Google Voice over BLE spec 1.0

Version: 1.0
Status: Published
Authors: sujithrk@google.com skill@google.com
Contributors: marvinramin@google.com
Last Update: 2020-07-11

Version	*Modified by*	*Date*	*Changes*
1.0	*skill@google.com*	*2020-07-11*	*Clarification of behaviour of MIC_CLOSE command;* *Updated description of characteristics behavior;*
Draft 0.Next2	*skill@google.com*	*2019-12-28*	*Changed Characteristics properties;*
Draft 0.Next1	*skill@google.com*	*2019-07-23*	*Initial draft of spec 1.0:* *payload format of all commands has changed;* *header is removed from audio payload;* *new voice interaction models (PTT/ HTT);* *MIC_EXTEND message added*
Draft 0.4e	*sujithrk@google.com*	*2018-09-18*	
Draft 0.4	*sujithrk@google.com*	*2017-08-15*	
Draft 0.3	*sujithrk@google.com*	*2017-06-19*	
Draft 0.2	*sujithrk@google.com*	*2017-06-06*	
Draft 0.1	*sujithrk@google.com*	*2017-04-24*	

Table of Content

1 Summary

This document describes the voice specific software requirements for Bluetooth Low Energy (BLE) based remote control to be used with an Android TV Device (ATV). The Android TV Remote will provide D-Pad based navigation and voice input through an onboard microphone. Voice captured on the microphone will be transmitted to the Android TV using BLE GATT characteristics based on a sequence of operations shown below.

Note: Advertising, Pairing, Battery, DPAD operations are out of scope for this document. See References for more details on those topics.

2 ATV Voice Service

The Android TV Remote is a Bluetooth Low Energy (BLE) device and will operate as a peripheral. The Android TV Device will operate as a central.

ATV Voice Service shall be instantiated as a Primary Service.
The service UUID shall be set to “ATV Voice Service”.
The Service and characteristic UUIDs are defined below:

2.1 GATT profile UUIDs for ATV Voice Service

The Remote Device should implement and advertise these characteristics with the corresponding properties.

Type	Short-form	UUID	Properties
ATV Voice Service	ATVV_SERVICE_UUID	AB5E0001-5A21-4F05-BC7D-AF01F617B664	
Write Characteristic	ATVV_CHAR_TX	AB5E0002-5A21-4F05-BC7D-AF01F617B664	Write without Response
Read Characteristic	ATVV_CHAR_AUDIO	AB5E0003-5A21-4F05-BC7D-AF01F617B664	Notify
Control Characteristic	ATVV_CHAR_CTL	AB5E0004-5A21-4F05-BC7D-AF01F617B664	Notify

2.2 Characteristic Behavior

The **ATVV_CHAR_TX** characteristic is written to by Android TV Device and is used to send information to the Remote Device.

The **ATVV_CHAR_CTL** characteristic is read using the GATT *Characteristic Value Notification* sub-procedure and returns control data. In most cases, the first byte contains necessary information but depending on the control type, other bytes can also be used.
Every notification for this characteristic should be sent to and confirmed by the Host.

The **ATVV_CHAR_AUDIO** characteristic is read using the GATT *Characteristic Value Notification* sub-procedure and returns audio data.
Under certain circumstances some notifications for this characteristic can be skipped (see section 4.3.2).

Both these characteristic can be configured for notification using the GATT *Write Characteristic Descriptors* sub-procedure on the *Client Characteristic Configuration* descriptor. When configured for notification, this characteristic can be notified while in a Connection. No notifications of this characteristic shall be sent when disconnected. The ATV Voice service shall not initiate a connection if a notification would have been sent when connected.

The value of the *Client Characteristic Configuration* descriptor is persistent for bonded devices when not in a connection.

2.3 Characteristics and Commands

Characteristic	Direction and Message	Command and Data
ATVV_CHAR_TX	ATV ---> Remote	
	GET_CAPS	**0x0A**, payload(5)
	MIC_OPEN	**0x0C**, payload(1)
	MIC_CLOSE	**0x0D**, payload(1)
	MIC_EXTEND	**0x0E**, payload(1)
ATVV_CHAR_AUDIO	Remote ---> ATV	
	{Audio Data, no message}	
ATVV_CHAR_CTL	Remote ---> ATV	
	AUDIO_STOP	**0x00**, payload(1)
	AUDIO_START	**0x04**, payload(3)
	START_SEARCH	**0x08**
	AUDIO_SYNC	**0x0A**, payload(6)
	CAPS_RESP	**0x0B**, payload(8+)
	MIC_OPEN_ERROR	**0x0C**, payload(2)

Network byte order (big-endian) is used to encode/decode 16-bit integers.

The remote should ignore any unrecognized extra data in the payload. This is required for capability with the future versions of Voice over BLE spec.

3 Capability

After the Remote Device is paired and connected, Android TV Device will send a command to get capabilities of the Remote Device. Android TV Device will include the highest version of the spec it supports. The Remote should then respond back with its supported spec version and capabilities.

GET_CAPS payload	
version (2)	The highest spec version that Android TV Device supports. **0x0100:** version 1.0.
legacy_0x0003 (2)	The legacy constant to comply with spec 0.4e. **0x0003:** constant value that is used for compatibility with the previous spec.
supported assistant interaction models (1)	Supported Assistant interaction model. The value of this parameter depends on the platform. See *Assistant interaction model*. **0x00:** only On-request model is supported; **0x01:** Press-to-Talk and On-request models are supported; **0x03:** Hold-to-Talk, Press-to-Talk and On-request models are supported.
CAPS_RESP payload	
version (2)	The spec version that is implemented on the Remote Device. **0x0100**: version 1.0; **0x0004**: version 0.4e.
codecs supported (1)	Audio codecs that are supported on the Remote Device. **0x01:** [0000_0001B] ADPCM 8khz/16bit;

