Discovery, connection and control specification for talkers and listeners

January 22, 2019

Revision 1.1a

Author

Marc Illouz (L-Acoustics)

Contributors

Andrew Elder (AudioScience)
Christophe Calmejane (L-Acoustics)
Cole Peterson (Meyer Sound)
Fabian Braun (d&b audiotechnik)
Frank An (Avid)
Genio Kronauer (L-Acoustics)
Henning Kaltheuner (d&b audiotechnik)
Morten Lave (Adamson)
Ray Dippert (Biamp)
Ray Tantzen (Presonus)
Rimas Avizienis (Meyer Sound)
Richard Bugg (Meyer Sound)
Vicent Perales (d&b audiotechnik)

And all members of the Avnu Pro Audio Technical Workgroup

Table of Contents

1. Introduction

This document describes the Avnu required method for providing interoperable and certified discovery, connection and control mechanisms for Professional Audio AVB devices.

This specification relies on a disambiguated subset of the IEEE 1722.1-2013 standard in order to achieve interoperability between Professional Audio AVB devices and controllers.

2. References

Name	*Reference*
802.1AS	IEEE 802.1AS-2011 “IEEE Standard for Local and metropolitan area networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks”.
802.1BA	IEEE 802.1BA-2011 “IEEE Standard for Local and metropolitan area networks - Audio Video Bridging (AVB) Systems”.
802.1Q	IEEE 802.1Q-2014 “IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks”.
AVDECC	IEEE 1722.1-2013, “IEEE Standard for Device Discovery, Connection Management, and Control Protocol for IEEE 1722™Based Devices”.
AVNU.IO.BASELINE	Milan Baseline interoperability specification.
AVNU.IO.FORMATS	Milan Formats interoperability specification – Revision 2.1.
AVNU.IO.MEDIACLOCKING	Milan Media clocking functional & interoperability specification – Revision 2.1.
AVNU.IO.REDUNDANCY	Milan Network redundancy interoperability specification – Revision 1.3.
AVTP	IEEE 1722-2016, “IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridges Local Area Networks”.

3. Glossary

Term	*Meaning*
Accumulated Latency	Worst case latency (in nanoseconds) that a Stream can encounter in its path from the Ingress Time Reference Plane of the Talker to a given point before or to

	the Presentation Time Reference Plane of a Listener. Refer to [AVTP, Clause 4.3.3] for the definition of the reference planes.
ACMP	AVDECC connection management protocol as described in [AVDECC, Clause 8].
ADP	AVDECC discovery protocol as described in [AVDECC, Clause 6].
AECP	AVDECC enumeration and control protocol as described in [AVDECC, Clause 9].
AECP AEM	The AECP-based protocol used to get and set AEM parameters.
AECP MVU	The AECP-based protocol used to get and set MVU parameters.
AEM	AVDECC Entity Model.
Attribute	An MRP attribute, as described in [802.1Q, Clause 10].
Audio Cluster	A group of audio channels processed as a whole in a PAAD. In this specification specifically, Audio Clusters always have exactly one channel.
Auto Connect	The process by which a Bound Sink is automatically performing all the necessary actions to receive a Stream from the remote Source identified by the Entity ID of the Talker and the index of the Source within this Talker. This process is described in section 8.2.6.
AVB	Audio Video Bridging. A set of specifications that allow time-synchronized, low-latency streaming services through IEEE 802 networks, as defined by [802.1BA].
AVB Domain	See [802.1BA].
AVB Interface	An AVB capable network interface capable of sourcing or sinking Streams and participating in an AVB Domain.
AVDECC	Audio/video discovery, enumeration, connection management, and control. A network protocol described in [AVDECC].
AVDECC Controller	A logical object within an end station, that implements the requirements of [AVDECC, Clause 5.4].
AVDECC Entity	A logical object within an end station, that implements the requirements of [AVDECC, Clause 5.3].
Avnu Pro Audio AAF Audio Stream Format	The AVTP Stream format defined by Avnu to transmit audio samples on a Pro Audio AVB network, as described in [AVNU.IO.FORMATS].

Avnu Pro Audio CRF Media Clock Stream Format	The AVTP Stream format defined by Avnu to transmit a media clock on a Pro Audio AVB network, as described in [AVNU.IO.MEDIACLOCKING].
Bound Sink	A Sink which has been successfully bound to a remote Source via the Connection Management protocol, as described in section 6.8.2.
Clock Domain	A source of a common clock signal within an AVDECC Entity.
Clock Source	A source of timing information for a Clock Domain.
Configuration	A collection of objects that defines the internal structure for a particular setup of the AVDECC Entity.
Controller	An AVDECC Controller.
Dynamic mapping	A Mapping which can be changed by an AVDECC Controller.
End station	See IEEE Std 802.
Entity ID	The EUI-64 identifier of an AVDECC Entity.
Entity Model ID	EUI-64 identifying the static model of the PAAD-AE and reported in the entity_model_id field as described in [AVDECC, Clause 6.2.1.9].
EUI-64	64-bit Extended Unique Identifier as defined by the IEEE. A EUI-64 is used to uniquely identify an object in a given context. It is composed of a part assigned by the IEEE Registration Authority to an organization (typically the 24 or 36 first bits), and a part assigned by the organization itself (typically the 40 or 28 last bits). Please refer to this document for more information: http://standards.ieee.org/develop/regauth/tut/eui.pdf
Input Stream	A Stream, from the point of view of the consuming Listener.
Listener	A PAAD which is able to receive and consume a Stream.
Listener attribute	An MSRP Listener attribute as described in [802.1Q, Clause 35].
Listener Asking Failed attribute	A Listener attribute which subtype is “Asking Failed”.
Listener Ready attribute	A Listener attribute which subtype is “Ready”.
Listener Ready Failed attribute	A Listener attribute which subtype is “Ready Failed”.

Locked Entity	An AVDECC Entity which has been locked by a Controller with the LOCK_ENTITY command.
MAAP	Mac Address Acquisition Protocol as described in [AVTP, Annex B].
Mapping	The mapping between a channel of an Audio Cluster and a channel of a Stream. Input mappings map Input Stream's channels to Audio Cluster's channels. Output mappings map Audio Cluster's channels to Output Stream's channels.
Milan	Audio networking solution designed and promoted by the Avnu Alliance, based on the AVB and AVDECC standards.
MRP	Multiple Registration Protocol as described in [802.1Q, Clause 10].
MSRP	Multiple Stream Registration Protocol as described in [802.1Q, Clause 35].
MVRP	Multiple VLAN registration protocol as described in [802.1Q, Clause 11.2].
MVU	Milan Vendor Unique.
Output Stream	A Stream, from the point of view of the producing Talker.
PAAD	A professional audio device with Ethernet AVB functionality.
PAAD-AE	The AVDECC Entity under consideration inside the PAAD.
PDU	Protocol Data Unit.
Probing Sink	A Sink which is currently probing a remote Source via the Connection Management protocol, as described in section 6.8.5.
R-PAAD	A PAAD that implements Redundancy as described in [AVNU.IO.REDUNDANCY].
R-PAAD-AE	The AVDECC Entity under consideration inside the R-PAAD.
Registered Controller	An AVDECC Controller which has registered to a PAAD-AE with the REGISTER_UNSOLICITED_NOTIFICATION command, and which is receiving unsolicited messages from this PAAD-AE each time its AEM state changes.
Settled Sink	A Sink which has successfully probed a remote Source via the Connection Management protocol, as described in section 6.8.5.

Sink	A component within an AVDECC Entity that consumes a Stream. In a given CONFIGURATION, each sink is identified by the index of the corresponding STREAM_INPUT descriptor.
Source	A component within an AVDECC Entity that produces a Stream. In a given CONFIGURATION, each source is identified by the index of the corresponding STREAM_OUTPUT descriptor.
SRP	Stream Reservation Protocol.
Started Sink	A Sink which is bound and which is set to process incoming AVTP Stream packets.
Static Mapping	A Mapping which cannot be changed by an AVDECC Controller.
Stopped Sink	A Sink which is bound and which is set to ignore incoming AVTP Stream packets.
Stream	A unidirectional flow of AVTP frames with the same Stream ID.
Stream Format	The set of parameters that describe the encoding of the audio or clock data of a Stream, as described in [AVNU.IO.FORMATS] and [AVNU.IO.MEDIACLOCING].
Stream ID	A 64-bit field that uniquely identifies a Stream, as defined in [802.1Q, Clause 35].
Stream's Destination MAC Address	The Destination MAC Address used by all the AVTP frames of the Stream.
Stream's Presentation time offset	The time offset (in nanoseconds) added by the Talker to the Ingress Time Reference Plane crossing time defined in [AVTP, Clause 4.3.3] to determine the AVTP timestamp of each transmitted sample.
Stream's Priority Code point	The value of PCP used by all the AVTP frames of the Stream.
Stream's Sampling Rate	The frequency (in Hz) at which the data of the Stream have been sampled by the Talker.
Stream's VLAN ID	The value of VLAN ID used by all the AVTP frames of the Stream.
Talker	A PAAD which is able to produce and transmit a Stream.
Talker Advertise attribute	An MSRP Talker Advertise attribute as described in [802.1Q, Clause 35].
Talker Attribute	Talker Advertise or Talker Failed attribute.
Talker Failed attribute	An MSRP Talker Failed attribute as described in [802.1Q, Clause 35].

VLAN	Virtual Local Area Network.

4. Scope

The intent of this specification is to define the set of AVDECC mechanisms that a PAAD must implement, and also to clarify some points of the AVDECC standard with regards to Professional Audio usage.

The main points of the AVDECC standard that are addressed in this specification are ADP (discovery), ACMP (connection management) and AECP (control). This specification introduces three clearly identified concepts for connection management: the “bound state”, the “settled state” and the “registering state”. These are successive states that allow a Sink to reach the final “connected state”. The “connected state” is a state where the device is successfully receiving and processing a Stream without errors; what constitutes “success” is vendor-specific and beyond the scope of this specification.

This specification considers a single AVDECC entity within a PAAD. If a physical device contains multiple AVDECC entities, they are viewed as independent PAADs by this specification. The AVDECC entity under consideration in the PAAD is referred to as the PAAD-AE.

When the PAAD-AE supports multiple CONFIGURATIONs, they must all comply with this specification.

5. Overview

5.1. Features

This specification defines a set of requirements that support the following features:

- Automatic discovery of the addition and removal of PAADs on the network
- Retrieval of the entity model of the discovered PAADs
- The ability to connect and disconnect streams between discovered PAADs
- The ability to get status information about the discovered PAADs and their connections
- The ability to control the discovered PAADs

5.2. Transport

The protocol defined in this specification operates on layer 2 Ethernet.

5.3. Exposed model

Each PAAD-AE exposes a public entity model to the network. This model follows the structure described in [AVDECC, Clause 7] and allows a generic controller to determine:

- The compatibility of the PAAD-AE with the current version of this specification
- How many input streams the PAAD-AE can sink, and the properties of each of them
- How many output streams the PAAD-AE can source, and the properties of each of them
- The redundant associations between the input/output streams of the PAAD-AE
- The clocking capabilities of the PAAD-AE, in terms of clock sources and sampling rates

- The static mappings, if any, between the internal processing channels of the PAAD-AE and the AVB stream channels

6. Entity model

6.1. Introduction

The Entity model is composed of static properties and dynamic properties. The static properties may not change while the PAAD-AE is online. The dynamic properties may change due to controller requests or various external events.

Note: the static properties of a PAAD-AE may be changed while the PAAD-AE is offline (due to a firmware upgrade for example). In this case, the PAAD-AE reports a different Entity Model ID so that controllers do not mistakenly use cached static properties the next time the PAAD-AE goes online.

6.2. Configurations and descriptors trees

As per [AVDECC], the static model exposed by an entity consists of descriptors organized in a tree structure. Each descriptor describes a specific functional block of the model.

This specification focuses on the following subset of the descriptors specified in [AVDECC, Clause 7]:

- ENTITY (1)
- CONFIGURATION (1..*)
    - STREAM_INPUT (0..*)
    - STREAM_OUTPUT (0..*)
    - AVB_INTERFACE (1..*)
    - CLOCK_DOMAIN (1..*)
    - CLOCK_SOURCE (1..*)
    - AUDIO_UNIT (0..*)
        - STREAM_PORT_INPUT (0..*)
            - AUDIO_CLUSTER (1..*)
        - STREAM_PORT_OUTPUT (0..*)
            - AUDIO_CLUSTER (1..*)
            - AUDIO_MAP (0..*)
    - “IDENTIFY” CONTROL (0..*)

A descriptor from one subtree shall not be contained in another subtree. In other words, each of the descriptors above, except the ENTITY and CONFIGURATION descriptors, shall have one, and only one, parent descriptor.

6.3. Descriptors

6.3.1. ENTITY

The PAAD-AE shall have exactly one ENTITY descriptor. This descriptor shall have the format specified in [AVDECC, Clause 7.2.1], with the following clarifications:

- The entity_model_id field shall be a valid EUI-64 (neither all zeros nor all ones).

- In the entity_capabilities field, the AEM_SUPPORTED, VENDOR_UNIQUE_SUPPORTED, CLASS_A_SUPPORTED and GPTP_SUPPORTED bits shall be set to 1. The AEM_PERSISTENT_ACQUIRE_SUPPORTED, GENERAL_CONTROLLER_IGNORE and ENTITY_NOT_READY bits shall be set to 0.
- The talker_stream_sources shall be set to the maximum number of Streams the AVDECC Talker is capable of sourcing simultaneously, among all possible CONFIGURATIONs.
- The listener_stream_sinks shall be set to the maximum number of Streams the AVDECC Listener is capable of sinking simultaneously, among all possible CONFIGURATIONs.

6.3.2. CONFIGURATION

The PAAD-AE may have any number of CONFIGURATION descriptors. Each CONFIGURATION descriptor shall have the format specified in [AVDECC, Clause 7.2.2].

6.3.3. AUDIO_UNIT

If at least one of the STREAM_INPUTs or at least one of the STREAM_OUTPUTs of a CONFIGURATION supports the Avnu Pro Audio AAF Audio Stream Format, then this CONFIGURATION shall contain at least one AUDIO_UNIT descriptor. Each AUDIO_UNIT descriptor shall have the format specified in [AVDECC, Clause 7.2.3].

The list of supported sampling rates of each AUDIO_UNIT shall correctly report the sampling rates supported by the AUDIO_UNIT.

An AUDIO_UNIT descriptor shall always report a sampling rate in the current_sampling_rate field that is one of the supported sampling rates described by the sampling rates list.

6.3.4. STREAM_INPUT and STREAM_OUTPUT

A CONFIGURATION shall contain at least one STREAM_INPUT or STREAM_OUTPUT descriptor. These descriptors shall have the format specified in [AVDECC, Clause 7.2.6], with the following clarifications.

In a STREAM_INPUT descriptor, the buffer_length field shall contain a valid value and this value shall be greater than or equal to 2126000 ns (2.126 ms), as per [AVTP, Clause 4.3.3, Equation (2)].

In the stream_flags field, the CLASS_A bit shall be set to 1.

The list of supported formats of each STREAM_INPUT/OUTPUT shall correctly report the formats supported by the stream, see [AVNU.IO.FORMATS] and [AVNU.IO.MEDIACLOCKING].

Each STREAM_OUTPUT referenced by one or more static mappings shall support only stream formats where these mappings are valid.

If a STREAM_INPUT/OUTPUT supports the Avnu Pro Audio CRF Media Clock Stream Format, it shall not support the Avnu Pro Audio AAF Audio Stream Format, and vice versa.

A STREAM_INPUT/OUTPUT descriptor shall always report a format in the current_format field that is one of the supported formats described by the formats list. Note that a single entry in the formats list can describe a range of formats when using the “up to” bit as described in [AVTP, Annex I.2.4].

As described in [AVNU.IO.REDUNDANCY, Annex A], a PAAD-AE may use the following extension for any of its STREAM_INPUT/OUTPUT descriptors, and shall use it for the STREAM_INPUT/OUTPUT descriptors that are part of a redundant pair:

Offset (Octets)	Length (Octets)	Name	Description
0	2	descriptor_type	The type of the descriptor. Always set to STREAM_INPUT or STREAM_OUTPUT.
2	2	descriptor_index	The index of the descriptor. This is the index of the Stream.
4	64	object_name	64-octet UTF-8 string containing a Stream name.
68	2	localized_description	The localized string reference pointing to the localized Stream name. See 7.3.6.
70	2	clock_domain_index	The descriptor_index of the Clock Domain providing the media clock for the Stream. See 7.2.9.
72	2	stream_flags	Flags describing capabilities or features of the Stream. See Table 7.9.
74	8	current_format	The Stream format of the current format, as defined in 7.3.2.
82	2	formats_offset	The offset from the start of the descriptor for the first octet of the formats. This field is 136 for this version of AEM.
84	2	number_of_formats	The number of formats supported by this audio Stream. The value of this field is referred to as N. The maximum value for this field is 47 for this version of AEM.

86	8	backup_talker_entity_id_0	The primary backup AVDECC Talker's Entity ID.
94	2	backup_talker_unique_id_0	The primary backup AVDECC Talker's Unique ID.
96	8	backup_talker_entity_id_1	The secondary backup AVDECC Talker's Entity ID.
104	2	backup_talker_unique_id_1	The secondary backup AVDECC Talker's Unique ID.
106	8	backup_talker_entity_id_2	The tertiary backup AVDECC Talker's Entity ID.
114	2	backup_talker_unique_id_2	The tertiary backup AVDECC Talker's Unique ID.
116	8	backedup_talker_entity_id	The Entity ID of the AVDECC Talker that this Stream is backing up.
124	2	backedup_talker_unique_id	The Unique ID of the AVDECC Talker that this Stream is backing up.
126	2	avb_interface_index	The descriptor_index of the AVB_INTERFACE from which this Stream is sourced or to which it is sinked.
128	4	buffer_length	The length in nanoseconds of the MAC's ingress or egress buffer as defined in Figure 5.4 of IEEE Std 1722-2011. For a STREAM_INPUT this is the MAC's ingress buffer size, and for a STREAM_OUTPUT this is the MAC's egress buffer size. This is the length of the buffer between the IEEE Std 1722-2011 reference plane and the MAC.
132	2	redundant_offset	The offset from the start of the descriptor for the first octet of the redundant_streams array. This field is 136+8*N for this version of AEM.

134	2	number_of_redundant_streams	The number of redundant streams supported by this Stream. The value of this field is referred to as R. The maximum value for this field is 8 for this version of AEM.
136	8*N	formats	Array of Stream formats of the supported formats, as defined in 7.3.2.
136+8*N	2*R	redundant_streams	Array of redundant STREAM_INPUT STREAM_OUTPUT descriptor indices. The present document doesn't specify any order for the elements of this array.

6.3.5. AVB_INTERFACE

Each CONFIGURATION of a PAAD-AE shall contain at least one AVB_INTERFACE descriptor. Each AVB_INTERFACE descriptor shall have the format specified in [AVDECC, Clause 7.2.8], with the following clarifications:

- In the interface_flags field, the GPTP_SUPPORTED and SRP_SUPPORTED bits shall be set to 1.

The PAAD-AE shall always use the same index, in all CONFIGURATIONs, for the AVB_INTERFACE descriptors that represent the same physical AVB interface of the PAAD.

6.3.6. CLOCK_SOURCE

Each CLOCK_DOMAIN of a CONFIGURATION shall have at least one associated CLOCK_SOURCE descriptor. Each CLOCK_SOURCE descriptor shall have the format specified in [AVDECC, Clause 7.2.9], with the following clarifications/changes.

For each STREAM_INPUT that supports the Avnu Pro Audio CRF Media Clock Stream Format as defined in [AVNU.IO.MEDIACLOCKING], or for the single STREAM_INPUT that supports the Avnu Pro Audio AAF Media Format as defined in [AVNU.IO.FORMATS] when the CONFIGURATION doesn't support CRF input, there shall be exactly one associated CLOCK_SOURCE descriptor with the following fields:

- clock_source_type = INPUT_STREAM
- clock_source_location_type = STREAM_INPUT
- clock_source_location_index = index of the STREAM_INPUT
- clock_source_flags: value not imposed by this specification
- clock_source_identifier: value not imposed by this specification

In addition, if the CONFIGURATION has at least one STREAM_OUTPUT, then there shall be at least one CLOCK_SOURCE descriptor associated with an internal clock, with the following fields:

- clock_source_type = INTERNAL

- clock_source_location_type: value not imposed by this specification
- clock_source_location_index: value not imposed by this specification
- clock_source_flags: value not imposed by this specification
- clock_source_identifier: value not imposed by this specification

There may be zero, one or several CLOCK_SOURCE descriptors associated with external sources:

- clock_source_type = EXTERNAL
- clock_source_location_type: value not imposed by this specification
- clock_source_location_index: value not imposed by this specification
- clock_source_flags: value not imposed by this specification
- clock_source_identifier: value not imposed by this specification

6.3.7. STREAM_PORT_INPUT and STREAM_PORT_OUTPUT

If a CONFIGURATION has at least one STREAM_INPUT which supports the Avnu Pro Audio AAF Audio Stream Format, then at least one of the AUDIO_UNITs of this CONFIGURATION shall contain a STREAM_PORT_INPUT descriptor. Each STREAM_PORT_INPUT descriptor shall have the format specified in [AVDECC, Clause 7.2.13].

If a CONFIGURATION has at least one STREAM_OUTPUT which supports the Avnu Pro Audio AAF Audio Stream Format, then at least one of the AUDIO_UNITs of this CONFIGURATION shall contain a STREAM_PORT_OUTPUT descriptor. Each STREAM_PORT_OUTPUT descriptor shall have the format specified in [AVDECC, Clause 7.2.13].

The STREAM_PORT_INPUT descriptors shall not have any attached AUDIO_MAP descriptors.

Note: this is because a PAAD-AE has to support dynamic mappings on input.

A PAAD that supports sample rate conversion between AUDIO_UNIT and STREAM_INPUT/OUTPUT shall indicate this by setting at least one of the SYNC_SAMPLE_RATE_CONV or ASYNC_SAMPLE_RATE_CONV bits.

Note: the interpretation of the distinction between the two bits is left open to the manufacturer.

6.3.8. AUDIO_CLUSTER

Each STREAM_PORT_INPUT of a CONFIGURATION shall contain at least one AUDIO_CLUSTER.

Each STREAM_PORT_OUTPUT of a CONFIGURATION shall contain at least one AUDIO_CLUSTER.

Each AUDIO_CLUSTER descriptor shall have the format specified in [AVDECC, Clause 7.2.16], with the following clarifications:

- The signal_type, signal_index, signal_output and path_latency fields are not required to be valid.
- The channel_count field shall be equal to 1.

Note: the requirement about channel_count is to allow the user to assign a different name to each individual channel.

6.3.9. AUDIO_MAP

The STREAM_PORT_INPUT descriptors of a CONFIGURATION shall not contain any AUDIO_MAP descriptor.

Note: this means that a PAAD-AE implements dynamic mappings on all of its STREAM_PORT_INPUT descriptors.

The STREAM_PORT_OUTPUT descriptors of a CONFIGURATION may contain zero, one or several AUDIO_MAP descriptors. Each AUDIO_MAP descriptor shall have the format specified in [AVDECC, Clause 7.2.19].

Note: this means that a PAAD-AE is free to implement static or dynamic mappings on its STREAM_PORT_OUTPUT descriptors.

If a CONFIGURATION has some AUDIO_MAP descriptors, there shall not exist more than 1 static mapping, among all the AUDIO_MAPs, referencing a given channel of a given STREAM_OUTPUT.

Note: the PAAD-AE is allowed to organize the list of static mappings as it prefers, in any number of AUDIO_MAP descriptors, with any number of static mappings (including zero) in each AUDIO_MAP and any ordering of the static mappings within the AUDIO_MAPs For example, a given implementation could decide to return a total of 8 static mappings in 3 AUDIO_MAPs: one AUDIO_MAP (index 0) with 2 static mappings, one AUDIO_MAP (index 1) with zero static mapping and one AUDIO_MAP (index 2) with 6 static mappings.

6.3.10. “IDENTIFY” CONTROL

A CONFIGURATION may contain zero, one or several “IDENTIFY” CONTROL descriptors. These descriptors shall have the format specified in [AVDECC, Clause 7.2.22] and [AVDECC, Clause 7.3.4.2]. This specification however doesn’t require the block_latency, control_latency, signal_type, signal_index and signal_output fields to be valid.

Each “IDENTIFY” CONTROL descriptor should be associated with a manufacturer-defined visual/audible effect on the PAAD.

When the PAAD-AE contains one or several "IDENTIFY" CONTROL descriptors, one of them is referred by the manufacturer as being the primary "IDENTIFY" CONTROL, and this primary "IDENTIFY" CONTROL shall exist in all CONFIGURATIONs and shall have always the same index.

