IEEE 1722.1

2016-03-31

AVDECC Clarifications

Marc Illouz
marc.illouz@l-acoustics.com

Table of contents

1 - Presentation of L-Acoustics

2 - Examples of issues with AVDECC

- 2.1 - Inconsistency
- 2.2 - Ambiguity
- 2.3 - Lack

3 - Proposal of an « AVDECC clarification »

4 - List of very important questions

- 4.1 - ACMP
- 4.2 - AEM commands
- 4.3 - AEM model

1 - Presentation of L-Acoustics

- Major actor in the Pro Audio industry
  - Founded in 1984
  - Inventor of the WST in 1992
  - Covers big events over the world
- R&D
  - Fundamental acoustics
  - Signal processing
  - Electronics
  - AVB
- Today, we miss confidence in the interoperability of our AVDECC implementation

2.1 - Example of inconsistency: connection count field

- Definition of the « connection_count » field:

8.2.1.15 connection_count field

The connection_count field is used by the state commands to return the number of connections an AVDECC Talker thinks it has on its Stream source […]

- Definition of the « ListenerStreamInfo » structure:

8.2.2.2.2 ListenerStreamInfo

[…] - connected: 1 bit […]

Function called by the Talker on a CONNECT_TX_COMMAND:

8.2.2.6.2.2 connectTalker(command)

[…] The connectTalker function returns a response structure of type ACMPCommandResponse filled with the contents of the command parameter, with the stream_id, stream_dest_mac, stream_vlan_id and connection_count field set to the values for the AVDECC TalkerStreamInfos entry associated Stream. […]

Function called by the Listener on a GET_RX_STATE command:

8.2.2.5.2.10 getState(command)

The getState function returns a response structure of type ACMPCommandResponse filled with the contents of the command parameter, with the stream_id, stream_dest_mac, stream_vlan_id, connection_count, flags, talker_entity_id, and talker_unique_id fields set to the values for the AVDECC ListenerStreamInfos entry associated with the Stream identifier by the command structure. […]

Reference: AVDECC Clarifications – Part 1 ACMP, section 3.5 GET_RX_STATE_COMMAND and GET_RX_STATE_RESPONSE

2.2 - Example of ambiguity: SAVED_STATE flag

Definition of the SAVED_STATE field:

8.2.1.17 flags field

[…] SAVED_STATE: Connection has saved state (used in Get State only) […]

Reference to SAVED_STATE in the "Fast Connect" description:

8.2.2.1.1 Fast Connect

Fast connect is used in rapid boot mode when the AVDECC Listener has saved state, indicating that its input Stream sink is connected to a specific Entity ID and unique identifier. […]

Reference to SAVED_STATE in the "Controller Disconnect" description:

8.2.2.1.4 Controller Disconnect

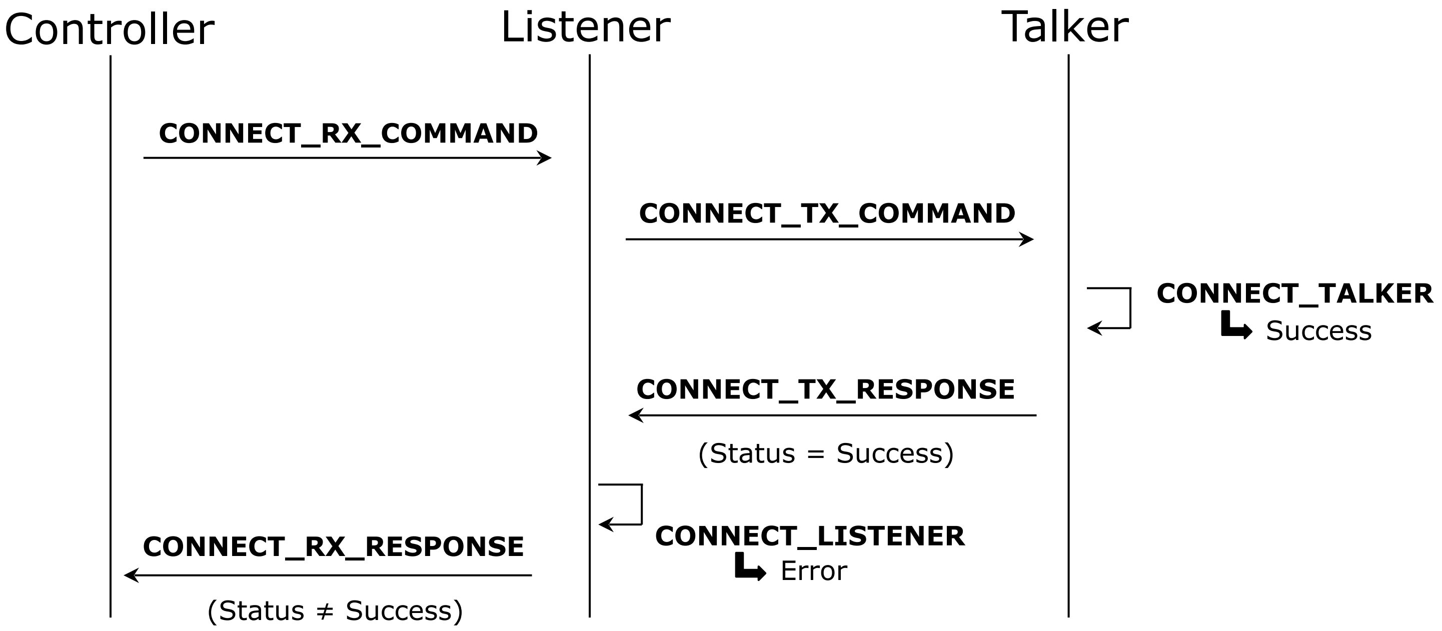
Controller disconnect mode is the normal of operation for tearing down a connection. In this mode, the AVDECC Controller shall send a DISCONNECT_RX_COMMAND to the AVDECC Listener. The AVDECC Listener removes any saved state and tears down the Stream. […]