	0x02: [0000_0010B] ADPCM 16khz/16bit (recommended); **0x03:** [0000_0011B] ADPCM 8khz/16bit & 16khz/16bit. The value should be used if the Remote Device supports "*Dynamic Bandwidth adjustment*".
assistant interaction model (1)	Assistant interaction model that is used by this Remote Device. The returned value should be aligned with "*supported assistant interaction models*" value provided by the AndroidTV Device. See *Assistant interaction model.* **0x00:** On-request model (should be supported by all remotes); **0x01:** Press-to-Talk model enabled; **0x03:** Hold-to-Talk model enabled.
audio frame size (2)	The desired audio packet size. The value is used in audio frame counting and usually matches the maximum payload size of a single notification, but can be any arbitrary number. *Some examples:* **0x0014:** 20 bytes (default); **0x00A0:** 160 bytes (recommended for 16kHz audio codec and 20ms connection interval).
extra configuration (1)	**0x01:** [0000_0001B] If set, host will attempt to enable DLE (and negotiate a new ATT_MTU size) which would allow to transfer an entire audio frame in a single bluetooth packet. *It's recommended for the Remote to implement the "GATT Client" role and negotiate ATT MTU and send DLE before sending CAPS_RESP message. Then there is no need to request DLE from the host side.*
reserved(1)	**0x00**: not used
remote fw data (any)	Optional field that remote can use to export data about the firmware installed on the remote. This information will be attached to any bug report and could help to identify problems specific to the FW version of the remote.

Android TV Device supports all spec versions up to the reported one. The Remote Device is expected to support at least one version that it reports back to the Android TV Device.

The Android TV Device won't communicate with the Remote devices if the major version number is outside of the supported range. (In other words Android TV Device reporting spec version 1.0 will work with any Remote Devices version up to 1.0, will attempt to work with Remote Devices versions up to 1.127, and won't work with any Remote Devices version 2.0 or higher)

For improved interoperability it's recommended that so called “universal” remotes implement the most common version of the spec that is supported by the majority of Android TVs (which is 0.4e at the moment). The remotes that are part of ATV packaging could implement any version of the spec that is supported by host Android TV device.

Note: The command to get capabilities can be sent multiple times depending on the state of Android TV Device. The Remote is expected to respond to each command appropriately.

The Remote device should send correct replies to the GET_CAPS payload specified in Google Voice over BLE specification 0.4e.

4 Audio

4.1 Recording

The audio recording feature of the Android TV Remote shall conform to the audio recording requirements in Section 5.4 of the Android 7.1 Compatibility Definition.
Check the Performance section in the “ATV Remote ERS” document for more information on microphone requirements.

4.2 Audio Streaming Format

The data rate is 8kHz [16kHz] at 16-bits per sample so 128kbps [256kbps] raw and then compressed 4:1 to 32kbps [64kbps] (IMA ADPCM).

Top nibble (bits 4-7) is decoded first, followed by bottom nibble (bits 0-3).

4.2.1 Encoding on the remote

The firmware for the remote should include software to encode the raw audio data using an IMA ADPCM encoder with a 4:1 compression ratio.

4.2.2 Audio Frame

Audio frames are transferred via **ATVV_CHAR_AUDIO** characteristic and contain encoded audio data.
The size of frame usually depends on the MTU. The default length is 20, but can be adjusted using DLE and “*bytes per characteristic*” field in the **CAPS_RESP** message.

4.3 Voice Transmission

The microphone on the Android TV Remote is intended to be used for voice input. If the remote has an LED it should be turned on for the whole duration of audio capture. Audio capture should not start if the notifications on the **ATVV_CHAR_AUDIO** characteristic are not configured or disabled.

4.3.1 Audio Start/Stop Messages

When the Remote Device is ready to transmit audio it sends **AUDIO_START** command followed by encoded audio frames.

AUDIO_START payload	
reason (1)	The reason for the audio stream to start. **0x00:** Audio transfer is triggered by MIC_OPEN request; **0x01:** PTT Audio transfer is triggered by “Assistant” button press; **0x03:** HTT Audio transfer is triggered by “Assistant” button press and will stop once the button is released.
codec used (1)	The audio codec that is used for this audio stream. **0x01:** [0000_0001B] ADPCM, 8kHz/16bit **0x02:** [0000_0010B] ADPCM 16kHz/16bit
stream id (1)	Audio stream identifier. **0x00:** if the ***reason*** field is *0x00;* **0x01..0x80:** an auto-incremented value if the ***reason*** field is not 0x00. The stream identifier is used by the **MIC_CLOSE** and **MIC_EXTEND** commands to prevent closing unintended audio sessions.

When the audio transmission finishes, **AUDIO_STOP** command should be sent by the Remote Device.

AUDIO_STOP payload	
reason (1)	The reason for the audio stream to finish.

	0x00: stop was triggered by **MIC_CLOSE** message; **0x02:** triggered by releasing an Assistant button during HTT interaction; **0x04:** triggered by an upcoming **AUDIO_START** command; **0x08:** triggered by "Audio Transfer Timeout"; **0x10:** triggered by disabling notifications for **ATVV_CHAR_AUDIO** characteristic; **0x80:** triggered by some other reason.

***Important: AUDIO_START** command is sent only when notifications are enabled for **ATVV_CHAR_AUDIO** characteristic. If notifications are being disabled while audio data transfer is in progress, **AUDIO_STOP** command shall be sent immediately with the reason: **0x10**.*

4.3.2 Audio Sync Message

AUDIO_SYNC message might be sent by the remote prior to any Audio Frame to synchronize the state of the audio stream.

The remote is expected to send the **AUDIO_SYNC** message when:

- some audio frames are not delivered (due to network congestion and insufficient internal memory to queue captured audio data);
- when audio stream configuration is changed (i.e. switching from 16kHz to 8kHz);
- periodic synchronization.

AUDIO_SYNC payload	
codec used (1)	The audio codec that will be used for the audio stream after this sync point. **0x01:** [0000_0001B] ADPCM, 8kHz/16bit; **0x02:** [0000_0010B] ADPCM 16kHz/16bit.
frame no (2)	Sequence number of the audio frame which is about to be sent. Monotonically increasing until rollover (starts at 0).
pred value (2)	Predicted ADPCM value.
step index (1)	Index in ADPCM step size table.

