JSON API for ST 2110 Capable Products

Studio Technologies, Inc.

Version 0.4

17 Jan 2019

This page has been intentionally left blank.

Important Note

This document is based on the "Coveloz JSON API Documentation minuet_aes67_release-v.2.2.0-104" provided by Ross Video®. This document can be located on any ST 2110 capable device at: `http://[DEVICE_IP]/JsonAPI.html` where `[DEVICE_IP]` is the ST 2110 IPv4 address of the device.

The JSON API commands, responses, and information outlined in this document are subject to change at any time. Be sure to keep up to date to guarantee you have the latest revision.

For questions regarding Studio Technologies, Inc.'s ST 2110 enabled products or this document, please contact:

STUDIO TECHNOLOGIES INC.

7440 Frontage Rd.,
Skokie, IL 60077 USA
(847) 676-9177

Gordon Kapes
President
gkapes@studio-tech.com

Randall Kelso
Senior Engineer
rkelso@studio-tech.com

Nick Clark
Engineer
nclark@studio-tech.com

Table of Contents

1 Introduction

This document outlines the format of the JavaScript Object Notation (JSON) API used to communicate with ST 2110 capable products developed by Studio Technologies, Inc.

The "device" in this document refers to any ST 2110 capable product developed by Studio Technologies, Inc.

This JSON API uses the HTTP POST method to communicate with a device. The HTTP POST should include:

- The POST URL is `'cgi-bin/handleCommands'`
- The content-type is `'application/json'`
- The POST data contains at least two descriptors: `'command'` and `'json'`

The JSON should POST to `http://[DEVICE_IP]/cgi-bin/handleCommands` where `[DEVICE_IP]` is the ST 2110 IPv4 address of the device.

In response to the POST command, the device will respond with a message in JSON form with a success or failure indicator, and the requested data or error message.

A command message structure should be as follows:

```
{
     "command" : "[COMMAND]",
     "json" :
     {
          [JSON_DESCRIPTORS_DATA]
     }
}
```

`[COMMAND]` is one of the commands outlined in this document (e.g. `update_stream`). `[JSON_DESCRIPTORS_DATA]` are the JSON descriptors and data used to configure the device (e.g. `"id" : 14156936383406`). This may have no value depending on the command.

A response message structure will be as follows:

```
{
     "auxmsg" : null,
     "command" : "[COMMAND]",
     "error" : [ERROR_INT],
     "error_string" : "[ERROR_STRING]",
     "[REQUESTED_DATA_LABEL]" :
     {
          [REQUESTED_DESCRIPTORS_DATA]
     },
     "success" : [SUCCESS_BOOL]
}
```

`[COMMAND]` is the command sent to the device in the command message (e.g. `get_alarm_status`).
`[ERROR_INT]` is the error integer value. Zero indicates no error.
`[ERROR_STRING]` is the error message associated with the error integer value.
`[REQUESTED_DATA_LABEL]` is an optional command response field. This will may or may not be included in the response message, and will have a varying descriptor value (e.g. `global-status`).
`[REQUESTED_DESCRIPTORS_DATA]` are the response JSON descriptors and data requested in the command message (e.g. `"type" : 4`).
`[SUCCESS_BOOL]` is the success of the command message in boolean form (`true` or `false`).

2 Commands

This section outlines the commands used to configure the device. This will include example command and response messages for each command.

2.1 Hardware Module Commands

These commands control the physical ST 2110 interface hardware of the device.

2.1.1 reboot

Description: Reboots the ST 2110 interface on the device.

Parameters:

None.

Command Message to reboot the device's ST 2110 interface:

```
{
     "command" : "reboot",
     "json" : ""
}
```

Expected Response Message:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

2.2 System-Wide Management and Configuration Commands

These commands configure the device's network properties, including IP address, PTP profile, and NMOS.

2.2.1 sysmgr_set_system_parameter

Description: Set system parameters. Each parameter can be individually set. Changes to all parameters, except for "ptp-profile", require a system reboot to take effect.

Parameters:

- **ptp-profile**: [string] PTP Profile to use, available options are `"default"` IEEE1588:Default or AES67:Default, `"media"` for AES67:Media, `"smpte"` for SMPTE:2059-2, `"gptp"` for GPTP:L2.
- **offset-from-master-threshold**: [integer] Specifies the threshold value for PTP offset from master to deem PTP connection as stable.
- **board-name-prefix**: [string] Advertised board prefix
- **advert-tcp-port**: [integer] TCP port to use for advertisements
- **advert-eth-port**: [string] Ethernet interface to use for advertisements [`"eth0"` | `"eth1"` | `"eth2"`]
- **rtsp-tcp-port**: [integer] TCP port to use for RTSP
- **rtsp-eth-port**: [string] Ethernet interface to use for RTSP [`"eth0"` | `"eth1"` | `"eth2"`]
- **redundancy-enable**: [boolean] Enable or disable Ethernet redundancy.
- **add-eth-cfg**: [object] Change the configuration of the Ethernet interface
    - **eth**: [string] The name of the Ethernet interface to modify [`"eth0"` | `"eth1"` | `"eth2"`]
    - **mode**: [object]
        - **mode**: [string] Mode to configure the IP address [`static` | `DHCP`]
    - **static**: [object] Static IP information, when static mode is specified
        - **ip**: [string] static IP address for the Ethernet interface
        - **gw**: [string] Gateway IP address, default is `"0.0.0.0"`
        - **mask**: [string] Network mask for the interface, default is `"255.255.255.0"`
- **rem-eth-cfg**: [string] Remove current configuration for a specified Ethernet interface
- **tdm-ch-per-line**: [integer] Number of channels per TDM line
- **nmos**: [Object] Change the nmos properties
    - **advertisement-interface**: [string] Interface used to advertise
    - **device-name**: [string] Device name used in advertisement
    - **enable**: [boolean] Enable or disable nmos feature
    - **http-sdp-port**: [integer] Port used to retrieve SDP from http
    - **node-name**: [string] Node name used in advertisement

- **node-port**: [integer] Port used for node url
- **query-port**: [integer] Port used for query url
- **registration-port**: [integer] Port used for registration url
- **shelf-id**: [integer] Physical shelf id of device
- **slot-id**: [integer] Physical slot id of device

Command Message to set static IP address and subnet mask:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "add-eth-cfg" :
          {
               "eth" : "eth0",
               "mode" :
               {
                    "mode" : "static"
               },
               "static" :
               {
                    "ip" : "192.168.0.25",
                    "mask" : "255.255.255.0"
               }
          }
     }
}
```

Expected Response Message:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

2.3 PTP Commands

These commands request and set the IEEE 1588-2008 Precision Time Protocol Version 2 (PTPv2) settings on the device.

2.3.1 ptp_get

Description: Requests the current PTP status and configuration from the device.

Parameters (Response Only):

- **domainNumber**: [unsigned integer] The domain in which the PTPv2 traffic is being transmitted. Default value is `0`, range `0-127`.
- **meanPathDelay**: [unsigned integer] Average delay between Master transmit and slave receive in ns.
- **offsetFromMaster**: [signed integer] offset between the master and the slave in ns.
- **offsetScaledLogVariance**: [integer] stability of the clock.
- **stepsRemoved**: [unsigned integer] Number of boundary clocks between local clock and Grandmaster.
- **grandMaster**: [object] Grandmaster PTPv2 parameters.
    - **clockAccuracy**: [unsigned integer] Clock accuracy.
    - **clockAccuracyTable6**: [string] Clock accuracy in descriptive string form.
    - **clockClass**: [unsigned integer] The traceability of the time or frequency from the grandmaster clock.
    - **id**: [string] Unique ID of the clock.
    - **priority1**: [unsigned integer] User defined value, range `0-255`.
    - **priority2**: [unsigned integer] User defined value, range `0-255`.
    - **offsetScaledLogVariance**: [integer] stability of the clock.
    - **slaveOnly**: [boolean] Is the device set to slave only mode (ST 2110 requirement).
- **local**: [object] Local PTPv2 parameters.
    - **clockAccuracy**: [unsigned integer] Clock accuracy.
    - **clockAccuracyTable6**: [string] Clock accuracy in descriptive string form.
    - **clockClass**: [unsigned integer] The traceability of the time or frequency from the grandmaster clock.
    - **id**: [string] Unique ID of the clock.
    - **priority1**: [unsigned integer] User defined value, range `0-255`.
    - **priority2**: [unsigned integer] User defined value, range `0-255`.
    - **offsetScaledLogVariance**: [integer] stability of the clock.
    - **slaveOnly**: [boolean] Is the device set to slave only mode (ST 2110 requirement).

- **port**: [object array] Each port's PTPv2 parameters on the device.
    - **announceInterval**: [unsigned integer] Time between announce messages in seconds.
    - **announceReceiptTimeout**: [unsigned integer] Timeout for announce messages in seconds.
    - **delayMechanism**: [string] Delay mechanism, either `"E2E"` or `"P2P"`.
    - **id**: [string] ID of local clock for specific port.
    - **logMinDelayReqInterval**: [integer] TBD.
    - **logMinPdealyReqInterval**: [integer] TBD.
    - **peerMeanPathDelay**: [signed integer] Delay between local clock and transparent clock in ns.
    - **portId**: [integer] TBD.
    - **roleStatus**: [string] The type of clock.
    - **roleStatusId**: [integer] The type of clock in integer form.
    - **syncInterval**: [unsigned integer] Time between sync messages in seconds.
- **slaveOnly**: [boolean] Is the device set to slave only mode (ST 2110 requirement).
- **profile**: [string] PTP Profile to use, available options are `"IEEE1588:Default"` or `"AES67:Default"` or `"default"`, `"AES67:Media"` or `"media"`, `"SMPTE:2059-2"` or `"smpte"`, `"GPTP:L2"` or `"gptp"`.

Command Message to get PTP configuration:

