Common Ember+ Sub-Tree for Receivers and Senders in SMPTE ST2022-6 and ST 2110 Endpoints

VERSION 1.3

The propose of this document is to provide a lightweight straightforward interface for control systems to create and manage multicast connections between SMPTE ST 2110 devices, or SMPTE ST 2022-6 devices. The interface described herein uses the well-documented Ember+ protocol to manage information flow between the controller and the device; a specific sub-tree structure is defined which can be common across devices. The controller uses this sub-tree to understand the sender and receiver configurations and to enable the control system to direct the receivers to the correct signals, through the exchange of properly formatted SDP values.

Ember + descriptions and documentations, and licensing information (it's free) can be found at https://github.com/Lawo

This document is presently a draft for comments – but the intent is that the common interface sub-tree and a guide to its use will be maintained at the above Github and change-controlled there.

Scope of proposal

The scope of this interface sub-tree is to allow for the exchange of SDP information via Ember+ from a transmitter to receiver for SMPTE ST 2110 and/or ST 2022-6 IP flows. It is also to be used to expose the receiver structure in term of signal capacity and structure of video, audio and metadata. This will enable a control system to know the signal information of the various video & audio transmitters within a device (from their exposed SDPs) and to convey that information to other receivers in an organized way. This interface enables the switching signals by switching SDP values, enabling transition from one stream to another. This interface also provides limited status of the state of a receiver and whether the streams are present or impaired.

Out of Scope

This proposed interface does not replace the normal configuration, status, alarm, or user-facing interfaces which any product has. Setup and configuration of the device, outside of receiver SDP values, remains the responsibility of existing device interfaces. This proposed interface only augments existing capability.

This proposed interface also does not control or manage network switches, though many systems will be built on the assumption that receivers will utilize IGMP in order to join multicast streams in response to SDP values written through this interface.