6.3.11. CLOCK_DOMAIN

A CONFIGURATION shall contain at least one CLOCK_DOMAIN descriptor. Each CLOCK_DOMAIN descriptor shall have the format specified in [AVDECC, Clause 7.2.32].

6.4. ENTITY dynamic state

6.4.1. Locked state

At a given time, a PAAD-AE may be either locked by a controller or not locked by any controller. A PAAD-AE cannot be locked by two different controllers at the same time.

A PAAD-AE is locked and unlocked by using the LOCK_ENTITY command.

When a PAAD-AE is locked by a controller, it shall keep the following information about the locking controller:

- The Entity ID of the locking controller

While a PAAD-AE is locked, it shall not allow non-AVDECC controllers (proprietary remote control software, front-panel of the device, …) to change the state of the Entity.

The locked state is cleared by a power cycle.

6.4.2. List of registered controllers

A PAAD-AE shall maintain a list of registered controllers. Each entry of this list contains the following elements:

- The Entity ID of the registered controller
- The MAC address of the registered controller
- The port number of the PAAD on which the controller has registered
- The Sequence ID of the next unsolicited notification

This list contains one entry per controller which has successfully registered using the REGISTER_UNSOLICITED_NOTIFICATION command, and not been deregistered yet.

This list doesn't contain duplicates: two different entries shall have a different combination of (Entity ID, MAC Address, port number).

Due to resource limitations, a PAAD-AE may not be capable of registering any number of controllers. This specification requires that a PAAD-AE be able to register at least 1 controller.

The list of registered controllers is cleared by a power cycle.

6.5. AUDIO_UNIT dynamic state

6.5.1. Sampling rate

At any given time, an AUDIO_UNIT must be running at one of the sampling rates it supports.

The current sampling rate shall be saved in a non-volatile memory and restored after a power cycle.

Note: the sampling rate of an AUDIO_UNIT is the frequency at which audio samples are processed in the unit. It's also the frequency at which samples enter and exit the unit. Depending on the presence of sample rate converters, a PAAD-AE may or may not be able to receive/transmit streams at a given sampling rate and process samples at a different sampling rate.

6.6. AVB_INTERFACE dynamic state

There are a number of dynamic values that the PAAD-AE shall maintain and expose through the Control layer.

6.6.1. gPTP

For each AVB-capable interface:

- gPTP grandmaster clock ID
- gPTP path sequence to the grandmaster
- gPTP domain number
- gPTP propagation delay

6.6.2. MSRP

For each AVB-capable interface:

- MSRP Domain parameters (for each supported SR Class: priority and default VLAN ID)

6.6.3. Diagnostic counters

For each AVB-capable interface, the PAAD-AE shall keep track of the following counters (each counter is a 32-bit unsigned integer that wraps over to zero when it reaches the maximum value):

Symbol	**Meaning**
LINK_UP	Number of times the link state has changed from down to up, since boot.
LINK_DOWN	Number of times the link state has changed from up to down, since boot. The PAAD-AE shall ensure that at any time, either LINK_UP=LINK_DOWN (in this case, the link is currently down), or LINK_UP=LINK_DOWN+1 (in this case, the link is currently up).
GPTP_GM_CHANGED	Number of gPTP GM changes, since boot.

The PAAD-AE may keep track of the following counters (each counter is a 32-bit unsigned integer that wraps over to zero when it reaches the maximum value):

Symbol	**Meaning**
FRAMES_TX	Total number of frames sent out of this interface, since boot. All frames are counted here, no matter their type.

FRAMES_RX	Total number of frames of frames received through this interface, since boot. All frames are counted here, no matter their type.
RX_CRC_ERROR	Total number of frames received through this interface, with an incorrect CRC, since boot.

6.7. STREAM_OUTPUT dynamic state

6.7.1. Format

The format is the set of parameters that describe the encoding of the audio or clock data, as described in [AVNU.IO.FORMATS] and [AVNU.IO.MEDIACLOCKING].

A STREAM_OUTPUT descriptor shall always be using a format that is one of the supported formats described by the formats list. Note that a single entry in the formats list can describe a range of formats when using the “up to” bit as described in [AVTP, Annex I.2.4].

The current format shall be saved in a non-volatile memory and restored after a power cycle.

6.7.2. SRP state

At a given time, for each STREAM_OUTPUT of the currently set CONFIGURATION, the PAAD may:

- Be declaring an MSRP Talker attribute (Talker Advertise or Talker Failed) corresponding to the stream
- Or not be declaring any Talker attribute

When the PAAD is declaring a Talker attribute, it may:

- Be registering a Listener Ready or Listener Ready Failed attribute
- Or be registering a Listener Asking Failed attribute
- Or not be registering any Listener attribute

For each of the STREAM_OUTPUTs of its currently set CONFIGURATION, a PAAD shall always declare an MSRP Talker attribute as soon as it has valid SRP parameters for this stream (see section 6.7.4 SRP parameters).

6.7.3. Streaming state

As long as a PAAD is declaring a Talker Advertise attribute and receiving a Listener Ready or Listener Ready Failed attribute for a STREAM_OUTPUT, it shall be streaming AVTP packets. This specification excludes the possibility for a STREAM_OUTPUT to be stopped (STREAMING_WAIT state shall not be implemented).

6.7.4. SRP parameters

Each STREAM_OUTPUT shall maintain the following SRP parameters:

- Stream ID
- Stream Destination MAC Address
- Stream VLAN ID
- Stream Priority code point

At a given time, each of these SRP parameters may be valid or not. The meaning of “valid” is left open to the implementer. When all of the SRP parameters associated to a STREAM_OUTPUT of the currently set CONFIGURATION are valid, the PAAD shall declare a Talker attribute (Talker Advertise or Talker Failed) for this STREAM_OUTPUT, and the valid SRP parameters shall be used to populate the First Value of the attribute.

6.7.5. Plug-and-play SRP parameters (informative)

This section describes a possible plug-and-play implementation of a Talker PAAD-AE, but doesn’t preclude any other implementation.

In this implementation, the PAAD-AE builds automatically the Stream ID based on the MAC Address of the AVB interface used to source the stream, it dynamically allocates the Stream Destination MAC Address using the MAAP protocol, and it takes the Stream VLAN ID and Stream Priority code point from the Domain attribute.

Below is the method this PAAD uses to allocate the SRP parameters of a declared SRP Talker attribute:

SRP parameter	**Allocation mechanism / validity**
Stream ID	The PAAD decides by itself the value of the Stream ID used for its stream. The first 6 bytes are equal to the MAC Address of the AVB interface associated with this STREAM_OUTPUT. The 2 last bytes can be used indifferently by the PAAD to distinguish between the different Streams it is sourcing from this AVB interface. *Note: the PAAD may always use the same Stream ID for a given STREAM_OUTPUT descriptor, or change each time it starts declaring an MSRP Talker attribute for this stream.* This parameter is always valid.

Stream Destination MAC Address	The PAAD uses MAAP to allocate a unique Destination MAC Address for each of its STREAM_OUTPUTs. A STREAM_OUTPUT cannot be declaring a Talker attribute until MAAP has run and has allocated a Destination MAC Address for this STREAM_OUTPUT. While a STREAM_OUTPUT is declaring a Talker attribute, if MAAP reports a conflict, the PAAD withdraws the Talker attribute, waits for 2 LeaveAll periods and allocates a new Destination MAC Address, then declares a new Talker attribute with the new Destination MAC Address. *Note: at that time, the PAAD may decide to use the same Stream ID as before, or another Stream ID.* This parameter is valid as long as MAAP has successfully allocated an address and is not reporting a conflict. When invalid, the Stream Destination MAC Address is set to zero.
Stream VLAN ID	The PAAD uses, at the time it starts declaring the Talker attribute, the Default VLAN ID associated with the SR Class of the STREAM_OUTPUT on the AVB interface used to source this Stream. While a STREAM_OUTPUT is declaring a Talker attribute, even if the Default VLAN ID changes in the AVB Domain, the PAAD does not change the VLAN ID of the STREAM_OUTPUT. This parameter is always valid.
Stream Priority code point	The PAAD uses, at the time is starts declaring the Talker attribute, the Priority code point associated with the SR Class of the STREAM_OUTPUT on the AVB interface used to source this Stream. While a STREAM_OUTPUT is declaring the Talker attribute, if the Priority code point changes in the AVB Domain, the PAAD withdraws the Talker attribute, waits for 2 LeaveAll periods and then declares a new Talker attribute with the new Priority code point. *Note: at that time, the PAAD may decide to use the same Stream ID as before, or another Stream ID.* This parameter is always valid.

6.7.6. Presentation time offset

The Presentation time offset is the value a PAAD adds to the Ingress Time Reference Plane crossing time defined in [AVTP, Clause 4.3.3] to determine the AVTP timestamp of each transmitted sample. As per [AVTP, Clause 4.3.3], the PAAD additionally adds some fixed delay value (between 0 and 125 microseconds) to compensate the uncertainty about the estimation of the crossing time.

By default, the PAAD-AE shall use a 2ms Presentation time offset for all its STREAM_OUTPUTs. *Note: this means that, supposing the timing uncertainty of the PAAD is 125us, the PAAD determines the AVTP timestamp of each sample by adding 2.125 ms to the estimated crossing time of the sample.*

The Presentation time offset can be changed to any value in the range between 0x0 and 0x7FFFFFFF nanoseconds by using the SET_STREAM_INFO command.

The Presentation time offset shall be saved in a non-volatile memory and restored after a power cycle.

6.7.7. Diagnostic counters

For each STREAM_OUTPUT, the PAAD-AE shall keep track of the following counters (each counter is a 32-bit unsigned integer that wraps over to zero when it reaches the maximum value):

Symbol	**Meaning**
STREAM_START	Incremented each time the Talker starts streaming.
STREAM_STOP	Incremented each time the Talker stops streaming. At any time, the PAAD-AE shall ensure that either STREAM_START=STREAM_STOP+1 (in this case, the Talker is currently streaming), or STREAM_START=STREAM_STOP (in this case, the Talker is not currently streaming).
MEDIA_RESET	Incremented at the end of every observation interval during which the “mr” bit has been toggled in any of the transmitted Stream Data AVTPDUs. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second. Reset to 0 each time the Talker starts streaming.
TIMESTAMP_UNCERTAIN	Incremented at the end of every observation interval during which the “tu” bit has been set in any of the transmitted Stream Data AVTPDUs. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second. Reset to 0 each time the Talker starts streaming.

FRAMES_TX	Incremented at the end of every observation interval during which at least one Stream Data AVTPDU has been transmitted on this STREAM_OUTPUT. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second. Reset to 0 each time the Talker starts streaming.

6.8. STREAM_INPUT dynamic state

6.8.1. Format

The format is the set of parameters that describe the encoding of the audio or clock data, as described in [AVNU.IO.FORMATS] and [AVNU.IO.MEDIACLOCKING].

A STREAM_INPUT descriptor shall always be using a format that is one of the supported formats described by the formats list. Note that a single entry in the formats list can describe a range of formats when using the “up to” bit as described in [AVTP, Annex I.2.4].

The current format shall be saved in a non-volatile memory and restored after a power cycle.

6.8.2. Bound state

A STREAM_INPUT is said to be bound when it has successfully been associated with a remote STREAM_OUTPUT through the Controller Bind operation described in section 8.2.3. Controller Bind.

Note: “bound” doesn’t necessarily mean that data is flowing or even that bandwidth is reserved.

At a given time, a STREAM_INPUT is either bound to one remote STREAM_OUTPUT, or not bound.

The current bound state shall be saved in a non-volatile memory and restored after a power cycle.

6.8.3. Binding parameters

For each bound STREAM_INPUT, a PAAD-AE shall save the following information, called the binding parameters, in a non-volatile memory:

- Entity ID of the talker to which it is currently bound
- Index of the talker’s source to which it is currently bound
- Entity ID of the controller that requested the binding
- Started/stopped state

The binding parameters are cleared when the STREAM_INPUT gets unbound.

6.8.4. Talker's discovered state

A PAAD shall run an instance of the ADP Discovery state machine for each of its bound STREAM_INPUTs. The state machine shall be as described in section 9.4 Listener's discovery state machine (*note: it differs from [AVDECC, Clause 6.2.6]*).

The current state may be either Discovered or Not discovered.

6.8.5. Probing/settled state

At a given time, a bound STREAM_INPUT may be either probing the bound source, or settled on the bound source. The PAAD-AE automatically enters the probing state when it needs to query stream parameters just after binding, or after the stream has been disrupted for some reason (including a power cycle of the PAAD). When the stream source returns and stream parameters have been probed, the PAAD-AE becomes settled.

Note: the probing/settled state is undefined when the STREAM_INPUT *is not bound.*

Note: "settled" doesn't necessarily mean that data is flowing or even that bandwidth is reserved.

A PAAD-AE having a settled STREAM_INPUT shall be monitoring for a corresponding MSRP Talker Advertise or Talker Failed attribute which matches the SRP parameters stored by the PAAD-AE for this STREAM_INPUT. If a Talker Advertise attribute with matching SRP parameters is being received, the PAAD-AE shall be declaring an MSRP Listener Ready attribute. If no Talker Advertise attribute with matching SRP parameters is being received, the PAAD-AE may either be declaring a Listener Asking Failed attribute, or not be declaring anything.

For further details, please refer to section 8 Connection Management.

6.8.6. Probing status and ACMP status

For each STREAM_INPUT of the currently set CONFIGURATION, the PAAD-AE shall maintain a Probing status variable and an ACMP status variable. The aim is to give information to the user about the current status of the probing process of this STREAM_INPUT.

The Probing status variable is a 3-bit integer; the possible values are as follows:

Value	Name	Description
0	PROBING_DISABLED	The sink is not probing because it is not bound. This corresponds to the UNBOUND state of the Listener's sink state machine (refer to section 8.3).
1	PROBING_PASSIVE	The sink is probing passively. It waits until the bound talker has been discovered.

		This corresponds to the PRB_W_AVAIL state of the Listener's sink state machine (refer to section 8.3).
2	PROBING_ACTIVE	The sink is probing actively. It is querying the stream parameters to the talker. This corresponds to the PRB_W_DELAY, PRB_W_RESP, PRB_W_RESP2 and PRB_W_RETRY states of the Listener's sink state machine (refer to section 8.3).
3	PROBING_COMPLETED	The sink is not probing because it is settled. This corresponds to the SETTLED_NO_RSV and SETTLED_RSV_OK states of the Listener's sink state machine (refer to section 8.3).
4 to 7	-	Reserved for future use.

The ACMP status variable is a 5-bit integer which is defined only when the Probing status is PROBING_ACTIVE. In this case, it reports any errors encountered during the last probe sequence. Otherwise, it is 0. The possible values are as defined in [AVDECC, Table 8.2]. Please refer to section 8.3 for rules that define when the ACMP status variable is updated by the PAAD-AE.

6.8.7. Started/stopped state

At a given time, a bound STREAM_INPUT may be either started or stopped.

A PAAD-AE having a started STREAM_INPUT shall process the incoming stream data as soon as the SRP reservation is done and the Stream AVTPDUs are received. A PAAD-AE having a stopped STREAM_INPUT shall discard the Stream AVTPDUs it receives.

Note: the started/stopped state is undefined when the STREAM_INPUT *is not bound.*

The current started/stopped state shall be saved in a non-volatile memory and restored after a power cycle.

6.8.8. SRP state

At a given time, for each settled STREAM_INPUT of the currently set CONFIGURATION, the PAAD may:

- Either be registering a Talker Advertise attribute that matches the SRP parameters probed for the settled stream (Stream ID, Stream Destination MAC Address and Stream VLAN ID)

- Or be registering a Talker Failed attribute that matches the SRP parameters probed for the settled stream (Stream ID, Stream Destination MAC Address and Stream VLAN ID)
- Or not be registering any of the attributes mentioned above

Note: the SRP state is not registering when the STREAM_INPUT *is not settled.*

6.8.9. SRP parameters

Each settled STREAM_INPUT shall record the following SRP parameters:

- Stream ID
- Stream Destination MAC Address
- Stream VLAN ID

In case the STREAM_INPUT is not settled, the Stream ID, Stream Destination MAC Address and Stream VLAN ID shall be all zeros.

In case the STREAM_INPUT is settled, the Stream ID, Stream Destination MAC Address and Stream VLAN ID shall be exactly as in the last received PROBE_TX_RESPONSE (refer to section 8 Connection management for more details about these messages).

Please note that the values of the SRP parameters used by the STREAM_INPUT are not dependent upon the content of any potentially registered SRP Talker attribute. If, at some point, the SRP parameters declared by the talker differ from the SRP parameters expected by the STREAM_INPUT, the PAAD-AE will ignore the Talker attribute and return to the probing state in order to get up-to-date information from a subsequent PROBE_TX_RESPONSE message.

6.8.10. Diagnostic counters

For each STREAM_INPUT of the currently set CONFIGURATION, the PAAD-AE shall keep track of the following counters (each counter is a 32-bit unsigned integer that wraps over to zero when it reaches the maximum value):

Symbol	**Meaning**
MEDIA_LOCKED	Incremented each time the input stream gets synchronized on the media clock (whatever where the media clock comes from). *Note: the definition of "synchronized" is left open to each manufacturer.*

MEDIA_UNLOCKED	Incremented each time the input stream gets unsynchronized on the media clock (whatever where the media clock comes from). *Note: the definition of "unsynchronized" is left open to each manufacturer*. At any time, the PAAD-AE shall ensure that either MEDIA_LOCKED=MEDIA_UNLOCKED (in this case, the input stream is not synchronized on the media clock), or MEDIA_LOCKED=MEDIA_UNLOCKED+1 (in this case, the input stream is synchronized on the media clock).
STREAM_INTERRUPTED	Incremented each time the stream playback is interrupted for any reason other than a Controller Unbind operation. *Note: depending on the implementation, this can include the loss of several AVTPDUs, wrong timestamps, buffer overrun/underrun, …*
SEQ_NUM_MISMATCH	Incremented at the end of every observation interval during which a Stream Data AVTPDU has been received with a non-sequential sequence_num field. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
MEDIA_RESET	Incremented at the end of every observation interval during which the "mr" bit was toggled in any of the received Stream Data AVTPDUs. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
TIMESTAMP_UNCERTAIN	Incremented at the end of every observation interval during which the "tu" bit was set in any of the received Stream Data AVTPDUs. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
UNSUPPORTED_FORMAT	Incremented at the end of every observation interval during which a Stream Data AVTPDU has been received with a format that did not match the current format of the STREAM_INPUT.

	The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
LATE_TIMESTAMP	Incremented at the end of every observation interval during which a Stream Data AVTPDU has been received with an avtp_timestamp field that was in the past. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
EARLY_TIMESTAMP	Incremented at the end of every observation interval during which a Stream Data AVTPDU has been received with an avtp_timestamp field that was too far in the future to process. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.
FRAMES_RX	Incremented at the end of every observation interval during which at least one Stream Data AVTPDU has been received on this STREAM_INPUT. The duration of the observation interval is implementation-specific and shall be less than or equal to 1 second.

The PAAD-AE shall reset all of these counters to zero each time the STREAM_INPUT changes its state from not bound to bound.

Note: the PAAD-AE does not reset these counters when the STREAM_INPUT changes its state from bound to not bound.

6.9. STREAM_PORT_OUTPUT dynamic state

6.9.1. Channels mapping

The ultimate goal of a Talker PAAD is to transmit one or several audio channels into output AVB streams. As per [AVNU.IO.FORMATS], an audio stream transmitted by a PAAD may contain one or several channels.

At a given time, each channel of each STREAM_OUTPUT (in the current format) is either not mapped or mapped to a channel of an AUDIO_CLUSTER of a STREAM_PORT_OUTPUT. When the STREAM_PORT_OUTPUT supports dynamic mappings (i.e. when it has no AUDIO_MAP), mappings to a STREAM_OUTPUT can be added or removed dynamically, but only when the STREAM_OUTPUT is not streaming.

The PAAD-AE shall maintain a list of all its output channel mappings. This list shall be saved in a non-volatile memory and restored after a power cycle.

6.10. STREAM_PORT_INPUT dynamic state

6.10.1. Channels mapping

The ultimate goal of a Listener PAAD is to receive one or several audio channels from input AVB streams. As per [AVNU.IO.FORMATS], an audio stream received by a PAAD may contain one or several channels.

At a given time, each channel of each AUDIO_CLUSTER of each STREAM_PORT_INPUT is either not mapped, or mapped to a channel of a STREAM_INPUT (in this case, the index of the mapped STREAM_INPUT's channel shall be lower than the number of channels in the current format of the STREAM_INPUT). In an R-PAAD-AE, each channel of each AUDIO_CLUSTER of each STREAM_PORT_INPUT, when mapped, will be mapped simultaneously to all the equivalent channels of the redundant STREAM_INPUTs.

The PAAD-AE shall maintain a list of all its input channel mappings. This list shall be saved in a non-volatile memory and restored after a power cycle.

A PAAD-AE shall support changing mappings from a STREAM_INPUT at any time (even when it is bound).

6.11. CLOCK_DOMAIN dynamic state

6.11.1. Clock source

The clock source determines the source of the clock used in a clock domain. This specification uses two categories of clock sources: internal or input stream. When there are several possible clock sources for a given audio unit/clock domain, the user is expected to correctly set the clock source of each audio unit of each PAAD-AE in order to exchange audio data with the correct synchronization.

At any given time, a CLOCK_DOMAIN must be using one of the associated CLOCK_SOURCE descriptors. The PAAD-AE is able to dynamically change the clock source to any of the CLOCK_SOURCE descriptors associated with the CLOCK_DOMAIN.

The current clock source shall be saved in a non-volatile memory and restored after a power cycle.

6.11.2. Diagnostic counters

Symbol	**Meaning**
LOCKED	Incremented each time the media clock used in the clock domain gets locked. *Note: the definition of "locked" is left open to each manufacturer.*

UNLOCKED	Incremented each time the media clock used in the clock domain gets unlocked. *Note: the definition of "unlocked" is left open to each manufacturer*. At any time, the PAAD-AE shall ensure that either LOCKED=UNLOCKED (in this case, the media clock is currently locked), or LOCKED=UNLOCKED+1 (in this case, the media clock is currently unlocked).

6.12. "IDENTIFY" CONTROL dynamic state

The "IDENTIFY" CONTROL is used for "Controller to Entity" identification. Please refer to section 7.5.4 Identification notification for the reverse mechanism ("Entity to Controller" identification).

The "IDENTIFY" CONTROL has value 0 when the entity is not in identification mode (default mode after reset), and value 255 when the entity is in identification mode.

This value can be changed by a Controller using the SET_CONTROL command, or by a vendor-specific action on the PAAD.

When value is 255, the PAAD is expected to show, by some visual/audible effect (for example blinking some LEDs), that it is in identification mode. The PAAD remains in identification mode until the value of the "IDENTIFY" CONTROL is set back to 0.

6.13. User names

The PAAD-AE shall maintain the following list of names:

- entity_name of the ENTITY descriptor
- group_name of the ENTITY descriptor
- object_name of each CONFIGURATION descriptor
- object_name of each AUDIO_UNIT descriptor
- object_name of each STREAM_INPUT descriptor
- object_name of each AVB_INTERFACE
- object_name of each CLOCK_SOURCE
- object_name of each AUDIO_CLUSTER descriptor
- object_name of each CLOCK_DOMAIN
- object_name of each "IDENTIFY" CONTROL

All the names in the list above shall have a default value which is vendor-specific and shall be modifiable by the user. The PAAD-AE shall save them in a non-volatile memory and restore them after a power cycle.

7. Control

7.1. General

A PAAD-AE shall implement the AECP common format as described in [AVDECC, Clause 9.2.1], with the following change:

- A PAAD-AE is allowed to transmit AECP response packets with no limitation of the control_data_length field.

Note: this is voluntarily in contradiction to [AVDECC, Clause 9.2.1.1.7]. This specification allows a PAAD-AE to transmit AECP response packets as big as the maximum-sized Ethernet frame on the underlying network. This brings more flexibility to the PAAD-AE when sending the variable-length AECP AEM response messages. These messages are READ_DESCRIPTOR, GET_AVB_INFO, GET_AS_PATH, GET_AUDIO_MAP, ADD_AUDIO_MAPPINGS and REMOVE_AUDIO_MAPPINGS.

A PAAD-AE shall implement AECP AEM as described in [AVDECC, Clause 9.2.1.2] and also the AECP vendor unique extension defined in the following section 7.2 AECP Milan Vendor Unique format.

