It is also known as the host processor and sometimes loosely referred to the as the CPU.

Audio Algorithms
Techniques such as AEC, noise reduction, sound source separation and beam forming implemented in software.

Audio Format
The bit layout of sampled audio data for one or more input or output channels in Little- or Big-Endian byte order. It may be encoded, compressed or include a header with metadata about the audio, and formatted into a common format such as Pulse Coded Modulation (PCM) or Opus with single or multiple outputs. Voice input data is typically 16-bits wide data and sampled at 16,000 samples/second.

Audio Front End (AFE)
The audio front end is a separate hardware or software component that implements audio algorithms to improve speech content and quality as required by specific agents.

Automatic Speech Recognition (ASR)
The identification and translation of spoken language into text.

Cloud-Based Wake Word Verification (CBWWV)
A cloud service that verifies a wake word. When wake word verification fails, a notification is sent to the agent software on the device to exit the listening state and return to the idle state. Otherwise, cloud processing proceeds on behalf of the agent whose wake word was detected.

Digital Signal Processor (DSP)
A specialized processor whose architecture is optimized for performing mathematical functions such as “add”, “subtract”, “multiply”, “divide” and the most common, “MAC” (Multiply and Accumulate). DSPs can run AFE audio algorithms and WWEs with lower power consumption than applications processors.

Noise Reduction (NR)
An audio processing step that uses acoustic signal processing techniques to reduce undesirable content such as noise to improve the speech to noise ratio.

Single Agent Listening State
A condition where only one agent on a multi-agent device is in the listening state.

Voice Activity Detection (VAD)
The detection of the presence or absence of human speech to facilitate speech processing.

Wake Word
A word or phrase a user speaks to “wake up” (invoke) a specific agent, get its attention and have it start listening.

Wake Word Engine (WWE)
A component that is responsible for detecting agent-specific wake words in a voice audio stream. A WWE can be embedded in a separate processor, such as a digital signal processor or co-processor with/without a neural accelerator, or run on the applications processor.

Wake Word Model
A neural network or machine learning data representation of one or more wake words.

Wake Word Triggers

A wake word spoken at the beginning of a user interaction is intended to wake the corresponding agent such that it responds to the subsequent inquiry or request (the utterance). After the wake word is detected, the voice utterance data are (1) captured and streamed to its cloud services (2) for natural language processing, where (3) a text-to-speech (TTS) response may be streamed back to the device and output by the speakers. These phases correspond to an active agent in the listening, thinking or speaking states, respectively. At all other times, the agent is in the idle state.

Wake Words within Utterances

Wake words may also be spoken within the utterance, such as when a user is asking an agent about another agent. Developers of VII multi-agent devices should be careful to ensure that such mid-utterance wake words do not invoke the corresponding agent, where undesirable behaviors may occur.

Table 1 provides examples of undesirable behaviors that may occur when listening and speaking states are not coordinated for two fictitious agents *Albert* and *Amanda*. Wake words are shown underlined and trigger the respective WWE in all cases.

Use Case	User Speech	Desired Behavior	Undesired Behavior
1	Albert, is it true that agent Amanda is better for driving directions?	Albert triggers on “Albert”, receives the full utterance, and responds with “Yes, it is true.”	Amanda triggers on “Amanda”, receives “is better for driving directions”, and simultaneously responds with “I do not understand” at the same time Albert is responding for the desired behavior.
2	Albert, what do you know about the Amanda voice assistant?	Albert triggers on “Albert”, receives the full utterance, and responds with “Amanda is a voice assistant that is great at providing driving directions.”	Amanda triggers on “Amanda”, receives “voice assistant” and responds with “I do not understand” at the same time Albert is responding for the desired behavior. Amanda triggers again for Albert’s TTS output of “Amanda” via the speakers and receives “is a voice assistant that is great at providing driving directions.”, to which it responds again with “I do not understand.”

Table 1 - Wake Words within Utterance Use Cases

These confusing customer experiences demonstrate the need for device software to aid in the coordination and handling of agent listening and speaking states. For example, agents may use middleware to coordinate a single agent listening state, where only one agent may be in the listening state at any time.

Wake Words within Audio Output

An agent's wake word spoken within the audio output by the device speakers may get captured by the microphones, pass through the audio front end to the WWE, and result in a trigger. The wake word may originate from media content such as a TV ad about the agent or an agent's TTS. On multi-agent devices, there is increased potential for unintentional wakes due to more wake words that may be spoken through TTS or media. Preventing unintentional wakes and invocations is important in order to reduce frustration and maintain Customer Trust.

An agent's cloud services may embed metadata (a "fingerprint") for wake words in the generated TTS sent to the device. This fingerprint is detected by the agent's WWE, which acts as a signal to ignore the wake word when the agent is already in the listening state.