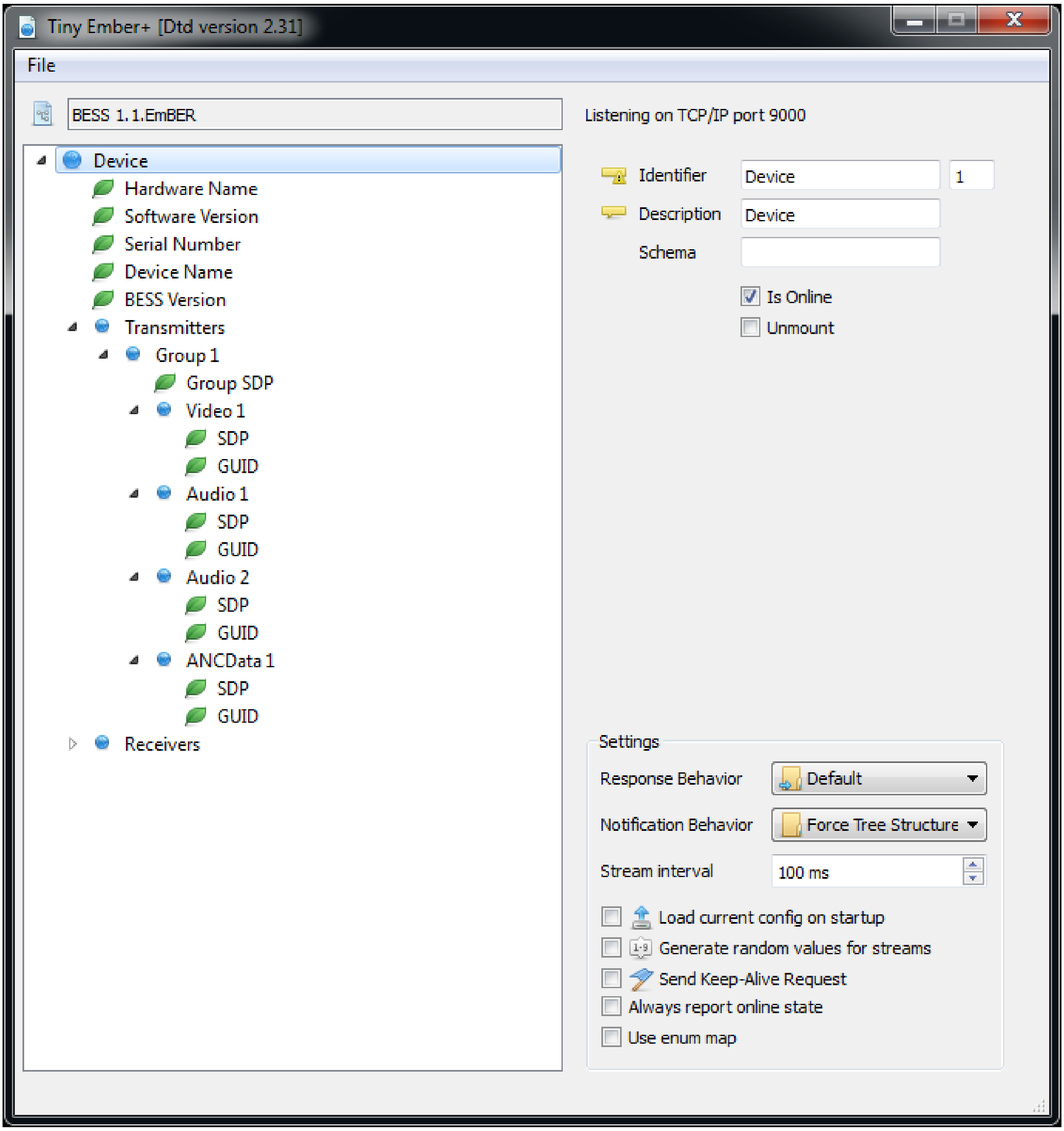
Relationship to AMWA IS04 and IS05

This proposed interface is designed specifically to co-exist well with AMWA NMOS in systems which implement NMOS.

Sub-Tree Details

The subtree is composed of a “Transmitters” branch and a “Receivers” branch. The Receivers branch is shown below. The Transmitters branch and Receivers branch shall each be children of the device's

Ember+ root. Other child nodes of the Ember+ root may exist.



BESS 1.1.EmBER

Listening on TCP/IP port 9000

- Device
  - Hardware Name
  - Software Version
  - Serial Number
  - Device Name
  - BESS Version
  - Transmitters
    - Group 1
      - Group SDP
      - Video 1
        - SDP
        - GUID
      - Audio 1
        - SDP
        - GUID
      - Audio 2
        - SDP
        - GUID
      - ANCData 1
        - SDP
        - GUID
  - Receivers

Identifier: Device 1

Description: Device

Schema:

☑ Is Online

☐ Unmount

Settings

Response Behavior: Default

Notification Behavior: Force Tree Structure

Stream interval: 100 ms

☐ Load current config on startup

☐ Generate random values for streams

☐ Send Keep-Alive Request

☐ Always report online state

☐ Use enum map

Device

Schema Identifier: com.lawo.emberplus.nmos-interop.device

Hardware Name

Identifier = “HardwareName”

Type: String

Access: Read Only

Deivce name that includes a manufacterer identification

Software Version

Identifier = “SoftwareVersion”

Type: String
Access: Read Only
Version ID of Current Software

Serial Number

Identifier = “SerialNumber”

