VII Architecture Best Practices

Foundational concepts

January 19th, 2021

Contents

Abstract

In a world with multiple voice services each with different capabilities, customers should have the freedom to choose their preferred services for any task. This means enabling simultaneous access to multiple voice services on the same product. This whitepaper is the first of a series that provide recommendations and architectural suggestions for consideration when building products that support multiple simultaneous voice agents and the 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 the Voice Interoperability Initiative and the design guide. Many of the terms and concepts used in this whitepaper refer to terms that are detailed in the design guide. It is important to note that the concepts in this document are primarily focused on achieving the Universal Device Commands and Agent Transfer experiences in the Multi-agent Design guide.

Overview

The Voice Interoperability Initiative (VII) is committed to providing customers choice and flexibility to interact with multiple voice agents. In pursuit of this objective, Amazon has published the Multi-agent Design Guide, which offers an initial set of design recommendations for developing products that supports multiple simultaneous wake words.
The recommendations for multi-agent experiences laid out in the design guide focus on delighting customers, while in parallel addressing privacy. To support you on the initial steps to creating a device that aligns with the design guide, this paper presents foundational architectural concepts for creating a scalable solution for building a multi-agent device.

To create a delightful experience for customers, a device maker may consider elements of interoperability for the voice agents they integrate on the device. These voice assistants may share a common client software stack or have independent client software stacks. Devices come in a variety of physical configurations, each requiring a customized implementation.

You will first learn of the terminology that is used in this document to help define a common understanding. The next section will go into detail on the opportunities for improving the customer experience for devices with multiple simultaneous voice agents. You will be guided through scenarios that deeply investigate the implications of creating frustration-free experiences and maintaining customer trust with a strong privacy foundation. You will then be presented a recommendation for some foundational concepts that should be present in a VII Middleware component to enable interoperability between agents in a scalable and flexible way.

Before You Begin: Terminology

Also refer to terms in the design guide.

Activity
An Activity captures the lifecycle of any audible agent feature which is not part of a Dialog. This can include making a call, delivering an alarm, and playing music.

Control
A Control is a device-global action with a type matching a supported Universal Device Command (UDC) on the device.

Dialog
A Dialog captures the lifecycle of any agent interaction which involves the user or agent continuously interacting with each other, analogous to a conversation between two people. It may consist of many back and forth responses between the agent and user before the Dialog is complete. A Dialog may be initiated by the user or an agent.

Experience
An Experience is a representation of the state of an agent which may be shared with the Device to be rendered visibly, physically, or audibly for the user. The same type of Experience may be rendered differently per agent according to the Device and agent makers. For example, the Listening attention state may be rendered in a different way for a given agent. An Experience can also represent non-Dialog states of an agent, such as an active timer being played for the user.

Focus Management
It is important to accurately determine what's happening on a device at a given time, in order to create the desired UX in a device with multiple simultaneously available agents. Focus management allows the device to intelligently control the Activities of agents to enable a graceful coordinated response to user requests.

Main Application

This is the software program that is in charge of operating the device and all of its functions. It is responsible for maintaining the source of truth for entities such as the volume level and interfacing with the physical elements of the device such as visual indicators. It may also be responsible for operating processes such as agent clients.

Summoning
The action of invoking another voice agent by another so a Dialog can be initiated with the user to fulfill a user request.

VII Middleware
This is the heart of multi-agent coordination and cooperation. It presents a layer for agents and the Device for initialization and interaction and is composed of a collection of elements that define dialog session and activity rules, as well as virtual mappings to hardware on the device.

Wake Word Engine (WWE)
A component that is responsible for detecting specific wake words within a provided audio stream. A WWE can either be embedded in the hardware DSP chip, or it could be an application-driven software component, among other possibilities.

Challenges to achieving interoperability

The design guide discusses why interoperability between agents is important for the best user experience when multiple agents are simultaneously available on a device. This implies that some effort is needed to enable interoperability on a device. Let's examine some of the challenges, and a proposed way to go about solving them.

