Voice Interoperability Initiative Architecture Series

Universal Device Commands

July 21th, 2021

Contents

Abstract

This whitepaper is a part of a series that provides recommendations and architectural suggestions for consideration when building products that support simultaneous availability of multiple voice agents and align with the Voice Interoperability Initiative (VII) Multi-Agent Design Guide (MADG). It is intended for technical architects, device maker engineers, and voice agent developers. You will benefit by already being familiar with the Voice Interoperability Initiative and the VII Multi-Agent Design Guide. Many of the terms and concepts used in this whitepaper refer to terms that are detailed in the Multi-Agent Design Guide. This document provides technical guidance to implement Universal Device Commands (UDCs). UDCs are those commands and controls that a customer may use with any compatible agent to control certain device functions, even if the agent was not used to initiate the function.

Overview

Universal Device Commands (UDCs) are an important feature for devices with multiple voice agents because they help avoid frustration and foster a delightful experience for customers. This paper will guide you through scenarios where UDCs offer customer flexibility, explains how you can approach implementing UDCs in a device, and what may be needed to satisfy the customer expectations in a device with simultaneously available voice agents.

UDCs are an aspect of interoperability that allows a customer to affect a limited set of actions on a device using any available voice agent. There are important considerations for implementing UDCs such as ensuring that no unnecessary data is shared between voice agents to achieve the goal. This paper will describe methods and considerations to implement UDCs on devices and voice agents in a way that minimizes data sharing.

This whitepaper includes details involved in supporting scenarios for the different types of UDCs discussed in the Design Guide. Finally, it discusses potential implementations for device makers and voice agent designers based on the concepts in the “Foundational Concepts” whitepaper.

It is important to note that this whitepaper concerns UDCs when they are used within a single physical device, and does not attempt to cover multiple devices.

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.

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

Dialog
A dialog captures the lifecycle of any Agent interaction which involves the user or agent continuously speaking to 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. The agent that is involved is called the “focused agent”.

Natural Language understanding (NLU)
The process responsible for transforming human language into a machine-readable format.

Text-To-Speech (TTS)
The technology that enables text to be converted into speech sounds imitative of the human voice.

UDCs under the Microscope

Privacy-First Design

In order to fulfill a customer's spoken intent, voice agents may require technology able to perform Automated Speech Recognition (ASR), Natural Language understanding (NLU) and Text-To-Speech responses, in order to fulfill customer requests for specific activities such as initiating media and operating timers/alarms. Secure interoperability – developing voice services that can work alongside others while protecting the privacy and security of customers – is one of VII's four pillars. So, what is minimally necessary to enable a voice agent to know about the device's state to satisfy a customer's intent to execute a UDC? Let's begin by classifying the types of UDCs and detailing what devices and voice agents may need to know to fulfill the voice request of the customer.

Classifying UDCs

UDCs are those commands and controls that a customer may use with any compatible agent to control certain device functions, even if the agent was not used to initiate the function. There are broadly two types of UDCs:

- **Device Global commands** are implemented by each agent independently, for example changing the volume of the device's output speaker. No information of any kind is required to be surfaced from one agent to another. In order to be perform this command and to deliver the best customer experience, an agent may need to disambiguate whether the user's command is intended to command a specific device, another device, or control the volume on multiple devices.
- **Cross-Agent commands** require an action to be taken by a voice agent on behalf of another. These commands may require state information to be communicated from the device to a voice agent to fulfill the customer's intent. An example of this type of command is a request to stop a sounding timer from Agent A that was initiated from Agent B.

UDC Priorities

Customers increasingly expect voice agents to behave naturally and understand the context in which a command was given. In settings where multiple people use a device and may have their own preferred assistant, UDCs are a key part of a device's functionality. An example is the simple utterance "Stop" which is one of the most popular commands that customers use with

any voice agent to stop the activity that holds the highest focus and aligns with the customer's intent.

Customers increasingly expect voice agents to behave naturally and understand the context in which a command was given. In settings where multiple people use a device and may have their own preferred assistant, correctly interpreting the customer's intent for a command may require a voice agent to implement a carefully built set of rules that keep track of the "focus" of the device. Consider the example of the utterance 'Stop', which customers could use with any voice agent to stop the activity that holds the highest focus and aligns with the customer's intent. If an activity such as an alarm sounding interrupts another activity such as media playback, then a global foreground 'Stop' command needs to correspond to an intent to stop the sounding alarm instead stopping music playback because the alarm has preferential focus.

Here is an example prioritization schema, from highest to lowest:

1. Communications: A calling activity with a remote device such as a phone call.
2. Dialogs: a user actively engaged in an interaction with an agent that has the current focus of the device.
3. Alerts: Functions such as reminders and timers that "alert" the user of an activity or status change, such as a reminder or timer.
4. Content: Media such as audio playback.