Type: String
Access: Read Only
Serial Number of Device

Device Name

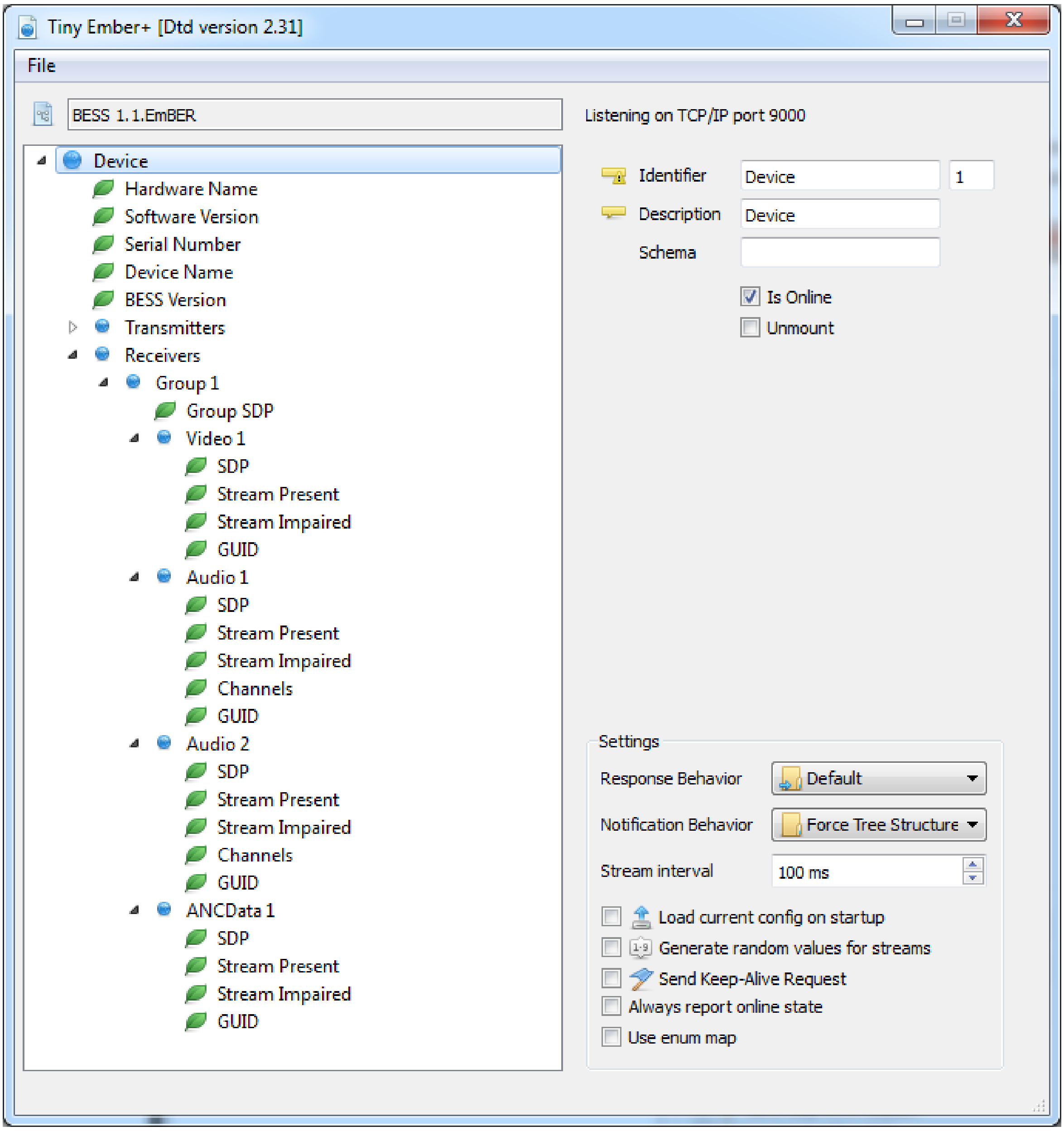
Identifier = “DeviceName”

Type: String
Access: Read Only
User definable system name

BESS Version

Identifier = “BessVersion”

Type: String
Access: Read Only
Version of BESS Schema



Tiny Ember+ [Dtd version 2.31]

File

BESS 1.1.EmBER

Listening on TCP/IP port 9000

- Device
  - Hardware Name
  - Software Version
  - Serial Number
  - Device Name
  - BESS Version
  - Transmitters
  - Receivers
    - Group 1
      - Group SDP
      - Video 1
        - SDP
        - Stream Present
        - Stream Impaired
        - GUID
      - Audio 1
        - SDP
        - Stream Present
        - Stream Impaired
        - Channels
        - GUID
      - Audio 2
        - SDP
        - Stream Present
        - Stream Impaired
        - Channels
        - GUID
      - ANCData 1
        - SDP
        - Stream Present
        - Stream Impaired
        - GUID