Promoting Frustration-Free Experiences

In order to create a delightful customer experience when multiple agents are simultaneously available on a device as contemplated by the design guide, device makers may want to take care to address scenarios that create frustration for customers. For example, there are circumstances where the UX can be improved to avoid frustration by the user. Achieving coordinated graceful agent interoperability is something that customers can expect from a voice enabled device with multiple simultaneously available agents. Let's examine scenarios where customer frustration is possible using multiple agents, and what needs to be done to address them.

1. An alarm is sounding from agent 1 and the user uses agent 2 to stop it using a Global UDC command such as "agent 2, stop".

   This is an easy concept, but has implications for the device maker implementing the solution and the agent provider. As agent 2, I must first understand the user's Stop intent, realize that I cannot address this intent, then deliver a Stop intent message to the device. The device must then understand this Stop intent, determine what currently available activities on the device may apply to this Stop intent, and determine which activity currently has the highest focus. As an agent provider, I must build support for "unhandled intents" and have a facility by which it may be communicated to the device. As a device maker, I must be aware of the activities of all the active agents and understand which ones have the

highest focus or priority. Having a system that is able to perform Focus Management is a requirement to achieving this type of experience.

2. I am listening to my favorite podcast initiated via agent 1 agent 2 wants to inform me that a package has arrived using audible speech (TTS).

An example of good UX here is to attenuate or pause the podcast from agent 1, play the TTS from agent 2, and then resume the podcast at normal volume. If I did nothing but applied a "mixer", the audio output would be garbled, and I would potentially miss content from my podcast and not understand the TTS. In order to implement this, a device must understand the states and activities of the agents, and it must be able to coordinate the flow of actions such as ducking/pausing in order to provide a delightful customer experience.

3. Agent 1 is reciting a recipe to me via TTS. Agent 2 wants to barge-in to inform me that someone is at the door of my house and then ask if I want to engage in a conversation with the visitor.

This is an example of two active Dialogs, each from a different agent. The recommended UX here is to stop the recipe Dialog from agent 1, and make agent 2 active to engage the user in the newly active Dialog. This is an important example as it illustrates the need for Focus Management and is a way to ascertain when a specific Dialog from an agent should be allowed to interrupt another. What would happen if I had no such capability? In this example, a device's behavior may be that a currently active Dialog never be interrupted. As a result, the customer would never know that they had a visitor at the house door. This kind of coordination encourages graceful behavior, and it is important to ensure customers feel comfortable using more than one agent on a device.

The following example matrix can help you think about the proper UX when different activities involving multiple agents come into play. The suggestions here are not meant to be absolutes, and may vary according to the desired UX.

Agent State	**Agent 2: TTS**	**Agent 2: Media Playback**	**Agent 2: Alarm**
Agent 1: TTS	Previously active TTS is interrupted	Pause or Duck Media playback	Alarm is slightly ducked while TTS is engaged.
Agent 1: Media Playback	Pause or Duck Media playback	Previously active media should stop, new media should play.	Pause or Duck Media playback
Agent 1: Alarm	Alarm is slightly ducked while TTS is engaged.	Pause or Duck Media playback	Both alarms sound, but not simultaneously.

Customer Privacy

The Multi-Agent Design Guide emphasizes that strong privacy and security guidelines are key aspects to earning and maintaining customer trust. In order to achieve this correctly, one important consideration is that device makers should carefully design the microphone audio paths and the agent invocation methods (such as wake words and action buttons) to ensure that the customer's voice goes to the intended agent only. The customer should always be aware of which agent is active and listening through sound cues and/or UX patterns. Let's examine a subset of important privacy considerations around how to properly process microphone audio, and consider what is required on the device to ensure alignment with the design guide. It should be noted that microphone audio is not the only core privacy concern. There are other concerns such as how to properly implement UDCs and respect the user's privacy.

1. Another agent's wakeword *should not trigger* when another agent is already active in a Dialog and is in the Listening state.