The most common method for addressing audio output feedback into the audio input system is Acoustic Echo Cancellation (AEC), implemented by the Audio Front End (AFE). AEC essentially subtracts out the speaker output "heard" by the microphones. However, AEC effectiveness can vary considerably. When media playback volume level is high relative to the speech content, AEC may fail to cancel enough of the wake word and result in an unintended WWE trigger.

To counter failures by AEC to remove a wake word from audio output, the output to the speakers can also be channeled as input to additional connected WWEs. Each "output" WWE can then detect its supported wake words and signal its client software to ignore the coincident detection, separate from the WWEs used for wake words spoken by the user. However, this approach does require additional system resources (CPU, memory) to run the additional WWEs.

Audio Front End

The audio front end (AFE) is a separate component on the device that typically applies acoustic signal processing techniques to the audio input signal from the microphones to improve the quality of the speech content for ASR and potentially a WWE. On multi-agent devices, the AFE may need to provide multiple, simultaneously available voice inputs for the enabled agents, each potentially requiring different data processing and formats. When an agent is invoked, the appropriately processed audio input data must be directed to the agent's cloud voice services.

An AFE may be a hardware or software component. A hardware AFE is typically comprised of one or more digital signal processors (DSPs) that run the audio algorithms for AEC, noise reduction (NR), sound source separation and beam forming to hone in on the direction of the user's speech. DSPs are optimized for the acoustic signal processing algorithms and consume less power than an applications processor (AP), making them ideal for portable devices and low-power, always-on modes. Software AFEs run the same or similar audio algorithms, but on the AP, trading power efficiency for the benefit of not incurring the additional cost of DSPs.

A hardware AFE may also run one or more WWEs on its DSPs. These are particularly beneficial for low-powered devices and when the operating system on the AP is unable to simultaneously share audio input across multiple agent applications. Audio input data switching techniques implemented in the WWE DSP may be used to overcome this limitation, as described in WWE on a DSP.

The flexibility of an agent's voice-dependent components may also affect the audio pathways and handling of voice input data. For example, an agent's WWE may tolerate or even benefit from more intensively processed audio with AEC and NR, whereas its cloud ASR may require raw or AEC-only multi-microphone input audio. These aspects should be taken into consideration when designing the audio routing of voice input data provided by the AFE.

WWE on a DSP

A WWE may run on a DSP for power conservation or to provide an application program running on the AP simultaneous access to audio input for certain operating systems. A WWE running on a DSP must coordinate the timing and transfer of voice data from the AFE to software running on the AP. This coordination involves several operational modes. The operational modes control whether the DSP is in a sleep mode to conserve power. Some WWE DSPs may include other modes to control whether the WWE is temporarily disabled until a stop capture signal is received from the client software, indicating that the listening state is complete. WWE DSPs may also offer to buffer or stream utterance data. Buffered utterance data accumulate audio input for short intervals and then send it in bursts to the AP to reduce AP overhead and converse power, but introduce slight latencies. Whereas streamed audio data require continual interactivity with software on the AP to transfer data in real-time, but may introduce additional AP overhead and consume additional power. While a WWE on a DSP may be ideal for a portable device, it adds software complexity on the AP to coordinate operational modes.

Voice Audio Routing

Several key components on multi-agent devices play a crucial role in ensuring that voice input data are directed to the intended agent's cloud services. Client software on the device controls the selection of appropriately processed and formatted voice input data from the AFE and directs it to the appropriate agent's cloud voice services.

Agent providers offer software development kits or APIs for interfacing with their cloud voice services, and may require a separate software stack. In such cases, the degree to which agents on the device coordinate, cooperate and interoperate may be limited. Agents that are able to use common middleware for enforcing a single agent listening state benefit in providing users coordinated agent experiences.

Developers should consider using a common framework and device middleware such as a VII Library to aid in the handling and routing of voice input data. Figure 1 shows a viable conceptual architecture for voice input audio flow on a multi-agent device that supports three simultaneously available agents using two WWEs. The agent policies for this example require that only their respective cloud services receive voice input data whenever their agent is invoked. WWE_1 embeds the wake words for $Agent_1$ and $Agent_2$ and the other WWE embeds a single wake word for $Agent_3$. All three agents utilize an agent-neutral VII Library that helps enforce single-agent listening states, during which only the invoked agent receives the voice input data. With this library, agents are expected to request entry into a dialog with the user.