```
{
     "command" : "ptp_get",
     "json" : ""
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "ptp_get",
     "error" : 0,
     "error_string" : "",
     "domainNumber" : 0,
     "grandMaster" :
     {
          "clockAccuracy" : 33,
          "clockAccuracyTable6" : "within 100ns",
          "clockClass" : 6,
          "id" : "ec-46-70-ff-fe-00-9f-0c",
          "priority1" : 10,
          "priority2" : 12,
          "scaledLogVariance" : 13563
```

```
          "slaveOnly" : false
     },
     "local" :
     {
          "clockAccuracy" : 33,
          "clockAccuracyTable6" : "within 100ns",
          "clockClass" : 255,
          "id" : "ce-e3-87-ff-fe-f5-13-c9",
          "priority1" : 127,
          "priority2" : 128,
          "scaledLogVariance" : 17258
          "slaveOnly" : true
     },
     "meanPathDelay" : 1609,
     "offsetFromMaster" : -18,
     "offsetScaledLogVariance" : 0,
     "port" :
     [
          {
               "announceInterval" : 0,
               "announceReceiptTimeout" : 3,
               "delayMechanism" : "E2E",
               "id" : "ce-e3-87-ff-fe-f5-13-c9",
               "logMinDelayReqInterval" : -3,
               "logMinPdealyReqInterval" : -3,
               "peerMeanPathDelay" : 0,
               "portId" : 1,
               "roleStatus" : "Slave",
               "roleStatusId" : 9,
               "syncInterval" : -3
          }
     ],
     "profile" : "SMPTE:2059-2",
     "stepsRemoved" : 1,
     "slaveOnly" : true,
     "success" : true
}
```

2.3.2 ptp_set_parameter

Description: Sets one or more user configurable PTPv2 parameters.

Parameters:

- **domain**: [unsigned integer] PTP domain number.
- **profile**: [string] PTP Profile to use, available options are `"default"` IEEE1588:Default or AES67:Default, `"media"` for AES67:Media, `"smpte"` for SMPTE:2059-2, `"gptp"` for GPTP:L2.
- **priority1**: [unsigned integer] priority1, lower number indicates higher priority.
- **priority2**: [unsigned integer] priority2, lower number indicates higher priority.
- **port**: [unsigned integer] Specify the PTP port ID. The following parameters `"announceInterval"`, `"announceReceiptTimeout"`, and `"syncInterval"` are port based. To modify these parameters, the port value needs to be specified, if not, then default value of `1` is used.
- **announceInterval**: [signed integer] PTPv2 announce interval interval. This is set per port and appropriate value should be provided in the "port" selection.
- **announceReceiptTimeout**: [signed integer] Announce receipt timeout interval. This is set per port and appropriate value should be provided in the "port" selection.
- **syncInterval**: [signed integer] Sync interval. This is set per port and appropriate value should be provided in the "port" selection.
- **slaveOnly**: [boolean] Sets the board in slave only mode. Will force the `"priority1"` to `255` when set.

Command Message to set PTP parameters:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "domain" : 0,
          "profile" : "smpte",
          "priotity1" : 1,
          "priotity2" : 1,
          "port" : 1,
          "announceInterval" : -1,
          "announceReceiptTimeout" : -1,
          "syncInterval" : -3,
          "slaveOnly" : true
     }
}
```

The local clock was set to the highest priority in the default domain, however it was set to be a slave only, nullifying the setting of the higher priority. It does not announce anything, as the announce values are all set to a number lower than `1`.

Expected Response Message:

```
{
      "error" : 0,
      "error_string" : "",
      "success" : true
}
```

2.4 RTP Source/Sync Configuration Commands

These commands set, modify, and remove sessions and session parameters. Sessions consist of a source stream and a destination stream.

2.4.1 session_remove

Description: Removes a specified session from the system.

Parameters:

- **id**: [unsigned integer 64-bit] Session ID of the session to be removed.

Command Message to remove session `34561664715862`:

```
{
     "command" : "session_remove",
     "json" :
     {
          "id" : 34561664715862
     }
}
```

Expected Response Message:

```
{
     "command" : "session_remove",
     "error" : 0,
     "error_string" : "",
     "id" : 34561664715862,
     "success" : true
}
```

2.4.2 stream_add_destination

Description: Creates a new destination session on the device based on the remote source session ID. The destination session receives the packetized data from the network and outputs it onto the physical interface.

Parameters:

- **channel-list**: [array of unsigned integers] List of channels to be used to play out the destination session (mandatory).
- **link-offset**: [unsigned integer] The time in us between the moment where the media was sampled on the source and when it will be played on the destination. If 0 is selected then we will use the default based on Packet time. Default value `0`, range is `[2 31]*packet time`.
- **name**: [string] mandatory Session name, which needs to be unique to avoid conflicts in advertisement. Forbidden characters are: "`~!*()\/'"[]|`"
- **rem-sid**: [object] Information regarding the advertised remote stream used to create the destination session.
    - **id**: [unsigned integer 64-bit] The advertisement ID of the source session. Command `get_advertised_session_list` can be used to obtained the list of the advertised sessions (mandatory)
    - **media-idx**: [unsigned integer] Specifies the media definition to use inside the SDP, as an SDP can have more than one media definition provided in the description. Default value is `0` which indicates the first media definition to use.
    - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.
- **type**: [string] The type of session to create, audio (mandatory).
- **codec**: [string] Audio codec used by session. Default value is `"AM824"`. Values can be: `"L16"`, `"L24"`, and `"AM824". Users should select "L24"`.
- **user-sdp**: [string] This field is meant to be used when an advertisement is not present but the SDP is generated on the source. If the user specifies an SDP on this field then we discard the info provided (if any) on the `"rem-sid"` and we use this SDP instead. By default this parameter is empty.

Command Message to create destination session "`my_dst_sssn`" (8 channels):

```
{
     "command" : "stream_add_destination",
     "json" :
     {
          "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
          "type" : "audio",
          "codec" : "L24",
```

```
          "link-offset" : 0,
          "name" : "my_dst_sssn",
          "rem-sid" :
          {
               "id" : [REM_SES_ID],
               "media-idx" : 0,
               "payload-type" : 98
          }
     }
}
```

`[REM_SES_ID]` [unsigned integer 64-bit] is the remote source session ID the device is connecting the destination session to.