   In a single agent device, the WWE should always be active and ready to notify the agent to engage the user in a Dialog. This rule no longer applies when two agents or more are active. What would happen if I were able to invoke another agent by triggering its wakeword while another agent is in the Listening state? I would be faced with a situation where two agents are now in a Listening state, each expecting user voice data. To avoid such situations, the device must be able to ascertain the state of available agents and recognize when the associated wake words should be considered active, based on the states of the active agents.

2. Voice agent clients should only have access to the microphone data when they are in the Listening state.

   Consumers have come to expect that voice agents will only transition to the Listen state via the user speaking the wakeword or an action button on the device. The microphone audio must only be supplied to the agent client software when it is in that Listening state. This is crucial to meeting the privacy expectations of the user. When multiple agents can be awakened, the underlying audio subsystem must understand which agent is in the Listening state and only supply microphone audio to the corresponding client software.

It is also possible that some voice agent activities, such as voice calling handled by a specific voice agent client's software, should disable the wakeword for other agents to meet the user's privacy expectations. Let's look at an example scenario. A user is actively engaged on a voice call with a friend that was initiated using voice agent 1's client software. To best meet the privacy expectations of the user for a private call, it may be preferable to not only restrict the microphone audio to voice agent 1, but to also disable wake words from other agents during this activity. This is less confusing for users and reduces the risk that audio is sent to wrong

agent from false accepts by another agent's wakeword for example. This kind of control of device and agent functions can only be done using logic that understands the activities of the available agents.

The examples above illustrate the challenges device makers face to ensure that multi-agent devices do not have flaws that negatively affect the customer's experience. It is through this lens that the following VII Middleware architectural concepts below have been formed. The aim is to provide a foundation by which the critical privacy and UX considerations for devices with multiple agents can be addressed.

VII Middleware Overview

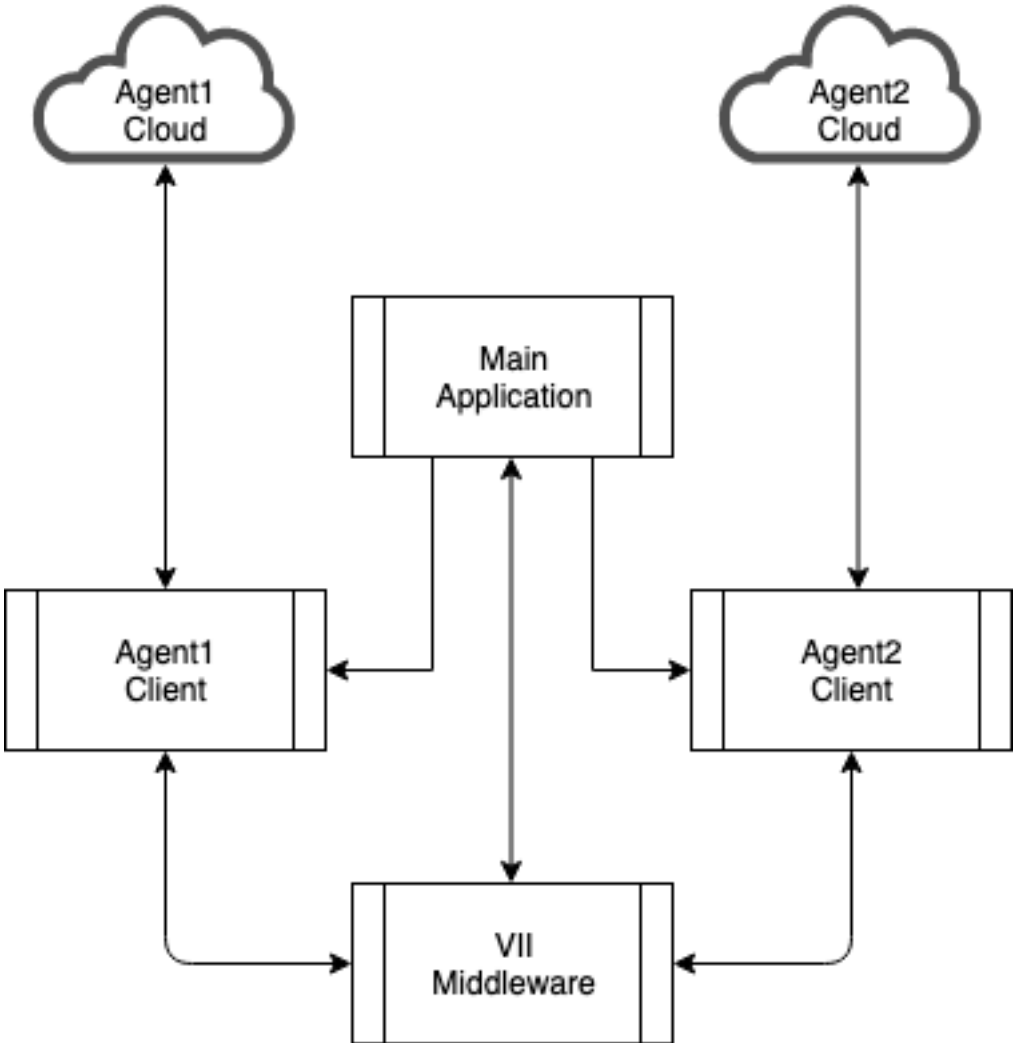
It is recommended that the VII middleware should be structured in a way to use integration patterns that minimize the need for information exchange between agents and expose on-device APIs to enable control of multiple agents. These APIs enable developers to coordinate multi-agent experiences both visually and audibly, implement Universal Device Commands (UDCs), and allow agent Transfer experiences between agents. It is the responsibility of the agent-makers and device-makers to use these APIs to provide the optimal multi-agent experience. For example, if agent 1 wants to initiate an agent Transfer to agent 2, it must communicate its intent via an API to the VII middleware, which then performs the necessary validation and proceeds with managing the agent Transfer experience.