The list of AECP AEM commands which are relevant to this specification is defined in section 7.3 AECP AEM commands and responses. The list of AECP MVU commands which are relevant to this specification is defined in section 7.4 AECP MVU commands and responses. Section 7.5 describes the unsolicited notifications.

7.2. AECP Milan Vendor Unique format

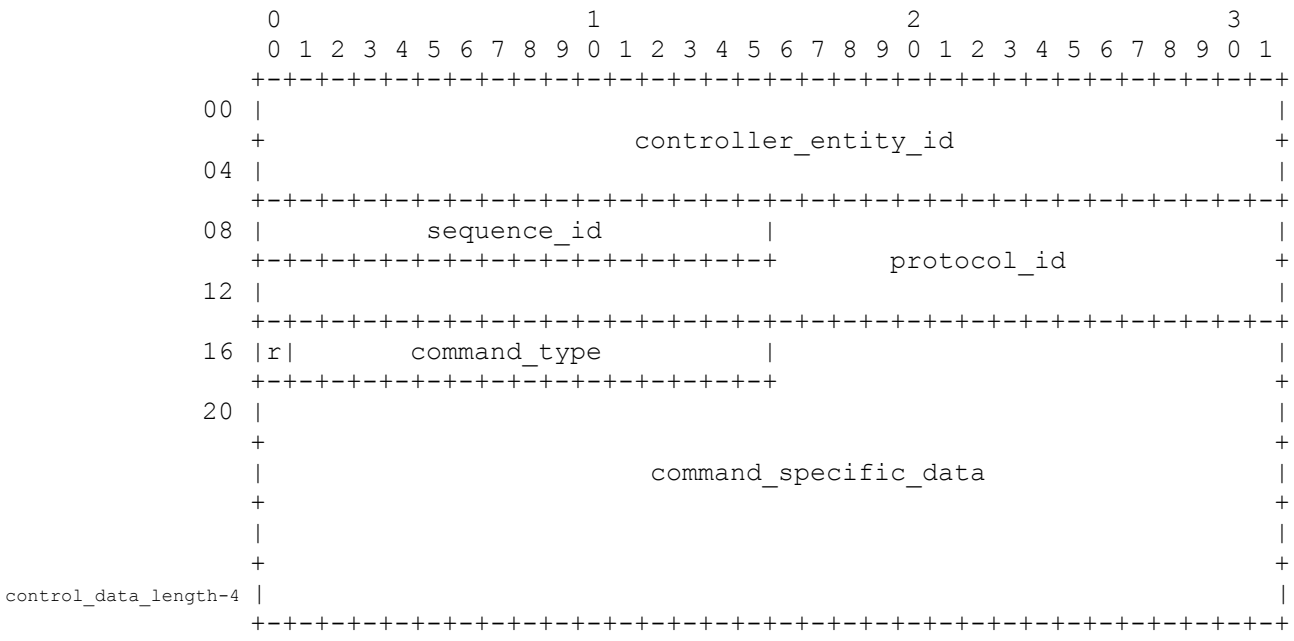
7.2.1. Introduction

This specification uses the AECP extensibility feature described in [AVDECC, Clause 9.2.1.5]. It defines a vendor unique AECP format called MVU (Milan Vendor Unique) that allows an AVDECC Entity to expose Milan-specific information. Following sections describe the format of an MVU message, the possible status codes and the command timeouts.

7.2.2. MVU payload format

MVU adds the following fields to the AECPDU following the sequence_id field:

- protocol_id: 48 bits
- r: 1 bit
- command_type: 15 bits
- command_specific_data: command specific length



If the command or response frame is shorter than the minimum transmission unit, then the frame is padded to the minimum length. These padding octets are not included in the control_data_length field value.

7.2.2.1. protocol_id field

The protocol_id field identifies the AECP vendor unique protocol. It is composed of the OUI-36 owned by Avnu (00-1B-C5-0A-C) appended with a 12-bit protocol unique identifier (0x100 for MVU). The result is 00-1B-C5-0A-C1-00.

7.2.2.2. r field

This is a reserved field and shall be set to zero (0) in all AECP MVU command and response messages.

7.2.2.3. command_type field

The command_type field is set to one of the values from the following table:

Value	Name	Description	Clause
0x0000	GET_MILAN_INFO	Get Milan-specific information of a given CONFIGURATION.	7.4.1
0x0001 to 0x7fff	-	Reserved for future use	-

7.2.2.4. command_specific_data field

The command_specific_data field is defined by the message_type and command_type. The format of this field is defined per command in 7.4 AECP MVU commands and responses.

7.2.3. MVU Status codes

The status field is set to the appropriate value from the following table. In a command, the status field is set to SUCCESS.

Value	Status code	Meaning
0	SUCCESS	The PAAD-AE successfully performed the command and has valid results.
1	NOT_IMPLEMENTED	The PAAD-AE does not support the command type.
2 to 31	-	Reserved for future use.

7.2.4. MVU Command timeouts

The AECP MVU protocol follows the same principle as AEM for command timeouts.

All MVU messages shall have a 250 ms timeout. If a response is not received within 250 milliseconds, then the transaction is considered to have timed out.

Entities shall respond to all MVU commands within 240 ms.

7.3. AECP AEM commands and responses

7.3.1. ACQUIRE_ENTITY

The PAAD-AE shall not implement the ACQUIRE_ENTITY command (NOT_IMPLEMENTED shall be returned).

Note: this is voluntarily in contradiction to [AVDECC, Clause 7.4]. This specification considers that the only valid reason to prevent a PAAD-AE from accepting set requests from other controllers is to perform on it a short sequence of atomic operations. This feature is fulfilled by the LOCK_ENTITY command.

7.3.2. LOCK_ENTITY

The PAAD-AE shall implement the LOCK_ENTITY command as specified in [AVDECC, Clause 7.4.2].

The PAAD-AE shall support the UNLOCK flag.

The PAAD-AE shall not allow locking another descriptor than the ENTITY descriptor (NOT_SUPPORTED shall be returned in this case).

Note: the UNLOCK flag set in a LOCK_ENTITY command can be used by a controller to simply query the current locked state of an entity without requesting to lock it.

Note: in case of automatic unlock by the PAAD-AE itself after the 1-minute timeout period, an unsolicited notification is sent to all registered controllers, and only to these controllers. If the locking controller had not registered, it will not be informed of the automatic unlock action taken by the PAAD-AE.

7.3.3. ENTITY_AVAILABLE

The PAAD-AE shall implement the ENTITY_AVAILABLE command as specified in [AVDECC, Clause 7.4.3].

7.3.4. READ_DESCRIPTOR

The PAAD-AE shall implement the READ_DESCRIPTOR command as specified in [AVDECC, Clause 7.4.5].

7.3.5. SET_CONFIGURATION

The PAAD-AE shall implement the SET_CONFIGURATION command as specified in [AVDECC, Clause 7.4.7].

The PAAD-AE shall not accept a SET_CONFIGURATION command if one of the STREAM_INPUTs is bound or one of the STREAM_OUTPUTs is streaming. In this case, the STREAM_IS_RUNNING error code shall be returned.

If the PAAD-AE is locked by a controller, it shall not accept a SET_CONFIGURATION command from a different controller, and it shall also not change its current configuration by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.6. GET_CONFIGURATION

The PAAD-AE shall implement the GET_CONFIGURATION command as specified in [AVDECC, Clause 7.4.8].

7.3.7. SET_STREAM_FORMAT

For each STREAM_INPUT/OUTPUT descriptor, the PAAD-AE shall implement the SET_STREAM_FORMAT command as specified in [AVDECC, Clause 7.4.9].

The PAAD-AE shall not accept a SET_STREAM_FORMAT command on a STREAM_INPUT if the STREAM_INPUT is bound, nor on a STREAM_OUTPUT if the STREAM_OUTPUT is streaming. In these cases, the STREAM_IS_RUNNING error code shall be returned.

In addition, before accepting a SET_STREAM_FORMAT command, the PAAD-AE shall check that all channels of the STREAM_INPUT/OUTPUT that are referenced by existing mappings (static or dynamic) still exist in the new format. If not, the PAAD-AE shall refuse the command with the BAD_ARGUMENTS error code.

For an R-PAAD-AE, the requirements of [AVNU.IO.REDUNDANCY, Clause 6.2.3.3] also apply.

If the PAAD-AE is locked by a controller, it shall not accept a SET_STREAM_FORMAT command from a different controller, and it shall also not change any stream format by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.8. GET_STREAM_FORMAT

For each STREAM_INPUT/OUTPUT descriptor, the PAAD-AE shall implement the GET_STREAM_FORMAT command as specified in [AVDECC, Clause 7.4.10].

7.3.9. SET_STREAM_INFO

The PAAD-AE shall not implement the SET_STREAM_INFO command on STREAM_INPUT descriptors (NOT_SUPPORTED shall be returned).

Note: the reason for that is that the Connection management protocol (described in section 8) is used to dynamically set the stream parameters of the listeners from the PROBE_TX_RESPONSE message.

The PAAD-AE shall implement the SET_STREAM_INFO command on STREAM_OUTPUT descriptors as specified in [AVDECC, Clause 7.4.15], with the following changes/clarifications.

If the STREAM_OUTPUT is streaming then the PAAD-AE shall refuse the command with the STREAM_IS_RUNNING error code.

The XXX_VALID flags in the command indicate which parameters the PAAD-AE shall process from the command.

A PAAD-AE shall support the MSRP_ACC_LAT_VALID sub-command and set the Presentation Time offset to the value of the msrp_accumulated_latency field. The PAAD-AE shall support any value of the Presentation time offset in the range between 0x0 and 0x7FFFFFFF nanoseconds. If the value is outside this range, the PAAD-AE shall return BAD_ARGUMENTS.

A PAAD-AE may support other sub-commands to set other STREAM_OUTPUT parameters but the expected behavior is not defined by this specification.

If the PAAD-AE receives a SET_STREAM_INFO command with any unsupported sub-commands, the PAAD-AE shall refuse the entire command by returning the NOT_SUPPORTED error code.

In a successful response, the PAAD-AE shall set the MSRP_ACC_LAT_VALID flag to the same value as in the command, and when the flag is set, the returned msrp_accumulated_latency is the same value as in the command.

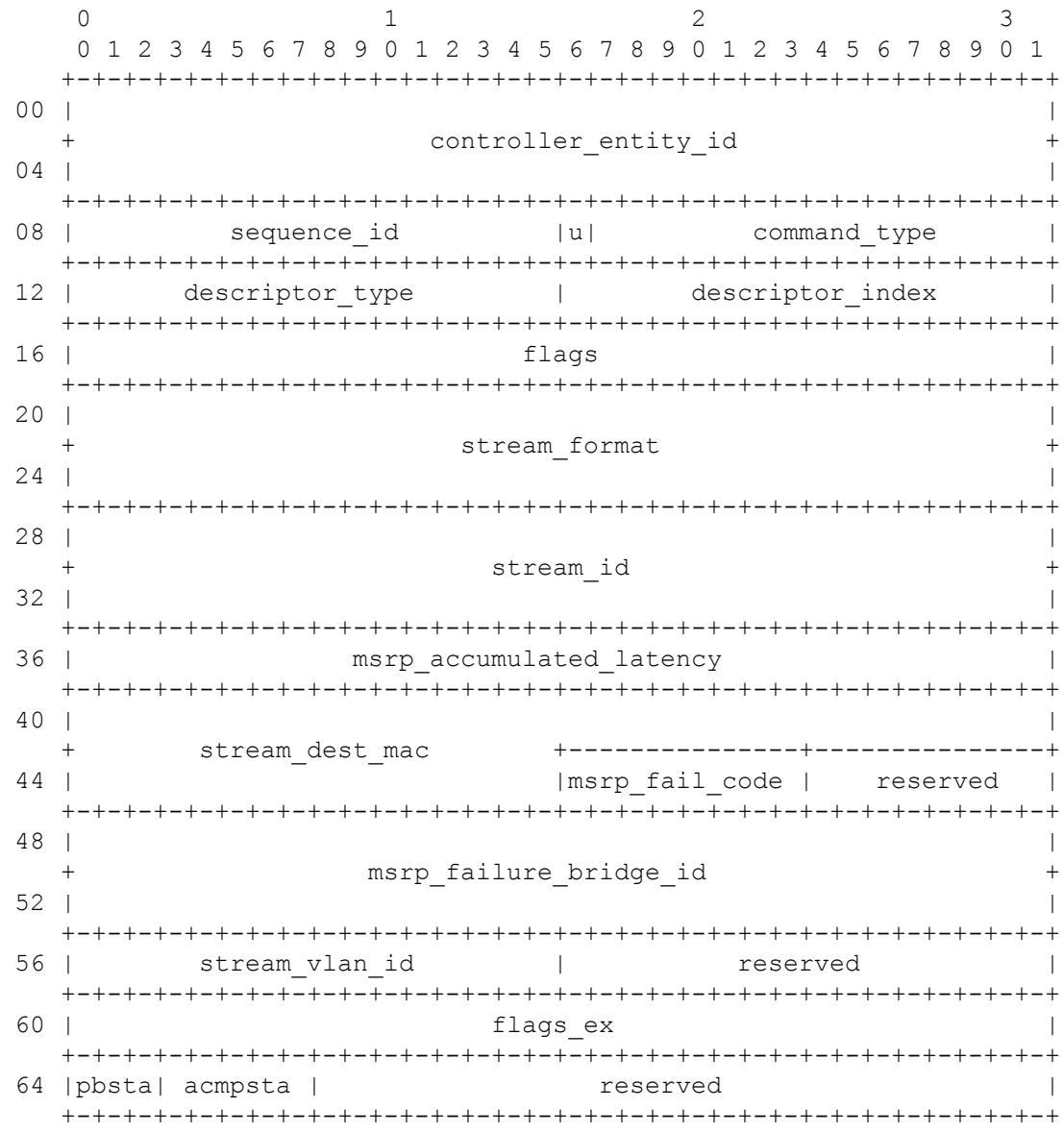
For an R-PAAD-AE, the requirements of [AVNU.IO.REDUNDANCY, Clause 6.2.3.4] also apply.

If the PAAD-AE is locked by a controller, it shall not accept a SET_STREAM_INFO command from a different controller, and it shall also not change the Presentation time offset of its STREAM_OUTPUTs by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.10. GET_STREAM_INFO

For each STREAM_INPUT/OUTPUT descriptor, the PAAD-AE shall implement the GET_STREAM_INFO command as specified in [AVDECC, Clause 7.4.16], with the following changes/clarifications:

- The CONNECTED flag is renamed to BOUND.
- The TALKER_FAILED flag is renamed to REGISTERING_FAILED.
- The GET_STREAM_INFO response message is extended by 8 bytes as shown in the diagram below.



The flags_ex field is set to a combination of values as appropriate from the following table:

Bit #	Field value	Name	Meaning
31	0x00000001	REGISTERING	For a STREAM_INPUT: indicates that the STREAM_INPUT is registering either a matching Talker Advertise or a matching Talker Failed attribute. For a STREAM_OPUTPUT: indicates that the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute and registering a matching Listener attribute (Listener Ready, Listener Ready Failed or Listener Asking Failed).
0 to 30	-	-	Reserved for future use.

Note: a 1722.1 controller which is not aware of this specification will ignore the new fields of the GET_STREAM_INFO response message. A 1722.1 controller which is aware of this specification will use the control_data_length field of the AECPDU containing the GET_STREAM_INFO response message to determine if the new fields are present.

7.3.10.1. Requirements for a STREAM_INPUT

For a STREAM_INPUT, the PAAD-AE shall set the flags as follows:

Bit #	Field Value	Name	Value
31	0x00000001	CLASS_B	Not defined by this specification. Should be set to 0.
30	0x00000002	FAST_CONNECT	This specification requires that this field be set to 1 if the STREAM_INPUT is bound, 0 otherwise.
29	0x00000004	SAVED_STATE	This specification recommends that this field be set to 1 if the STREAM_INPUT is bound, 0 otherwise.

28	0x00000008	STREAMING_WAIT	0 if the STREAM_INPUT is bound and started, 1 if the STREAM_INPUT is bound and stopped, undefined if the STREAM_INPUT is not bound.
27	0x00000010	SUPPORTS_ENCRYPTED	Not defined by this specification. Should be set to 0.
26	0x00000020	ENCRYPTED_PDU	Not defined by this specification. Should be set to 0.
25	0x00000040	REGISTERING_FAILED	1 if the STREAM_INPUT is registering a matching Talker Failed attribute.
7 to 24	-	-	Reserved for future use.
6	0x02000000	STREAM_VLAN_ID_VALID	Indicates whether the stream_vlan_id field contains valid information. The stream_vlan_id field shall contain valid information if, and only if, the STREAM_INPUT is settled.
5	0x04000000	BOUND	1 if the STREAM_INPUT is bound, 0 otherwise.
4	0x08000000	MSRP_FAILURE_VALID	Indicates whether the msrp_failure_bridge_id and msrp_failure_code fields contain valid information. The msrp_failure_bridge_id and msrp_failure_code fields shall contain valid information when the STREAM_INPUT is registering a matching Talker Failed attribute. *Note: this bit has the same value as the REGISTERING_FAILED bit.*

3	0x10000000	STREAM_DEST_MAC_VALID	Indicates whether the stream_dest_mac field contains valid information. The stream_dest_mac field shall contain valid information if, and only if, the STREAM_INPUT is settled.
2	0x20000000	MSRP_ACC_LAT_VALID	Indicates whether the msrp_accumulated_latency field contains valid information. The msrp_accumulated_latency field shall contain valid information if, and only if, the STREAM_INPUT is registering either a matching Talker Advertise or a matching Talker Failed attribute. *Note: this bit has the same value as the REGISTERING bit of the flags_ex field.*
1	0x40000000	STREAM_ID_VALID	Indicates whether the stream_id field contains valid information. The stream_id field shall contain valid information if, and only if, the STREAM_INPUT is settled.
0	0x80000000	STREAM_FORMAT_VALID	Indicates whether the stream_format field contains valid information. The stream_format field shall always contain valid information.

The PAAD-AE shall set the extended flags as follows in the flags_ex field:

Bit #	**Field Value**	**Name**	**Value**
31	0x00000001	REGISTERING	1 if the STREAM_INPUT is registering either a matching Talker Advertise or a matching Talker Failed attribute.

The stream_format field shall be set to the current format of the STREAM_INPUT.

The pbsta field shall be set to the Probing status of the STREAM_INPUT and the acmpsta field shall be set to the ACMP status of the STREAM_INPUT (see section 6.8.4).

If the STREAM_INPUT is settled then the stream_id, stream_dest_mac and stream_vlan_id fields shall be set to the corresponding values received in the PROBE_TX_RESPONSE from the talker. Otherwise, the values of these fields shall be set to zero.

If the STREAM_INPUT is registering either a matching Talker Advertise or a matching Talker Failed attribute then the msrp_accumulated_latency field shall be set to the accumulated_latency extracted from the registered Talker attribute plus any internal implementation-dependent max delay an AVTP Stream frame will encounter before reaching the MAC's ingress buffer (see [AVTP, Figure 6]). Otherwise, the field shall be set to 0.

If the STREAM_INPUT is registering a matching Talker Failed attribute then the msrp_failure_bridge_id and msrp_failure_code fields shall be set to the bridge_id and failure_code extracted from the registered Talker Failed attribute. Otherwise, these fields shall be set to 0.

7.3.10.2. Requirements for a STREAM_OUTPUT

For a STREAM_OUTPUT, the PAAD-AE shall set the flags as follows:

Bit #	Field Value	Name	Value
31	0x00000001	CLASS_B	Not defined by this specification. Should be set to 0.
30	0x00000002	FAST_CONNECT	This specification requires that this field be always set to 0.
29	0x00000004	SAVED_STATE	This specification requires that this field be always set to 0.
28	0x00000008	STREAMING_WAIT	This specification requires that this field be always set to 0.
27	0x00000010	SUPPORTS_ENCRYPTED	Not defined by this specification. Should be set to 0.

26	0x00000020	ENCRYPTED_PDU	Not defined by this specification. Should be set to 0.
25	0x00000040	REGISTERING_FAILED	1 if the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute and registering a matching Listener Asking Failed attribute.
7 to 24	-	-	Reserved for future use.
6	0x02000000	STREAM_VLAN_ID_VALID	Indicates whether the stream_vlan_id field contains valid information. If the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute then the stream_vlan_id field shall contain valid information. This specification doesn't define the validity of this field in other cases.
5	0x04000000	BOUND	This specification requires that this field be always set to 0.
4	0x08000000	MSRP_FAILURE_VALID	Indicates whether the msrp_failure_bridge_id and msrp_failure_code fields contain valid information. The msrp_failure_bridge_id and msrp_failure_code fields shall contain valid information when the STREAM_OUTPUT is declaring a Talker Failed attribute.
3	0x10000000	STREAM_DEST_MAC_VALID	Indicates whether the stream_dest_mac field contains valid information. If the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute then the stream_dest_mac field shall contain valid

			information. This specification doesn't define the validity of this field in other cases.
2	0x20000000	MSRP_ACC_LAT_VALID	Indicates whether the msrp_accumulated_latency field contains valid information. The msrp_accumulated_latency field shall always contain valid information.
1	0x40000000	STREAM_ID_VALID	Indicates whether the stream_id field contains valid information. If the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute then the stream_id field shall contain valid information. This specification doesn't define the validity of this field in other cases.
0	0x80000000	STREAM_FORMAT_VALID	Indicates whether the stream_format field contains valid information. The stream_format field shall always contain valid information.

The PAAD-AE shall set the extended flags as follows in the flags_ex field:

Bit #	**Field Value**	**Name**	**Value**
31	0x00000001	REGISTERING	1 if the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute and registering a matching Listener attribute (Listener

			Ready, Listener Ready Failed or Listener Asking Failed).

The stream_format field shall be set to the current format of the STREAM_OUTPUT.

The pbsta and acmpsta fields shall be set to 0.

If the STREAM_OUTPUT is declaring a Talker Advertise or a Talker Failed attribute then the stream_id, stream_dest_mac and stream_vlan_id fields shall be set to the corresponding values of the declared Talker attribute. Otherwise, the values of these field are not defined by this specification.

The msrp_accumulated_latency field shall be set to the presentation time offset currently used by the talker for this STREAM_OUTPUT.

If the STREAM_OUTPUT is declaring a Talker Failed attribute then the msrp_failure_bridge_id and msrp_failure_code fields shall contain the declared failure_bridge_id and failure_code. Otherwise, they shall be set to 0.

Note: The combination {REGISTERING=1, REGISTERING_FAILED=0, MSRP_FAILURE_VALID=0} means that the STREAM_OUTPUT is currently streaming.

7.3.11. SET_NAME

The PAAD-AE shall implement the SET_NAME command as specified in [AVDECC, Clause 7.4.17].

The PAAD-AE shall support setting all the names contained in the descriptors it implements.

If the PAAD-AE is locked by a controller, it shall not accept a SET_NAME command from a different controller, and it shall also not change any user name by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.12. GET_NAME

The PAAD-AE shall implement the GET_NAME command as specified in [AVDECC, Clause 7.4.18].

The PAAD-AE shall support getting all the names contained in the descriptors it implements.

7.3.13. SET_SAMPLING_RATE

For each AUDIO_UNIT descriptor of the currently set CONFIGURATION, the PAAD-AE shall implement the SET_SAMPLING_RATE command as specified in [AVDECC, Clause 7.4.21].

If there are existing mappings to/from STREAM_OUTPUT/STREAM_INPUT which sampling rate is different from the target sampling rate of the command, and at least one of these mappings references a channel of a STREAM_PORT_INPUT/OUTPUT that has none of the SSRC/ASRC bits set, then the PAAD-AE may refuse the command and return the UNSUPPORTED error code.

If the PAAD-AE is locked by a controller, it shall not accept a SET_SAMPLING_RATE command from a different controller, and it shall also not change the sampling rate of any AUDIO_UNIT by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.14. GET_SAMPLING_RATE

For each AUDIO_UNIT descriptor, the PAAD-AE shall implement the GET_SAMPLING_RATE command as specified in [AVDECC, Clause 7.4.22].

7.3.15. SET_CLOCK_SOURCE

For each CLOCK_DOMAIN descriptor, the PAAD-AE shall implement the SET_CLOCK_SOURCE command as specified in [AVDECC, Clause 7.4.23].

For an R-PAAD-AE, the requirements of [AVNU.IO.REDUNDANCY, Clause 6.2.3.5] also apply.

If the PAAD-AE is locked by a controller, it shall not accept a SET_CLOCK_SOURCE command from a different controller, and it shall also not change the clock source of any CLOCK_DOMAIN by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.16. GET_CLOCK_SOURCE

For each CLOCK_DOMAIN descriptor, the PAAD-AE shall implement the GET_CLOCK_SOURCE command as specified in [AVDECC, Clause 7.4.24].