There are additional sub-types of qualifying activities that are important to classify appropriately. For example, a device-to-device call or phone call for an emergency should perhaps not be interrupted. Devices with displays may also have behaviors and activities associated with a visual focus that is not considered in the above. It is also possible for certain activities to hold focus for only a period of time, or even share the focus with other activities with alternating behavior, for example an alarm sounding periodically if the device is engaged in a calling activity.

A UDC may also be initiated from a non-voice interaction such as a button press. The assignment of a button press to an action may be dependent on what is currently in the highest priority of focus on the device. An example of this would be the assignment of a button to stop an alert when it is in focus vs media playback.

UDCs and Implementation Requirements

It is helpful to define the roles and responsibilities of the Device Maker and the Voice Agent in a Multi-Agent device:

- **Device Maker**

    - **Low-level**: Typically provides the hardware and low-level interfaces needed for the device functions such as speaker volume control.
    - **VII middleware**: May develop or integrate an application that interfaces with the voice agent client software, the low-level interfaces mentioned above, and other components such as wake word engines (WWEs).
- **Voice Agent**
    - **Client**: May make a client/SDK available for the device maker to interface with the voice agent's backend service.
    - **Voice Agent Service**: May make available interfaces and functionality associated with the agent, such as an interface to ingest audio data.

The following table presents the UDCs included in the Design Guide, and a set of potential device maker and agent developer responsibilities when fulfilling the command.

UDC	Device Maker	Voice Agent Developer
Stop/Dismiss Sounding Alert	• Track the state of active alerts and other activities of the device. • Provide an interface by which the command can be given. • Take action to stop/dismiss the alert by exercising the corresponding interfaces.	• Inbound: Provide an interface by which an alert can be stopped/dismissed. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the Stop Alert UDC.
Stop Media	• Track the state of active media playback and other activities of the device. • Provide an interface by which the command can be given. • Take action to stop media by exercising the corresponding interfaces.	• Inbound: Provide an interface by which media playback can be stopped. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the Stop Media UDC.
Stop Camera Feed	• Track the state of active camera feeds and other activities of the device. • Provide an interface by which the command can be given. • Take action to stop the camera feed by exercising the corresponding interfaces.	• Inbound: Provide an interface by which the camera feed can be stopped. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the state of camera feeds.

Reject Incoming Call	• Track the state of active calls and other activities of the device. • Provide an interface by which the command can be given. • Take action to reject the call by exercising the corresponding interfaces.	• Inbound: Provide an interface by which the incoming call can be rejected. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the Reject Call UDC.
Stop Agent Speaking	• Track the state of active speech by voice agents and other activities of the device. • Provide an interface by which the command can be given. • Take action to stop speech by exercising the corresponding interfaces.	• Inbound: Provide an interface by which speech can be stopped. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the Stop Agent Speaking UDC.
Global Foreground Stop	• Track the state of all activities that can be controlled by UDCs. • Provide an interface by which the command can be given. • Take action by exercising the corresponding interfaces.	• Inbound: Provide an interface by which the relevant activity can be stopped. • Outbound: Provide an interface to communicate the device's current activity in focus, and/or an interface to communicate the Global Foreground Stop UDC.
Volume Up/Down/Control/Mute	• Track the state of the volume level. • Provide an interface by which the command can be given. • Take action by exercising the corresponding interface(s).	• Outbound: Provide an interface to communicate the volume control request, and report the state as necessary.

You can observe common themes in what should be supported for UDCs to function:

Device Maker Low-level

- Provide low-level interfaces to change the physical state of the device, such as the speaker volume.
- Provide an interface to communicate state changes of buttons and other physical controls on the device such as screen brightness.

Device Maker VII middleware

- Provide an interface for agents or applications to register/deregister activities such as Dialogs, Alerts, Communications, and Media.
- Provide an interface by which an agent can stop the registered activities of other agents/applications.
- Provide an interface to change and report the state of physical functions of the device, such as the speaker volume.

Voice Agent Client SDK

- Implement the VII middleware interface to register/deregister activities of different types such as Dialogs, Alerts, Communications, and Media.
- Implement the VII middleware interface to change and report the state of physical devices properties such as the speaker volume.
- Implement the execution of the UDC if it should be invoked by the VII middleware.

Voice Agent Service

- Provide an interface by which the device state of activities in focus can be collected, and/or a way to deliver the UDC to the Client SDK, which will then interface with the VII middleware.
    - An example implementation of this could be achieved by providing an invoked agent the anonymized map of necessary activities in focus at the time of the request on the device when the agent enters the Listening state. This option provides limited additional context regarding the activities on the device may not be aware of to ensure the proper action is taken.
    - Another implementation of this could be achieved by providing an invoked agent the list of UDCs available at the time of the request that an agent could action when the agent enters the Listening state.