- Interpretation 1: the Listener sink is currently connected
- Interpretation 2: the Listener sink is currently connected and has saved the connection info (talker entity id, talker unique id, controller id) for use in a future Fast Connect.
- Interpretation 3: the Listener is capable of performing Fast Connect (whatever the current state of the connection)

When a sink is currently attempting "Fast Connect": is it connected? Does it have saved state?

Reference: AVDECC Clarifications – Part 1 ACMP, section 3.5 GET_RX_STATE_COMMAND and GET_RX_STATE_RESPONSE

2.3 - Example of lack: half open connection



The Talker is connected, the Listener is not connected. The Controller cannot disconnect the Talker (a DISCONNECT_RX_COMMAND to the Listener will raise a NOT_CONNECTED error).

Similar scenario: after a successful connection, the Listener is removed from the network.

Reference: AVDECC Clarifications – Part 1 ACMP, sections "4.5 Connection succeeded on Talker and failed on Listener" and "4.7 Talker connected to a ghost Listener".

3 - Proposal of an “AVDECC clarification”

How:

- Make 3 documents : ACMP, AEM commands, AEM descriptors
- Build them incrementally
- Describe as precisely as possible the tricky points of the specification. Might contain errors in initial revisions
- Clearly mention the imprecisions and lacks of the IEEE 1722.1-2013 standard, so that everybody is aware of them

When:

- As soon as possible; several devices are AVB-ready but not fully AVDECC-compatible
- Before, or in parallel of, the new revision of the 1722.1 specification

Who:

- I can be in charge of writing the documents
- I will need support from the authors of IEEE 1722.1-2013 + experience and feedback from the implementors of AVDECC entities

4.1 - List of very important questions: ACMP (1/2)

- Clarifications of all the fields of the messages, especially:
  - "connection_count"
  - "flags" (CLASS_B, FAST_CONNECT, SAVED_STATE, STREAMING_WAIT, SUPPORTS_ENCRYPTED, ENCRYPTED_PDU)
- Connection/disconnection command accepted if locked/acquired by another controller?
- Is there a way to disable "Fast Connect" capability of a Listener?
- Is there a way to request a Listener to stop trying to connect in "Fast Connect" mode after a reboot (other than disconnecting it from its talker)?
- While trying to connect in "Fast Connect" mode, should the listener report that its sink is connected or not?
- If the "connectListener" function of the Listener state machine returns an error, should the Listener send a DISCONNECT_TX_COMMAND to the Talker (it is not conform to the specification, but seems more logical)?

4.1 - List of very important questions: ACMP (2/2)

- Never use the TALKER_NO_BANDWIDTH error code?
- Talker shall refuse a connection if MAAP failed (TALKER_DEST_MAC_FAIL error code)? What if MAAP succeeds but the allocated range is lost after connection has been established? Is the Talker allowed to allocate a new dest_adr for the connected source?
- On the Listener, how to behave if SRP parameters different from ACMP parameters (dest_adr, vlan_id, class)?
- Does the “Unique Id” part of the Stream ID chosen by the Talker have to be equal to the “Unique Id” of the ACMP source?

4.2 - List of very important questions: AEM commands

- SET_SAMPLING_RATE command is it allowed if one of the sinks/sources of the entity is active? connected?
- SET_STREAM_FORMAT command is it allowed if the source is connected?
- SET_STREAM_INFO command to a Listener sink: no interest?
- SET_STREAM_INFO command to a Talker source:
    - stream_id: allowed only if source not connected? Is it possible to return to default behavior after setting the stream_id?
    - msrp_accumulated_latency: not taken into account if source not connected?
    - stream_dest_mac: allowed if source already connected?
    - stream_vlan_id: allowed if source already connected?
    - CLASS_B: is it possible for the controller to select the SR Class used by a Talker source? Allowed if source already connected?
- GET_ASSOCIATION_ID: return 0x00000000 or 0xFFFFFFFF if association_id not valid yet?
- SET_CLOCK_SOURCE: is it ok not to implement this command? (if the entity automatically selects its clock source)

4.3 - List of very important questions: AEM descriptors

- STREAM_INPUT: how to advertise that a Listener supports any number of channels for its input streams?
- CLOCK_SOURCE: “clock_source_identifier” can be modified by a WRITE_DESCRIPTOR? What’s the meaning of “clock_source_flags” (STREAM_ID, LOCAL_ID)?