7.3.17. SET_CONTROL

For the “IDENTIFY” CONTROL descriptor, the PAAD-AE shall implement the SET_CONTROL command as specified in [AVDECC, Clause 7.4.25] and [AVDECC, Clause 7.3.4.2].

If the PAAD-AE is locked by a controller, it shall not accept a SET_CONTROL command on the “IDENTIFY” CONTROL from a different controller, and it shall also not change the identification state of the entity by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.18. GET_CONTROL

For the “IDENTIFY” CONTROL descriptor, the PAAD-AE shall implement the GET_CONTROL command as specified in [AVDECC, Clause 7.4.26].

7.3.19. START_STREAMING

For each STREAM_INPUT descriptor, the PAAD-AE shall implement the START_STREAMING command as specified in [AVDECC, Clause 7.4.35]. The PAAD-AE shall not support the START_STREAMING command for a STREAM_OUTPUT descriptor (NOT_SUPPORTED shall be returned).

A PAAD-AE receiving START_STREAMING on a bound and stopped STREAM_INPUT shall change the state of the STREAM_INPUT to started.

Note: this command has no effect on a STREAM_INPUT that is not already bound or already started.

If the PAAD-AE is locked by a controller, it shall not accept a START_STREAMING command from a different controller, and it shall also not change the started state of any bound STREAM_INPUT by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.20. STOP_STREAMING

For each STREAM_INPUT descriptor, the PAAD-AE shall implement the STOP_STREAMING command as specified in [AVDECC, Clause 7.4.36]. The PAAD-AE shall not support the STOP_STREAMING command for a STREAM_OUTPUT descriptor (NOT_SUPPORTED shall be returned).

A PAAD-AE receiving STOP_STREAMING on a bound and started STREAM_INPUT shall change the state of the STREAM_INPUT to stopped.

Note: this command has no effect on a STREAM_INPUT that is not already bound or stopped.

If the PAAD-AE is locked by a controller, it shall not accept a STOP_STREAMING command from a different controller, and it shall also not change the started state of any bound STREAM_INPUT by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.21. REGISTER_UNSOLICITED_NOTIFICATION

The PAAD-AE shall implement the REGISTER_UNSOLICITED_NOTIFICATION command as specified in [AVDECC, Clause 7.4.37].

If the PAAD-AE has not enough resources to create a new entry in the list of registered controllers, it may send CONTROLLER_AVAILABLE commands to its registered controllers to check if they can be deregistered. *Note: as per [AVDECC, Clause 9.2.2.3.2], a retry is always done before declaring a command has timed out.* If a controller doesn't reply, the PAAD-AE may deregister it and send an unsolicited DEREGISTER_UNSOLICITED_NOTIFICATION to this controller only. If one or more controllers is deregistered this way, the PAAD-AE can accept the new registration.

After the previous step, if the PAAD-AE still has not enough resources to create a new entry in the list of registered controllers, it shall return the NO_RESOURCES error code. The entity shall not automatically deregister another controller that is responding to the CONTROLLER_AVAILABLE command.

If a new entry is created, the Sequence ID associated with this entry is initialized to zero.

7.3.22. DEREGISTER_UNSOLICITED_NOTIFICATION

The PAAD-AE shall implement the DEREGISTER_UNSOLICITED_NOTIFICATION command as specified in [AVDECC, Clause 7.4.38].

7.3.23. GET_AVB_INFO

The PAAD-AE shall implement the GET_AVB_INFO command as specified in [AVDECC, Clause 7.4.40].

7.3.24. GET_AS_PATH

The PAAD-AE shall implement the GET_AS_PATH command as specified in [AVDECC, Clause 7.4.41].

7.3.25. GET_COUNTERS

The PAAD-AE shall implement the GET_COUNTERS command as specified in [AVDECC, Clause 7.4.42]. The PAAD-AE shall support this command for each AVB_INTERFACE, CLOCK_DOMAIN, STREAM_INPUT and STREAM_OUTPUT descriptor of the currently set CONFIGURATION.

For each AVB_INTERFACE descriptor, the PAAD-AE shall implement and return the following counters:

Bit #	Bit value	Symbol
31	0x00000001	LINK_UP
30	0x00000002	LINK_DOWN
26	0x00000020	GPTP_GM_CHANGED

For each AVB_INTERFACE descriptor, the PAAD-AE may implement and return the following counters:

Bit #	Bit value	Symbol
29	0x00000004	FRAMES_TX
28	0x00000008	FRAMES_RX
27	0x00000010	RX_CRC_ERROR

For each CLOCK_DOMAIN descriptor, the PAAD-AE shall implement and return the following counters:

Bit #	Bit value	Symbol
31	0x00000001	LOCKED
30	0x00000002	UNLOCKED

For each STREAM_INPUT descriptor, the PAAD-AE shall implement and return the following counters:

Bit #	Bit value	Symbol
31	0x00000001	MEDIA_LOCKED
30	0x00000002	MEDIA_UNLOCKED
29	0x00000004	STREAM_INTERRUPTED
28	0x00000008	SEQ_NUM_MISMATCH
27	0x00000010	MEDIA_RESET
26	0x00000020	TIMESTAMP_UNCERTAIN
23	0x00000100	UNSUPPORTED_FORMAT
22	0x00000200	LATE_TIMESTAMP
21	0x00000400	EARLY_TIMESTAMP
20	0x00000800	FRAMES_RX

For each STREAM_OUTPUT descriptor, the PAAD-AE shall implement and return the following counters:

Bit #	Bit value	Symbol
31	0x00000001	STREAM_START
30	0x00000002	STREAM_STOP

29	0x00000004	MEDIA_RESET
28	0x00000008	TIMESTAMP_UNCERTAIN
27	0x00000010	FRAMES_TX

7.3.26. GET_AUDIO_MAP

For each STREAM_PORT_INPUT descriptor and for each STREAM_PORT_OUTPUT descriptor that has no AUDIO_MAP descriptor, the PAAD-AE shall implement the GET_AUDIO_MAP command as specified in [AVDECC, Clause 7.4.44]. If a PAAD-AE receives a GET_AUDIO_MAP command for a STREAM_PORT_OUTPUT that has AUDIO_MAP descriptor(s), the PAAD-AE shall reply with the NOT_SUPPORTED error code.

This specification requires a PAAD-AE to follow the rules below for each STREAM_PORT_INPUT:

- The PAAD-AE shall partition the set of AUDIO_CLUSTERs' channels into disjoint subsets which size does not exceed 176 (88 for redundant devices). This partitioning shall be fixed for a given CONFIGURATION. Let's call N the number of subsets in the current CONFIGURATION.
- The PAAD-AE shall always return N in the number_of_maps field of the GET_AUDIO_MAP response, no matter the actual count of dynamic mappings.
- In response to a GET_AUDIO_MAP command with map_index=P (P<N), the PAAD-AE shall return all the dynamic mappings referencing the AUDIO_CLUSTERs' channels which are in subset P, and only these ones. *Note: As there is at most one (two in case of a redundant device) dynamic mapping per AUDIO_CLUSTER's channel, the PAAD-AE will return no more than 176 mappings, which fits in a GET_AUDIO_MAP response message. The PAAD-AE will return 0 mapping in the GET_AUDIO_MAP response if there is no dynamic mapping referencing the AUDIO_CLUSTERs' channels which are in subset P.*

This specification requires a PAAD-AE to follow the rules below for each STREAM_PORT_OUTPUT:

- The PAAD-AE shall partition the set of STREAM_OUTPUTs' channels into disjoint subsets which size does not exceed 176. Redundant STREAM_OUTPUTs' channels shall be in the same subset. This partitioning shall be fixed for a given CONFIGURATION. Let's call N the number of subsets in the current CONFIGURATION.
- The PAAD-AE shall always return N in the number_of_maps field of the GET_AUDIO_MAP response, no matter the actual count of dynamic mappings.
- In response to a GET_AUDIO_MAP command with map_index=P (P<N), the PAAD-AE shall return all the dynamic mappings referencing the STREAM_OUTPUTs' channels which are in subset P, and only these ones. *Note: As there is at most one dynamic mapping per STREAM_OUTPUT's channel, the PAAD-AE will return no more than 176 mappings, which fits in a GET_AUDIO_MAP response message. The PAAD-AE will return 0 mapping in the GET_AUDIO_MAP response if there is no dynamic mapping referencing the STREAM_OUTPUTs' channels which are in subset P.*

Note*: the purpose of the requirements above is to allow a simple controller to enumerate the dynamic mappings of a PAAD-AE with more than 176 dynamic mappings, without the need to lock it during the enumeration.*

7.3.27. ADD_AUDIO_MAPPINGS

For each STREAM_PORT_INPUT descriptor and for each STREAM_PORT_OUTPUT descriptor that has no AUDIO_MAP descriptor, the PAAD-AE shall implement the ADD_AUDIO_MAPPINGS command as specified in [AVDECC, Clause 7.4.45]. If a PAAD-AE receives an ADD_AUDIO_MAPPINGS command for a STREAM_PORT_OUTPUT that has AUDIO_MAP descriptor(s), the PAAD-AE shall reply with the NOT_SUPPORTED error code.

A PAAD-AE shall return the BAD_ARGUMENTS error code if one or more of the mappings in the command was invalid. In this case, no mapping shall be added.

A PAAD-AE shall treat as invalid a mapping that references a channel of a STREAM_INPUT/OUTPUT that doesn't exist in the currently set format for this STREAM_INPUT/OUTPUT.

A PAAD-AE shall treat as invalid a mapping that references a channel of a currently streaming STREAM_OUTPUT.

A PAAD-AE should treat as invalid a mapping that references a channel of a STREAM_INPUT/OUTPUT which sampling rate is different from the sampling rate of the AUDIO_UNIT, if none of the SSRC and ASRC bits are set in the STREAM_PORT_INPUT/OUTPUT descriptor.

A PAAD-AE shall return the BAD_ARGUMENTS error code to an ADD_AUDIO_MAPPINGS command on a STREAM_PORT_INPUT if the command contains two different mappings that reference the same cluster's channel and two different stream's channels that are not redundant. In this case, no mapping shall be added.

A PAAD-AE shall return the BAD_ARGUMENTS error code to an ADD_AUDIO_MAPPINGS command on a STREAM_PORT_OUTPUT if the command contains two different mappings that reference the same stream's channel and two different cluster's channels. In this case, no mapping shall be added.

For an R-PAAD-AE, the requirements of [AVNU.IO.REDUNDANCY, Clause 6.2.3.8] also apply.

If the PAAD-AE is locked by a controller, it shall not accept an ADD_AUDIO_MAPPINGS command from a different controller, and it shall also not add any mapping of any STREAM_PORT_INPUT/OUTPUT by non-AVDECC means (proprietary remote control software, front-panel, …).

7.3.28. REMOVE_AUDIO_MAPPINGS

For each STREAM_PORT_INPUT descriptor and for each STREAM_PORT_OUTPUT descriptor that has no AUDIO_MAP descriptor, the PAAD-AE shall implement the REMOVE_AUDIO_MAPPINGS command as specified in [AVDECC, Clause 7.4.46]. If a PAAD-AE receives a REMOVE_AUDIO_MAPPINGS command for a STREAM_PORT_OUTPUT that has AUDIO_MAP descriptor(s), the PAAD-AE shall reply with the NOT_SUPPORTED error code.

A PAAD-AE shall ignore duplicate mappings that may be present in a REMOVE_AUDIO_MAPPINGS command.

A PAAD-AE shall not allow removing a mapping that references a channel of a currently streaming STREAM_OUTPUT. In this case, the PAAD-AE shall return a BAD_ARGUMENTS error code and no mapping shall be removed.

For an R-PAAD-AE, the requirements of [AVNU.IO.REDUNDANCY, Clause 6.2.3.9] also apply.

If the PAAD-AE is locked by a controller, it shall not accept an REMOVE_AUDIO_MAPPINGS command from a different controller, and it shall also not remove any mapping of any STREAM_PORT_INPUT/OUTPUT by non-AVDECC means (proprietary remote control software, front-panel, …).

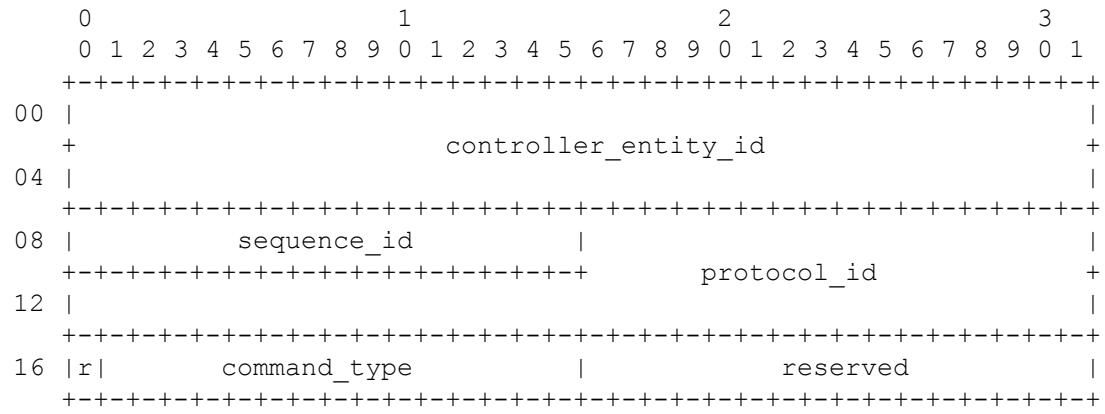
7.4. AECP MVU commands and responses

7.4.1. GET_MILAN_INFO

The PAAD-AE shall implement the GET_MILAN_INFO command as specified in this section.

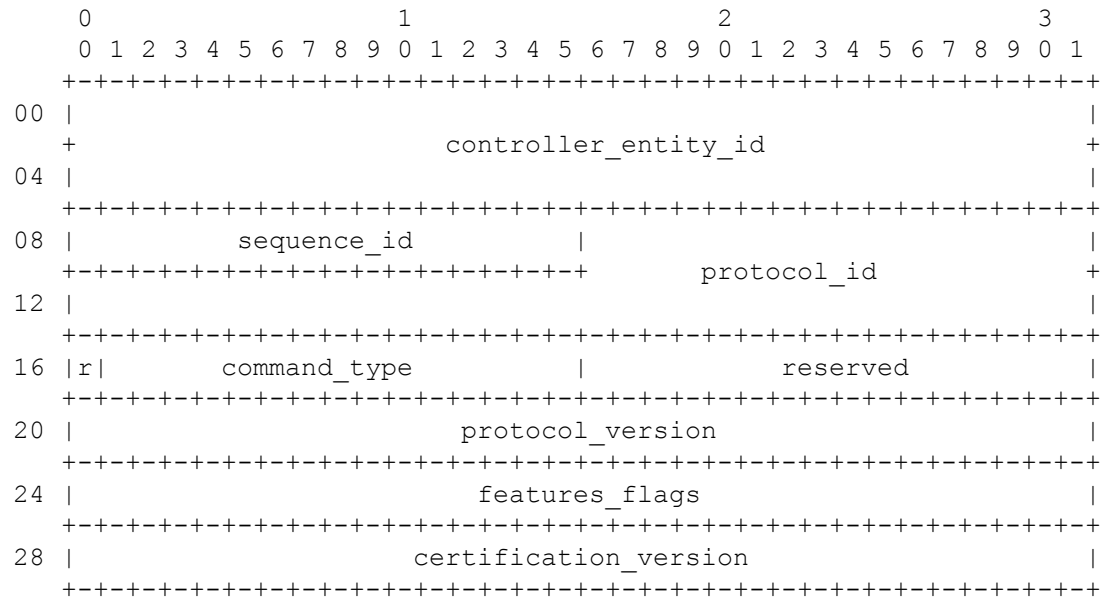
The GET_MILAN_INFO command is used to get Milan-specific information about the PAAD-AE.

The GET_MILAN_INFO command AECPDU format is shown below:



The reserved field shall be set to 0 by the sender and ignored by the receiver of the command.

The GET_MILAN_INFO response AECPDU format is shown below:



The reserved field shall be set to 0 by the sender and ignored by the receiver of the response.

The protocol_version field shall be set to the Milan protocol version supported by the PAAD-AE. Unless superseded by the [AVNU.IO.BASELINE] document, the value of this field is 1.

The features_flags field shall be set to a combination of values as appropriate from the following table:

Bit #	Field value	Name	Meaning
31	0x00000001	REDUNDANCY	The AVDECC Entity supports the Milan redundancy scheme as specified in [AVNU.IO.REDUNDANCY].
0 to 30	-	-	Reserved for future use.

The certification_version field shall be set to the version number of the Milan certification that the PAAD-AE has passed. It is composed of four 8-bit numbers which, when read from MSB to LSB and noted as dot-separated decimal numbers, are the human-readable representation of the version. This field is set to 0 if the PAAD-AE has not passed any Milan certification.

7.5. Notifications

7.5.1. General

A PAAD-AE shall implement unsolicited notifications as described in [AVDECC, Clause 7.5].

Below are some clarifications that shall apply to a PAAD-AE:

- When an unsolicited notification is transmitted, the PAAD-AE sends one unsolicited notification message to each registered controller.
- The transmitted unsolicited notification messages all contain the same information, but the following fields shall be populated with the unique parameters associated with each registered controller (see section 6.4.2): destination_mac_address, controller_entity_id and sequence id. The port mentioned in the entry shall be used to transmit an unsolicited notification message.
- After sending an unsolicited notification message, the PAAD-AE increments by 1 the Sequence ID variable associated with the entry. *Note: the PAAD-AE increments the variable, no matter whether the packet is actually sent on the wire or not. "Sending an unsolicited notification" here means "transmitting the packet to the network stack", that's all.*

7.5.2. List of unsolicited notifications

Each time a PAAD-AE sends a successful response to one of the following commands and the command modifies the state of the PAAD-AE, the PAAD-AE shall immediately send an unsolicited notification to all registered controllers:

- SET_CONFIGURATION
- SET_STREAM_FORMAT
- SET_STREAM_INFO
- SET_NAME
- SET_SAMPLING_RATE
- SET_CLOCK_SOURCE
- SET_CONTROL
- START_STREAMING
- STOP_STREAMING
- ADD_AUDIO_MAPPINGS
- REMOVE_AUDIO_MAPPINGS
- LOCK_ENTITY

In addition, if the PAAD-AE is not locked and it changes its state by non-AVDECC means (proprietary remote control software, front-panel,…) but the result of the change is the same as if a controller had sent one of the commands above, then the PAAD-AE shall send unsolicited notification(s) to the registered controllers.

If, as a result of a successful SET_STREAM_FORMAT, SET_STREAM_INFO, ADD_AUDIO_MAPPINGS or REMOVE_AUDIO_MAPPINGS command, an R-PAAD-AE automatically changes the state of the associated redundant stream/mappings beyond what the controller strictly requested, then the R-PAAD-AE shall send unsolicited notification(s) to the registered controllers.

Finally, the following unsolicited notifications shall be sent asynchronously of any command when the state of the entity changes:

Unsolicited notification	**Description**
GET_STREAM_INFO	Sent when one of these pieces of information changes: - Stream ID - MSRP accumulated latency (for a STREAM_INPUT only) - Stream destination MAC Address - MSRP Talker attribute registration state (STREAM_INPUT only) - MSRP failure code - MSRP failure bridge ID - MSRP Listener attribute registration state (STREAM_OUTPUT only) - MSRP Talker attribute declaration state (STREAM_OUTPUT only) - Stream VLAN ID - Bound state (STREAM_INPUT only) - Started/stopped state (STREAM_INPUT only)

GET_AVB_INFO	Sent when one of these pieces of information changes: - gPTP grandmaster ID - gPTP propagation delay - gPTP domain number - asCapable - Priority associated with SR Class A - Default VLAN ID associated with SR Class A
GET_AS_PATH	Sent when the gPTP path sequence changes.
GET_COUNTERS	Sent when one of the counters is updated, with the restriction of not sending more than one unsolicited notification per descriptor per second.
LOCK_ENTITY	Sent when the entity automatically unlocks itself after the one (1) minute timeout.
DEREGISTER_ UNSOLICITED_NOTIFICATION	Sent when the entity automatically deregisters a registered controller (sent only to this controller).

7.5.3. Detection of departing controllers

The PAAD-AE shall monitor the availability of each of its registered controllers:

- After each valid AECP command received from a registered controller (including the REGISTER_UNSOLICITED_NOTIFICATION command), the PAAD-AE shall (re)set the monitor timer associated with this controller. The duration of the monitor timer is a random value between 30 seconds and 60 seconds.
- At expiration of the monitor timer, the PAAD-AE shall send a CONTROLLER_AVAILABLE command to the controller (and a retry if necessary, as described in [AVDECC, Clause 9.2.2.3.2]). If the controller replies (no matter the value of the status code) then the PAAD-AE shall set a new monitor timer with a random value between 30 seconds and 60 seconds. If the controller doesn't reply then the PAAD-AE shall remove it from the list of registered controllers and send an unsolicited DEREGISTER_UNSOLICITED_NOTIFICATION to this controller only.

7.5.4. Identification notification

When a PAAD provides a way to report itself to the AVDECC Controllers (typically by providing a button on the front-panel), it should implement the Identification Notification as described in [AVDECC, Clause 7.5.1]. Please note that this mechanism is an “Entity to Controller” identification mechanism, as opposed to the “’IDENTIFY” CONTROL which is a “Controller to Entity” identification mechanism (refer to “6.12 IDENTIFY CONTROL dynamic state”).

The “identifyButtonPressed” variable mentioned in [AVDECC, Clause 7.5.1.3] is TRUE when the PAAD is in a state where the user wants it to report itself to the AVDECC Controllers, FALSE otherwise. The mapping of this variable to the exact action of the user is vendor-specific.

Note: although the “Entity to Controller” identification is in principle independent from the “Controller to Entity” identification, a specific implementation may choose to use the same button on the PAAD to trigger an Identification notification and to set/toggle the state of the “IDENTIFY” CONTROL.

8. Connection management

8.1. Concepts and principles

8.1.1. Introduction

This chapter defines a protocol used to manage connections between Talkers and Listeners on a network. The protocol is based on ACMP [AVDECC, Clause 8] with a few differences that are detailed in section 8.2.

8.1.2. Binding

In an attempt to ensure that PAADs on a network behave in a plug-and-play and user-friendly manner, this specification defines the concept of a "binding" from the perspective of a Listener. At any given time, each STREAM_INPUT of a Listener can be in either a bound or an unbound state. When a STREAM_INPUT is in a bound state, the Listener will do its best to maintain a continuous flow of packets from a specific stream (identified by STREAM_OUTPUT index) of a particular talker (identified by Entity ID). A sink (on a Listener) can only be bound to a single source (from a Talker) at a time. Multiple sinks can simultaneously be bound to the same source.

The steps involved in binding a Listener's source are as follows. First, a controller verifies that the source and sink streams are compatible (support the same stream format) and are located in the same gPTP domain (gPTP grandmaster IDs are the same). Next, the controller issues a request to the Listener to bind to the desired source on the Talker. At this point, all further actions required to establish and maintain the flow of audio data from Talker to Listener are to be performed by the Listener.

If a STREAM_INPUT is in a bound state, it will retain that state across power cycles and will only transition into an unbound state due to the intervention of a controller.

This specification doesn't define the concept of a binding from the perspective of a talker.

8.1.3. Settlement

While the binding is driven by the controller, the actual connection is done by the Listener. The probing is the process by which the Listener is requesting up-to-date SRP stream parameters to the Talker. The "settlement" is the result of the probing process, that is the fact that the Listener's sink is associated with the set of SRP parameters the Talker is using for its stream.

Once a binding has been done, the Listener continuously attempts to establish and maintain the settlement, even in the face of network disruptions and/or power outages. To make a Listener stop attempting to receive a stream, a controller must issue an unbinding request to that Listener. After acknowledging the unbinding request, the Listener stops processing packets from the stream, withdraws its Listener Ready attribute (if applicable) and clears any internal state related to the binding.

8.1.4. Auto Connect

A design goal of this protocol is that Listeners be able to maintain established connections autonomously, without relying on intervention from a controller. Once a controller has issued a binding request to a Listener and received an acknowledgement, it has no further involvement in the connection process until an unbinding is desired. Once it has acknowledged a binding request, a Listener must perform the following steps:

- Probe the SRP parameters of the talker's source stream
- Perform the necessary SRP transactions to reserve the required resources in the network
- Continuously monitor for SRP failures

If an SRP failure is detected or the Listener is reset due to a power cycle, this process must be restarted, with one additional preliminary step:

- Wait for an ADP ENTITY_AVAILABLE message from the talker to ensure that both talker and listener are still in the same gPTP domain.