The diagram below illustrates the interactions between agents, the main application of the device, and the VII Middleware. Through the use of the VII Middleware, the user experience is coordinated through the use of a manager that is aware of multiple agents. It is important that the VII Middleware be aware of the activities of the agents on a device in order to provide the proper multi-agent experience. For example, a TTS response by an agent from a voice query should duck or pause media that may be playing from another agent. This would not be possible without a layer that is aware of the current media-playing activity. The Main application is responsible for rendering these experiences and handling user input. Both agents and the Main application make calls to and from the VII Middleware. The Main application also makes calls to agents for operations that do not require use of the VII Middleware. For example, handling a button which triggers a specific agent's feature.

The following design principles inform this architectural approach:

1. **Agents should avoid direct communications with each other**. The VII Middleware enables agents to interact via an on-device software for use cases that require it, such as for Universal Device Commands, without the need for direct communications with each other.

2. **The VII Middleware should not interact with the agent backend services.** It only interacts with the device side implementation for each agent's client.
3. **Agents should only directly communicate with the Main application via the VII Middleware.** This ensures that multi-agent experiences that are consistent with the design guide can be correctly rendered.

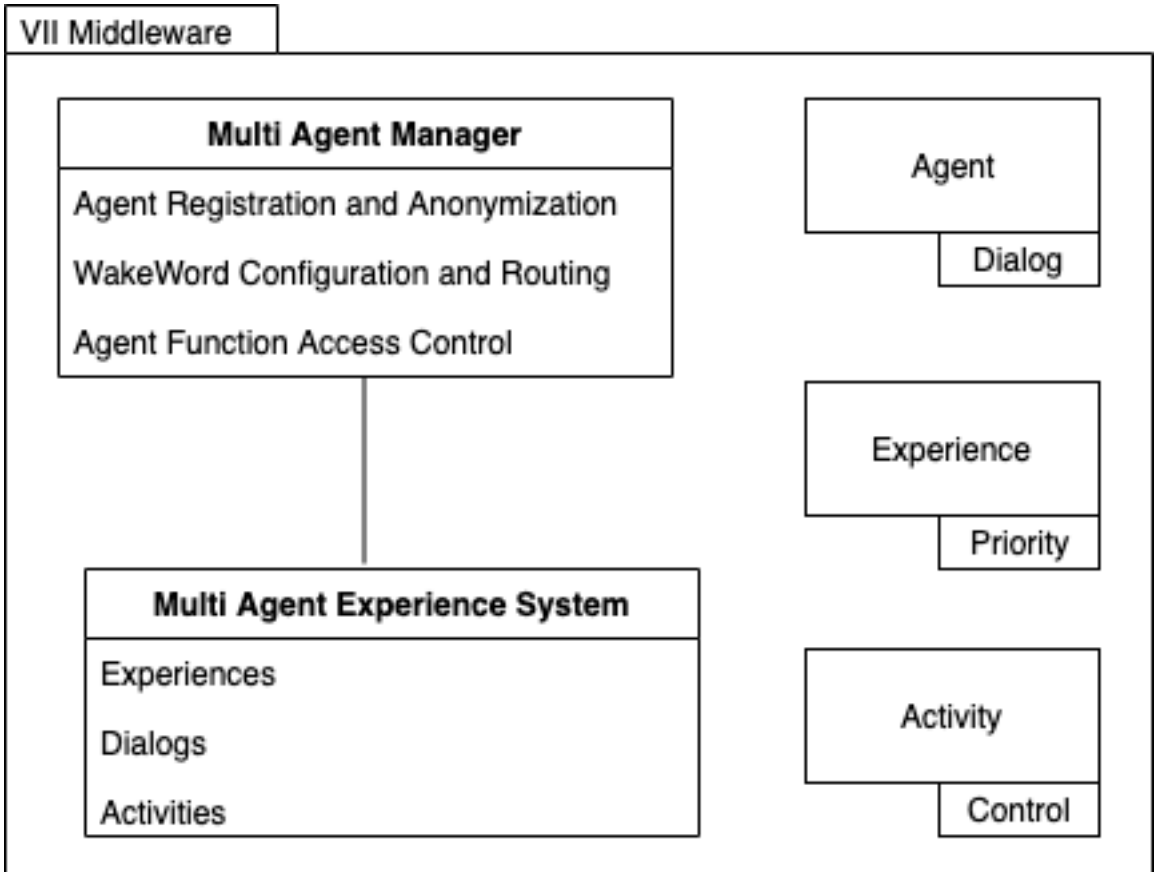


VII Middleware Components

A device can use the VII Middleware to enable seamless support for multiple voice agents working simultaneously. The following concepts should be present to allow the interoperability patterns laid out by the design guide to be implemented:

- **Multi Agent Manager (MAM).** This component mostly interfaces with the Main Application of a device and provides a way to register agents, inform agents of Wake Word detections, and handle agent user Experience requests.
- **Multi Agent Experience System (MAES).** This is an agent-facing component that is used to render the proper UX through the device's Attention System. This Experience System should be accessed through the Multi Agent Manager.

Let's examine the following diagram to see what should be present in these components to enable seamless interoperability.



Multi agent Manager (MAM)

The MAM concept is at the heart of enabling agents to achieve interoperability, in charge of the administration of agents and WWE's. This component should control or track the following:

- **Agent registration**. The Main Application must register agents to the MAM, but not to other agents.
- **Manage the wakeword configuration associated with each respective agent.** The Main application should present agents and WWE details as parameters to the MAM, so that MAM explicitly knows which wake words are associated with which agents. Wake Word Engines come in a variety of configurations. You should consider that a device can have multiple WWEs, and that a single WWE could be used for more than one agent, and that multiple wake words may be used for a specific agent.
- **Enable and Control access to agent specific functions.** Functions such as invoking an agent as part of an agent Transfer are necessary to support the interoperability patterns laid out in the design guide. Access to these Functions should only be made available to another agent as needed, for example when a Dialog is active.

Multi agent Experience System (MAES)

The MAES should be seen as the runtime of agents working together. It is the centralized point for cross-agent interactions and correctness of behavior as defined by the design guide. It is important to have a single entity that is used to collect the Activities of the voice agents, and can therefore act accordingly to coordinate the proper UX.