4.3.3 Dynamic Bandwidth adjustment

Due to limited Bluetooth bandwidth it’s hard to guarantee sustained data transfer speed at 64kbps. The remote can send an **AUDIO_SYNC** message and adjust configuration of audio stream.
The remote manufacturers have a few options to mitigate the problem and improve reliability of their Remote Devices:

1. Always use 8kHz sample rate which requires a fixed 32kbps channel.
2. Start 8kHz transfer and upgrade to 16kHz if there is enough bandwidth and only a limited number of retransmissions happen.
3. Attempt to transfer 16kHz audio, and fall back to 8kHz as when internal buffer is full and audio frames are about to be dropped. (Although some data is still dropped, but at least no more drops should happen in the future).
4. Use more sophisticated methods of switching between 8kHz and 16kHz streams without losing audio data.

The configuration of the last audio transmission can be reused for the next transmission if needed.

4.4 Requesting audio stream

MIC_OPEN command is sent by Android TV Device when it wants to access microphone data to which the Remote Device should reply with either **AUDIO_START** or **MIC_OPEN_ERROR** message.

MIC_OPEN payload	
mic mode (1)	Audio consumption mode. **0x00:** Playback mode. The audio data is intended for realtime audio consumption. The Remote Device should keep a reduced internal audio buffer worth of a few connection intervals of data. It should more aggressively adjust audio bandwidth and/or drop audio frames if needed to preserve time domain; **0x01:** Capture mode. The audio data is intended for non-realtime use cases. The Remote Device is allowed to use an increased internal audio buffer to avoid losses of audio packets by all means.

If **MIC_OPEN** command is received while microphone is already open, 2 following outcomes are possible:

1. if microphone is open because of a previously sent MIC_OPEN command, then the Remote should restart an audio stream and notify Android TV host with AUDIO_START message;
2. if microphone is open because of ongoing PTT/HTT interaction, the Remote should reply with **MIC_OPEN_ERROR(0x0F80)**. The ongoing audio stream should not be interrupted.

MIC_CLOSE message is sent by Android TV device to stop the voice transmission.
The message should only be handled by the remote control if the microphone is in open state and the requested stream id matches the stream id of the active audio stream.
If the message is handled, the remote control is expected to disable the microphone and has to stop sending any audio data via **ATVV_CHAR_AUDIO** characteristic. This action should be acknowledged by the remote control by replying with an **AUDIO_STOP** message.

MIC_CLOSE payload	
stream id (1)	The identifier of an audio stream to close. **0x00:** an audio stream which was initiated by the MIC_OPEN command; **0x01..0x80:** an audio stream which was not initiated by MIC_OPEN command (but rather by HTT or PTT interactions); **0xFF:** any ongoing audio stream.

MIC_EXTEND message is used to reset *"Audio Transfer Timeout"* if the microphone is still open. The Remote Device should not reply to the MIC_EXTEND message.

MIC_EXTEND payload	
stream id (1)	The identifier of an audio stream to extend. **0x00:** an audio stream which was initiated by the MIC_OPEN command; **0x01..0x80:** an audio stream which was not initiated by MIC_OPEN command (but rather by HTT or PTT interactions); **0xFF:** any ongoing audio stream.

MIC_OPEN_ERROR message is sent when a request to **MIC_OPEN** cannot be fulfilled.

MIC_OPEN_ERROR payload

error code (2)	The reason why the MIC_OPEN command has failed. **0x0F01:** reserved; **0x0F02:** remote is not active (see Active Remote Timeout); **0x0F03:** notifications are not enabled for **ATVV_CHAR_AUDIO** characteristic; **0x0F80:** ongoing PTT/HTT interaction is in progress; **0x0FFF:** internal error.

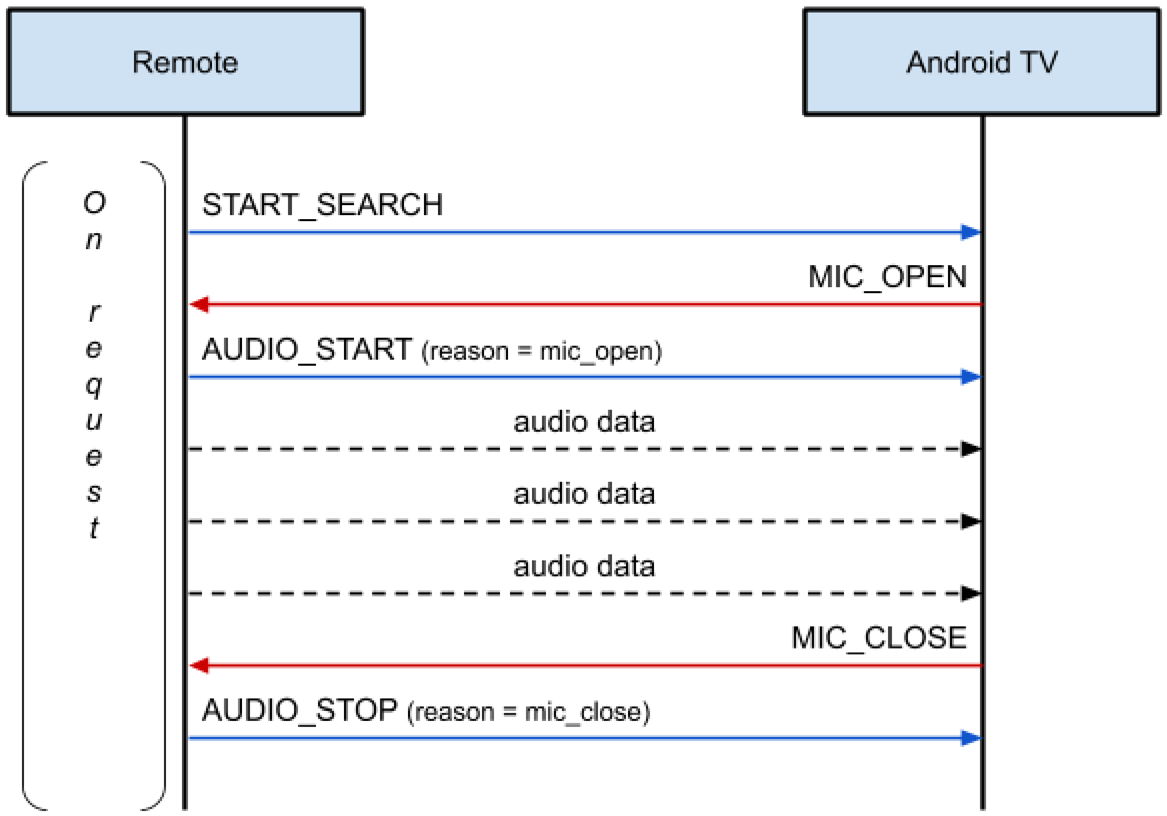
4.5 Assistant interaction model

The manufacturer of the remote defines the behaviour of the “Assistant” button. The On-request assistant interaction model **must be** supported by all Remote Devices and it should be used by default (once the remote is connected) before any other interaction model is negotiated by **GET_CAPS**, **CAPS_RESP** messages exchange.