The state where a Listener's sink is bound but has not yet determined the SRP parameters used by the Talker, or has to renew its knowledge of the SRP parameters, is called the probing state in this specification.

By means of the mechanism described above, a listener is able to reestablish a stream after any of the following events, without the intervention of a controller:

- Power cycle of the listener
- Power cycle of the talker
- Power cycle or replacement of any switch on the path from the talker to the listener

However, in the following situations the listener doesn't have to (and will likely not be able to) reestablish a stream:

- If the listener is restarted for some reason other than a power cycle or a simple reset (firmware upgrade, reset to factory defaults, etc.)
- If, while the listener was off, or while one of the switches on the path from the talker to the listener was off, the configuration of the talker has changed (firmware upgrade, reset to factory defaults, or any change explicitly requested by a controller)

In the two situations described above, it is expected that a controller will be used to reconfigure the listener and establish a new binding if necessary.

8.1.5. Binding, settlement and SRP reservation

It is important to understand that binding, settlement (as defined here) and stream reservation (as defined by SRP) are not created or destroyed synchronously. A Listener with a bound sink makes every effort to connect to the Talker's source and ensure that it will receive packets from the Talker's source stream. However, due to timing, connectivity and/or resource constraints on the network, the Talker may not be sending packets or even have reserved bandwidth at a particular moment in time. Similarly, when a Listener's sink gets unbound, it will take some time for the associated SRP bandwidth reservations to be released. However, once things are in a steady state, a stream will never flow from a Talker to a Listener when the Listener's sink is not bound.

8.2. Usage of ACMP

8.2.1. Preliminary note

Although the connection management protocol defined here is largely based on ACMP, it differs in some key points. Whenever the protocol defined by this specification differs from ACMP (as defined by the AVDECC standard), the definition in this specification takes precedence.

8.2.2. PDUs

The connection management protocol defined by this specification makes use of the same PDUs as ACMP, with the following changes:

- The CONNECT_RX_COMMAND/RESPONSE messages are renamed to BIND_RX_COMMAND/RESPONSE. The DISCONNECT_RX_COMMAND/RESPONSE are renamed to UNBIND_RX_COMMAND/RESPONSE.
- The CONNECT_TX_COMMAND/RESPONSE messages are renamed to PROBE_TX_COMMAND/RESPONSE.
- The DISCONNECT_TX_COMMAND/RESPONSE and GET_TX_CONNECTION_COMMAND/RESPONSE messages are not used.
- The “talker_unique_id” field of the PDUs always represents the index of a STREAM_OUTPUT descriptor of the current CONFIGURATION of a Talker PAAD-AE.
- The “listener_unique_id” field of the PDUs always represents the index of a STREAM_INPUT descriptor of the current CONFIGURATION of a Listener PAAD-AE.
- The TALKER_FAILED flag is renamed to REGISTERING_FAILED:

Bit #	Field value	Name	Meaning
25	0x00000040	REGISTERING_FAILED	In a GET_RX_STATE_RESPONSE: indicates that the sink is registering a matching Talker Failed attribute. In a GET_TX_STATE_RESPONSE: indicates that the source is declaring a Talker Advertise or a Talker Failed attribute and registering a matching Listener Asking Failed attribute.

The next sections of this specification define the legal values for the PDU fields listed below, which vary based on PDU type and the state of the sending PAAD-AE:

message_type
status
controller_entity_id
talker_entity_id
listener_entity_id
talker_unique_id
listener_unique_id
sequence_id
connection_count
FAST_CONNECT
STREAMING_WAIT
REGISTERING_FAILED
stream_id
stream_dest_mac
stream_vlan_id

When the value of one of the fields above does not make sense in a particular ACMPDU, this specification will mark it as "undefined". This means that the sender is allowed to put any value and the receiver shall ignore it.

The following fields of the ACMPDUs are not used by the protocol defined by this specification and should always be set to 0 by the sender:

CLASS_B
SAVED_STATE
SUPPORTS_ENCRYPTED
ENCRYPTED_PDU
Reserved

8.2.3. Command timeouts

The protocol defined by this specification uses modified values of the ACMP command timeouts defined in [AVDECC, Clause 8.2.2]. The values required by this specification are as follows:

Command	**Timeout value**
PROBE_TX_COMMAND	200 ms
GET_TX_STATE_COMMAND	200 ms
BIND_RX_COMMAND	200 ms
UNBIND_RX_COMMAND	200 ms
GET_RX_STATE_COMMAND	200 ms

8.2.4. Controller Bind

The protocol defined by this specification uses a modified version of the "Controller Connect" mechanism defined by [AVDECC, Clause 8], and names it Controller Bind. The Controller Bind mechanism is used to set up a new binding by requesting that a Listener binds to a specified Talker. After a successful Controller Bind transaction, the Listener is responsible for performing all further actions necessary to receive the requested stream from the Talker. The sequence of events in the Controller Bind process is as follows:

1) The Controller requests that a new binding be established by sending a BIND_RX_COMMAND message to the Listener
2) The Listener acknowledges the binding request by sending a BIND_RX_RESPONSE message to the Controller

Note: the Controller Bind process doesn't involve the Talker.

The Listener stores the binding parameters contained in the BIND_RX_COMMAND message (talker_entity_id, talker_unique_id, controller_unique_id, STREAMING_WAIT) in non-volatile memory in order to enable the stream to be quickly reestablished after a power cycle and/or network disruption.

Once the binding request has been acknowledged, the Listener invokes the Auto Connect mechanism to obtain the stream parameters and performs the necessary SRP transactions to begin receiving the stream. If at any future time the stream stops flowing or is otherwise disrupted, the Listener again invokes Auto Connect to attempt to reestablish the stream. Please refer to section 8.2.6 Auto Connect for more details.

This specification requires that the Listener check whether it is in a locked state before processing a binding request from a Controller. If the PAAD-AE is locked by a Controller other than the one issuing the request, the PAAD-AE rejects the request by returning the CONTROLLER_NOT_AUTHORIZED error code.

8.2.5. Controller Unbind

The protocol defined by this specification uses a modified version of the “Controller Disconnect” mechanism defined by [AVDECC, Clause 8], and names it Controller Unbind. The sequence of events in the Controller Unbind process is as follows:

1) The Controller requests that an existing binding be terminated by sending an UNBIND_RX_COMMAND to the Listener.
2) The Listener acknowledges the unbinding request by sending an UNBIND_RX_RESPONSE message to the Controller.

Note: the Controller Unbind process doesn’t involve the Talker.

Once the unbinding request has been acknowledged, the Listener discontinues the associated SRP reservation (revokes Listener Ready declaration) for the specified stream, and shortly thereafter will stop receiving packets from the stream. The Talker is not explicitly informed of the unbinding via an ACMP message, however it can detect the presence of interested Listeners by monitoring SRP registrations. When no active Listeners remain, the Talker will stop transmitting the stream.

This specification requires that the Listener check whether it is in a locked state before processing an unbinding request from a Controller. If the PAAD-AE is locked by a Controller other than the one issuing the request, the PAAD-AE rejects the request by returning the CONTROLLER_NOT_AUTHORIZED error code.

8.2.6. Auto Connect

The protocol defined by this specification uses a modified version of the “Fast Connect” mechanism defined by [AVDECC, Clause 8], and names it Auto Connect. Auto Connect is the mechanism by which a Listener, with a STREAM_INPUT that is in the bound state but is not currently receiving SRP registration from the associated stream, attempts to initiate or reestablish the SRP registration. This is accomplished by periodically sending PROBE_TX_COMMAND messages to query the associated Talker about the parameters of the specified stream, and performing the necessary SRP transactions once the relevant stream parameters are known. If an established stream is interrupted for any reason (power cycle, network outage, etc), the Listener shall perform the following steps:

1. Discontinue the SRP reservation associated with the stream, and clear any saved SRP stream parameters (Stream ID, Stream Destination MAC Address, Stream VLAN ID).
2. Listen for ADP messages advertising the presence of the associated Talker on the same gPTP domain as the Listener.
3. Send a PROBE_TX_COMMAND message to the associated Talker to query the parameters of the stream.
4. Wait for a PROBE_TX_RESPONSE message from the Talker, or timeout and repeat step 3.
5. Initiate a new SRP reservation for the stream.

Please note that steps 1 and 2 are not done when Auto Connect is performed as an immediate consequence of a Controller Bind operation.

While a Listener is performing Auto Connect, a Controller can obtain details about the current state of the Auto Connect process by examining the probing_status and acmp_status fields of the Listener's response to an AECP AEM GET_STREAM_INFO message (see section 6.8.6).

The complete specification of the Auto Connect mechanism is in section 8.3 Specification of the Listener's behavior.

8.2.7. Talker's behavior

The protocol defined by this specification modifies certain aspects of the behavior of Talkers that are specified or implied by [AVDECC, Clause 8]:

- Talkers begin declaring MSRP Talker Advertise or Talker Failed attributes for each of their STREAM_OUTPUTs as soon as possible. Talkers rely only on SRP (not ACMP) to determine whether any Listeners are interested in their STREAM_OUTPUTs by monitoring whether any matching Listener Ready or Listener Ready Failed attributes have been registered.
- Talkers do not maintain any internal state related to bound/settled Listeners. The DISCONNECT_TX_COMMAND always returns SUCCESS without modifying the state of the Talker, and the GET_TX_CONNECTION_COMMAND is not implemented. A PAAD-AE will never issue the aforementioned commands. The PROBE_TX_COMMAND is only used to query the Talker about the parameters of a stream - it has no impact on the internal state of the Talker.

The complete specification of the talker's behavior is in section 8.4 Specification of the Talker's behavior.

8.3. Specification of the Listener's behavior

8.3.1. Treatment of an incoming ACMP message

On reception of a BIND_RX_COMMAND / GET_RX_STATE_COMMAND / UNBIND_RX_COMMAND message with listener_entity_id field being equal to the Entity ID of the PAAD-AE, the PAAD-AE shall:

1) Extract the listener's sink from the listener_unique_id field and check that it is valid. If not, send a BIND_RX_RESPONSE / GET_RX_STATE_RESPONSE / UNBIND_RX_RESPONSE message formatted as follows, and then exit.

Field	**Value**
message_type	BIND_RX/GET_RX_STATE/UNBIND_RX_RESPONSE
status	LISTENER_UNKNOWN_ID
controller_entity_id	Same value as in the command message.
talker_entity_id	Same value as in the command message.
listener_entity_id	Same value as in the command message.
talker_unique_id	Same value as in the command message.
listener_unique_id	Same value as in the command message.
sequence_id	Same value as in the command message.
connection_count	Undefined.
FAST_CONNECT	Undefined.

STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Continue processing the message according to the sink state machine (RCV_BIND_RX_CMD / RCV_GET_RX_STATE / RCV_UNBIND_RX_CMD event, see the next sections below).

Upon reception of a PROBE_TX_RESPONSE message with listener_entity_id field equal to the Entity ID of the PAAD-AE, the PAAD-AE shall:

1) Extract the listener's sink from the listener_unique_id field and check that it is valid. If not, ignore the message and exit.
2) Continue processing the message according to the sink state machine (RCV_PROBE_TX_RESP event, see the next sections below).

The listener shall ignore all other ACMP messages.

8.3.2. States of a sink

State	Description
UNBOUND	The sink is not bound.
PRB_W_AVAIL	The sink is probing. The PAAD-AE is waiting for the ADP state machine to report that the talker has been discovered. In this state, the PAAD-AE is running the ADP Discovery state machine.
PRB_W_DELAY	The sink is probing. The PAAD-AE is ready to send a PROBE_TX_COMMAND but is waiting for a random delay before sending the message. This is to avoid that all listeners send the message simultaneously. In this state, the PAAD-AE is running the ADP Discovery state machine.

PRB_W_RESP	The sink is probing. The PAAD-AE has just sent a PROBE_TX_COMMAND message to the talker and is waiting for the response. In this state, the PAAD-AE is running the ADP Discovery state.
PRB_W_RESP2	The sink is probing. The PAAD-AE has just sent a second PROBE_TX_COMMAND due to a timeout on the first command. In this state, the PAAD-AE is running the ADP Discovery state machine.
PRB_W_RETRY	The sink is probing. The PAAD-AE has timed out on the two sent PROBE_TX_COMMAND messages and is now waiting for the TMR_RETRY timer to expire before trying again. In this state, the PAAD-AE is running the ADP Discovery state machine.
SETTLED_NO_RSV	The sink is settled (has up-to-date SRP parameters). The PAAD-AE has not registered any SRP Talker attribute yet for the settled stream. In this state, the PAAD-AE is running the ADP Discovery state machine and monitoring the SRP Talker attribute.
SETTLED_RSV_OK	The sink is settled (has up-to-date SRP parameters). The PAAD-AE is registering an SRP Talker attribute for the settled stream. In this state, the PAAD-AE is running the ADP Discovery state machine and monitoring the SRP Talker attribute.

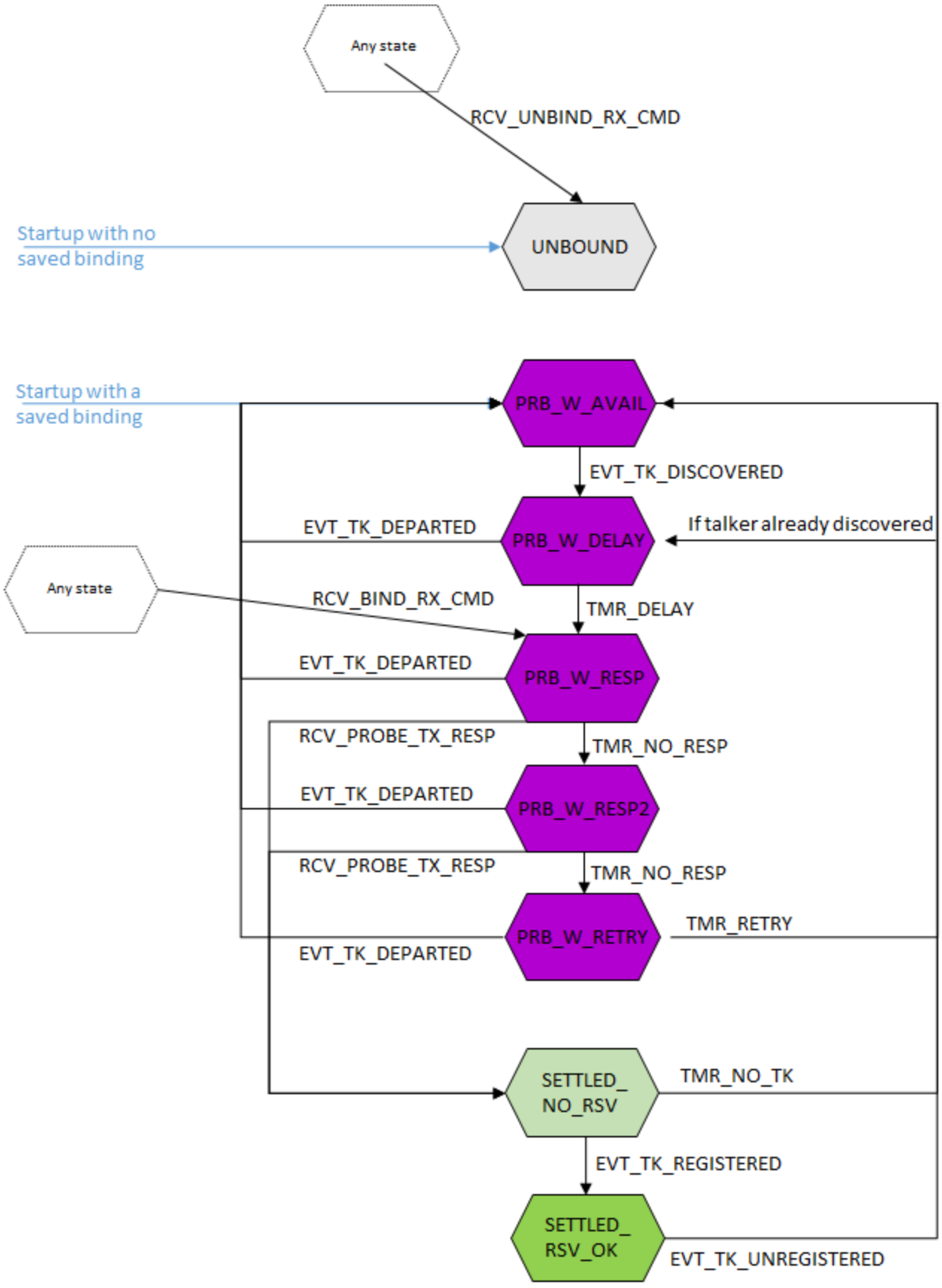
8.3.3. Events that trigger the state machine

Event	Description
TMR_NO_RESP	The talker's response timeout has expired. The value of the timeout is 200 milliseconds.
TMR_RETRY	The retry timeout has expired. The value of this timeout is 4 seconds.
TMR_DELAY	The random delay timeout has expired. The value of this timeout is between 0 and 1 second.
TMR_NO_TK	The Talker attribute registration timeout has expired. The value of this timeout is 10 seconds.
RCV_BIND_RX_CMD	Received a BIND_RX_COMMAND from a controller.
RCV_PROBE_TX_RESP	Received a PROBE_TX_RESPONSE from the talker.
RCV_GET_RX_STATE	Received a GET_RX_STATE_COMMAND from a controller.
RCV_UNBIND_RX_CMD	Received an UNBIND_RX_COMMAND from a controller.
EVT_TK_DISCOVERED	The ADP state machine has just reported that the talker has been discovered. This event is triggered only once on the transition Not discovered → Discovered.

EVT_TK_DEPARTED	The ADP state machine has just reported that the talker has been departed. This event is triggered only once on the transition Discovered → Not discovered.
EVT_TK_REGISTERED	The SRP state machine has just registered a Talker attribute (either Talker Advertise or Talker Failed) which matches the Stream ID, Stream Destination MAC Address and Stream VLAN ID that are associated with the settled stream. This event is triggered only once on the transition Not registered → Registered.
EVT_TK_UNREGISTERED	The SRP state machine has just unregistered the Talker attribute that was registered for the settled stream. This event is triggered only once on the transition Registered → Not registered.

8.3.4. Diagram of the state machine (informative)

Please note that the following diagram is a simplified overview of the real state machine that a sink implements. The normative description of the transitions of the state machine is given in the next section.


Any state
RCV_UNBIND_RX_CMD
Startup with no saved binding
UNBOUND
Startup with a saved binding
PRB_W_AVAIL
EVT_TK_DISCOVERED
EVT_TK_DEPARTED
PRB_W_DELAY
If talker already discovered
Any state
RCV_BIND_RX_CMD
TMR_DELAY
EVT_TK_DEPARTED
PRB_W_RESP
RCV_PROBE_TX_RESP
TMR_NO_RESP
EVT_TK_DEPARTED
PRB_W_RESP2
RCV_PROBE_TX_RESP
TMR_NO_RESP
PRB_W_RETRY
TMR_RETRY
EVT_TK_DEPARTED
SETTLED_ NO_RSV
TMR_NO_TK
EVT_TK_REGISTERED
SETTLED_ RSV_OK
EVT_TK_UNREGISTERED

The table below summarizes the possible transitions and indicates which section describes each transition:

	UNBOUND	PRB_W_AVAIL	PRB_W_DELAY	PRB_W_RESP	PRB_W_RESP2	PRB_W_RETRY	SETTLED_NO_RSV	SETTLED_RSV_OK
TMR_NO_RESP	x	x	x	8.3.5.16	8.3.5.23	x	x	x
TMR_RETRY	x	x	x	x	x	8.3.5.30	x	x
TMR_DELAY	x	x	8.3.5.10	x	x	x	x	x
TMR_NO_TK	x	x	x	x	x	x	8.3.5.36	x
RCV_BIND_RX_CMD	8.3.5.3	8.3.5.6	8.3.5.11	8.3.5.17	8.3.5.24	8.3.5.31	8.3.5.37	8.3.5.43
RCV_PROBE_TX_RESP	-	-	-	8.3.5.18	8.3.5.25	-	-	-
RCV_GET_RX_STATE	8.5.3.4	8.3.5.7	8.3.5.12	8.3.5.19	8.3.5.26	8.3.5.32	8.3.5.38	8.3.5.44
RCV_UNBIND_RX_CMD	8.3.5.5	8.3.5.8	8.3.5.13	8.3.5.20	8.3.5.27	8.3.5.33	8.3.5.39	8.3.5.45
EVT_TK_DISCOVERED	x	8.3.5.9	8.3.5.14	8.3.5.21	8.3.5.28	8.3.5.34	8.3.5.40	8.3.5.46
EVT_TK_DEPARTED	x	x	8.3.5.15	8.3.5.22	8.3.5.29	8.3.5.35	8.3.5.41	8.3.5.47
EVT_TK_REGISTERED	x	x	x	x	x	x	8.3.5.42	x
EVT_TK_UNREGISTERED	x	x	x	x	x	x	x	8.3.5.48

x = this event cannot happen in this state

- = this event is ignored in this state

8.3.5. Details of events processing

8.3.5.1. Startup of the PAAD-AE with no saved binding for this sink

1) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
2) Go to the UNBOUND state.

8.3.5.2. Startup of the PAAD-AE with a saved binding for this sink

1) Start the ADP Discovery state machine for the talker we want to connect to (at this point, talker has not been discovered).
2) Set the Probing status to PROBING_PASSIVE and ACMP status to 0.
3) Go to the PRB_W_AVAIL state.

8.3.5.3. UNBOUND state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, check that the controller_entity_id field of the BIND_RX_COMMAND is the Entity ID of the locking controller. If not, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.

listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
3) Send the following BIND_RX_RESPONSE message.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

4) Start the ADP Discovery state machine (at this point, talker has not been discovered).
5) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00

stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

6) Save a copy of the sent PROBE_TX_COMMAND message.
7) Set a 200-millisecond timer TMR_NO_RESP.
8) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
9) Go to the PRB_W_RESP state.

8.3.5.4. UNBOUND state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8.3.5.5. UNBOUND state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, check that the controller_entity_id field of the UNBIND_RX_COMMAND is the Entity ID of the locking controller. If not, send the following UNBIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.

stream_vlan_id	Undefined.

2) Send the following UNBIND_RX_RESPONSE message:

Field	Value
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8.3.5.6. PRB_W_ AVAIL state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, check that the controller_entity_id field of the BIND_RX_COMMAND is the Entity ID of the locking controller. If not, send the following BIND_RX_RESPONSE message and exit.

Field	Value
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine
3) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
4) Send the following BIND_RX_RESPONSE message.

Field	Value

message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

5) Start the ADP Discovery state machine (at this point, talker has not been discovered).
6) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Save a copy of the sent PROBE_TX_COMMAND message.
9) Set a 200-millisecond timer TMR_NO_RESP.
10) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
11) Go to the PRB_W_RESP state.

8.3.5.7. PRB_W_ AVAIL state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS

controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	0
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

8.3.5.8. PRB_W_ AVAIL state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, check that the controller_entity_id field of the UNBIND_RX_COMMAND is the Entity ID of the locking controller. If not, send the following UNBIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine.
3) Clear the saved binding parameters.
4) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
5) Send the following UNBIND_RX_RESPONSE message.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.

sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

6) Go to the UNBOUND state.

8.3.5.9. PRB_W_ AVAIL state / EVT_TK_DISCOVERED event

1) Note that the talker has been discovered.
2) Set a random delay timer (between 0 and 1 second) TMR_DELAY.
3) Go to the PRB_W_DELAY state.

8.3.5.10. PRB_W_DELAY state / TMR_DELAY event

1) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

12) Save a copy of the sent PROBE_TX_COMMAND message.
13) Set a 200-millisecond timer TMR_NO_RESP.
14) Go to the PRB_W_RESP state.

8.3.5.11. PRB_W_DELAY state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the BIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE

status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine.
3) Stop the TMR_DELAY timer.
4) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
5) Send the following BIND_RX_RESPONSE message.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

6) Start the ADP Discovery state machine (at this point, talker has not been discovered).
7) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.

sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Save a copy of the sent PROBE_TX_COMMAND message.
9) Set a 200-millisecond timer TMR_NO_RESP.
10) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
11) Go to the PRB_W_RESP state.

8.3.5.12. PRB_W_DELAY state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	0
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

8.3.5.13. PRB_W_DELAY state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the UNBIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following UNBIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.

sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine.
3) Clear the saved binding parameters.
4) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
5) Stop the TMR_DELAY timer.
6) Send the following UNBIND_RX_RESPONSE message.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

7) Go to the UNBOUND state.

8.3.5.14. PRB_W_DELAY state / EVT_TK_DISCOVERED event

Note that the talker has been discovered.