Identifier: Device 1

Description: Device

Schema:

☑ Is Online

□ Unmount

Settings

Response Behavior: Default

Notification Behavior: Force Tree Structure

Stream interval: 100 ms

□ Load current config on startup

□ Generate random values for streams

□ Send Keep-Alive Request

□ Always report online state

□ Use enum map

Syntax Elements in the Receiver Subtree

SDP

Identifier = “SDP”

Type: String

Access: Read/Write

SDP values as defined by SMPTE ST 2110-20 (for video), 2110-30/31 (for audio), or 2110-40 (for ANC data). For the case of SMPTE 2022-6, the SDP definition of SMPTE 2022-8 is used. Example SDP values are shown later in the document.

At the “Group” node, the SDP can contain multiple section for video, audio, and data streams. At the “Video”, “Audio” or “Data” node level, the SDP value is expected to be a single-stream SDP (or a duplicated pair of streams for redundancy purposes).

GUID

Identifier = "GUID"

Type: String

Access: Read Only

As defined in AMWA NMOS. GUIDs are exposed through this interface in order to identify receivers which might also be discovered through the AMWA "NMOS" IS04 specification. The Group GUID does not have equivalency in NMOS and is for future enhancement.

Stream Present

Identifier = "StreamPresent"

Type: Integer

Enumeration: 0=none, 1=primary, 2=secondary, 3=both

Access: Read Only

- both == the receiver is receiving (perhaps with some errors) both the primary and secondary network streams
- primary == the receiver is receiving (perhaps with errors) the primary but NOT the secondary. Receivers which are only capable of receiving one stream (no protection implemented) would signal primary when getting the one stream that they can.
- secondary == the receiver is receiving (perhaps with errors) the secondary but NOT the primary
- none == the receiver is not receiving either stream

Stream Impaired

Identifier = "StreamImpaired"

Type: Boolean

Access: Read Only

- true if the stream contains uncorrectable errors even after applying the packets from the protect stream when available.
- If Stream Present = none, then Stream Impaired should also be true.

Channels (Audio Only)

Identifier = "Channels"

Type: Integer

Access: Read Only

- Indicates the number of mono channels the audio stream receiver is setup to receive

Stream Switching

In order to switch an individual endpoint to a new stream, the control system writes a new SDP value to the SDP variable of the desired endpoint. The endpoint parses the new SDP value.

- If the SDP parses correctly and is appropriate to the endpoint
  - the endpoint will begin the process of transitioning to the new stream
  - the endpoint will reflect its current status with “Stream Present” and/or “Stream impaired”
  - and return the unchanged SDP to the control system after the transitioning has happened.
- If the SDP does not parse correctly or otherwise cannot be used by the endpoint (for example a media type which the endpoint cannot consume), the endpoint will take the following actions:
  - The endpoint will disconnect the previous stream.
  - The endpoint will return the unchanged SDP as received.
  - The endpoint will reflect its current status with “Stream Present” and/or “Stream impaired”

In order to “disconnect” an individual endpoint (tell it to receive no stream from the network), the control system writes an empty string to the SDP variable of the desired endpoint. This operation must always be successful.

  - The endpoint will return an empty SDP.
  - The endpoint will reflect its current status with “Stream Present” and/or “Stream impaired”

If deemed necessary, the device can implement an additional field to reflect its interpretation of the SDP. Yet, the actual SDP field must reflect the SDP as transmitted by e.g. the control system.

Devices supporting the group SDP operation, every essence SDP will be handled as mentioned above. The status messages for the grouped SDP is not defined yet.

Subtree Node Numbering/Naming Methodology

Ember+ tree nodes are identified by names. The names of the nodes in this tree are mechanically generated in order to reveal the relationships of the video and audio receivers. The groups will be numbered sequentially within the transmitter tree starting with “Group 1”, and similarly within the receiver tree starting with “Group 1”. Within each group, the video, audio, and data nodes are numbered sequentially starting with 1 inside each group.