4.5.1 On-request (legacy)

The remote notifies about *every* “Assistant” button press by sending **START_SEARCH** message in addition to a KEYCODE_ASSIST HID key event (HID: page 0x0C, command: 0x221) or KEYCODE_SEARCH.

The **START_SEARCH** message should be sent first followed by the HID event (*not shown on diagram*).

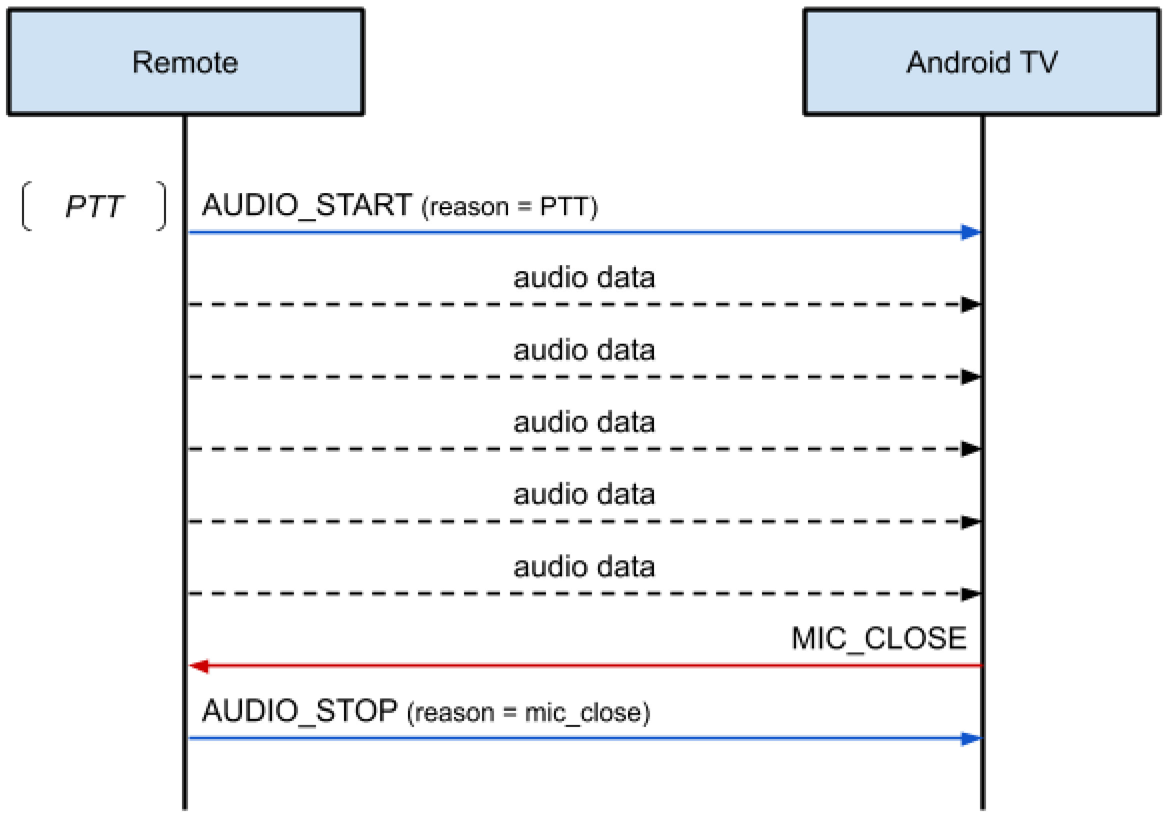


It is up to the Android TV Device to decide whether the audio data will be requested from remote or not using **MIC_OPEN** command.

If **MIC_OPEN** command was received before "Active Remote Timeout" completes, the Remote Device should send audio data and keep the microphone open until it receives a **MIC_CLOSE** message from the host device or "Audio Transfer Timeout" completes.
The Android TV Device might periodically (with 5-10 second interval) send **MIC_EXTEND** messages to restart "*Audio Transfer Timeout*" and extend audio recording duration.

4.5.2 Press-to-Talk

The remote sends **AUDIO_START** followed by audio data as soon as the "Assistant" button is pressed. In this interaction model KEYCODE_ASSIST key event **must not** be sent over HID interface.