8.3.5.15. PRB_W_DELAY state / EVT_TK_DEPARTED event

1) Note that the talker has not been discovered.
2) Stop the TMR_DELAY timer.
3) Set the Probing status to PROBING_PASSIVE and ACMP status to 0.
4) Go to the PRB_W_AVAIL state.

8.3.5.16. PRB_W_RESP state / TMR_NO_RESP event

1) Send a duplicate of the PROBE_TX_COMMAND message that was sent the first time, on the port associated with the STREAM_INPUT descriptor.
2) Set a 200-millisecond non-response timer TMR_NO_RESP
3) Go to the PRB_W_RESP2 state.

8.3.5.17. PRB_W_RESP state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the BIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine
3) Stop the TMR_NO_RESP timer
4) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
5) Send the following BIND_RX_RESPONSE message.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00

stream_vlan_id	0

6) Start the ADP Discovery state machine (at this point, talker has not been discovered).
7) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Save a copy of the sent PROBE_TX_COMMAND message.
9) Set a 200-millisecond timer TMR_NO_RESP.
10) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
11) Go to the PRB_W_RESP state.

8.3.5.18. PRB_W_RESP state / RCV_PROBE_TX_RESP event

1) Check that the controller_entity_id, talker_entity_id, talker_unique_id and sequence_id fields match the fields of the PROBE_TX_COMMAND message that has been sent. If not, ignore the message and exit.
2) Stop the TMR_NO_RESP timer.
3) If STATUS!=SUCCESS then set a 4-second timer TMR_RETRY, set the ACMP status to the value of status, go to the PRB_W_RETRY state and exit.
4) Otherwise, note the stream_id, stream_dest_mac and stream_vlan_id info, initiate SRP reservation and start listening for the stream packets. Then start a 10-second timer TMR_NO_TK, set the Probing status to PROBING_COMPLETED and ACMP status to 0 and go to the SETTLED_NO_RSV state.

8.3.5.19. PRB_W_RESP state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.

talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	0
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

8.3.5.20. PRB_W_RESP state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the UNBIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following UNBIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine.
3) Clear the saved binding parameters.
4) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
5) Stop the TMR_NO_RESP timer.
6) Send the following UNBIND_RX_RESPONSE message.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0

FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

7) Go to the UNBOUND state.

8.3.5.21. PRB_W_RESP state / EVT_TK_DISCOVERED event

Note that the talker has been discovered.

8.3.5.22. PRB_W_RESP state / EVT_TK_DEPARTED event

1) Note that the talker has not been discovered.
2) Stop the TMR_NO_RESP timer.
3) Set the Probing status to PROBING_PASSIVE and ACMP status to 0.
4) Go to the PRB_W_AVAIL state.

8.3.5.23. PRB_W_RESP2 state / TMR_NO_RESP event

1) Set a 4-second timer TMR_RETRY.
2) Set the ACMP status to 7 (LISTENER_TALKER_TIMEOUT).
3) Go to the PRB_W_RETRY state.

8.3.5.24. PRB_W_RESP2 state / RCV_BIND_RX_CMD event

Same treatment as in the PRB_W_RESP state.

8.3.5.25. PRB_W_RESP2 state / RCV_PROBE_TX_RESP event

Same treatment as in the PRB_W_RESP state.

8.3.5.26. PRB_W_RESP2 state / RCV_GET_RX_STATE event

Same treatment as in the PRB_W_RESP state.

8.3.5.27. PRB_W_RESP2 state / RCV_UNBIND_RX_CMD event

Same treatment as in the PRB_W_RESP state.

8.3.5.28. PRB_W_RESP2 state / EVT_TK_DISCOVERED event

Same treatment as in the PRB_W_RESP state.

8.3.5.29. PRB_W_RESP2 state / EVT_TK_DEPARTED event

Same treatment as in the PRB_W_RESP state.

8.3.5.30. PRB_W_ RETRY state / TMR_RETRY event

1) If the talker has not yet been discovered, set the Probing status to PROBING_PASSIVE and ACMP status to 0 and go to the PRB_W_AVAIL state.
2) Otherwise, start a random delay timer (between 0 and 1 second) TMR_DELAY and go to the PRB_W_DELAY state.

8.3.5.31. PRB_W_ RETRY state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the BIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine
3) Stop the TMR_RETRY timer
4) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
5) Send the following BIND_RX_RESPONSE message.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00

stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

6) Start the ADP Discovery state machine (at this point, talker has not been discovered).
7) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Save a copy of the sent PROBE_TX_COMMAND message.
9) Set a 200-millisecond timer TMR_NO_RESP.
10) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
11) Go to the PRB_W_RESP state.

8.3.5.32. PRB_W_ RETRY state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	0
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

8.3.5.33. PRB_W_ RETRY state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the UNBIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following UNBIND_RX_RESPONSE message and exit.

Field	Value
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Stop the ADP Discovery state machine.
3) Clear the saved binding parameters.
4) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
5) Stop the TMR_RETRY timer.
6) Send the following UNBIND_RX_RESPONSE message.

Field	Value
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

7) Go to the UNBOUND state.

8.3.5.34. PRB_W_ RETRY state / EVT_TK_DISCOVERED event

Note that the talker has been discovered.

8.3.5.35. PRB_W_RETRY state / EVT_TK_DEPARTED event

1) Note that the talker has not been discovered.
2) Stop the TMR_RETRY timer.
3) Set the Probing status to PROBING_PASSIVE and ACMP status to 0.
4) Go to the PRB_W_AVAIL state.

8.3.5.36. SETTLED_NO_RSV state / TMR_NO_TK event

1) Clear the SRP parameters and stop SRP.
2) If the talker has not yet been discovered then set the Probing status to PROBING_PASSIVE and ACMP status to 0 and go to the PRB_W_AVAIL state.
3) Otherwise, set the Probing status to PROBING_ACTIVE and ACMP status to 0, start a random delay timer (between 0 and 1 second) TMR_DELAY and go to the PRB_W_DELAY state.

8.3.5.37. SETTLED_NO_RSV state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the BIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Clear the SRP parameters and stop SRP.
3) Stop the ADP Discovery state machine.
4) Stop the TMR_NO_TK timer.
5) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
6) Send the following BIND_RX_RESPONSE message.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	SUCCESS

controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

7) Start the ADP Discovery state machine (at this point, talker has not been discovered).
8) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	**Value**
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

9) Save a copy of the sent PROBE_TX_COMMAND message.
10) Set a 200-millisecond timer TMR_NO_RESP.
11) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
12) Go to the PRB_W_RESP state.

8.3.5.38. SETTLED_NO_RSV state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	**Value**
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.

listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	0
stream_id	Value copied from the STREAM_INPUT's SRP parameters.
stream_dest_mac	Value copied from the STREAM_INPUT's SRP parameters.
stream_vlan_id	Value copied from the STREAM_INPUT's SRP parameters.

8.3.5.39. SETTLED_NO_RSV state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the UNBIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following UNBIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Clear the SRP parameters and stop SRP.
3) Stop the ADP Discovery state machine.
4) Clear the saved binding parameters.
5) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
6) Stop the TMR_NO_TK timer.
7) Send the following UNBIND_RX_RESPONSE message.

Field	**Value**
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.

sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Go to the UNBOUND state.

8.3.5.40. SETTLED_NO_RSV state / EVT_TK_DISCOVERED event

Note that the talker has been discovered.

8.3.5.41. SETTLED_NO_RSV state / EVT_TK_DEPARTED event

Note that the talker has not been discovered.

8.3.5.42. SETTLED_NO_RSV state / EVT_TK_REGISTERED event

1) Clear the TMR_NO_TK timer.
2) Go to the SETTLED_RSV_OK state.

8.3.5.43. SETTLED_RSV_OK state / RCV_BIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the BIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following BIND_RX_RESPONSE message and exit.

Field	**Value**
message_type	BIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Clear the SRP parameters and stop SRP.
3) Stop the ADP Discovery state machine.

4) Update the binding parameters with the controller_entity_id, talker_entity_id, talker_unique_id and STREAMING_WAIT fields, then save them to non volatile memory.
5) Send the following BIND_RX_RESPONSE message.

Field	Value
message_type	BIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_entity_id	Same value as in the BIND_RX_COMMAND message.
listener_entity_id	Same value as in the BIND_RX_COMMAND message.
talker_unique_id	Same value as in the BIND_RX_COMMAND message.
listener_unique_id	Same value as in the BIND_RX_COMMAND message.
sequence_id	Same value as in the BIND_RX_COMMAND message.
connection_count	1
FAST_CONNECT	0
STREAMING_WAIT	Same value as in the BIND_RX_COMMAND message.
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

6) Start the ADP Discovery state machine (at this point, talker has not been discovered).
7) Send the following PROBE_TX_COMMAND message on the port associated with the STREAM_INPUT descriptor.

Field	Value
message_type	PROBE_TX_COMMAND
status	SUCCESS
controller_entity_id	Value copied from the saved binding parameters.
talker_entity_id	Value copied from the saved binding parameters.
listener_entity_id	Entity ID of the PAAD-AE.
talker_unique_id	Value copied from the saved binding parameters.
listener_unique_id	Index of the sink.
sequence_id	Value selected by the PAAD-AE.
connection_count	0
FAST_CONNECT	1
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8) Save a copy of the sent PROBE_TX_COMMAND message.
9) Set a 200-millisecond timer TMR_NO_RESP.
10) Set the Probing status to PROBING_ACTIVE and ACMP status to 0.
11) Go to the PRB_W_RESP state.

8.3.5.44. SETTLED_RSV_OK state / RCV_GET_RX_STATE event

Send the following GET_RX_STATE_RESPONSE message:

Field	Value
message_type	GET_RX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_entity_id	Entity ID of the talker to which the sink is bound.
listener_entity_id	Same value as in the GET_RX_STATE_COMMAND message.
talker_unique_id	Source to which the sink is bound.
listener_unique_id	Same value as in the GET_RX_STATE_COMMAND message.
sequence_id	Same value as in the GET_RX_STATE_COMMAND message.
connection_count	1
FAST_CONNECT	1
STREAMING_WAIT	Value copied from the saved binding parameters.
REGISTERING_FAILED	1 if registering a Talker Failed attribute, 0 otherwise.
stream_id	Value copied from the STREAM_INPUT's SRP parameters.
stream_dest_mac	Value copied from the STREAM_INPUT's SRP parameters.
stream_vlan_id	Value copied from the STREAM_INPUT's SRP parameters.

8.3.5.45. SETTLED_RSV_OK state / RCV_UNBIND_RX_CMD event

1) If the PAAD-AE is locked, verify that the controller_entity_id field of the UNBIND_RX_COMMAND matches the Entity ID of the locking controller. If it doesn't, send the following UNBIND_RX_RESPONSE message and exit.

Field	Value
message_type	UNBIND_RX_RESPONSE
status	CONTROLLER_NOT_AUTHORIZED
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_entity_id	Same value as in the UNBIND_RX_COMMAND message.
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	Same value as in the UNBIND_RX_COMMAND message.
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Clear the SRP parameters and stop SRP.
3) Stop the ADP Discovery state machine.
4) Clear the saved binding parameters.
5) Set the Probing status to PROBING_DISABLED and ACMP status to 0.
6) Send the following UNBIND_RX_RESPONSE message.

Field	Value
message_type	UNBIND_RX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the UNBIND_RX_COMMAND message.

talker_entity_id	00:00:00:00:00:00:00:00
listener_entity_id	Same value as in the UNBIND_RX_COMMAND message.
talker_unique_id	0
listener_unique_id	Same value as in the UNBIND_RX_COMMAND message.
sequence_id	Same value as in the UNBIND_RX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

7) Go to the UNBOUND state.

8.3.5.46. SETTLED_RSV_OK state / EVT_TK_DISCOVERED event

Note that the talker has been discovered.

8.3.5.47. SETTLED_RSV_OK state / EVT_TK_DEPARTED event

Note that the talker has not been discovered.

8.3.5.48. SETTLED_RSV_OK state / EVT_TK_UNREGISTERED event

1) Clear the SRP parameters and stop SRP.
2) If the talker has not yet been discovered then set the Probing status to PROBING_PASSIVE and ACMP status to 0 and go to the PRB_W_AVAIL state.
3) Otherwise, set the Probing status to PROBING_ACTIVE and ACMP status to 0, start a random delay timer (between 0 and 1 second) TMR_DELAY and go to the PRB_W_DELAY state.

8.4. Specification of the Talker's behavior

8.4.1. Treatment of a PROBE_TX_COMMAND message

Upon reception of a PROBE_TX_COMMAND message with talker_entity_id field being equal to the Entity ID of the PAAD-AE, the PAAD-AE shall:

1) Extract the talker's source from the talker_unique_id field and check that it is valid. If not, send the following PROBE_TX_RESPONSE message and exit.

Field	**Value**
message_type	PROBE_TX_RESPONSE
status	TALKER_UNKNOWN_ID
controller_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_entity_id	Same value as in the PROBE_TX_COMMAND message.
listener_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_unique_id	Same value as in the PROBE_TX_COMMAND message.

listener_unique_id	Same value as in the PROBE_TX_COMMAND message.
sequence_id	Same value as in the PROBE_TX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Check that the command ingressed from the port attached to the target STREAM_OUTPUT. If not, either ignore the command, or send the following PROBE_TX_RESPONSE . Then exit.

Field	**Value**
message_type	PROBE_TX_RESPONSE
status	INCOMPATIBLE_REQUEST
controller_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_entity_id	Same value as in the PROBE_TX_COMMAND message.
listener_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_unique_id	Same value as in the PROBE_TX_COMMAND message.
listener_unique_id	Same value as in the PROBE_TX_COMMAND message.
sequence_id	Same value as in the PROBE_TX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

3) Check that a valid Destination MAC Address is available for this STREAM_OUTPUT. If not, send the following PROBE_TX_RESPONSE message and exit.

Field	**Value**
message_type	PROBE_TX_RESPONSE
status	TALKER_DEST_MAC_FAILED
controller_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_entity_id	Same value as in the PROBE_TX_COMMAND message.
listener_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_unique_id	Same value as in the PROBE_TX_COMMAND message.
listener_unique_id	Same value as in the PROBE_TX_COMMAND message.
sequence_id	Same value as in the PROBE_TX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

4) Send the following PROBE_TX_RESPONSE message.

Field	Value
message_type	PROBE_TX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_entity_id	Same value as in the PROBE_TX_COMMAND message.
listener_entity_id	Same value as in the PROBE_TX_COMMAND message.
talker_unique_id	Same value as in the PROBE_TX_COMMAND message.
listener_unique_id	Same value as in the PROBE_TX_COMMAND message.
sequence_id	Same value as in the PROBE_TX_COMMAND message.
connection_count	0
FAST_CONNECT	Same value as in the PROBE_TX_COMMAND message.
STREAMING_WAIT	Same value as in the PROBE_TX_COMMAND message.
REGISTERING_FAILED	0
stream_id	Stream ID of the stream transmitted by this source.
stream_dest_mac	Destination MAC Address used by the stream.
stream_vlan_id	VLAN ID used by the stream.

A Talker PAAD shall ignore the STREAMING_WAIT bit of the PROBE_TX_COMMAND message. No matter the value of the STREAMING_WAIT bit, a Talker PAAD shall always stream AVTP packets as long as bandwidth is reserved for its stream.

8.4.2. Treatment of a DISCONNECT_TX_COMMAND message

Upon reception of a DISCONNECT_TX_COMMAND message with talker_entity_id field being equal to the Entity ID of the PAAD-AE, the PAAD-AE shall:

1) Extract the talker's source from the talker_unique_id field and check that it is valid. If not, send the following DISCONNECT_TX_RESPONSE message and exit.

Field	Value
message_type	DISCONNECT_TX_RESPONSE
status	TALKER_UNKNOWN_ID
controller_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
talker_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
listener_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
talker_unique_id	Same value as in the DISCONNECT_TX_COMMAND message.
listener_unique_id	Same value as in the DISCONNECT_TX_COMMAND message.
sequence_id	Same value as in the DISCONNECT_TX_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Send the following DISCONNECT_TX_RESPONSE message.

Field	Value
message_type	DISCONNECT_TX_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
talker_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
listener_entity_id	Same value as in the DISCONNECT_TX_COMMAND message.
talker_unique_id	Same value as in the DISCONNECT_TX_COMMAND message.
listener_unique_id	Same value as in the DISCONNECT_TX_COMMAND message.
sequence_id	Same value as in the DISCONNECT_TX_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	0
stream_id	00:00:00:00:00:00:00:00
stream_dest_mac	00:00:00:00:00:00
stream_vlan_id	0

8.4.3. Treatment of a GET_TX_STATE_COMMAND message

Upon reception of a GET_TX_STATE_COMMAND message with talker_entity_id field being equal to the Entity ID of the PAAD-AE, the PAAD-AE shall:

1) Extract the talker's source from the talker_unique_id field and check that it is valid. If not, send the following GET_TX_STATE_RESPONSE message and exit.

Field	Value
message_type	GET_TX_STATE_RESPONSE
status	TALKER_UNKNOWN_ID
controller_entity_id	Same value as in the GET_TX_STATE_COMMAND message.
talker_entity_id	Same value as in the GET_TX_STATE_COMMAND message.
listener_entity_id	Same value as in the GET_TX_STATE_COMMAND message.
talker_unique_id	Same value as in the GET_TX_STATE_COMMAND message.
listener_unique_id	Same value as in the GET_TX_STATE_COMMAND message.
sequence_id	Same value as in the GET_TX_STATE_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

2) Send the following GET_TX_STATE_RESPONSE message.

Field	Value
message_type	GET_TX_STATE_RESPONSE
status	SUCCESS
controller_entity_id	Same value as in the GET_TX_STATE_COMMAND message.

talker_entity_id	Same value as in the GET_TX_STATE_COMMAND message.
listener_entity_id	00:00:00:00:00:00:00:00
talker_unique_id	Same value as in the GET_TX_STATE_COMMAND message.
listener_unique_id	0
sequence_id	Same value as in the GET_TX_STATE_COMMAND message.
connection_count	0
FAST_CONNECT	0
STREAMING_WAIT	0
REGISTERING_FAILED	1 if registering a Listener Asking Failed attribute, 0 otherwise.
stream_id	Stream ID of the stream transmitted by this source (undefined if the PAAD is not declaring any Talker attribute).
stream_dest_mac	Destination MAC Address used by the stream (undefined if the PAAD is not declaring any Talker attribute).
stream_vlan_id	VLAN ID used by the stream (undefined if the PAAD is not declaring any Talker attribute).

8.4.4. Treatment of a GET_TX_CONNECTION_COMMAND message

Upon reception of a GET_TX_CONNECTION_COMMAND message, the PAAD-AE shall send the following GET_TX_CONNECTION_RESPONSE message:

Field	**Value**
message_type	GET_TX_CONNECTION_RESPONSE
status	NOT_SUPPORTED
controller_entity_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
talker_entity_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
listener_entity_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
talker_unique_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
listener_unique_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
sequence_id	Same value as in the GET_TX_CONNECTION_COMMAND message.
connection_count	Undefined.
FAST_CONNECT	Undefined.
STREAMING_WAIT	Undefined.
REGISTERING_FAILED	Undefined.
stream_id	Undefined.
stream_dest_mac	Undefined.
stream_vlan_id	Undefined.

9. Discovery

9.1. General

A talker shall start ADP only when it is ready to accept AECP commands and probe requests from a listener.

A listener shall start ADP only when it is ready to accept AECP commands and binding requests from a controller.

9.2. ADPDUs

A PAAD-AE shall form ADPDUs as described in [AVDECC, Clause 6], with the following clarifications:

- The entity_model_id field shall be a valid EUI-64 (neither all zeros nor all ones).
- In the entity_capabilities field, the AEM_SUPPORTED, VENDOR_UNIQUE_SUPPORTED, CLASS_A_SUPPORTED, GPTP_SUPPORTED, AEM_IDENTIFY_CONTROL_INDEX_VALID and AEM_INTERFACE_INDEX_VALID bits shall be set to 1. The AEM_PERSISTENT_ACQUIRE_SUPPORTED, GENERAL_CONTROLLER_IGNORE and ENTITY_NOT_READY bits shall be set to 0.
- The talker_stream_sources shall be set to the maximum number of Streams the AVDECC Talker is capable of sourcing simultaneously, among all possible CONFIGURATIONs.
- The listener_stream_sinks shall be set to the maximum number of Streams the AVDECC Listener is capable of sinking simultaneously, among all possible CONFIGURATIONs.
- The gptp_grandmaster_id and gptp_domain_number fields shall be populated as per [AVDECC, Clause 6.2.1.17] and [AVDECC, Clause 6.2.1.18].
- The identify_control_index field shall be populated as per [AVDECC, Clause 6.2.1.19].
- The interface_index field shall be populated as per [AVDECC, Clause 6.2.1.20].
- The valid_time field shall be set to 10 in ENTITY_AVAILABLE messages. *Note: this results in the requirement that a PAAD-AE shall advertise itself every 5 seconds.*

Note: the fields of the ADPDUs are independent of the currently set CONFIGURATION of the PAAD-AE.

9.3. Advertise state machine

The PAAD-AE shall implement an independent instance of the Advertise state machine as described in this section, for each AVB_INTERFACE of the currently set CONFIGURATION.

9.3.1. Treatment of an incoming ADP message

Upon reception of an ADP ENTITY_DISCOVER message, the PAAD-AE shall:

1) Extract the entity_id field of the message.
2) If the entity_id is 0 or equal to the Entity ID of the PAAD-AE, then continue processing the message according to the state machine below (RCV_ADP_DISCOVER event). Otherwise, discard the message.

9.3.2. States of the port

State	Description

DOWN	The link is down.
WAITING	The link is up. The PAAD-AE is waiting for an event which will require the transmission of an ADP ENTITY_AVAILABLE or ADP ENTITY_DEPARTING message.
DELAY	The link is up. An event has required the transmission of an ADP ENTITY_AVAILABLE message but the PAAD-AE is waiting a random delay before transmitting the message on the wire. This is to avoid storms of ADP ENTITY_AVAILABLE messages, which could collapse the network, when several devices are powered up simultaneously or when a controller sends an ADP ENTTY_DISCOVER message with entity_id=0.

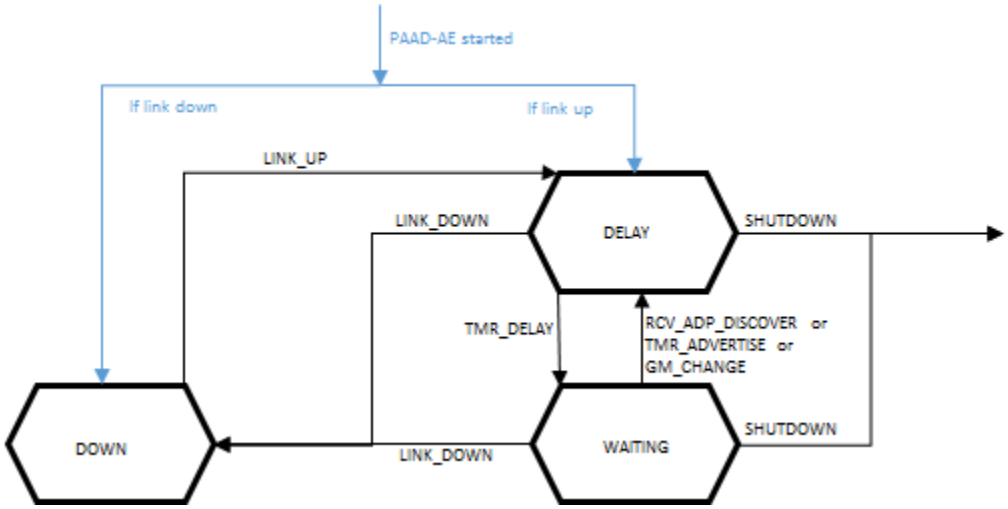
9.3.3. Events that trigger the state machine

Event	**Description**
RCV_ADP_DISCOVER	Received an ADP ENTITY_DISCOVER message with entity_id=0 or entity_id=the Entity ID of the PAAD.
TMR_ADVERTISE	ADP advertise timer expired. This is a fixed 5-second timer.
TMR_DELAY	ADP random delay timer expired. This is a random timer between 0 and 4 seconds.
LINK_UP	The link state has changed from Down to Up.
LINK_DOWN	The link state has changed from Up to Down.

GM_CHANGE	The ID of the current Grandmaster has changed.
SHUTDOWN	Request to shut down the PAAD-AE.

9.3.4. Diagram of the state machine (informative)

Please note that the following diagram is a simplified overview of the real state machine that a PAAD-AE implements. The normative description of the transitions of the state machine is given in the next section.