PLEASE NOTE:

It is stipulated that all tree labels be left as described. Transmitters, Receivers, Group, Video, Audio, ANCData should not be changed to a descriptor or other nomenclature.

Example

- Group 1
  - Video 1
  - Audio 1
  - Audio 2
  - Audio 3
  - ANCData 1

- Group 2
  - Video 1
  - Audio 1
  - Audio 2
  - Audio 3
  - Audio 4
  - ANCData 1
  - ANCData 2
- Group 3
  - Video 1
  - Audio 1
  - Audio 2
  - Audio 3
  - ANCData 1
  - ANCData 2

Addendum – Describing Ember+ acknowledgment and optional MAC field

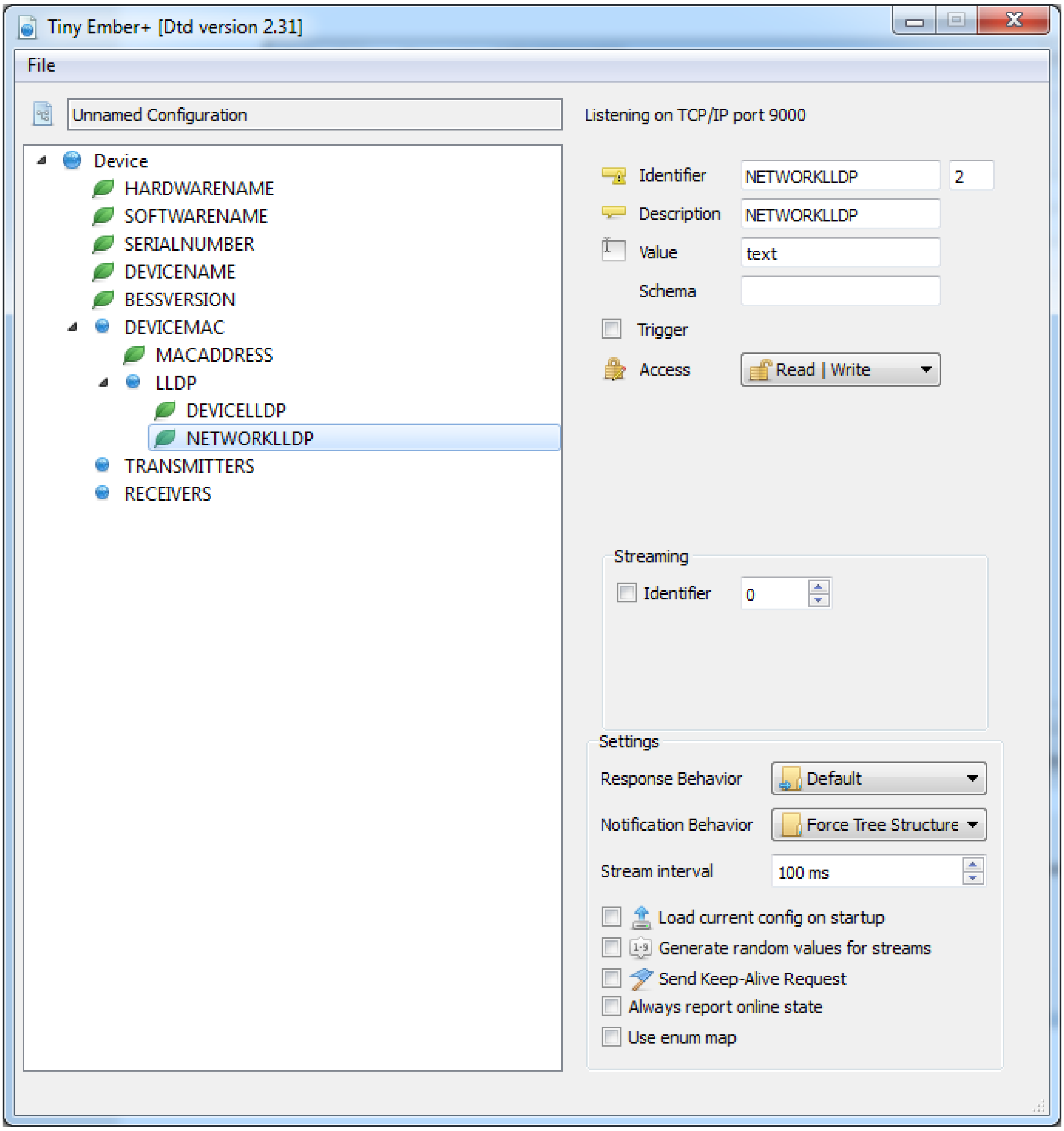
Ember+ acknowledgment section

A provider shall acknowledge that a value change e.g. to the SDP has been accepted by the provider by returning the original and unchanged value back to the consumer as soon as the value change becomes effective. At any time, the value returned or sent to a consumer must be valid by the same rules as described to the consumer by the provider (e.g. must be within specified minimum and maximum range). If the value is not accepted, the device may remain silent.

- This acknowledgment is also described in ember+ documentation.

Optional MAC field for dynamic devices

In order to accommodate devices that can move physical location and change IP addresses, a fixed address is needed to track the devices. The use of the MAC (Media Access Control) is recommended for this operation.



Unnamed Configuration

Listening on TCP/IP port 9000

- Device
  - HARDWARENAME
  - SOFTWARENAME
  - SERIALNUMBER
  - DEVICENAME
  - BESSVERSION
  - DEVICEMAC
    - MACADDRESS
    - LLDP
      - DEVICELLDP
      - NETWORKLLDP
  - TRANSMITTERS
  - RECEIVERS

Identifier: NETWORKLLDP 2

Description: NETWORKLLDP

Value: text

Schema:

□ Trigger

Access: Read | Write

Streaming

□ Identifier 0

Settings

Response Behavior: Default

Notification Behavior: Force Tree Structure

Stream interval: 100 ms

□ Load current config on startup

□ Generate random values for streams

□ Send Keep-Alive Request

□ Always report online state

□ Use enum map