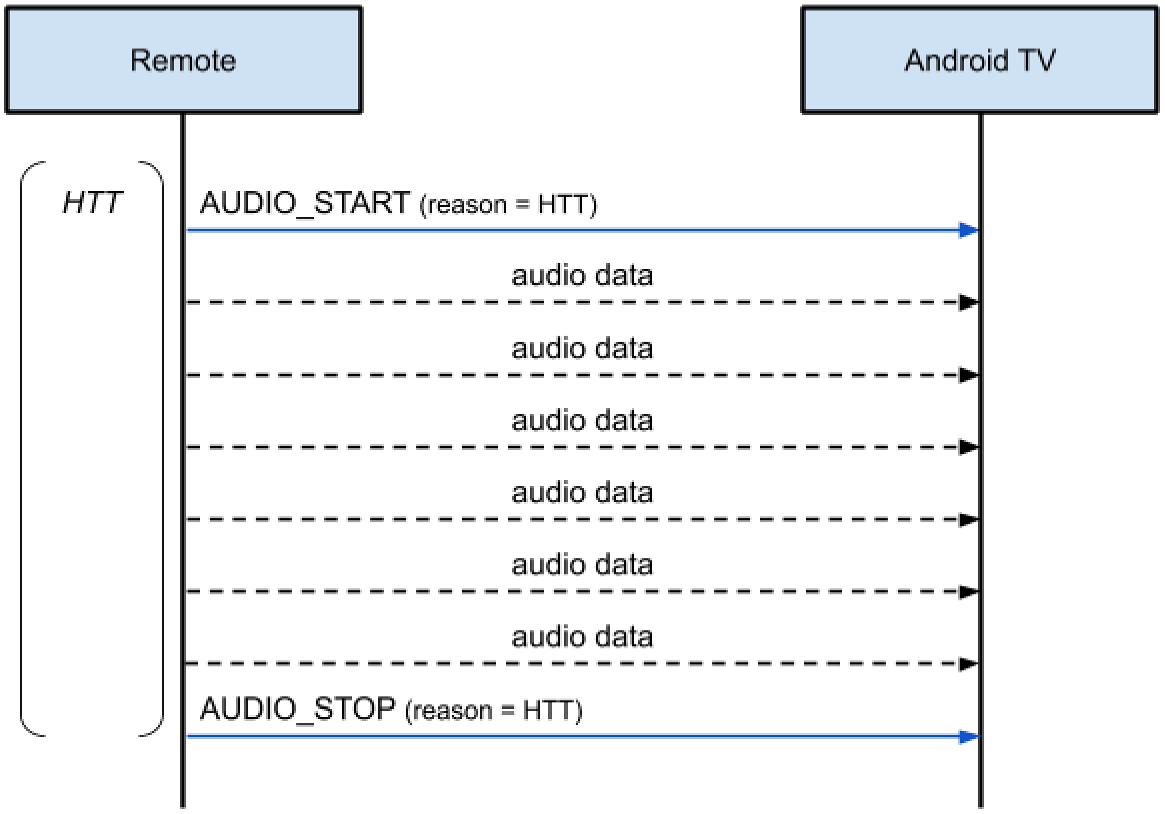


The Remote Device should send data and keep the microphone open until it receives a **MIC_CLOSE** message from the host device or “*Audio Transfer Timeout*” completes.
The Android TV Device might periodically (with 5-10 second interval) send **MIC_EXTEND** message to restart “*Audio Transfer Timeout*” and extend audio recording duration.

This interaction model might not be supported by the Android TV Device (see **GET_CAPS** message). The Remote Device should assume that Press-to-Talk interaction mode is not supported until **GET_CAPS** stated otherwise.

4.5.3 Hold-to-Talk

The remote sends **AUDIO_START** followed by audio data as soon as “Assistant” button is pressed. The remote sends **AUDIO_STOP** message as soon as “Assistant” button is released. In this interaction model KEYCODE_ASSIST key event **must not** be sent over HID interface.



The Android TV Device might still proactively stop recording by sending **MIC_CLOSE** command.

Note that "*Audio Transfer Timeout*" also applies to Hold-to-Talk model to avoid battery drain when Assistant button has been unintentionally pushed down. The Android TV Device might periodically (with 5-10 second interval) send **MIC_EXTEND** message to restart "*Audio Transfer Timeout*".

This interaction model might not be supported by the Android TV Device (see **GET_CAPS** message). The Remote Device should assume that Hold-to-Talk interaction mode is not supported until **GET_CAPS** stated otherwise.

4.6 Timeouts

4.6.1 Audio Transfer Timeout

This is a timeout on the Remote device, which is used to prevent battery drain when the mic is not closed correctly by the Android TV Device. (The timeout might differ for different voice interaction models)

Recommended value: between 15 seconds and 1 minute.

Timer start/reset: when **AUDIO_START** message is sent by Remote Device

Timer termination: when **MIC_CLOSE** command arrives.
Timer completion: send **AUDIO_STOP** message to Android TV device (no more audio packets should be sent)

4.6.2 Active Remote Timeout (optional)

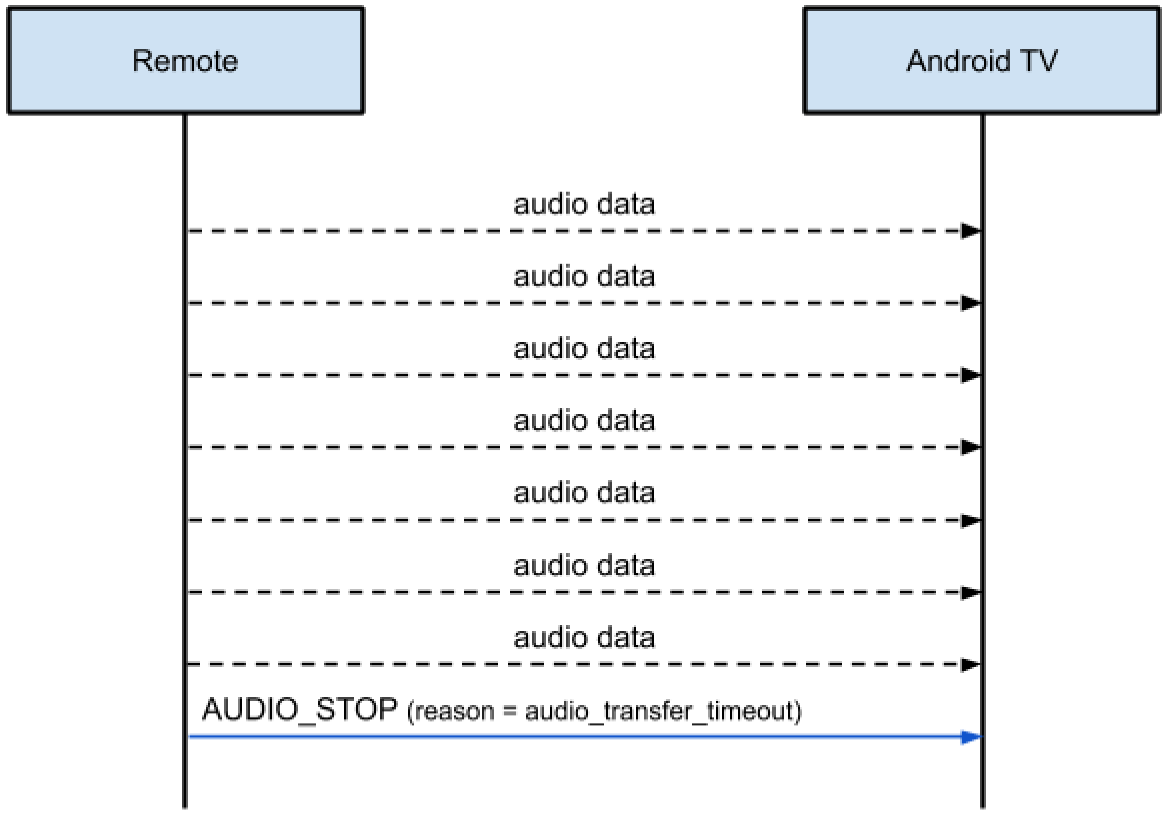
This timeout controls the interval when microphone can be opened by user request after the last user interaction with the remote. This is to address privacy concerns.