The table below summarizes the possible transitions and indicates which section describes each transition:

	DOWN	WAITING	DELAY
RCV_ADP_DISCOVER	-	9.3.5.4	-
TMR_ADVERTISE	x	9.3.5.5	x
TMR_DELAY	x	x	9.3.5.9
LINK_UP	9.3.5.3	x	x
LINK_DOWN	x	9.3.5.6	9.3.5.10
GM_CHANGE	-	9.3.5.7	-
SHUTDOWN	-	9.3.5.8	9.3.5.11

x = this event cannot happen in this state

- = this event is ignored in this state

9.3.5. Details of events processing

9.3.5.1. Startup of the PAAD-AE with link status down

Go to the DOWN state.

9.3.5.2. Startup of the PAAD-AE with link status up

1) Start the TMR_DELAY timer (random value between 0 and 2 seconds).
2) Go to the DELAY state.

9.3.5.3. DOWN state / LINK_UP event

1) Start the TMR_DELAY timer (random value between 0 and 4 seconds).
2) Go to the DELAY state.

9.3.5.4. WAITING state / RCV_ADP_DISCOVER event

1) Stop the TMR_ADVERTISE timer.
2) Start the TMR_DELAY timer (random value between 0 and 4 seconds).
3) Go to the DELAY state.

9.3.5.5. WAITING state / TMR_ADVERTISE event

1) Start the TMR_DELAY timer (random value between 0 and 4 seconds).
2) Go to the DELAY state.

9.3.5.6. WAITING state / LINK_DOWN event

1) Stop the TMR_ADVERTISE timer.
2) Go to the DOWN state.

9.3.5.7. WAITING state / GM_CHANGE event

1) Start the TMR_DELAY timer (random value between 0 and 4 seconds).
2) Go to the DELAY state.

9.3.5.8. WAITING state / SHUTDOWN event

1) Stop the TMR_ADVERTISE timer.
2) Send an ADP ENTITY_DEPARTING message.

9.3.5.9. DELAY state / TMR_DELAY event

1) Send an ADP ENTITY_AVAILABLE message.
2) Start the TMR_ADVERTISE timer (5 seconds).
3) Go to the WAITING state.

9.3.5.10. DELAY state / LINK_DOWN event

1) Stop the TMR_DELAY timer.
2) Go to the DOWN state.

9.3.5.11. DELAY state / SHUTDOWN event

1) Stop the TMR_DELAY timer.
2) Send an ADP ENTITY_DEPARTING message.

9.4. Listener's discovery state machine

The PAAD-AE shall implement an independent instance of the Listener's discovery state machine as described in this section, for each STREAM_INPUT of the currently set CONFIGURATION.

9.4.1. Treatment of an incoming ADP message

Upon reception of an ADP ENTITY_AVAILABLE/ENTITY_DEPARTING message, the listener shall:

1) Extract the entity_id field of the message.
2) Iterate through all the bound sinks of the listener. For each sink which is bound to the talker whose Entity ID matches the ADP message, continue processing the message according to the sink state machine below (RCV_ADP_AVAILABLE/RCV_ADP_DEPARTING event).

9.4.2. States of a sink

State	Description
TK_NOT_DISCOVERED	The talker has not been discovered.
TK_DISCOVERED	The talker has been discovered.

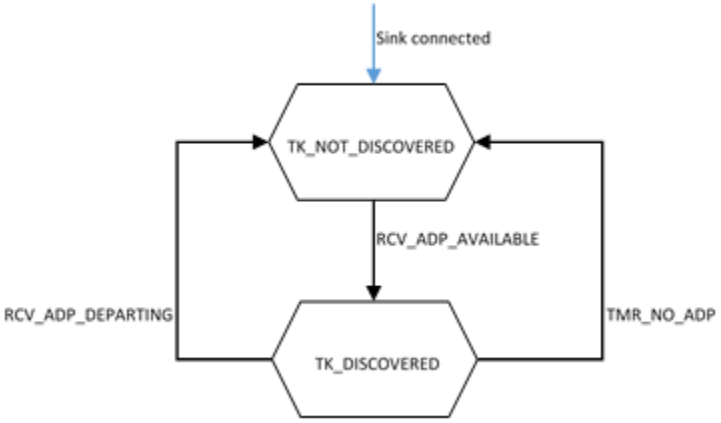
9.4.3. Events that trigger the state machine

Event	Description
RCV_ADP_AVAILABLE	Received an ADP ENTITY_AVAILABLE message from the talker.

RCV_ADP_DEPARTING	Received an ADP ENTITY_DEPARTING message from the talker.
TMR_NO_ADP	ADP timeout expired.

9.4.4. Diagram of the state machine (informative)

Please note that the following diagram is a simplified overview of the real state machine that a sink implements. The normative description of the transitions of the state machine is given in the next section.



The table below summarizes the possible transitions and indicates which section describes each transition:

	TK_NOT_DISCOVERED	TK_DISCOVERED
RCV_ADP_AVAILABLE	9.4.5.1	9.4.5.2
RCV_ADP_DEPARTING	-	9.4.5.3
TMR_NO_ADP	x	9.4.5.4

x = this event cannot happen in this state

- = this event is ignored in this state

9.4.5. Details of events processing

9.4.5.1. TK_NOT_DISCOVERED state / RCV_ADP_AVAILABLE event

1) Check the gptp_grandmaster_id and gptp_domain_number fields of the received ADP ENTITY_AVAILABLE message. If they don't match the current gPTP configuration/state of the PAAD on this port, then ignore the message and exit.
2) Save the values of the interface_index and available_index fields.
3) Set a TMR_NO_ADP timer according to the value of the valid_time field.
4) Go to the TK_DISCOVERED state and trigger the sink's connection state machine with the EVT_TK_DISCOVERED event.

9.4.5.2. TK_DISCOVERED state / RCV_ADP_AVAILABLE event

1) Check the interface_index field. If it is not equal to the saved value then ignore the message and exit.
2) Check the available_index field. If it is less than or equal to the value in the last received ADP ENTITY_AVAILABLE message then:
    a) Trigger the sink's connection state machine with the EVT_TK_DEPARTED event.
    b) Check the gptp_grandmaster_id and gptp_domain_number fields; if they don't match the current gPTP configuration/state of the PAAD on this port then stop the TMR_NO_ADP timer and go to the TK_NOT_DISCOVERED state without executing the following steps.
    c) Trigger the sink's connection state machine with the EVT_TK_DISCOVERED event.
3) Note the value of the available_index field and reset the TMR_NO_ADP timer according to the value of the valid_time field.

9.4.5.3. TK_DISCOVERED state / RCV_ADP_DEPARTING event

1) Check the interface_index field. If it is not equal to the saved value then ignore the message and exit.
2) Stop the TMR_NO_ADP timer.
3) Go to the TK_NOT_DISCOVERED state and trigger the sink's connection state machine with the EVT_TK_DEPARTED event.

9.4.5.4. TK_DISCOVERED state / TMR_NO_ADP event

Go to the TK_NOT_DISCOVERED state and trigger the sink's connection state machine with the EVT_TK_DEPARTED event.

Annex A (informative) Examples of AEM models

This Annex describes a few AEM models associated with concrete products.

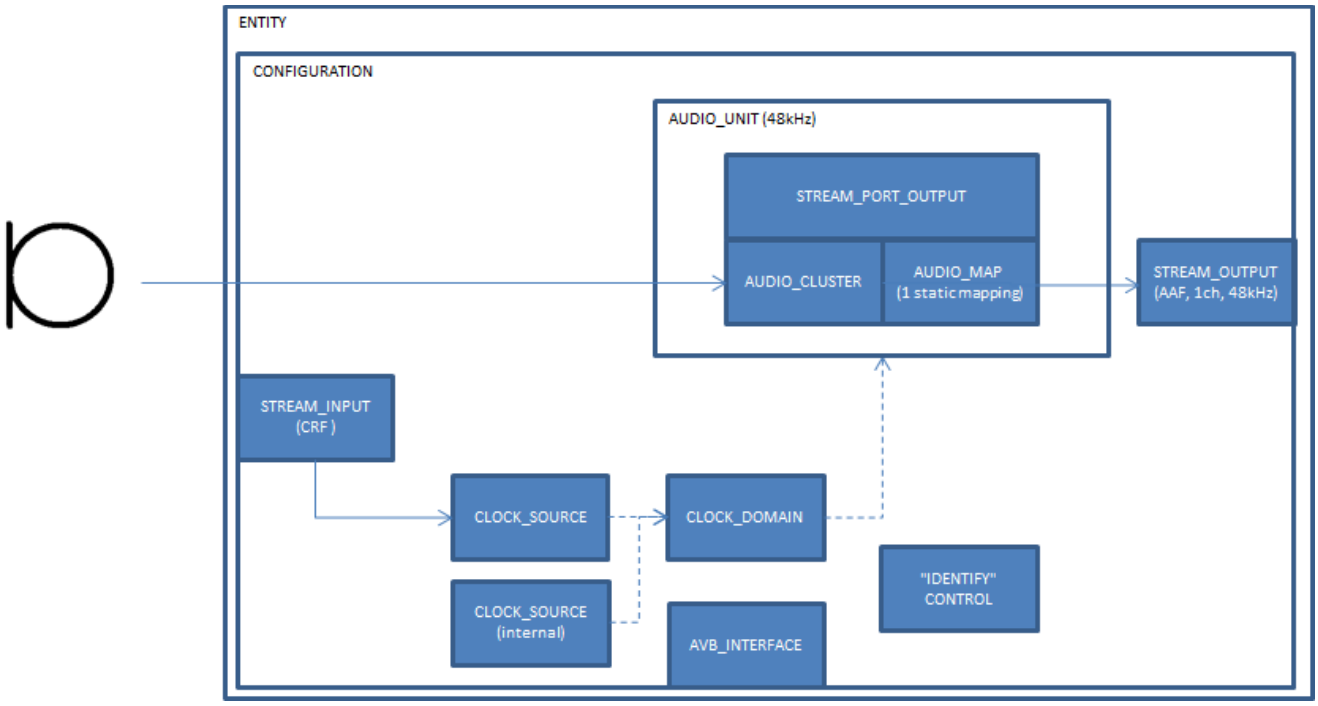
Please note that the content of this Annex is informative only. If the description in the informative chapters is not as in the normative chapters, the normative chapters always take precedence over the informative chapters.

A.1 Microphone

This is a simple microphone. From the external point of view, the device is composed of a microphone, an Ethernet connector and a LED. Its main usage is to stream the captured audio on an AVB network through a 48kHz 1-channel AAF stream.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting only one format: the Avnu CRF Media clock stream format, as specified in [AVNU.IO.MEDIACLOCKING].
- There is one STREAM_OUTPUT supporting only one format: the Avnu AAF Media Stream format at 48kHz with 1 channel.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input CRF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_OUTPUT, itself composed of one single-channel AUDIO_CLUSTER and one single-entry AUDIO_MAP. The potential signal processing made by the device is not represented by the Entity model. The single audio channel is directly and statically mapped to the AAF output stream.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.

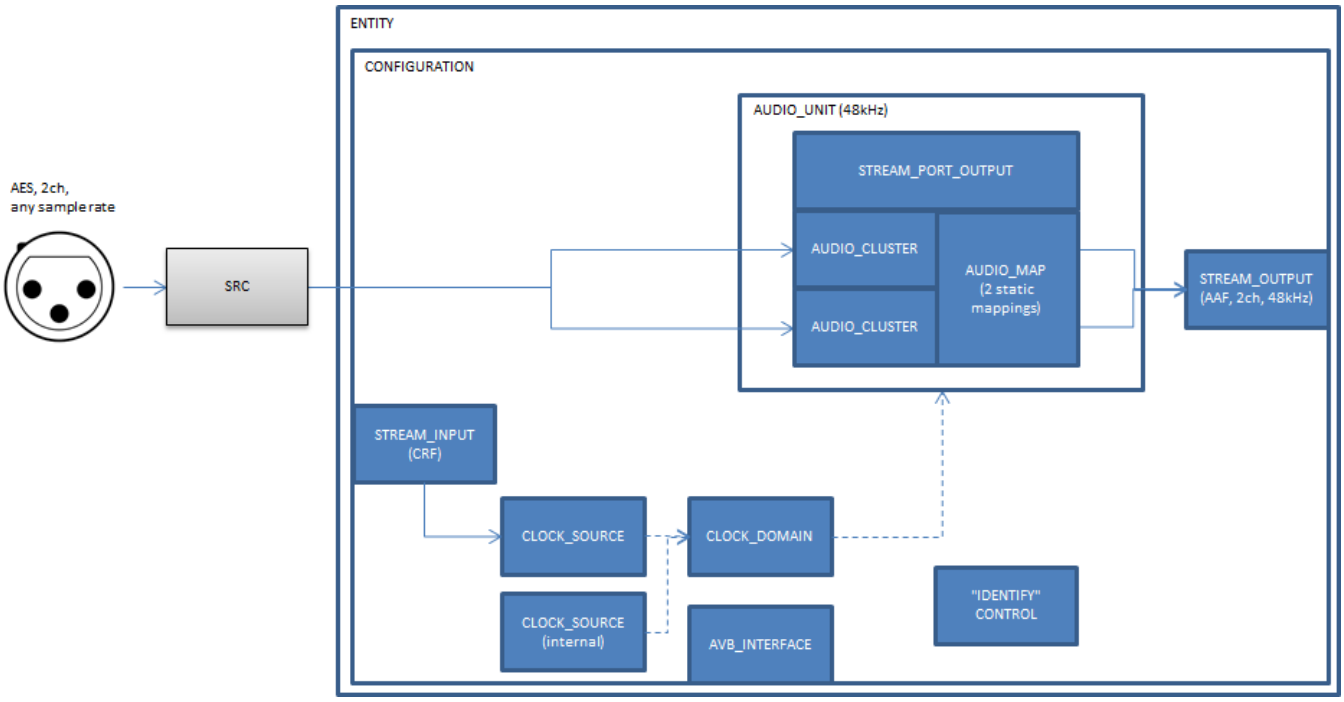


A.2 AES Break-In box with SRC

This is a simple AES Break-In box with Sample Rate Conversion. From the external point of view, the device is composed of an XLR connector for AES/EBU input, an Ethernet connector and a LED. Its main usage is to convert the 2-channel AES audio input (whatever the sampling rate) into a 48-kHz 2-channel AAF stream.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting only one format: the Avnu CRF Media clock stream format, as specified in [AVNU.IO.MEDIACLOCKING].
- There is one STREAM_OUTPUT supporting only one format: the Avnu AAF Media Stream format at 48kHz with 2 channels.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input CRF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_OUTPUT, itself composed of two single-channel AUDIO_CLUSTERs and one 2-entry AUDIO_MAP. The potential signal processing made by the device is not represented by the Entity model, in particular the sample rate conversion of the AES signal and the separation into two channels. The two audio channels are directly and statically mapped to the two channels of the AAF output stream.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.

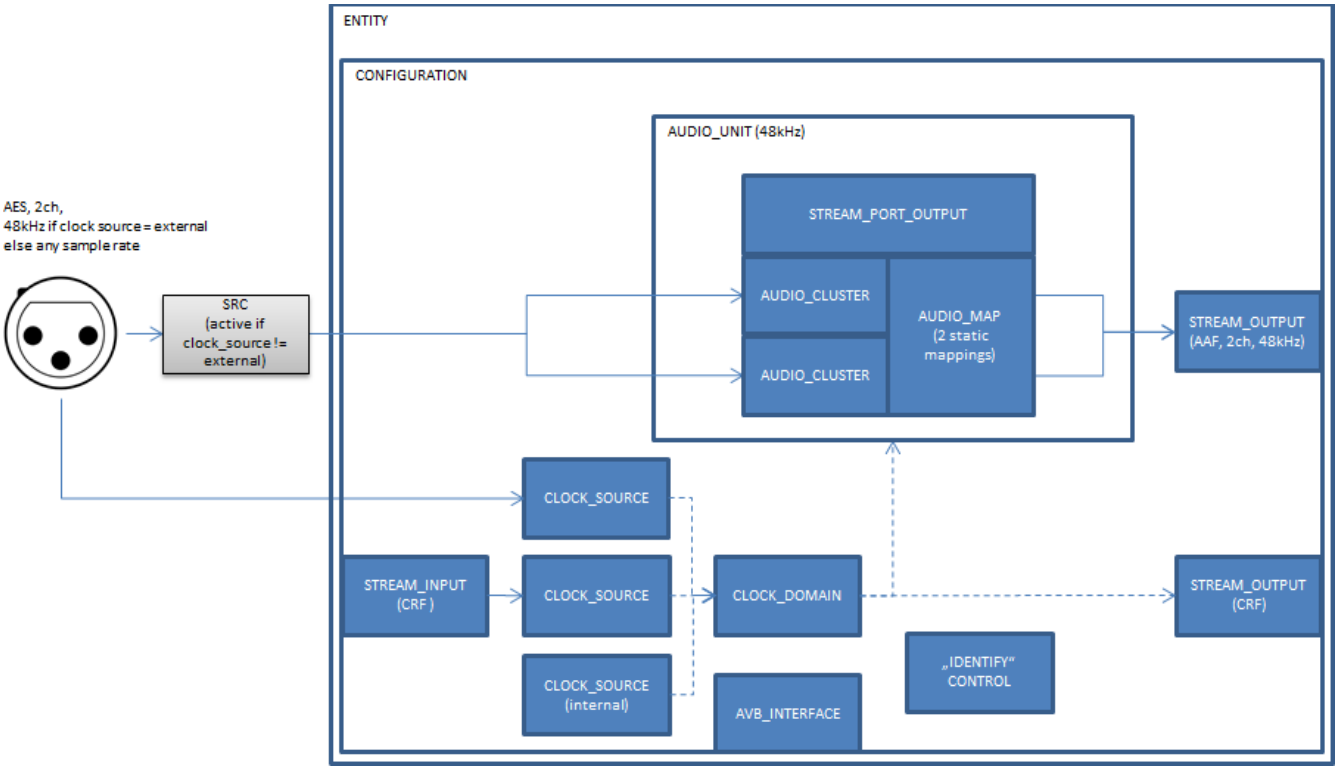


A.3 AES Break-In box with media clock master capability

This is an AES Break-In box with Sample Rate Conversion and the ability to act as a media clock master. From the external point of view, the device is composed of an XLR connector for AES/EBU input, an Ethernet connector and a LED. Its main usage is to convert the 2-channel AES audio input (whatever the sampling rate) into a 48-kHz 2-channel AAF stream and also convert the AES clock input into a separate CRF stream.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting only one format: the Avnu CRF Media clock stream format, as specified in [AVNU.IO.MEDIACLOCKING].
- There are two STREAM_OUTPUTs; the first one carries audio and supports only the Avnu AAF Media Stream format at 48kHz with 2 channels. The second one carries clock and supports only the Avnu CRF Media clock stream format.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and three CLOCK_SOURCEs. The device is able to either use its internal media clock, or derive it from the input CRF stream, or derive it from the input AES signal. Please note that in the latter case, the SRC at the AES input is disabled and all the unit is synchronized on the AES input.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_OUTPUT, itself composed of two single-channel AUDIO_CLUSTERs and one 2-entry AUDIO_MAP. The potential signal processing made by the device is not represented by the Entity model, in particular the sample rate conversion of the AES signal and the separation into two channels. The two audio channels are directly and statically mapped to the two channels of the AAF output stream.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.


ENTITY
CONFIGURATION
AUDIO_UNIT (48kHz)
STREAM_PORT_OUTPUT
AUDIO_CLUSTER
AUDIO_CLUSTER
AUDIO_MAP
(2 static
mappings)
AES, 2ch,
48kHz if clock source = external
else any sample rate
SRC
(active if
clock_source !=
external)
STREAM_OUTPUT
(AAF, 2ch, 48kHz)
CLOCK_SOURCE
STREAM_INPUT
(CRF)
CLOCK_SOURCE
CLOCK_DOMAIN
STREAM_OUTPUT
(CRF)
CLOCK_SOURCE
(internal)
AVB_INTERFACE
„IDENTIFY"
CONTROL

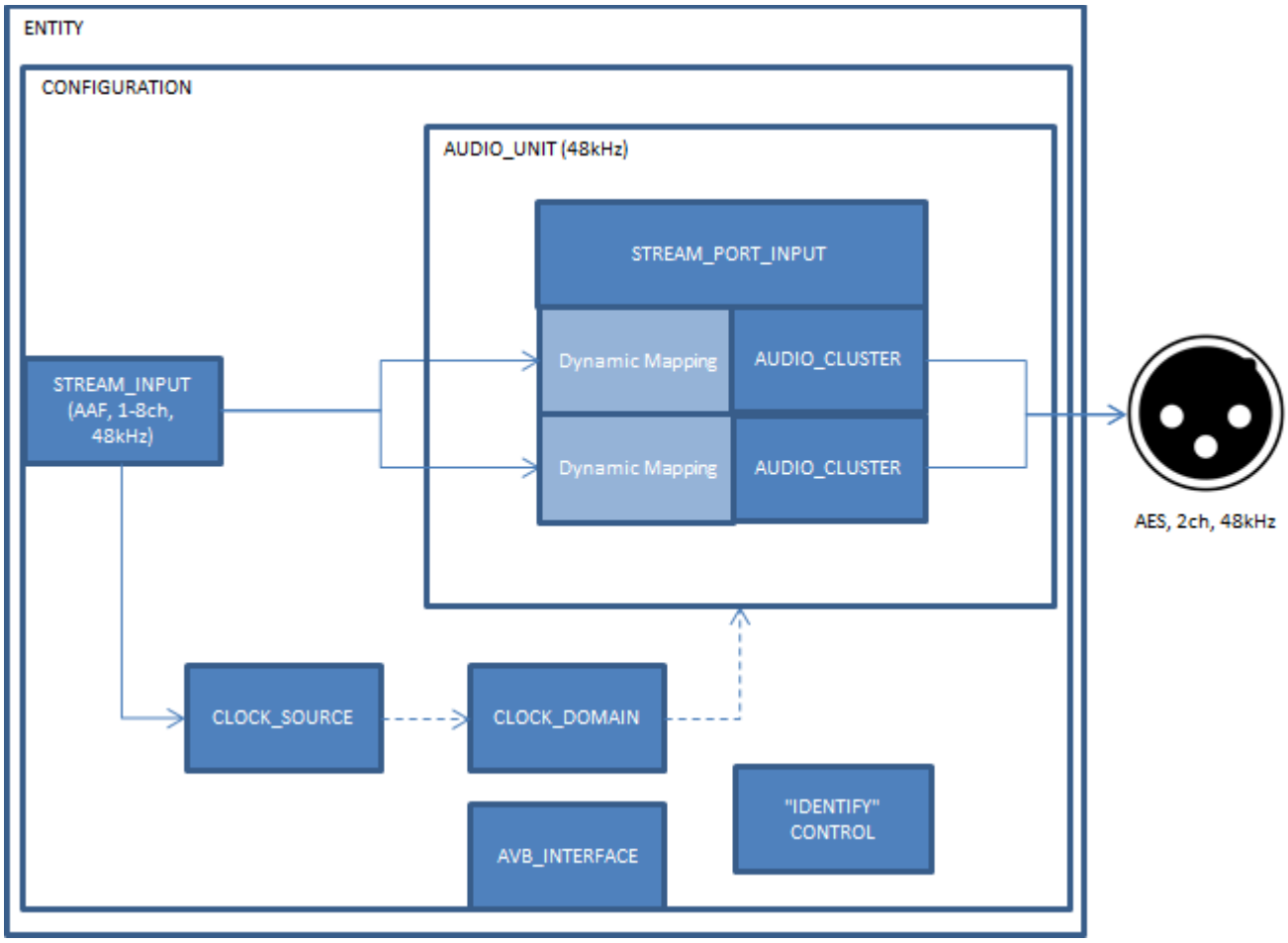
-

A.4 AES Break-Out box - simple

This is a simple AES Break-Out box. From the external point of view, the device is composed of an XLR connector for AES/EBU output, an Ethernet connector and a LED. Its main usage is to convert the input 48kHz AAF stream (1 to 8 channels) to a 48kHz 2-channel AES audio output.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting a range of formats: Avnu AAF Media stream format, 48kHz, from 1 channel to 8 channels.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input AAF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT, itself composed of two single-channel AUDIO_CLUSTERs. The two clusters channels are mapped from the channels of the input stream using dynamic mappings (there is no AUDIO_MAP descriptor). The potential signal processing made by the device is not represented by the Entity model.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.