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "stream_add_destination",
     "error" : 0,
     "error_string" : "",
     "id" : [NEW_SES_ID],
     "success" : true
}
```

`[NEW_SES_ID]` [unsigned integer 64-bit] is the new session ID for the destination session "`my_dst_sssn`".

2.4.3 stream_add_destination_manually

Description: Creates a new destination session. The destination session converts the packetized session into physical output audio.

Parameters:

- **session-id** : [unsigned integer 64-bit] Should be empty when creating a session. The reply message will contain the value of the newly created session.
- **name**: [string] session name (mandatory).
- **audio-sources**: [array of sources] These are the common attributes contained in each of the source lists.
    - ***channel-list**: [array of unsigned integers] List of channels to be used to create a source session (mandatory).
        - **id**: [unsigned integer] the output channel in the destination session.

        - **position**: [unsigned integer] the index of the source channel that is routed to an output channel in the destination session.
        - **type**: [string] the interface type can be selected (e.g. `"tdm"`).
    - **codec**: [string] Audio codec used by session. Default value is `"AM824"`. Values can be: `"L16"`, `"L24"`, and `"AM824"`. `Users should select "L24"`.
    - **dscp**: [unsigned integer] Sets the IP header DSCP field. Default value is `34`, range is `0-63`.
    - **name**: [string] session name, which needs to be unique to avoid conflicts in advertisement. Forbidden characters are: "`~!*()\/'"[]|`" (mandatory).
    - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.
    - **ptime**: [unsigned integer]. Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).
    - **transport**: [string] IP address used for transport, i.e. IP Header, destination IP address. If not specified, or value set to `0`, then the board will generate a random IP in the range `"239.x.y.z"` (mandatory).
    - **ttl**: [unsigned integer] TTL value used in the IP header. Default value is `64`, range is `0-127`.
    - **udp**: [signed integer] destination UDP port. If `-1` is specified, or absent, board will select an unused value starting on `5004`. Default value is `-1`, range is `1024-65535` (mandatory).
    - **media-clk-offset**: [signed integer] The media clock at epoch. Default value is `0`, must remain `0` to be ST 2110-10 compliant.
    - **redundant-type**: [string] Redundant type of the stream. Can be the following values: `"primary"`, `"secondary"`, or `"none"`. If omitted or empty none is assumed.
    - **channel-freq**: [unsigned integer] Defines the session channel frequency (`48000` or `96000`). If `0` or absent, then session will use the globally stored value.
    - **ptime**: [unsigned integer] Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).

***channel-list**: see `stream_add_source` for more information.

Command Message to create source session "`my_dst_sssn`" (8 channels):

```
{
     "command" : "stream_add_destination_manually",
     "json" :
     {
          "session-id" : 0,
          "name" : "my_dst_sssn",
          "audio-sources" :
          [
               {
                    "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
                    "id" : 0,
                    "type" : "audio",
                    "codec" : "L24",
                    "dscp" : 34,
                    "name" : "my_dst_strm",
                    "ptime" : 0,
                    "channel-freq" : 0,
                    "payload-type" : 98,
                    "transport" : 0,
                    "ttl" : 64,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "redundant-type" : "none"
               }
          ]
     }
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "stream_add_destination_manually",
     "error" : 0,
     "error_string" : "",
     "session-id" : [NEW_SES_ID],
     "name" : "my_dst_sssn",
     "audio-sources" :
     [
          {
               "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
               "id" : [NEW_STM_ID],
               "type" : "audio",
               "codec" : "L24",
               "dscp" : 34,
               "name" : "my_dst_strm",
```

```
                "ptime" : 0,
                "payload-type" : 98,
                "transport" : 0,
                "ttl" : 64,
                "udp" : -1,
                "media-clk-offset" : 0,
                "redundant-type" : "primary"
            }
        ]
}
```

`[NEW_SES_ID]` and `[NEW_STM_ID]` [unsigned integer 64-bit] are the new session and stream IDs for the source session "`my_src_sssn`".

2.4.4 stream_add_source

Description: Creates a new source session containing one or more streams that will be advertised on the network. The source session converts the physical input audio into the packetized session to send over the network.

Parameters:

- **session-id** : [unsigned integer 64-bit] Should be empty when creating a session. The reply message will contain the value of the newly created session.
- **name**: [string] session name (mandatory).
- **audio-sources**: [array of sources] These are the common attributes contained in each of the source lists.
    - ***channel-list**: [array of unsigned integers] List of channels to be used to create a source session (mandatory).
        - **id**: [unsigned integer] the output channel in the destination session.
        - **position**: [unsigned integer] the index of the source channel that is routed to an output channel in the destination session.
        - **type**: [string] the interface type can be selected (e.g. "`tdm`").
    - **codec**: [string] Audio codec used by session. Default value is "`AM824`". Values can be: "`L16`", "`L24`", and "`AM824`". `Users should select "L24".`
    - **dscp**: [unsigned integer] Sets the IP header DSCP field. Default value is `34`, range is `0-63`.
    - **name**: [string] session name, which needs to be unique to avoid conflicts in advertisement. Forbidden characters are: "`~!*()\/'"[]|`" (mandatory).
    - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.

- **ptime**: [unsigned integer]. Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).
- **transport**: [string] IP address used for transport, i.e. IP Header, destination IP address. If not specified, or value set to `0`, then the board will generate a random IP in the range `"239.x.y.z"` (mandatory).
- **ttl**: [unsigned integer] TTL value used in the IP header. Default value is `64`, range is `0-127`.
- **udp**: [signed integer] destination UDP port. If `-1` is specified, or absent, board will select an unused value starting on `5004`. Default value is `-1`, range is `1024-65535` (mandatory).
- **media-clk-offset**: [signed integer] The media clock at epoch. Default value is `0`, must remain `0` to be ST 2110-10 compliant.
- **redundant-type**: [string] Redundant type of the stream. Can be the following values: `"primary"`, `"secondary"`, or `"none"`. If omitted or empty none is assumed.
- **channel-freq**: [unsigned integer] Defines the session channel frequency (`48000` or `96000`). If `0` or absent, then session will use the globally stored value.
- **ptime**: [unsigned integer] Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).

***channel-list**: This can be defined two ways in this instance. It can be defined as an integer array (the simplest, and most common form):

```
{
     "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7]
}
```

It can also be defined as an array of objects, with the `id` and `position` parameters:

```
{
     "channel-list" :
     [
          {
               "id" : 14,
               "position" : 0,
```

```
                    "type" : "tdm"
               },
               {
                    "id" : 15,
                    "position" : 2,
                    "type" : "tdm"
               },
               {
                    "id" : 25,
                    "position" : 2,
                    "type" : "tdm"
               }
          ]
     }
```

[!] With channel-list defined as an object array above, channel 14 of the destination session will output TDM audio from channel 0 of the source session. Channel 15 and 25 of the destination session will output TDM audio from channel 2 of the source session. This definition of channel-list is not required, and it is suggested to use the integer array form instead.

Command Message to create source session "`my_src_sssn`" (8 channels):

```
{
     "command" : "stream_add_source",
     "json" :
     {
          "session-id" : 0,
          "name" : "my_src_sssn",
          "audio-sources" :
          [
               {
                    "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
                    "id" : 0,
                    "type" : "audio",
                    "codec" : "L24",
                    "dscp" : 34,
                    "name" : "my_src_strm",
                    "ptime" : 0,
                    "channel-freq" : 0,
                    "payload-type" : 98,
                    "transport" : 0,
                    "ttl" : 64,
                    "udp" : -1,
                    "media-clk-offset" : 0,
```

```
                    "redundant-type" : "primary"
               }
          ]
     }
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "stream_add_source",
     "error" : 0,
     "error_string" : "",
     "session-id" : [NEW_SES_ID],
     "name" : "my_src_sssn",
     "audio-sources" :
     [
          {
               "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
               "id" : [NEW_STM_ID],
               "type" : "audio",
               "codec" : "L24",
               "dscp" : 34,
               "name" : "my_src_strm",
               "ptime" : 0,
               "payload-type" : 98,
               "transport" : 0,
               "ttl" : 64,
               "udp" : -1,
               "media-clk-offset" : 0,
               "redundant-type" : "primary"
          }
     ]
}
```

`[NEW_SES_ID]` and `[NEW_STM_ID]` [unsigned integer 64-bit] are the new session and stream IDs for the source session "`my_src_sssn`".

2.4.5 update_global_cfg

Description: Update the global session configurations. These settings are applicable for all sessions on the board and specify the default values for all sessions, but they can also be overwritten during the creation of a specific session (except for red-udp-offset). Each setting can be changed individually.

Parameters:

- **channel-freq**: [unsigned integer] Channel sampling frequency in Hz. Possible values are `48000` and `96000`.
- **ptime**: [unsigned integer] Global packet time in us, possible values are `125`, `250`, and `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance). Amount of audio time inside each packet.
- **red-udp-offset**: [unsigned integer] Defines the UDP port offset between main traffic and the redundant one.

Command Message to set global session frequency, packet time, and redundant UDP offset:

```
{
     "command" : "update_global_cfg",
     "json" :
     {
          "global" :
          {
               "ptime" : 125,
               "channel-freq" : 48000,
               "red-udp-offset" : 2
          }
     }
}
```

Expected Response Message:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

2.4.6 update_session_state

Description: This command pauses or resumes an audio session. For a source session, it will stop/resume the traffic flow. For a destination session, it will mute/unmute the audio.

Parameters:

- **id**: [unsigned integer 64-bit] Session ID. Default value is `0`.
- **pause**: [boolean] When true pause the session, on false the session is resumed/released. Default value is `false`.

Command Message to stop audio on source session `790483520824`:

```
{
     "command" : "update_session_state",
     "json" :
     {
          "id" : 790483520824,
          "pause" : false
     }
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "update_session_state",
     "error" : 0,
     "error_string" : "",
     "id" : 790483520824,
     "success" : true
}
```

2.4.7 update_stream

Description: Updates a source or destination session configuration. This command is similar to `stream_add_source`.

Parameters:

- **session-id** : [unsigned integer 64-bit] Session ID of the session to be updated.
- **name**: [string] mandatory session name.
- **audio-sources**: [array of sources] These are the common attributes contained in each of the source lists.
    - ***channel-list**: [array of unsigned integers] List of channels to be used to create a source session (mandatory).
        - **id**: [unsigned integer] the output channel in the destination session.
        - **position**: [unsigned integer] the index of the source channel that is routed to an output channel in the destination session.
        - **type**: [string] the interface type can be selected (e.g. `"tdm"`).
    - **codec**: [string] Audio codec used by session. Default value is `"AM824"`. Values can be: `"L16"`, `"L24"`, and `"AM824"`. `Users should select "L24".`
    - **dscp**: [unsigned integer] Sets the IP header DSCP field. Default value is `34`, range is `0-63`.
    - **name**: [string] session name, which needs to be unique to avoid conflicts in advertisement. Forbidden characters are: "`~!*()\/'"[]|`" (mandatory).
    - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.
    - **ptime**: [unsigned integer]. Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).
    - **transport**: [string] IP address used for transport, i.e. IP Header, destination IP address. If not specified, or value set to `0`, then the board will generate a random IP in the range `"239.x.y.z"` (mandatory).
    - **ttl**: [unsigned integer] TTL value used in the IP header. Default value is `64`, range is `0-127`.
    - **udp**: [signed integer] destination UDP port. If `-1` is specified, or absent, board will select an unused value starting on `5004`. Default value is `-1`, range is `1024-65535` (mandatory).
    - **media-clk-offset**: [signed integer] The media clock at epoch. Default value is `0`, must remain `0` to be ST 2110-10 compliant.

- **redundant-type**: [string] Redundant type of the stream. Can be the following values: `"primary"`, `"secondary"`, or `"none"`. If empty `none` is assumed.
- **channel-freq**: [unsigned integer] Defines the session channel frequency (`48000` or `96000`). If `0` or absent, then session will use the globally stored value.
- **ptime**: [unsigned integer] Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. If `0` or absent, board will use the global ptime. If specified, it needs to match the global; there is currently no support for different ptimes on the board. Default value is `0`, range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).

***channel-list**: see `stream_add_source` for more information.

Command Message to update source session `340018917984`:

```
{
     "command" : "update_stream",
     "json" :
     {
          "session-id" : 340018917984,
          "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
          "type" : "audio",
          "codec" : "L24",
          "dscp" : 34,
          "name" : "my_new_src_sssn",
          "ptime" : 0,
          "channel-freq" : 0,
          "payload-type" : 98,
          "transport" : 0,
          "ttl" : 64,
          "udp" : 5012,
          "media-clk-offset" : 0,
          "redundant-type" : "primary"
     }
}
```

Expected Response Message:

```
{
     "command" : "update_stream",
     "error" : 0,
     "error_string" : "",
     "id" : 340018917984,
     "success" : true
}
```

2.5 RTP Session Status Commands

These commands report the status and configuration of current source and destination sessions.

2.5.1 get_alarm_status

Description: Returns the list of alarms and the global status of sessions. This command can be used to periodically poll the state of alarms on the device.

Parameters (response only):

- **global-status**: [object array] Global status of sessions and streams on the system
  - **type**: [integer] Global status value.
  - **description**: [string] Description of the status.
- **sessions**: [object array] current sessions and alarms for each session.
  - **id**: [unsigned integer 64-bit] Session ID.
  - **alarms**: [object array] alarms for the session
    - **type**: [integer] Alarm type value.
    - **level**: [string] Alarm level. Can be `"on"`, `"off"`, or `"disabled"`.
    - **severity**: [string] Alarm severity. Can be `"warning"`, `"error"`, `"critical"`, or `"ok"`.
    - **description**: [string] Description of the alarm.
  - **streams**: [object array] streams in the session.
    - **id**: [unsigned integer 64-bit] Stream ID.
    - **alarms**: [object array] alarms for the stream.
      - **type**: [integer] Alarm type value.
      - **level**: [string] Alarm level. Can be `"on"`, `"off"`, or `"disabled"`.
      - **severity**: [string] Alarm severity. Can be `"warning"`, `"error"`, `"critical"`, or `"ok"`.
      - **description**: [string] Description of the alarm.

Command Message to get alarms and session statuses of the device:

```
{
     "command" : "get_alarm_status",
     "json" : ""
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
```

```
     "command" : "get_alarm_status",
     "error" : 0,
     "error_string" : "",
     "global-status" :
     [
          "type" : 4,
          "description" : "Redundant link down"
     ],
     "sessions" :
     [
          {
               "alarms" :
               [
                    {
                         "description" : "No packets being
received",
                         "level" : "on",
                         "severity" : "error",
                         "type" : 0
                    }
               ],
               "id" : 2272615334874,
               "streams" :
               [
                    {
                         "alarms" :
                         [
                              {
                                   "description" : "No packets
being
                                                     received",
                                   "level" : "on",
                                   "severity" : "warning",
                                   "type" : 0
                              }
                         ],
                         "id" : 2272615401674
                    },
                    {
                         "id" : 159361304742
                    }
               ]
          }
     ],
     "success" : true
}
```

[!] This response is from a device with one destination session with two streams. There is a global alarm indicating that the redundant link of the device is down/not connected. The session has an error indicating that no audio is being received on one or more streams. Of the two streams in the session, one stream has an alarm indicating that it is the problematic stream, while the other has no issues.

2.5.2 get_advertised_session_list

Description: Gets the advertisement list. This command will list all talkers/senders found on the network using mDNS.

Parameters (response only):

- **list**: [object array] list of sessions
  - **host**: [object array] network information about the host of the session.
    - **address**: [string] location of session, can be either "local" or "remote".
    - **ip**: [string] IPv4 address of the ST 2110 interface of the device.
  - **inuse**: [boolean] active state of the session, can be `true` or `false`.
  - **sdp-raw**: [string] raw SDP session data.
  - **session**: [object array] session data parsed into parameters.
    - **id**: [unsigned integer 64-bit] Session ID.
    - **info**: [string] Session name.
    - **local-ip**: [string] IPv4 address of the ST 2110 interface of the device.
    - **media**: [object array] session specific information.
      - **clock-is-ptp**: [boolean] Device clock PTP state, can be `true` or `false`.
      - **group**: [string] TBD.
      - **info**: [string] Advertised session name.
      - **local-ip**: [string] IPv4 address of the ST 2110 interface of the device.
      - **mediaclk**: [integer] TBD.
      - **number-samples**: [integer] TBD.
      - **protocol**: [string] Protocol used for audio transport, default `"RTP/AVP"`.
      - **ptime**: [unsigned integer]. Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. Default value is `0` (global configuration), range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).
      - **ptp**: [object array] Basic PTP information.
        - domain: [unsigned integer] PTP domain.

            - gmid: [string] Device's Grandmaster ID.
        - **rtpmap**: [object array] RTP information for specific session.
            - **channel-freq**: [unsigned integer] Channel frequency, can be `48000` or `96000`.
            - **codec**: [string] Audio codec used by session. Default value is `"AM824"`. Values can be: `"L16"`, `"L24"`, and `"AM824". Users should select "L24".`
            - **number-channels**: [unsigned integer] Number of channels in the session.
            - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.
        - **transport**: [string] IP address used for transport, i.e. IP Header, destination IP address.
        - **type**: [string] The type of session to create, `"audio"`.
        - **ttl**: [unsigned integer] TTL value used in the IP header. Default value is `64`, range is `0-127`.
        - **udp**: [signed integer] destination UDP port. Range is `1024-65535`.
        - **video**: [object array] List of video parameters that is unused.
            - **colorimetry**: [string] null.
            - **depth**: [integer] null.
            - **format**: [string] null.
            - **sampling**: [string] null.
    - **protocol-version**: [integer] TBD.
    - **time**: [object array] TBD.
        - **start**: [integer] TBD.
        - **stop**: [integer] TBD.
    - **user-name**: [string] TBD.
    - **version**: [integer] TBD.
- **session-id**: [unsigned integer 64-bit] Session ID.
- **session-id-str**: [string] Session ID as a string.
- **status**: [string] alarm status of the session.
- **url**: [string] RTP protocol URL of the session.

Command Message to get all advertised source sessions on the network:

```
{
     "command" : "get_advertised_session_list",
     "json" : ""
}
```

Expected Response Message:


```
{
     "auxmsg" : null,
     "command" : "get_advertised_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "host" :
               {
                    "address" : "local",
                    "ip" : "192.168.0.100"
               },
               "inuse" : true,
               "sdp-raw" : "v=0\no=- 463993512032 0 IN IP4
192.168.0.100\ns=session-192.168.0.100-00\nt=0 0\nm=audio 5004
RTP/AVP 98\ni=source-192.168.0.100-01\nc=IN IP4
239.218.181.1/64\na=clock-domain:PTPv2
0\na=ts-refclk:ptp=IEEE1588-2008:52-28-bd-ff-fe-c9-cc-bd:0\na=media
clk:direct=0\na=source-filter: incl IN IP4 239.218.181.1
192.168.0.100\na=rtpmap:98
L24/48000/2\na=framecount:6\na=ptime:0.125\n",
               "session" :
               {
                    "id" : 463993512032,
                    "info" : "session-192.168.0.100-00",
                    "local-ip" : "192.168.0.100",
                    "media" :
                    [
                         {
                              "clock-is-ptp" : true,
                              "group" : "",
                              "info" : "source-192.168.0.100-01",
                              "local-ip" : "192.168.0.100",
                              "mediaclk" : 0,
                              "number-samples" : 6,
                              "protocol" : "RTP/AVP",
                              "ptime" : 0.125,
                              "ptp" :
                              {
                                   "domain" : 0,
                                   "gmid" :
"52-28-bd-ff-fe-c9-cc-bd"
                              },
                              "rtpmap" :
```

```
                              [
                                   {
                                        "channel-freq" : 48000,
                                        "codec" : "L24",
                                        "number-channels" : 2,
                                        "payload-type" : 98
                                   }
                              ],
                              "transport-ip" : "239.218.181.1",
                              "ttl" : 64,
                              "type" : "AUDIO",
                              "udp" : 5004,
                              "video" :
                              {
                                   "colorimetry" : "",
                                   "depth" : 0,
                                   "sampling" : ""
                              }
                         }
                    ],
                    "protocol-version" : 0,
                    "time" :
                    {
                         "start" : 0,
                         "stop" : 0
                    },
                    "user-name" : "-",
                    "version" : 0
               },
               "session-id" : 463993512032,
               "session-id-str" : "463993512032",
               "status" : "OK",
               "url" : "rtsp://192.168.0.100:8554/by-name
                        /session-192.168.0.100-00/"
          },
          {
               "host" :
               {
                    "address" : "remote",
                    "ip" : "192.168.0.105"
               },
               "inuse" : false,
               "sdp-raw" : "v=0\no=- 463993512033 0 IN IP4
192.168.0.105\ns=session-192.168.0.105-00\nt=0 0\nm=audio 5006
RTP/AVP 98\ni=source-192.168.0.105-00\nc=IN IP4
239.218.181.2/64\na=clock-domain:PTPv2
```

```
0\na=ts-refclk:ptp=IEEE1588-2008:52-28-bd-ff-fe-c9-cc-be:0\na=media
clk:direct=0\na=source-filter: incl IN IP4 239.218.181.2
192.168.0.105\na=rtpmap:98
L24/48000/2\na=framecount:6\na=ptime:0.125\n",
               "session" :
               {
                    "id" : 463993512033,
                    "info" : "session-192.168.0.105-00",
                    "local-ip" : "192.168.0.105",
                    "media" :
                    [
                         {
                              "clock-is-ptp" : true,
                              "group" : "",
                              "info" : "source-192.168.0.105-00",
                              "local-ip" : "192.168.0.105",
                              "mediaclk" : 0,
                              "number-samples" : 6,
                              "protocol" : "RTP/AVP",
                              "ptime" : 0.125,
                              "ptp" :
                              {
                                   "domain" : 0,
                                   "gmid" :
"52-28-bd-ff-fe-c9-cc-be"
                              },
                              "rtpmap" :
                              [
                                   {
                                        "channel-freq" : 48000,
                                        "codec" : "L24",
                                        "number-channels" : 2,
                                        "payload-type" : 98
                                   }
                              ],
                              "transport-ip" : "239.218.181.2",
                              "ttl" : 64,
                              "type" : "AUDIO",
                              "udp" : 5006,
                              "video" :
                              {
                                   "colorimetry" : "",
                                   "depth" : 0,
                                   "sampling" : ""
                              }
                         }
```

```
                    ],
                    "protocol-version" : 0,
                    "time" :
                    {
                         "start" : 0,
                         "stop" : 0
                    },
                    "user-name" : "-",
                    "version" : 0
               },
               "session-id" : 463993512033,
               "session-id-str" : "463993512033",
               "status" : "OK",
               "url" : "rtsp://192.168.0.105:8554/by-name
                        /session-192.168.0.105-00/"
          }
     ],
     "success" : true
}
```


[!] This response shows two advertised source sessions. The first is an active local session. The second is an inactive remote session. Both source session have 2 audio channels.

2.5.3 get_destination_session_list

Description: Returns the list of all of the device's local destination sessions.

Parameters (response only):

- **list**: [object array] list of destination sessions.
    - **name**: [string] Name of destination session.
    - **remote-sid**: [unsigned integer 64-bit] Remote source session ID.
    - **remote-sid-str**: [string] Remote source session ID as a string.
    - **session**: [unsigned integer 64-bit] Destination session ID.
    - **state**: [string] Current session state, `"Enabled"` or `"Disabled"`.
    - **stream**: [integer] index of the session.
    - **type**: [string] The type of session to create, `"audio"`.

Command Message to get a list of destination sessions on the device:

```
{
     "command" : "get_destination_session_list",
     "json" : ""
}
```

Expected Response Message:


```
{
     "auxmsg" : null,
     "command" : "get_destination_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "name" : "my_dst_strm_0",
               "remote-sid" : 834054116334,
               "remote-sid-str" : "834054116334",
               "session" : 940800754784,
               "state" : "Enabled",
               "stream" : 0,
               "type" : "audio"
          },
          {
               "name" : "my_dst_strm_1",
               "remote-sid" : 834182158514,
               "remote-sid-str" : "834182158514",
               "session" : 940958284204,
               "state" : "Disabled",
               "stream" : 1,
               "type" : "audio"
          }
     ],
     "success" : true
}
```


[!] The response message shows the device has two destination sessions. Both are connected to remote source sessions, but the second session is muted.

2.5.4 get_session_basic_info

Description: Gets session's basic information, such as stream configuration, internal status, and PTP information, for a specified local session on the device.

Parameters:

- **id**: [unsigned integer 64-bit] Session ID
- **name**: [string] Name of the session
- **ptp-domain**: [unsigned integer] PTP domain that board is running in
- **ptp-gmid**: [string] PTP Grandmaster ID

- **remote-sid**: [unsigned integer 64-bit] Session ID of the remote source session. Only applicable for receiver (destination) session. For sender (source) session, this value is 0.
- **remote-sid-str**: [string] String version of remote source session id (remote-sid). Only applicable for receiver (destination) session. For sender (source) session, this value is 0.
- **rtsp-url**: [string] URL of the RTSP of the session.
- **session-list**: [object array] List of session information.
    - **channel-freq**: [unsigned integer] Channel frequency for audio session. This is only displayed for audio sessions
    - **channel-list**: [array of unsigned integers] The channels used for this session. The channels correspond to physical ports on the board.
    - **class**: [string] Class of the session, either sender (source) or receiver (destination).
    - **codec**: [string] Codec for audio session. This is only displayed for audio sessions.
    - **ipv4**: [object] IP version 4 information about this session
        - **dscp**: [unsigned integer] Sets the IP header DSCP field. Default value is `34`, range is `0-63`.
        - **local**: [string] IPv4 address of the ST 2110 interface of the device.
        - **transport**: [string] IP address used for transport, i.e. IP Header, destination IP address.
        - **ttl**: [unsigned integer] TTL value used in the IP header. Default value is `64`, range is `0-127`.
    - **l1-bandwidth**: [unsigned integer 32-bit] Bandwidth for the traffic of this session, specified in bits/second
    - **link-offset**: [unsigned integer] Link offset for receiver (destination) session.
    - **media-clk-offset**: [unsigned integer] Offset for the media clock. Default value of `0`, must remain `0` to be ST 2110-10 compliant
    - **name**: [string] Name of the session
    - **packet-size**: [unsigned integer] Session packet size in bytes
    - **ptime**: [unsigned integer]. Defines the amount of time the packetizer will wait to fill the payload. This is the amount of audio inside the packet. Default value is `0` (global configuration), range is `0`, `125`, `250`, `1000` (choose `1000` for ST 2110-30 Level A compliance, or `125` for ST 2110-30 Level B compliance).
    - **rtp**: [object] RTP information about the session
        - **payload-type**: [unsigned integer] Index used on the RTP header. Default value is `98`, range is `0-255`.
        - **sscr**: [unsigned integer 32-bit] Unique synchronization source identifier for the source of a data stream.
    - **state**: [string] Current state of the session
    - **type**: [string] Type of the session, audio or video
    - **udp**: [object] UDP information about the session
        - **destination**: [unsigned integer] Destination UDP port.

        - **source**: [unsigned integer] Source UDP port.
    - **vlan**: [object] VLAN information about the session, such as VLAN id, priority, and tpid.
        - **id**: [unsigned integer] VLAN ID, range from `0` to `4094`.
        - **priority**: [unsigned integer] Priority, range from `0` to `7`.
        - **tpid**: [unsigned integer] Tag Protocol ID, default value `33024`.

Command Message to get basic information on local session `940958284204`:

```
{
     "command" : "get_session_basic_info",
     "json" :
     {
          "id" : 940958284204
     }
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "get_session_basic_info",
     "error" : 0,
     "error_string" : "",
     "id" : 940958284204,
     "name" : "my_dst_strm",
     "ptp_domain" : 127,
     "ptp-gmid" : "70-B3-D5-FF-FE-04-22-12",
     "remote-sid" : 834182158514,
     "remote-sid-str" : "834182158514",
     "rtsp-url" : "rtsp://169.254.4.106:8554/by-name/my_src_strm/",
     "session-list" :
     [
          {
               "channel-freq" : 48000,
               "channel-list" : [0, 1, 2, 3, 4, 5, 6, 7],
               "class" : "receiver",
               "codec" : "L24",
               "ipv4" :
               {
                    "dscp" : 34,
                    "local" : "169.254.4.76",
                    "transport" : "239.7.0.3",
                    "ttl" : 64
               },
```

```
                "l1-bandwidth" : 16960000,
                "link-offset" : 1600,
                "media-clk-offset" : 0,
                "name" : "my_dst_strm",
                "packet_size" : 198,
                "ptime" : 125,
                "rtp" :
                {
                     "payload-type" : 98,
                     "sscr" : 360446980
                },
                "state" : "Enabled",
                "type" : "audio",
                "udp" :
                {
                     "destination" : 10000,
                     "source" : 10000
                },


                "vlan" :
                {
                     "id" : 0,
                     "priority" : 0,
                     "tpid" : 33024
                }
           }
     ],
     "success" : true,
     "version" : 0
}
```


[!] This response shows the basic information for a destination source, "my_dst_strm", on a device, connected to "my_src_strm".

2.5.5 get_session_sdp

Description: Request the Session Description Protocol (SDP) for a specified session.

Parameters:

- id: [unsigned integer 64-bit] Session ID.

Command Message to get a list of destination sessions on the device:

```
{
     "command" : "get_session_sdp",
     "json" : ""
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "get_source_session_list",
     "error" : 0,
     "error_string" : "",
     "sdp" : "v=0\no=- 463993512033 0 IN IP4
192.168.0.105\ns=session-192.168.0.105-00\nt=0 0\nm=audio 5006
RTP/AVP 98\ni=source-192.168.0.105-00\nc=IN IP4
239.218.181.2/64\na=clock-domain:PTPv2
0\na=ts-refclk:ptp=IEEE1588-2008:52-28-bd-ff-fe-c9-cc-be:0\na=media
clk:direct=0\na=source-filter: incl IN IP4 239.218.181.2
192.168.0.105\na=rtpmap:98
L24/48000/2\na=framecount:6\na=ptime:0.125\n",
     "success" : true
}
```

2.5.6 get_source_session_list

Description: Returns the list of all of the device's local source sessions.

Parameters (response only):

- **list**: [object array] list of destination sessions.
  - **name**: [string] Name of destination session.
  - **remote-sid**: [unsigned integer 64-bit] Remote source session ID.
  - **remote-sid-str**: [string] Remote source session ID as a string.
  - **session**: [unsigned integer 64-bit] Destination session ID.
  - **state**: [string] Current session state, `"Enabled"` or `"Disabled"`.
  - **stream**: [integer] index of the session.
  - **type**: [string] The type of session to create, `"audio"`.

Command Message to get a list of destination sessions on the device:

```
{
     "command" : "get_source_session_list",
     "json" : ""
}
```

Expected Response Message:


```
{
     "auxmsg" : null,
     "command" : "get_source_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "name" : "my_src_strm_0",
               "remote-sid" : 0,
               "remote-sid-str" : "0",
               "session" : 940800754784,
               "state" : "Enabled",
               "stream" : 0,
               "type" : "audio"
          },
          {
               "name" : "my_src_strm_1",
               "remote-sid" : 0,
               "remote-sid-str" : "0",
               "session" : 940958284204,
               "state" : "Disabled",
               "stream" : 1,
               "type" : "audio"
          }
     ],
     "success" : true
}
```


[!] The response message shows the device has two source sessions. The second session is not sending audio.

2.6 Device Configuration Commands

Device Configuration allows the user to save current system setup, including running sessions, specify the startup configurations, and restore previously saved configurations.

2.6.1 filecfg_save_running

Description: Saves the current running configuration into a file configuration table at a specified position. If startup flag is specified, then this configuration will be used as the device's startup configuration.

Parameters:

- **pos**: [unsigned integer] Position in the file configuration table where the new configuration will be saved. If configuration is already present in the table, it will be overwritten with new information. Range is `0` to `10` (pos `10` is not writeable).
- **name**: [string] Name of the new configuration.
- **startup**: [boolean] True if a new configuration should be used as a start-up configuration. Default value is `false`.

Command Message to save the running device configuration:

```
{
     "command" : "filecfg_save_running",
     "json" :
     {
          "pos" : 0,
          "name" : "saved_config",
          "startup" : true
     }
}
```

Expected Response Message:

```
{
     "command" : "filecfg_save_running",
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

[!] The configuration `"saved_config"` was saved as the new starting configuration.

2.6.2 filecfg_set_startup

Description: Modifies the position of the start-up configuration.

Parameters:

- **pos**: [unsigned integer] Position in the file configuration table where the new start-up configuration will be loaded from. Range is `0` to `10`.

Command Message to set the new start-up configuration:

```
{
     "command" : "filecfg_set_startup",
     "json" :
     {
          "pos" : 0
     }
}
```