Recommended value: 1 minute
Timer starts/resets: Every time user interacts with the remote (i.e. pressing buttons, moving, using the voice capabilities of the remote).
On timer completion: any **MIC_OPEN** command should fail with 0x0F02 **MIC_OPEN_ERROR**.

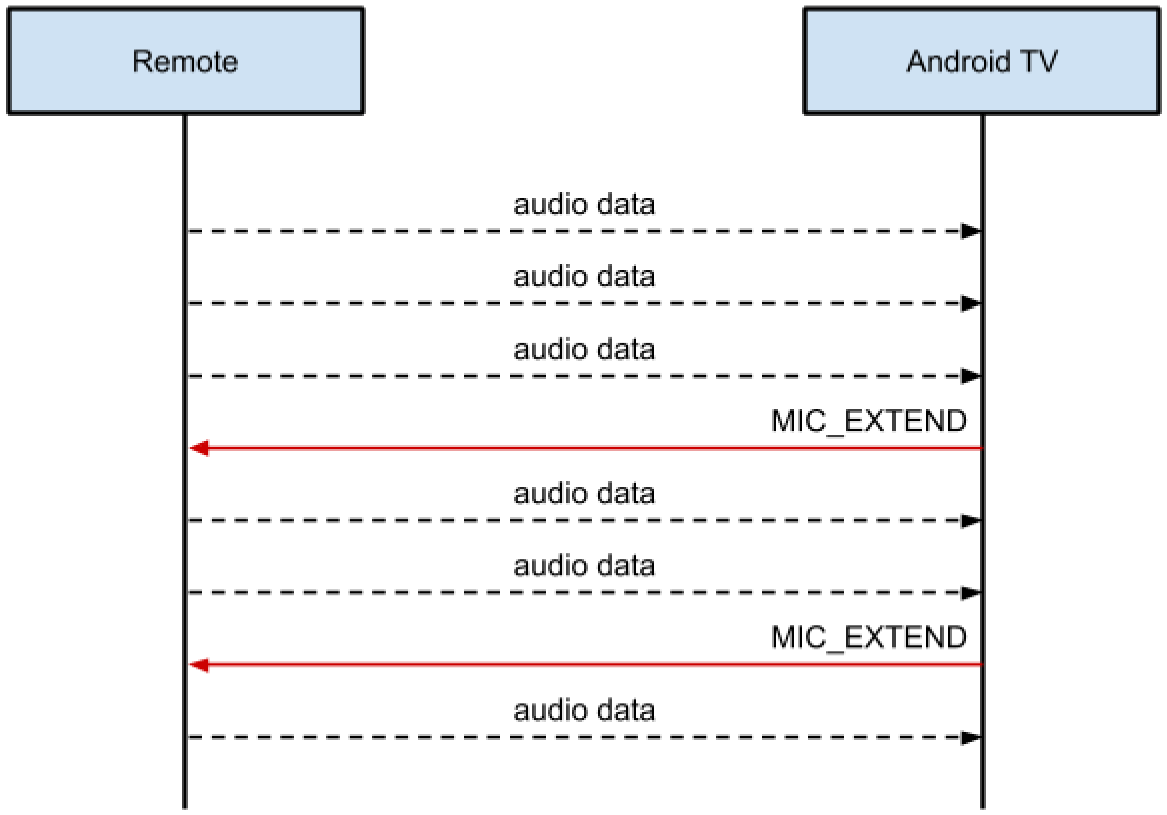
4.7 Non-trivial flow charts

4.7.1 Extended audio stream

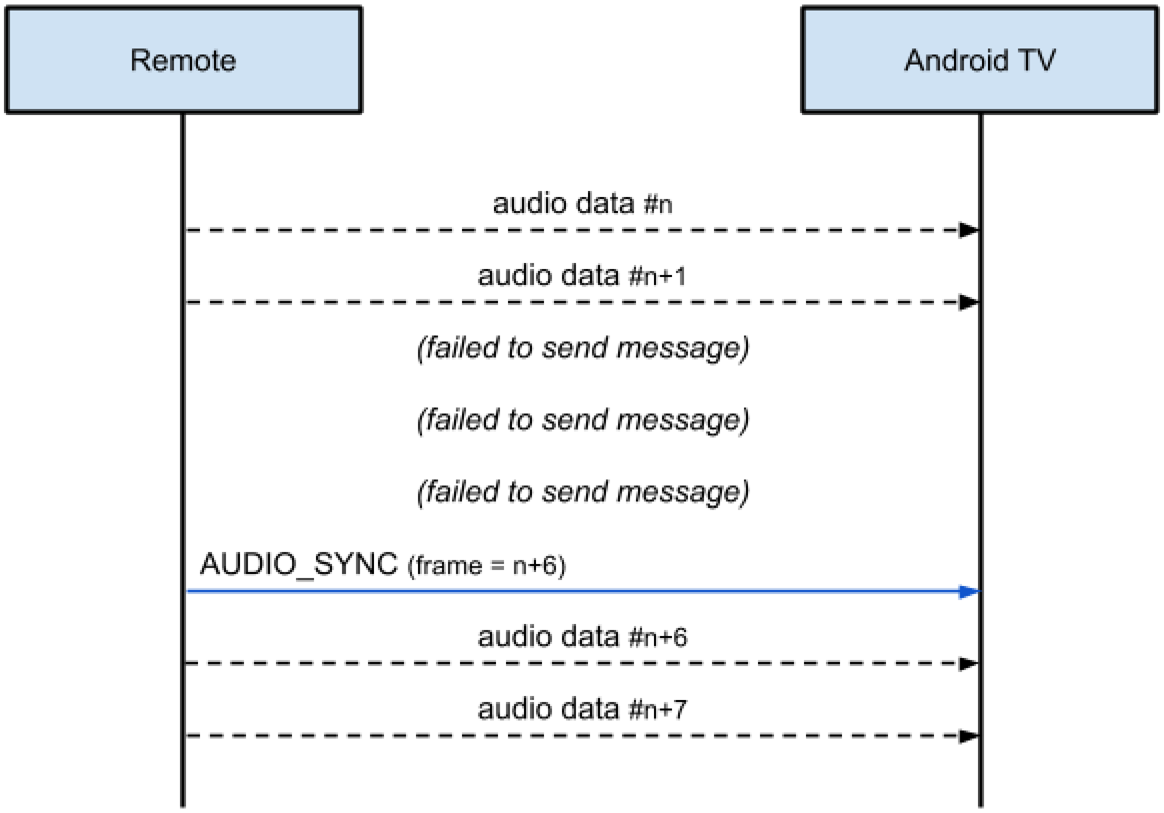
Audio Transfer Timeout prevents unclosed audio streams from running indefinitely and draining remote batteries and resources.