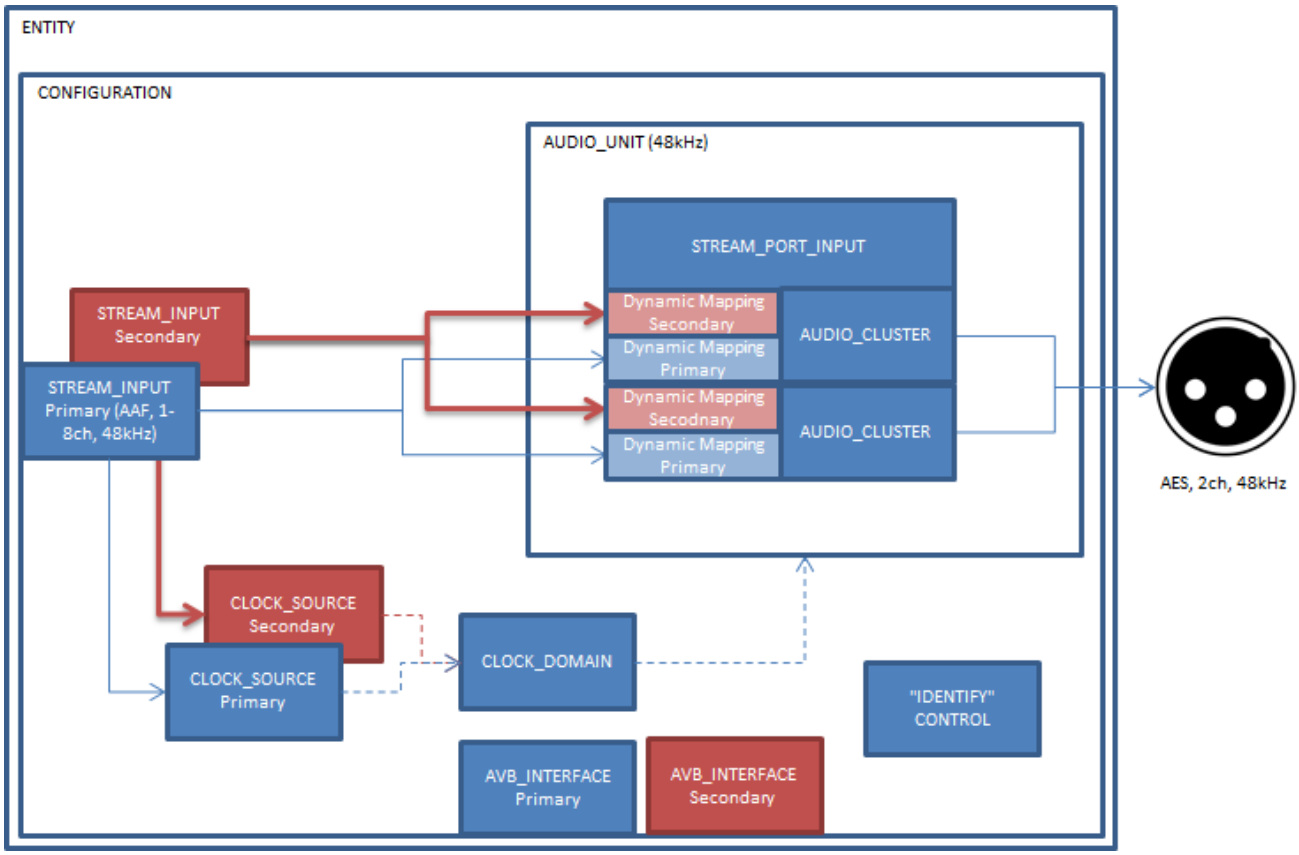


A.5 AES Break-Out box - Redundant

This is a redundant version of the previous AES Break-Out box. From the external point of view, the device is composed of an XLR connector for AES/EBU output, two Ethernet connectors and a LED. Its main usage is to convert the input 48kHz AAF stream (1 to 8 channels) to a 48kHz 2-channel AES audio output. Redundancy is provided thanks to the two Ethernet ports; one is plugged to the primary network and the other to the secondary network.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is a pair of two redundant STREAM_INPUTs supporting a range of formats: Avnu AAF Media stream format, 48kHz, from 1 channel to 8 channels.
- There is two AVB_INTERFACEs describing the two Ethernet ports of the device.
- There is one CLOCK_DOMAIN and three CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from one of the two input AAF streams.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT, itself composed of two single-channel AUDIO_CLUSTERs. The two clusters channels are mapped from the channels of the input streams using dynamic mappings (there is no AUDIO_MAP descriptor). Each cluster's channel has a mapping to the primary input stream and the equivalent mapping to the secondary. The potential signal processing made by the device is not represented by the Entity model.
- There is also an "IDENTIFY" CONTROL which is used to represent the state of the identifying LED which is present on the device.

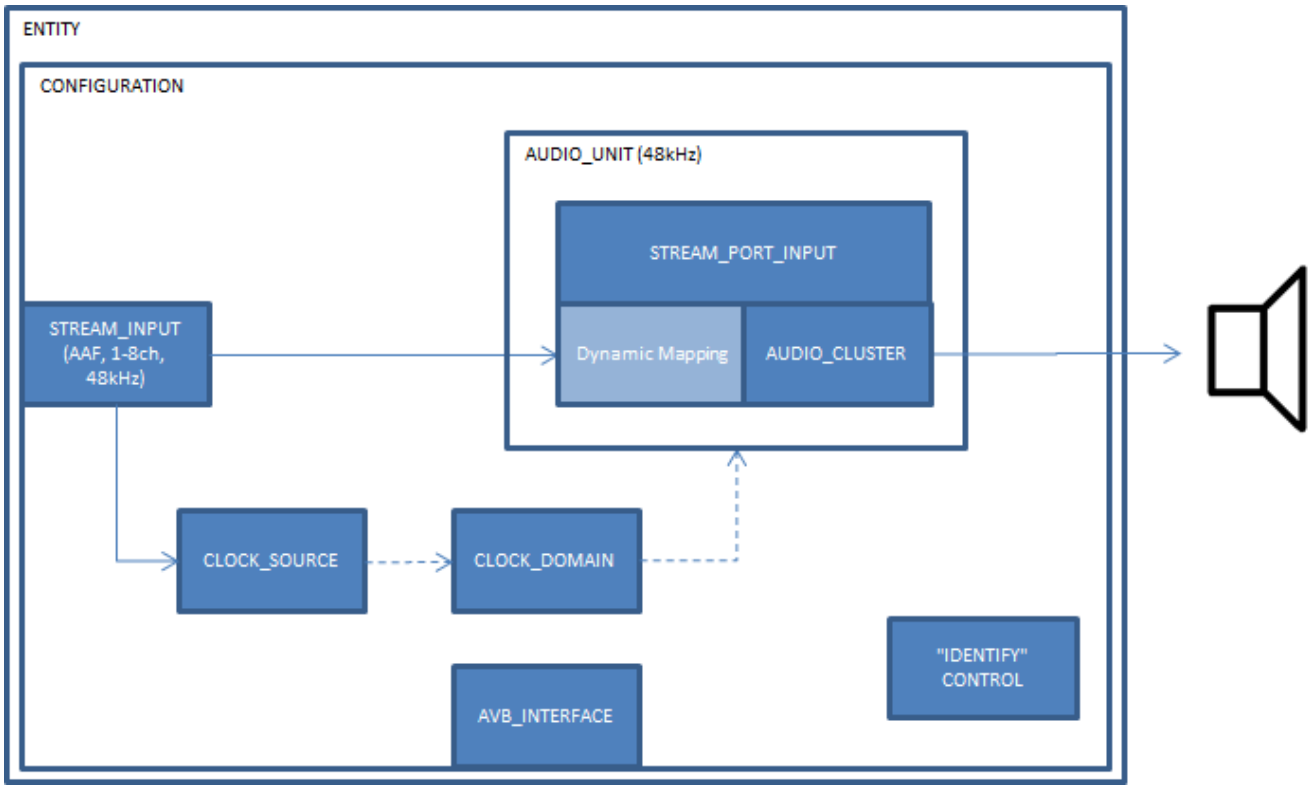


A.6 Simple speaker

This is a simple AVB speaker. From the external point of view, the device is composed of a speaker, an Ethernet connector and a LED. Its main usage is to render one audio channel from an input 48kHz AAF stream (1 to 8 channels).

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting a range of formats: Avnu AAF Media stream format, 48kHz, from 1 channel to 8 channels.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input AAF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT, itself composed of one single-channel AUDIO_CLUSTER. The cluster's channel is mapped from one channel of the input stream using dynamic mappings (there is no AUDIO_MAP descriptor). The potential signal processing made by the device is not represented by the Entity model.
- There is also an "IDENTIFY" CONTROL which is used to represent the state of the identifying LED which is present on the device.

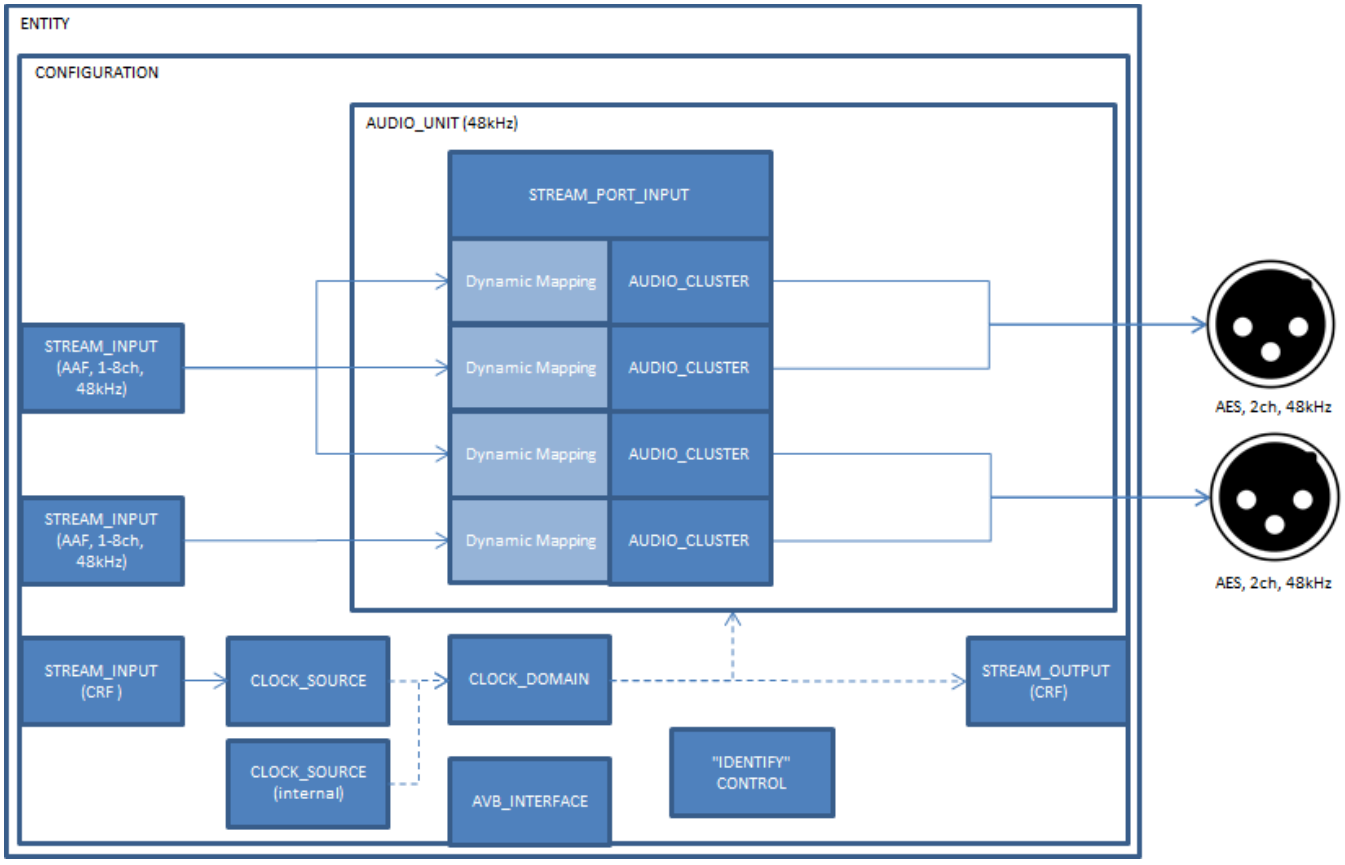


A.7 AES Break-Out box - advanced

This is an advanced AES Break-Out box. From the external point of view, the device is composed of two XLR connectors for AES/EBU outputs, an Ethernet connector and a LED. Its main usage is to convert the input 48kHz AAF streams (1 to 8 channels) to two 48kHz 2-channel AES audio outputs.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There are three STREAM_INPUTs; the two first ones carry audio and support the Avnu AAF Media stream format, 48kHz, from 1 channel to 8 channels. The third one carries clock and supports only the Avnu CRF Media clock stream format.
- There is one STREAM_OUTPUT supporting only one format: the Avnu CRF Media clock stream format, as specified in [AVNU.IO.MEDIACLOCKING].
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input CRF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT, itself composed of four single-channel AUDIO_CLUSTERs. The four clusters channels are mapped from the channels of the input streams using dynamic mappings (there is no AUDIO_MAP descriptor). The potential signal processing made by the device is not represented by the Entity model.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.

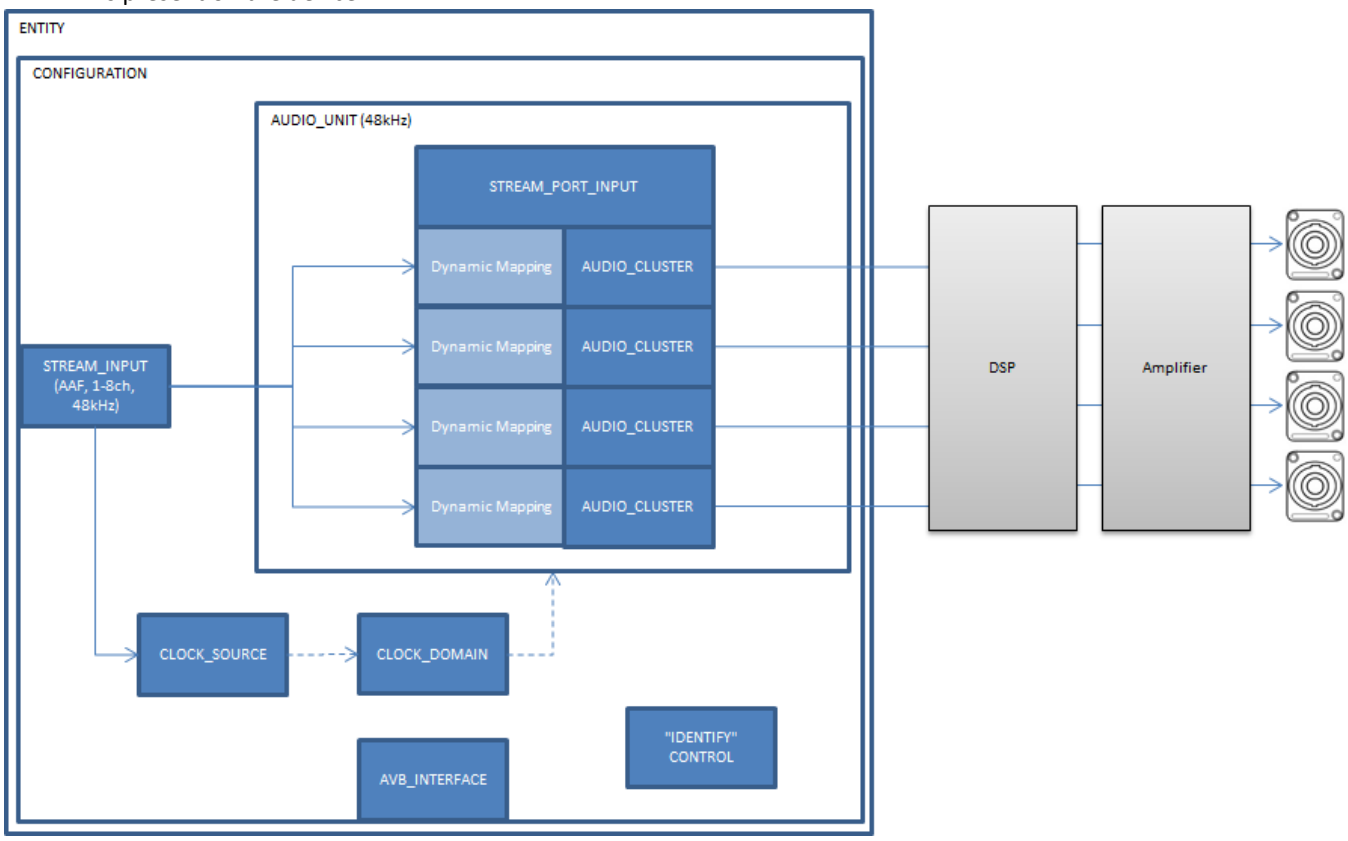

ENTITY
CONFIGURATION
AUDIO_UNIT (48kHz)
STREAM_PORT_INPUT
Dynamic Mapping
AUDIO_CLUSTER
Dynamic Mapping
AUDIO_CLUSTER
Dynamic Mapping
AUDIO_CLUSTER
Dynamic Mapping
AUDIO_CLUSTER
STREAM_INPUT (AAF, 1-8ch, 48kHz)
STREAM_INPUT (AAF, 1-8ch, 48kHz)
STREAM_INPUT (CRF)
CLOCK_SOURCE
CLOCK_SOURCE (internal)
CLOCK_DOMAIN
AVB_INTERFACE
"IDENTIFY" CONTROL
STREAM_OUTPUT (CRF)
AES, 2ch, 48kHz
AES, 2ch, 48kHz

A.8 Simple amplifier

This is a simple 4-channel amplifier. From the external point of view, the device is composed of four XLR connectors for amplified analog outputs, an Ethernet connector and a Display. Its main usage is to get four audio channels from the input AAF stream (48kHz or 96kHz, 1 to 8 channels), process and amplify them, and output them to the Loudspeakers.

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There is one STREAM_INPUT supporting a range of formats: the Avnu AAF Media Stream format at 48 kHz or 96kHz, with 1 to 8 channels.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input AAF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 96kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT, itself composed of four single-channel AUDIO_CLUSTERs. The STREAM_PORT_INPUT is equipped with a Synchronous Sample Rate Converter capable of converting audio data at 96kHz from the input stream to 96kHz to process them inside the AUDIO_UNIT. The four clusters channels are mapped from the channels of the input stream using dynamic mappings (there is no AUDIO_MAP descriptor). The potential signal processing and amplification made by the device is not represented by the Entity model.
- There is also an “IDENTIFY” CONTROL which is used to represent the blinking state of the identifying Display which is present on the device.

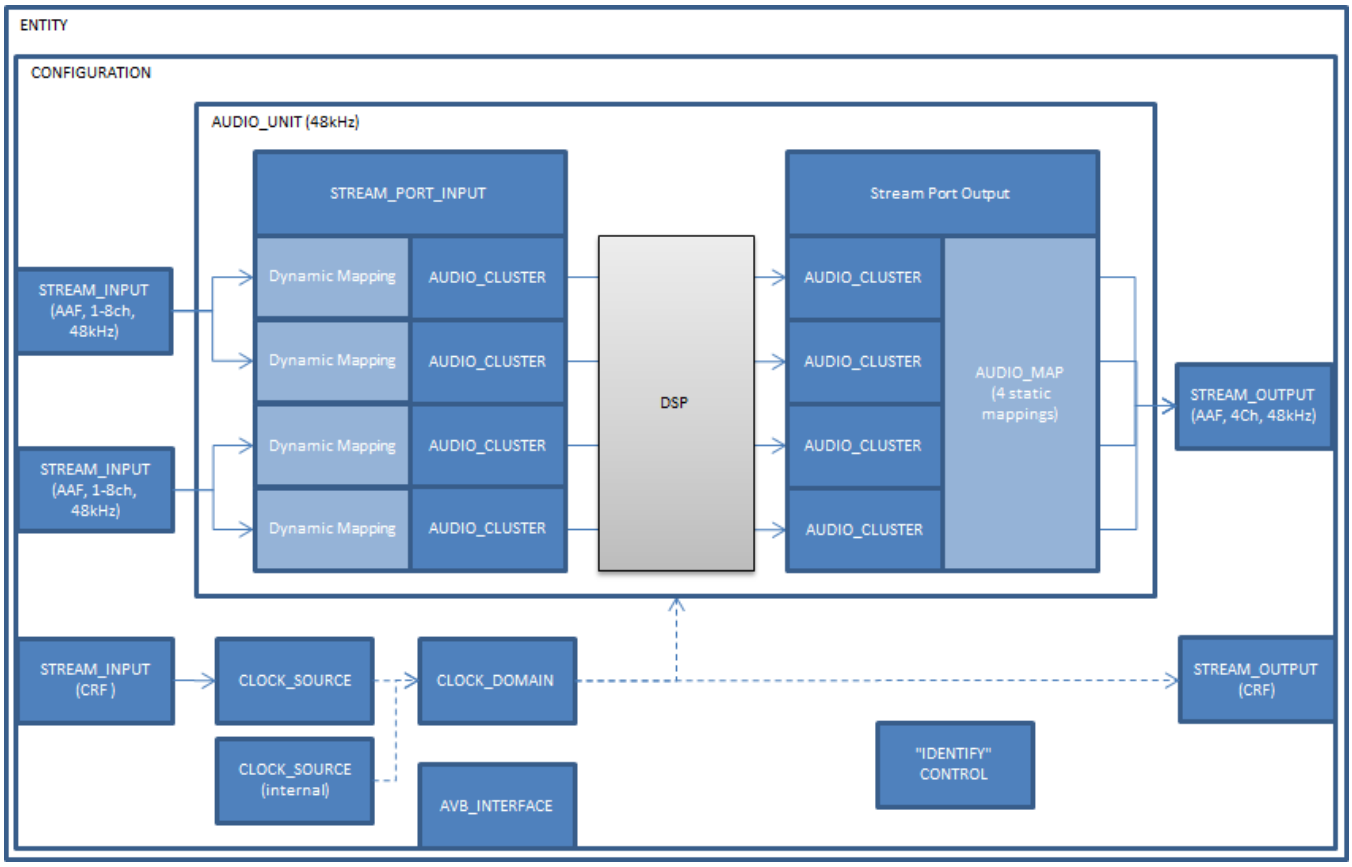


A.9 DSP Unit

This is a pure 4x4 DSP unit. From the external point of view, the device is composed of an Ethernet connector and a LED. Its main usage is to get four audio channels from the input AAF streams (48kHz, 1 to 8 channels), mix them, and output them to one output AAF stream (1 to 4 channels).

Below is a description of its internal Entity model:

- There is one CONFIGURATION.
- There are three STREAM_INPUTs; the two first ones carry audio and support the Avnu AAF Media stream format, 48kHz, from 1 channel to 8 channels. The third one carries clock and supports only the Avnu CRF Media clock stream format.
- There are two STREAM_OUTPUTs; the first one carries audio and supports a range of formats: the Avnu AAF Media Stream format at 48kHz with 1 to 4 channels. The second one carries clock and supports only the Avnu CRF Media clock stream format.
- There is one AVB_INTERFACE describing the single Ethernet port of the device.
- There is one CLOCK_DOMAIN and two CLOCK_SOURCEs. The device is able to either use its internal media clock or derive it from the input CRF stream.
- There is one AUDIO_UNIT clocked by the CLOCK_DOMAIN described above and working at 48kHz. The AUDIO_UNIT is composed of one STREAM_PORT_INPUT and one STREAM_PORT_OUTPUT.
    - The STREAM_PORT_INPUT is composed of four single-channel AUDIO_CLUSTERs. The four clusters channels are mapped from the channels of the input stream using dynamic mappings (there is no AUDIO_MAP descriptor).
    - The STREAM_PORT_OUTPUT is composed of four single-channel AUDIO_CLUSTERs. The four clusters channels are mapped to the channels of the output stream using dynamic mappings (there is no AUDIO_MAP descriptor).
- The potential signal processing made by the device is not represented by the Entity model.
- There is also an “IDENTIFY” CONTROL which is used to represent the state of the identifying LED which is present on the device.


ENTITY
CONFIGURATION
AUDIO_UNIT (48kHz)
STREAM_PORT_INPUT
Stream Port Output
STREAM_INPUT (AAF, 1-8ch, 48kHz)
STREAM_INPUT (AAF, 1-8ch, 48kHz)
Dynamic Mapping
AUDIO_CLUSTER
DSP
AUDIO_MAP (4 static mappings)
STREAM_OUTPUT (AAF, 4Ch, 48kHz)
STREAM_INPUT (CRF)
CLOCK_SOURCE
CLOCK_SOURCE (internal)
CLOCK_DOMAIN
AVB_INTERFACE
"IDENTIFY" CONTROL
STREAM_OUTPUT (CRF)