There are additional implementation considerations for the above UDCs:

- **Data Minimization**. UDCs can be implemented by providing minimal data regarding the on-device activities that are occurring at the time of a request to the extent necessary to disambiguate the customer's intent (and only for the invoked agent).

- **Voice Agent and Device Symmetry:** It can be confusing for customers if the agents on the device do not support the same UDCs, and for the ideal customer experience, the voice agents available on the device would be capable of supporting the same or similar set of UDCs on the device. Also, voice agents should consider the device capabilities for their support for UDCs. For example, if the device hardware supports a volume range of 0-63, the available voice agents should interpret volume control requests similarly to avoid confusing the customer. Voice agents should also consider how to best utilize the hardware capabilities of the device.

UDC Request Examples

The following table provides some examples of UDC requests and a possible flow of actions to be taken.

Stop Sounding Timer

	Action	VII Middleware	Voice Agent A	Voice Agent B
1	User sets timer by voice with Voice Agent A.		• Set timer	
2	Timer is due	• New activity of type “alert” registered from Voice Agent A	• Start timer alert • Register new activity of type “alert” with VII middleware	
3	User instructs Voice Agent B to “Stop Timer”	• Receives the request from Voice Agent B to stop an activity of type “timer”		• Instructs the VII middleware to determine which of multiple current activities should be stopped on the device.
4	Timer stops	• Requests Voice Agent A to stop timer • Timer activity is cleared.	• Receives and executes request to stop timer from VII middleware.	

There are some important points in this example:

- While Voice Agent B's backend service may need to be aware that a timer is active to fulfill the Stop Timer UDC, it does not need to know that this request was made by Voice Agent A.
- Even when the UDC is completed, Voice Agent B does not need to be made aware of the result of the UDC.
- The VII middleware should shield agents from knowing how many other agents are present and which agents they might be, and only expose the minimal state required to fulfill the UDC.

By following these points, there is minimal information shared between agents and the user's intent has been fulfilled.

Global Foreground Stop

	Action	VII Middleware	Voice Agent A	Voice Agent B
1	User sets a timer with Voice Agent A		• Set timer	
2	User issues a voice command to Voice Agent B to initiate a call	• New activity of type "communications" is registered from Voice Agent B.		• Call initiated
3	Timer is due	• New activity of type "alert" registered from Voice Agent A	• Start timer alert • Register new activity of type "alert" with VII middleware	
3	User instructs Voice Agent B to "Stop"	• Receives the request from Voice Agent B to stop the activity currently in the highest priority focus, in this case the sounding Timer.		• Instructs the VII middleware to determine which of multiple current activities should be stopped on the device.

4	Timer stops	• Requests Voice Agent A to stop timer • Timer activity is cleared.	• Receives and executes request to stop timer from VII middleware.	

Again, this example includes important points:

- As mentioned in the implementation requirements previously, Voice Agent B's backend service can be made aware that a timer activity is occurring. Voice Agent B can then make the determination that this timer activity should be stopped based on the map of activities that has been communicated by the device and its own context through the Voice Agent B's client SDK.
- The command to stop the timer should be sent to Voice Agent B's client SDK by Voice Agent B's backend service, which will then relay the request to the VII middleware.
- The VII middleware should then validate the UDC stop timer request and communicate the request to Voice Agent A's client SDK.

It is important for the voice agent Client SDK to register activities with the VII middleware to permit UDC's to function. Additionally, the VII middleware will need to determine which of multiple current activities should be stopped.

Conclusion

When using multiple simultaneously available voice agents, customers deserve devices that are intuitive to operate, behave naturally and can be trusted with their data. It is important to surface commonly used device functions via UDCs to enable a seamless customer experience with the available voice agents.

Voice agents and devices should consider which activity is in focus at any given time on the device. This enables the appropriate action to be executed when a UDC is performed, and aims to reduce frustration in the customer experience.

The example implementations of Universal Device Commands discussed in this document present device makers and agent developers insights into what may be needed to support cross-agent and global commands. In addition, the use of on-device middleware to facilitate interactions between the device and the voice agents available on the device is an important building block to enabling UDCs without sharing data about the activity beyond what is necessary to complete the request.

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
- VII Best Practices Foundation Concepts whitepaper: https://m.media-amazon.com/images/G/01/vii/VII_Architecture_Best_Practices_Foundational_Concepts_Whitepaper.pdf
- Voice Assistant Coexistence on the Sonos Platform: https://tech-blog.sonos.com/posts/voice-assistant-coexistence-on-the-sonos-platform/

Document Revisions

Date	Description
July 2021	First publication