But at times user input requires long audio sessions. This problem is solved by the host by periodically sending **MIC_EXTEND** messages.

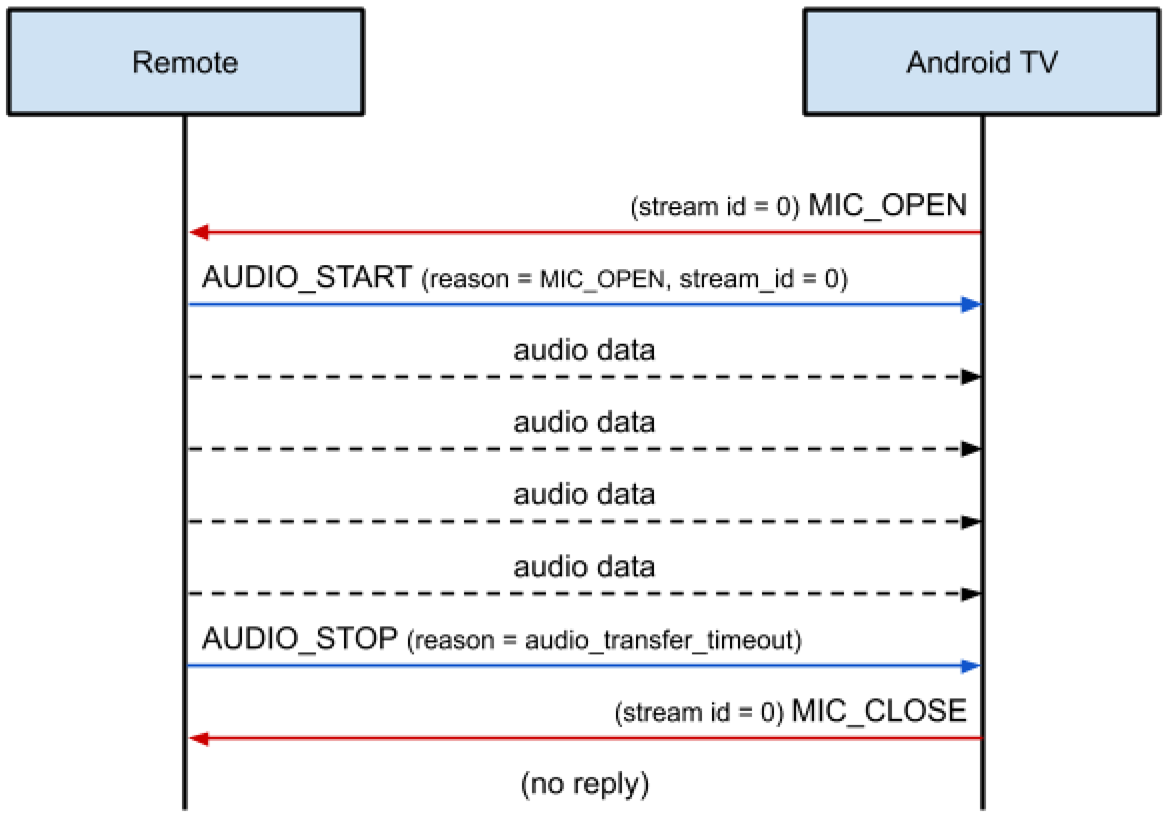


4.7.2 Audio sync messages



4.7.3 Closing non-existing audio stream

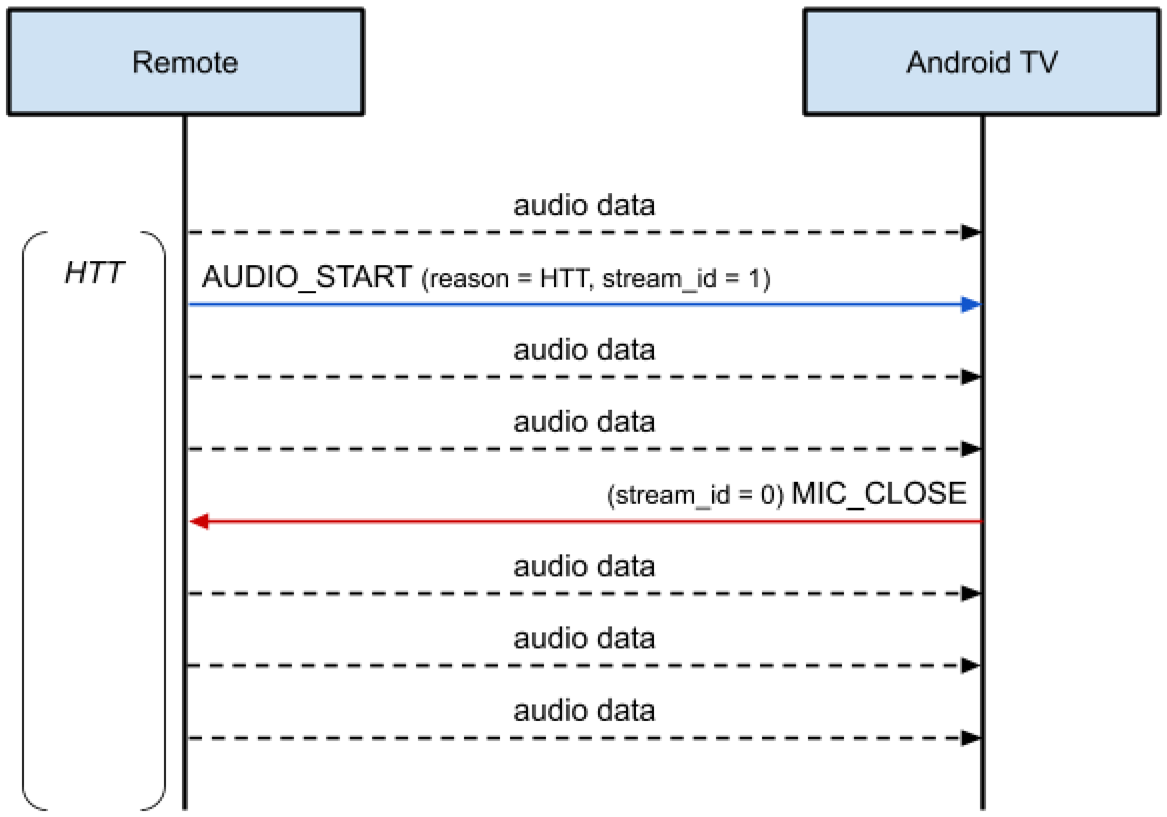
Because Bluetooth communication is asynchronous the Host does not always have up to date information about the state of the audio session the Remote owns. Could happen that the Remote and the Host are closing the audio stream at the same time, in that case the Remote can receive **MIC_CLOSE** command after the stream is already closed. The Remote shall ignore any MIC_CLOSE messages that do not match the current state of the audio stream.


Remote
Android TV
(stream id = 0) MIC_OPEN
AUDIO_START (reason = MIC_OPEN, stream_id = 0)
audio data
audio data
audio data
audio data