Expected Response Message:

```
{
     "auxmsg" : null,
     "command" : "filecfg_set_startup",
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

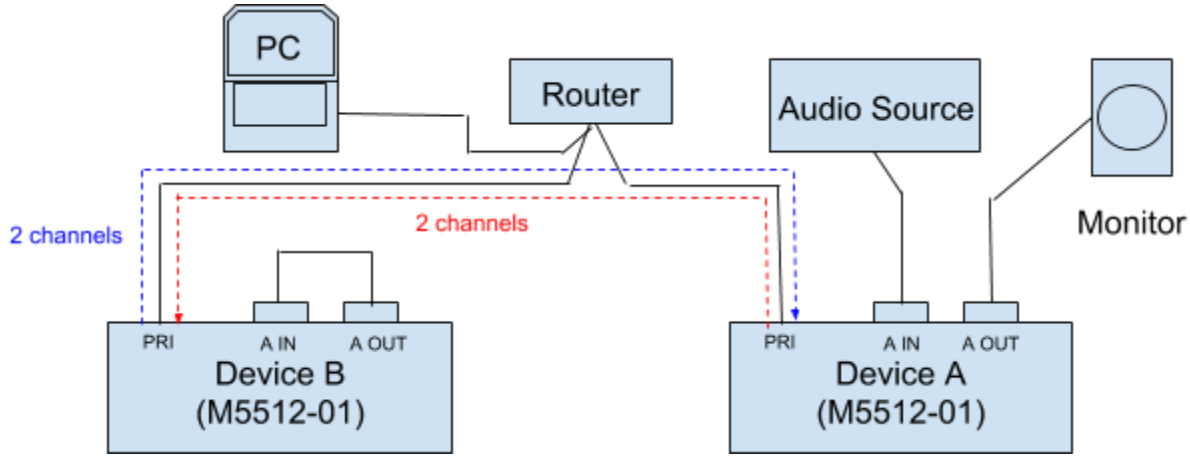
[!] The new start-up configuration is position `0`.

3 Appendix A

3.1 Examples

This section will discuss specific examples of the JSON API's use. This will include configuring the ST 2110 interface of a device and creating a stream between sessions. There will be two devices used in these examples, and are two Studio Technologies, Inc. Model 5512-01 Audio Interfaces, and will be referred to as Device A and Device B.

By the end of the Examples section, both devices will have two audio streams between them. The setup for the examples is as follows:



It is important to note that these examples are not strictly for two Studio Technologies, Inc. devices. These examples will also work for configuring and creating sessions for one device at a single point in a system that other ST 2110 or AES67 units can stream to.

For these examples, Device A will have an IP address of `192.168.0.99` and Device B will have an IP address of `192.168.0.100`.

3.1.1 Setting Static IP Addresses and other Parameters

In many professional systems, static IP addressing is important to keep an organized network. This example will discuss how to set a static IP address as well as other parameters such as PTP profile and NMOS configuration for discovery on the network. To do this, the `sysmgr_set_system_parameter` command must be sent.

For Device A, the static IP address will be set to `192.168.0.99`, with a subnet mask of `255.255.255.0`. To conform to the ST 2110 standard the PTP profile must be `SMPTE 2059-2`. Finally, most NMOS parameters are set by default, but the device and node names need to be set, and NMOS must be enabled. Device A's NMOS device and node names will be set to `"ST-M5512-A"`.

The following JSON message will be sent to `http://[DEVICE_IP]/cgi-bin/handleCommands` using the HTTP POST method (`[DEVICE_IP]` can be obtained from the front-panel display on the device under ST 2110 PRI IP Addr on Device A) and the header should include a content-type of `'application/json'`:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "add-eth-cfg" :
          {
               "eth" : "eth0",
               "mode" :
               {
                    "mode" : "static"
               },
               "static" :
               {
                    "ip" : "192.168.0.99",
                    "mask" : "255.255.255.0"
               }
          },
          "ptp-profile" : "smpte",
          "nmos" :
          {
               "device-name" : "ST-M5512-A",
               "node-name" : "ST-M5512-A",
               "enable" : true
          }
     }
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

According to the description of the `sysmgr_set_system_parameter` command the device must be rebooted if any parameter other than `ptp-profile` is modified. The `reboot` command must now be sent to Device A.

The following JSON message will be sent to `http://[DEVICE_IP]/cgi-bin/handleCommands` using the HTTP POST method (`[DEVICE_IP]` can be obtained from the front-panel display on the device under ST 2110 PRI IP Addr on Device A) and the header should include a content-type of `'application/json'`:

```
{
     "command" : "reboot",
     "json" : ""
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

The ST 2110 interface on Device A will now reboot. The SYS and SYNC LEDs on Device A will now go amber and off respectively, and will both return to green when the ST 2110 interface is running again. Device A will now be at IP address `192.168.0.99`.

Repeat the same steps for Device B, but set its IP address to `192.168.0.100`, and set its NMOS device and node names to `"ST-M5512-B"`. Send the commands to the ST 2110 PRI IP Addr found on the front-panel display of the device.

3.1.2 Configure Global Settings

Before audio is routed global session settings should be set. For compliance of Level A of ST 2110, packet time (the length of audio present in an audio packet) should be set for, at most, 1 ms and audio sampling rate should be 48 kHz. To do this, the `update_global_cfg` command should be sent. Device A will be configured first.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "update_global_cfg",
     "json" :
     {
          "global" :
          {
               "ptime" : 1000,
               "channel-freq" : 48000
          }
     }
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

Send the same message to Device B at `192.168.0.100`.

Now both devices are configured for Level A compliance of ST 2110.

3.1.3 Requesting and Configuring PTP Parameters

Timing is a very important part of ST 2110. It is important to know the current PTP configuration, and the ST 2110 interface on the devices allows for significant configuration of PTP parameters. Device A will first be queried for its PTP parameters using the `ptp_get` command.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_get",
     "json" : ""
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "ptp_get",
     "error" : 0,
     "error_string" : "",
     "domainNumber" : 0,
     "grandMaster" :
     {
          "clockAccuracy" : 33,
          "clockAccuracyTable6" : "within 100ns",
          "clockClass" : 6,
          "id" : "ec-46-70-ff-fe-00-9f-0c",
          "priority1" : 10,
          "priority2" : 12,
          "scaledLogVariance" : 13563
          "slaveOnly" : false
     },
     "local" :
     {
          "clockAccuracy" : 33,
          "clockAccuracyTable6" : "within 100ns",
          "clockClass" : 255,
          "id" : "ce-e3-87-ff-fe-f5-13-c9",
          "priority1" : 127,
          "priority2" : 128,
          "scaledLogVariance" : 17258
          "slaveOnly" : true
     },
```

```
     "meanPathDelay" : 1609,
     "offsetFromMaster" : -18,
     "offsetScaledLogVariance" : 0,
     "port" :
     [
          {
               "announceInterval" : 0,
               "announceReceiptTimeout" : 3,
               "delayMechanism" : "E2E",
               "id" : "ce-e3-87-ff-fe-f5-13-c9",
               "logMinDelayReqInterval" : -3,
               "logMinPdealyReqInterval" : -3,
               "peerMeanPathDelay" : 0,
               "portId" : 1,
               "roleStatus" : "Slave",
               "roleStatusId" : 9,
               "syncInterval" : -3
          }
     ],
     "profile" : "SMPTE:2059-2",
     "stepsRemoved" : 1,
     "slaveOnly" : true,
     "success" : true
}
```


The PTP parameters for Device A have been identified which can be found under the `"local"` and `"port"` objects. Grandmaster information is listed under the "grandMaster" parameter. Since this example uses only the PRI port on the devices the `"port"` object array only lists one port for the device. The command can now be sent to Device B.

There may be some PTP parameters that need to be changed. In that case, the PTP parameters will be changed on the devices using the `ptp_set_parameter` command. The command will be sent to Device A first. It is important to note that the PTP profile must be set first then followed by a command to set the remaining parameters.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "profile" : "smpte"
     }
}
```

This command message set the device to use the `SMPTE 2059-2` PTP profile to conform to the ST 2110 standard.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "domain" : 127,
          "priotity1" : 128,
          "priotity2" : 128,
          "port" : 1,
          "announceInterval" : -2,
          "announceReceiptTimeout" : 3,
          "syncInterval" : -3,
          "slaveOnly" : true
     }
}
```

This command message set the device to use domain `127` and to a `"priority1"` and `"priority2"` of `128` on port `1`. Even though the `"priority1"` is set to `128` in this command message that action is nullified by setting `"slaveOnly"` to `true` which sets the `"priority1"` to `255`.

It is important that the two devices share the same PTP configuration so that there is a reliable audio stream. Send the same command to Device B.

3.1.4 Saving the Running Configuration

The ST 2110 interface on these devices allow for up to ten stored device configurations that can be loaded at any time. These configurations include system parameters, global configurations, and even created sessions. This allows for quick and easy startup for repeated setups. To do this the `filecfg_save_running` command should be sent. In this example, the currently running configurations will be saved to the devices in the first configuration memory slot and set as the start-up configuration for future device boots. The Device A configuration will be saved first.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "filecfg_save_running",
     "json" :
     {
          "pos" : 0,
          "name" : "M5512_config",
          "startup" : true
     }
}
```

The configuration is to be saved as `"M5512_config"` in the zeroth configuration memory location and will now be the start-up configuration.

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

Send the same message to Device B at `192.168.0.100`.

It is not a bad idea to save a new configuration once all sessions and streams are established on the device. This will ensure that at the end of a production, or in the case of a power failure, the device will resume the same function on reboot.

3.1.5 Creating Source and Destination Sessions

Before audio can be transmitted between devices source (sender) and destination (receiver) sessions must be established on both devices. These sessions act as starting and ending points for audio streams between devices.