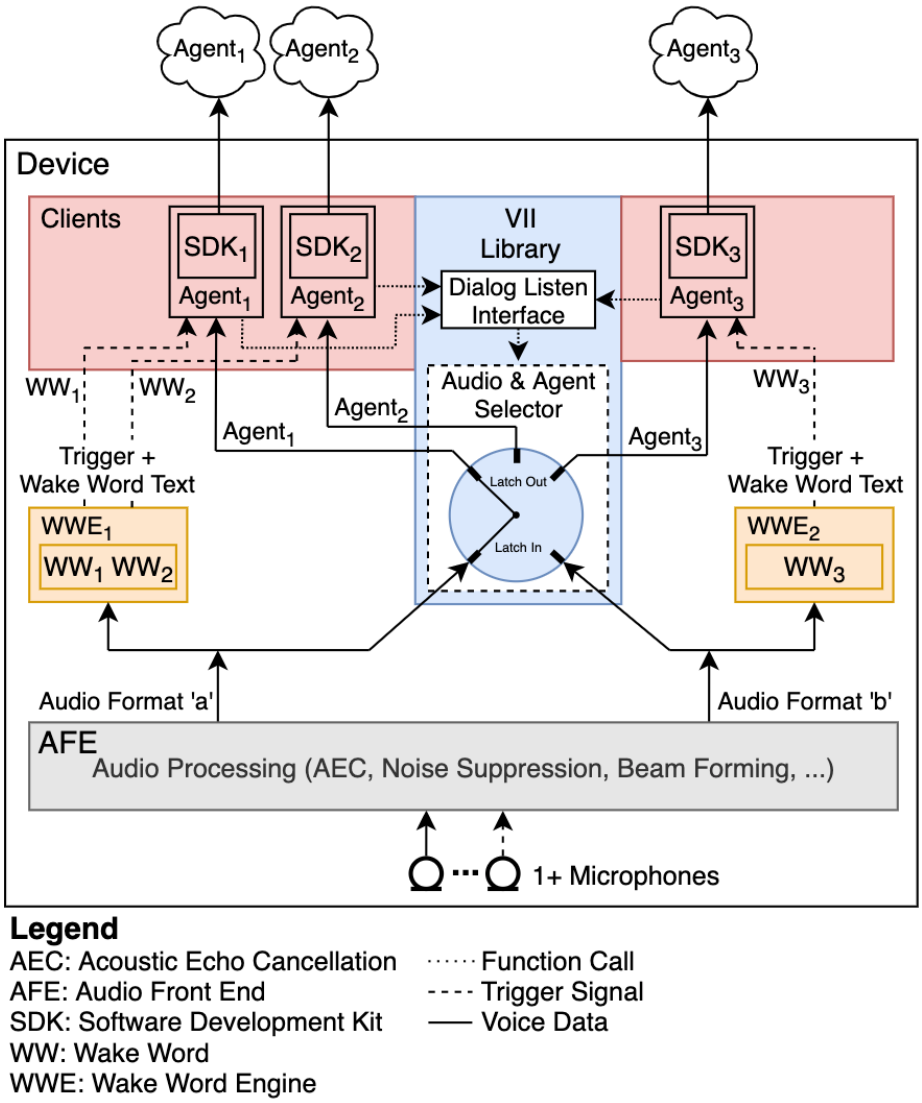

Figure 1 - - Device Voice Input Flow – 3 Agents, 2 WWEs

The library verifies whether another agent is already in a dialog with the user. If no other agent is currently in a dialog with the user, the invoked agent requesting the dialog is granted access

to the voice input data by an *Audio and Agent Selector* component, which selects the appropriately processed data from the AFE and routes it to the agent's client software. For the illustrated device, both $Agent_1$ and $Agent_2$ share the same processed and formatted audio, Audio Format 'a', and $Agent_3$ uses a uniquely processed and formatted audio, Audio Format 'b'. Both formatted audio sources are simultaneously available to both WWEs and the VII Library Audio and Agent Selector. When the user speaks an agent wake word, the corresponding WWE will trigger and notify the respective agent client software. The agent client software then makes a request to the VII Library to enter a dialog and, if granted, calls a dialog *listen* method to enter the device's Listening state. The VII Library then connects an input latch to the agent-compatible processed and formatted audio and an output latch to the invoked agent's audio input stream. The agent client software then starts reading the voice utterance data and streams it to its cloud services. The Audio and Agent Selector component ensures that no other agent client software is able to capture or stream audio while an agent is in its Listening state.

An agent may also be hosted by another agent's cloud services. Figure 2 shows a viable conceptual architecture that shares an agent's cloud infrastructure.