AUDIO_STOP (reason = audio_transfer_timeout)
(stream id = 0) MIC_CLOSE
(no reply)

4.7.4 On-request audio interrupted by HTT interaction

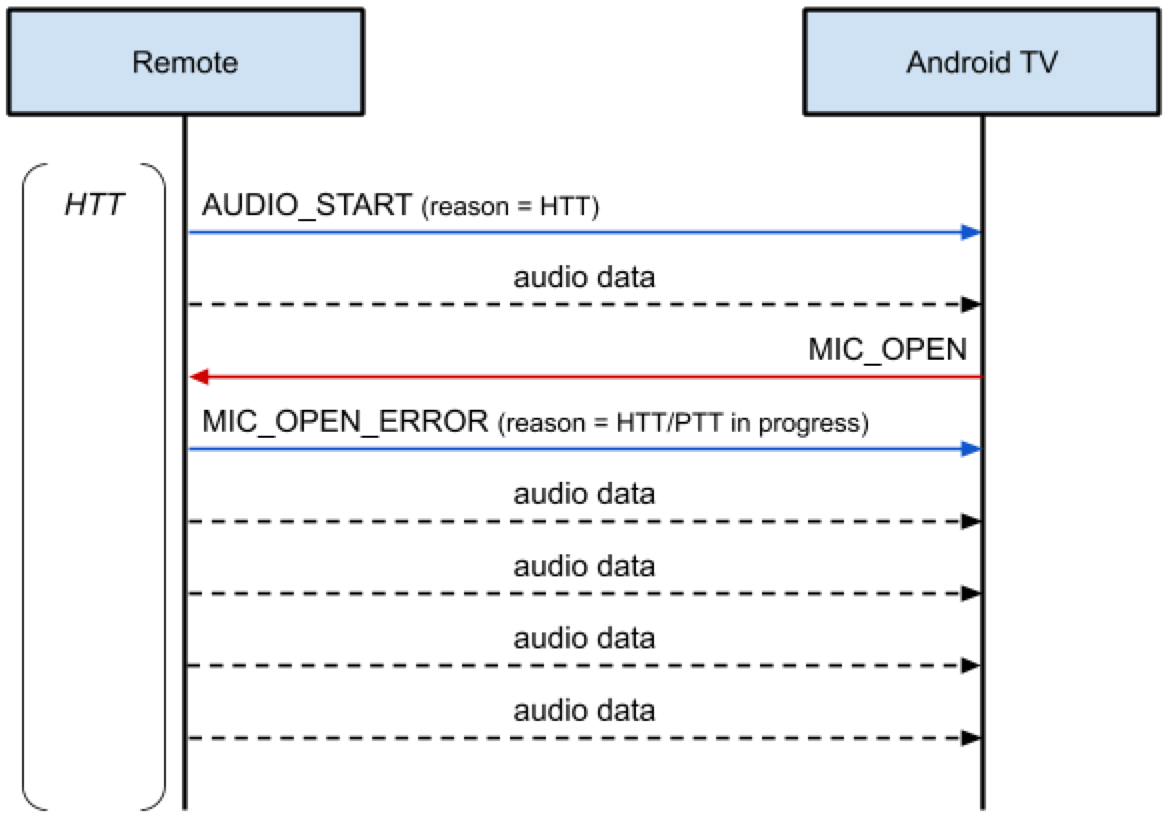
The following flow chart demonstrates the case when the “Assistant” button was used during an ongoing audio session.

Please note the “**MIC_CLOSE**” command that can theoretically be issued by the Host which is unaware of an incoming HTT interaction. In that case the MIC_CLOSE command is silently ignored by the remote as stream_id field doesn’t match the currently active audio stream.



4.7.5 HTT interaction at the same time as MIC_OPEN request

HTT and PTT sessions cannot be interrupted by MIC_OPEN request. When it happens **MIC_OPEN_ERROR** *0xF80* is returned without interrupting the audio data stream.



5 References

[Android 7.1 Compatibility Definition]
[Android TV Remote ERS]