The MAC for devices may be ether be a one or more simple parameter entries, where the identifier is a MAC address and the read only value is a descriptive name of the devices e.g. port associated with the MAC. Or, can be sub node entries where the identifier is again the MAC address and contained within this node the device and network LLDP entries are contained:

MAC Address

Identifier = "<MacAddress>"

Type: STRING

Access: Read Only

Descriptive name of port associated with above MAC address.

- The MAC addresses use colon delineation
- Example = "1A:23:45:87:C8:D9"

or

MAC Address

Identifier = "<MacAddress>"

Type: Node

- Example = "1A:23:45:87:C8:D9"

Device LLDP

Identifier = "DeviceLLDP"

Type: STRING

Access: Read Only

- Must follow LLDP mandatory message structure described in IEEE 802.1AB
- Device LLDP must contain the PORT ID and CHASSIS ID of the broadcast device

Network LLDP

Identifier = "NetworkLLDP"

Type: STRING

Access: Read Only

- Must follow LLDP mandatory message structure described in IEEE 802.1AB
- Device LLDP must contain the PORT ID and CHASSIS ID of the network neighbor device (port the device is connected to).

SDP Examples

SDP example for 2110 – 20

```
v=0
o=- 0 1 IN IP4 192.168.21.68
s=Video Stream 1
i=Streamed by V__matrix
t=0 0
a=group:DUP primary secondary
m=video 50020 RTP/AVP 97
c=IN IP4 239.21.68.1/255
a=source-filter:incl IN IP4 239.21.68.1 192.168.21.68
a=rtpmap:97 raw/90000
a=fmtp:97 sampling=YCbCr-4:2:2; depth=10; width=1920; height=1080;
exactframerate=30000/1001; colorimetry=BT709; TP=2110TPN; TCS=SDR; PM=2110GPM;
SSN=ST2110-20:2017; PAR=1:1; interlace=1
a=mediaclk:direct=0
a=ts-refclk:localmac=00-0B-72-06-0E-94
a=mid:primary
m=video 50120 RTP/AVP 97
c=IN IP4 239.121.68.1/255
a=source-filter:incl IN IP4 239.121.68.1 192.168.121.68
a=rtpmap:97 raw/90000
a=fmtp:97 sampling=YCbCr-4:2:2; depth=10; width=1920; height=1080;
exactframerate=30000/1001; colorimetry=BT709; TP=2110TPN; TCS=SDR; PM=2110GPM;
SSN=ST2110-20:2017; PAR=1:1; interlace=1
a=mediaclk:direct=0
a=ts-refclk:localmac=00-0B-72-06-0E-99
a=mid:secondary
```