The MAES should implement the following concepts:

- **Experiences.** The MAES should manage all active Experiences for a given device and be informed of any changes. A priority based on the type of Experience should be assigned so that the proper UX can be produced. It makes no assumptions about what Experiences an agent may wish to render to the user or how a device will choose to render a particular Experience. For example, an incoming call from one agent should have higher priority than music playing by another.
- **Dialogs.** The MAES should be able to track the Dialog state of each respective agent. This is important for ensuring customer privacy expectations. For example, it may be a customer's expectation that a request by one agent upon another be made only when a Dialog by the requesting agent is active. This guardrail is important for preventing unexpected use of the VII middleware to control an agent's activities by another. This places a condition that only allows an agent to affect another's activities when the customer is engaged in a Dialog.

- **Activities.** The MAES should be aware of Activities and track the life cycle of the Activities. It is responsible for avoiding agent collisions, ensuring that multiple agents with potential conflicting Activities do not interfere with each other, but follow rules that align with a harmonious behavior from a UX perspective. Activities should have types, such as: Dialog, Communications, Alerts, and Content. Agents should request an Activity which will be assigned the proper focus, for example whether the Activity should be in the foreground or the background. This enables MAES to create the proper UX. As an example, it may be appropriate that a background Activity of a certain type should be attenuated such as pausing or lowering the volume.
  An example of an Activity would be a Communications activity, which may be requested to be in the foreground. Another example of such an Activity would be an Alert. An activity may change from foreground to background any number of times.
- **Controls.** In order to satisfy a Universal Device Command such as a Global Foreground Stop, the MAES should understand what controls are available to be requested for a given Activity. For example, an available Control for an Activity of type Alert would be "Stop".

An Experience can be for example one of the core attention states conveyed by the design guide: Listening, Thinking, and Speaking. Each one of these states may have an audiovisual Experience that may need to be customized per agent.

The Main application of the device that is responsible for rendering the Experience should be notified when the list of active Experiences changes. For example, a barge-in by the user when another agent is in a Speaking state should trigger an Experience change that needs to be rendered. It is important for the Main application to respond quickly to the MAES to ensure a good customer experience.

The list of active Experiences should be kept in priority order, where the first Experience in the list has the highest priority and must be rendered if the device is capable of doing so. An example of this is a voice query initiated by the user to agent 2, while music was initiated and is currently playing via agent 1. The proper UX may be to pause or attenuate the volume level of the audio playback from agent 1 while agent 2 enters its Listening, Thinking and Speaking states.

Now that these foundational concepts have been defined, it is a good time to think about your specific device and its voice agents. Each device and voice agent may have unique use cases, and require special consideration in how they may fit in this proposed model. A good next step may be to document your multi-agent use cases carefully and how they may be able to leverage the concepts presented in this document.

Conclusion

For devices with multiple simultaneously available agents, device makers will create a more delightful and graceful user experience for their device if they address interoperability.

The architectural concepts proposed in this document emphasize an approach to enabling interoperability between agents that decouples agents from the underlying device. These concepts also allow coordinated multi-agent experiences through an orchestrating entity to achieve a quality UX while preserving user privacy.

The core high-level design concepts necessary for interoperability such as the Multi agent Manager have been advanced as fundamental building blocks to create a scalable and non-prescriptive architecture for devices to implement recommendations in the design guide. This paper has introduced conceptual pillars such as the use of a VII Middleware with a Multi-Agent Manager to manage agents, and the Multi Agent Experience System to manage multi-agent interactions to create delightful experience that customers expect from devices with multiple voice agents.

Contributors

Contributors to this document include:

- **Philippe Lantin**, Principal Solutions Architect, Alexa Voice Services

Additional Resources

- Voice Interoperability Initiative: https://developer.amazon.com/en-US/alexa/voice-interoperability
- Multi-agent Design Guide: https://developer.amazon.com/en-US/alexa/voice-interoperability/design-guide

Document Revisions

Date	Description
January 2021	First publication