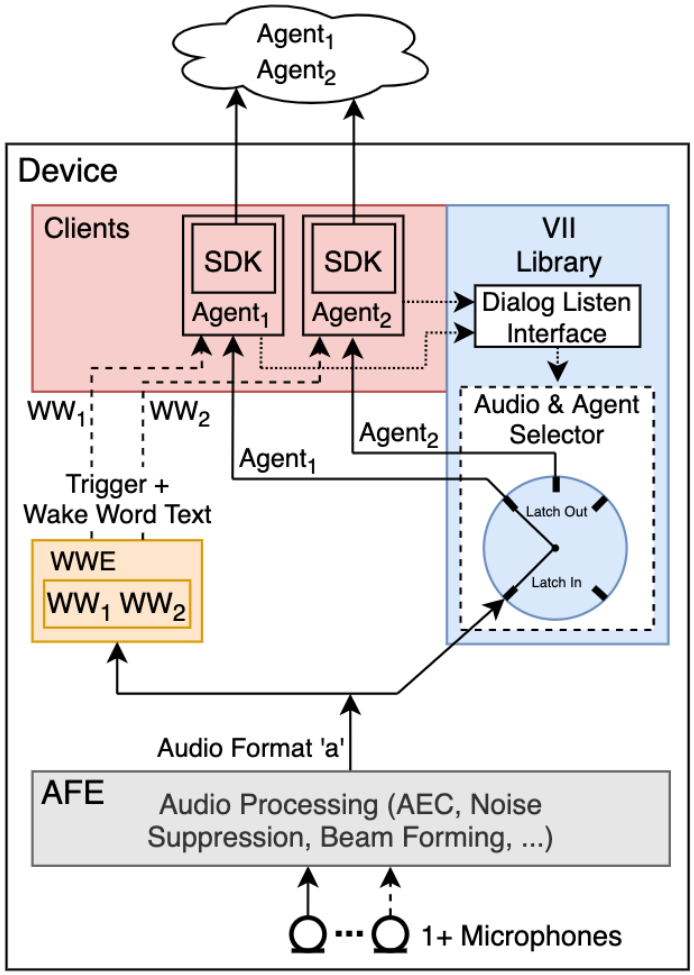

Figure 2 - Voice Input Data Flow - Native and Hosted Agents

In this case, $Agent_1$ and $Agent_2$ are implemented as distinct and separate agents that use the same software development kit (SDK) to communicate with common cloud services. This architecture provides a huge cost benefit for the agent that is hosted by another agent's cloud services as it avoids the need for separately building out the necessary secure cloud infrastructure for natural language processing, TTS, and AI. This approach also requires only a single set of processing and formatting of the voice input data because the WWE and cloud ASR are common to both agents. The common VII Library also benefits both agents in providing a single agent listening state to ensure voice input data pass through the appropriate agent's pathway to its respective cloud services.

Multi-agent product requirements, the locations of agent cloud services, agent dependencies on WWEs for wake word support, agent-specific SDKs and APIs, the AP operating system, and agent provider support ultimately govern the overall architecture. The above architectures are not all-encompassing, but the flexibility of a VII Library that utilizes this approach for handling and routing voice input data, independent of whether WWEs reside on the AP or DSP, makes it versatile. Likewise, when agents are able to coordinate their activities and use common software such as a VII Library, it increases system and data integrity.

These above architectures address the use case scenarios described in Wake Words within Utterances. They accomplish this by limiting a dialog with the user to one agent at any time. When an agent is invoked and requests a dialog, only when no other agent is currently in a dialog with the user is it provided access to voice input data. Use case 2 requires additional handling of any wake words output by the speaker and can be addressed by using WWEs that recognize wake word fingerprint data in the TTS, or by adding WWEs whose input is the audio output signal in order to ignore wake words in speaker output.

Conclusion

Multi-agent devices provide customers delightful ways to engage with a variety of experiences and features offered by agents and their services. However, building multi-agent devices requires thoughtful consideration of key components for proper handling and routing of voice input data and the potentially unique data formats required by WWEs, CBWWV, and agent ASR. Applications software on the device should also consider the techniques employed by auxiliary processors such as DSPs, where additional coordination and communication may be required. Lastly, developers benefit by using common multi-agent middleware on the device to help coordinate agent interoperability and properly handle and route voice input data.

Contributors

The contributors to this document are:

- Robert Mars, Principal Solutions Architect, Alexa Voice Services
- Eran Feld, Vice President Engineering - Smartware, DSPG
- Rony Rado, Linux/Android Software Team Leader, DSPG
- Joe Murphy, Vice President Marketing, Sensory, Inc.

Additional Resources

- Voice Interoperability Initiative: https://developer.amazon.com/en-US/alexa/voice-interoperability
- Multi-Agent Design Guide: https://build.amazonalexadev.com/rs/365-EFI-026/images/VII_Multi_Agent_Design_Guide.pdf
- VII Architecture Best Practices – Foundational Concepts: https://m.media-amazon.com/images/G/01/vii/VII_Architecture_Best_Practices_Foundational_Concepts_Whitepaper.pdf

Document Revisions

Date	Description
July 2021	First publication