SDP example for 2110-30 Two channel

```
v=0
o=- 123456 11 IN IP4 192.168.100.2
s=Example of a SMPTE ST2110-30 signal
i=this example is for 8 channel at 1 millisecond
t=0 0
a=group:DUP primary secondary
m=audio 5004 RTP/AVP 98
c=IN IP4 239.100.9.10/32
a=source-filter: incl IN IP4 239.100.9.10 192.168.100.2
a=rtpmap:98 L24/48000/2
a=framecount:48
a=ptime:1
a=fmtp:101 channel-order=SMPTE2110.(L,R)
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:37
a=ssrc:12345 cname:user1@example.com
a=mediaclk:direct=0
a=mid:primary
m=audio 5004 RTP/AVP 98
```

```
c=IN IP4 239.101.9.10/32
a=source-filter: incl IN IP4 239.100.9.11 192.168.101.2
a=rtpmap:98 L24/48000/2
a=framecount:48
a=ptime:1
a=fmtp:101 channel-order=SMPTE2110.(L,R)
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:37
a=ssrc:12345 cname:user1@example.com
a=mediaclk:direct=0
a=mid:secondary
```

<u>SDP example for Group 2110 – 20 and 30</u>

```
v=0
o=- 0 1 IN IP4 192.168.21.68
s=Video Stream 1
i=Streamed by V__matrix
t=0 0
a=group:DUP primary secondary
m=video 50020 RTP/AVP 97
c=IN IP4 239.21.68.1/255
a=source-filter:incl IN IP4 239.21.68.1 192.168.21.68
a=rtpmap:97 raw/90000
a=fmtp:97 sampling=YCbCr-4:2:2; depth=10; width=1920; height=1080;
exactframerate=30000/1001; colorimetry=BT709; TP=2110TPN; TCS=SDR; PM=2110GPM;
SSN=ST2110-20:2017; PAR=1:1; interlace=1
a=mediaclk:direct=0
a=ts-refclk:localmac=00-0B-72-06-0E-94
a=mid:primary
m=video 50120 RTP/AVP 97
c=IN IP4 239.121.68.1/255
a=source-filter:incl IN IP4 239.121.68.1 192.168.121.68
a=rtpmap:97 raw/90000
a=fmtp:97 sampling=YCbCr-4:2:2; depth=10; width=1920; height=1080;
exactframerate=30000/1001; colorimetry=BT709; TP=2110TPN; TCS=SDR; PM=2110GPM;
SSN=ST2110-20:2017; PAR=1:1; interlace=1
a=mediaclk:direct=0
a=ts-refclk:localmac=00-0B-72-06-0E-99
a=mid:secondary
a=group:DUP primary secondary
m=audio 50030 RTP/AVP 97
c=IN IP4 239.31.68.1/255
a=source-filter:incl IN IP4 239.31.68.1 192.168.21.68
a=rtpmap:97 L24/48000/16
a=ts-refclk:localmac=00-0B-72-06-0E-94
a=mediaclk:direct=0 rate=48000
a=clock-domain:local=0
```

```
a=framecount:6
a=ptime:0.125
a=mid:primary
m=audio 50130 RTP/AVP 97
c=IN IP4 239.131.68.1/255
a=source-filter:incl IN IP4 239.131.68.1 192.168.121.68
a=rtpmap:97 L24/48000/16
a=ts-refclk:localmac=00-0B-72-06-0E-99
a=mediaclk:direct=0 rate=48000
a=clock-domain:local=0
a=framecount:6
a=ptime:0.125
a=mid:secondary
```