Source sessions must be established first, as destination sessions require a remote source session to connect to which ultimately creates the stream. To add a source session the `stream_add_source` command will be used. On each device a 2-channel source session will be created with a `L24` codec. The session names will be `"M5512-A-src-session"` and `"M5512-B-src-session"` for Device A and B, respectively, and have audio source names of `"M5512-A-src-stream"` and `"M5512-B-src-stream"`. A source session will be created on Device A first.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "stream_add_source",
     "json" :
     {
          "session-id" : 0,
          "name" : "M5512-A-src-session",
          "audio-sources" :
          [
               {
                    "channel-list" : [0, 1],
                    "id" : 0,
                    "type" : "audio",
                    "codec" : "L24",
                    "dscp" : 34,
                    "name" : "M5512-A-src-stream",
                    "ptime" : 0,
                    "payload-type" : 98,
                    "transport" : 0,
                    "ttl" : 64,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "redundant-type" : "none"
               }
          ]
     }
}
```

It is important to note that "ptime" is set to 0 which tells the device to use the global ptime. "transport" is set to 0 which tells the device to assign a transport address automatically. The same can be said for "udp" which is set to -1 and the session will be assigned a UDP port automatically. "media-clk-offset" is set to 0 to be compliant with ST 2110-10. "dscp", "payload-type", and "ttl" are all set to default values.

The response from Device A is:


```
{
     "command" : "stream_add_source",
     "error" : 0,
     "error_string" : "",
     "session-id" : 9281879910736,
     "name" : "M5512-A-src-session",
     "audio-sources" :
     [
          {
               "channel-list" :
               [
                    {
                         "id" : 0,
                         "position" : 0,
                         "type" : "tdm"
                    },
                    {
                         "id" : 1,
                         "position" : 1,
                         "type" : "tdm"
                    },
               ],
               "id" : 9281888299480,
               "type" : "audio",
               "codec" : "L24",
               "dscp" : 34,
               "name" : "M5512-A-src-stream",
               "ptime" : 0,
               "payload-type" : 98,
               "transport" : "239.173.59.230",
               "ttl" : 64,
               "udp" : 5004,
               "media-clk-offset" : 0,
               "redundant-type" : "primary"
          }
     ]
}
```

The response from Device A is almost the same as the command message with the addition of the response message `error` and `error_string` parameters. The new `"session-id"` is returned as `9281879910736` and the new `"audio-sources" "id"` is returned as `9281888299480`. The transport address and UDP port were also set by the device.

Repeat these steps to get the `"session-id"` and `"audio-sources" "id"` for Device B. For Device B in this example the `"session-id"` is `9600022684968` and the `"audio-sources" "id"` is `9600023426852`.

In this example the session IDs are known, however, in most applications the session IDs will be unknown. It is good practice to query for all advertised source streams on the network and the appropriate remote session IDs can then be chosen for the destination sessions that will be initialized. To do this the `get_advertised_session_list` command should be sent to the devices.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "get_advertised_session_list",
     "json" : ""
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "get_advertised_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "host" :
               {
                    "address" : "local",
                    "ip" : "192.168.0.99"
               },
               "inuse" : false,
               "sdp-raw" : "v=0\no=- 9281879910736 0 IN IP4
192.168.0.99\ns=M5512-A-src-session\nt=0 0\nm=audio 5004 RTP/AVP
98\ni=M5512-A-src-stream\nc=IN IP4
239.173.59.230/64\na=clock-domain:PTPv2
0\na=ts-refclk:ptp=IEEE1588-2008:ec-46-70-ff-fe-00-9f-0c:0\na=media
```

```
clk:direct=0\na=source-filter: incl IN IP4 239.173.59.230
192.168.0.99\na=rtpmap:98
L24/48000/2\na=framecount:6\na=ptime:0.125\n",
               "session" :
               {
                    "id" : 9281879910736,
                    "info" : "M5512-A-src-session",
                    "local-ip" : "192.168.0.99",
                    "media" :
                    [
                         {
                              "clock-is-ptp" : true,
                              "group" : "",
                              "info" : "M5512-A-src-stream",
                              "local-ip" : "192.168.0.99",
                              "mediaclk" : 0,
                              "number-samples" : 6,
                              "protocol" : "RTP/AVP",
                              "ptime" : 0.125,
                              "ptp" :
                              {
                                   "domain" : 0,
                                   "gmid" :
"ec-46-70-ff-fe-00-9f-0c"
                              },
                              "rtpmap" :
                              [
                                   {
                                        "channel-freq" : 48000,
                                        "codec" : "L24",
                                        "number-channels" : 2,
                                        "payload-type" : 98
                                   }
                              ],
                              "transport-ip" : "239.173.59.230",
                              "ttl" : 64,
                              "type" : "AUDIO",
                              "udp" : 5004,
                              "video" :
                              {
                                   "colorimetry" : "",
                                   "depth" : 0,
                                   "sampling" : ""
                              }
                         }
                    ],
```

```
                    "protocol-version" : 0,
                    "time" :
                    {
                         "start" : 0,
                         "stop" : 0
                    },
                    "user-name" : "-",
                    "version" : 0
               },
               "session-id" : 9281879910736,
               "session-id-str" : "9281879910736",
               "status" : "OK",
               "url" : "rtsp://192.168.0.99:8554/by-name
                        /M5512-A-src-stream/"
          },
          {
               "host" :
               {
                    "address" : "remote",
                    "ip" : "192.168.0.100"
               },
               "inuse" : false,
               "sdp-raw" : "v=0\no=- 96000226849680 IN IP4
192.168.0.100\ns=M5512-B-src-session\nt=0 0\nm=audio 5004 RTP/AVP
98\ni=M5512-B-src-stream\nc=IN IP4
239.255.65.170/64\na=clock-domain:PTPv2
0\na=ts-refclk:ptp=IEEE1588-2008:ec-46-70-ff-fe-00-9f-0c:0\na=media
clk:direct=0\na=source-filter: incl IN IP4 239.255.65.170
192.168.0.100\na=rtpmap:98
L24/48000/2\na=framecount:6\na=ptime:0.125\n",
               "session" :
               {
                    "id" : 9600022684968,
                    "info" : "M5512-B-src-session",
                    "local-ip" : "192.168.0.100",
                    "media" :
                    [
                         {
                              "clock-is-ptp" : true,
                              "group" : "",
                              "info" : "M5512-B-src-stream",
                              "local-ip" : "192.168.0.100",
                              "mediaclk" : 0,
                              "number-samples" : 6,
                              "protocol" : "RTP/AVP",
                              "ptime" : 0.125,
```

```
                              "ptp" :
                              {
                                   "domain" : 0,
                                   "gmid" :
"ec-46-70-ff-fe-00-9f-0c"
                              },
                              "rtpmap" :
                              [
                                   {
                                        "channel-freq" : 48000,
                                        "codec" : "L24",
                                        "number-channels" : 2,
                                        "payload-type" : 98
                                   }
                              ],
                              "transport-ip" : "239.255.65.170",
                              "ttl" : 64,
                              "type" : "AUDIO",
                              "udp" : 5004,
                              "video" :
                              {
                                   "colorimetry" : "",
                                   "depth" : 0,
                                   "sampling" : ""
                              }
                         }
                    ],
                    "protocol-version" : 0,
                    "time" :
                    {
                         "start" : 0,
                         "stop" : 0
                    },
                    "user-name" : "-",
                    "version" : 0
               },
               "session-id" : 9600022684968,
               "session-id-str" : "9600022684968",
               "status" : "OK",
               "url" : "rtsp://192.168.0.100:8554/by-name
                        /M5512-B-src-stream/"
          }
     ],
     "success" : true
}
```

The response shows both advertised source sessions on the network, one session from each of the devices. The same command can be sent to Device B, however the response will be the same with the exception that the local and remote hosts will change in the message.

Now that the session IDs are known destination sessions can be initialized. To do this the `stream_add_destination` command should be sent to both of the devices. Device A will be sent the message first.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "stream_add_destination",
     "json" :
     {
          "channel-list" : [0, 1],
          "type" : "audio",
          "codec" : "L24",
          "link-offset" : 0,
          "name" : "M5512-A-dst-session",
          "rem-sid" :
          {
               "id" : 9600022684968,
               "media-idx" : 0,
               "payload-type" : 98
          }
     }
}
```

Since the destination session is being created on Device A `"id"` in `"rem-sid"` must be set to the session ID of Device B, `9600022684968`. The name of the destination session will be `"M5512-A-dst-session"`. Note that the session has two channels just as the source session does.

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "stream_add_destination",
     "error" : 0,
     "error_string" : "",
     "id" : 19366704770408,
     "success" : true
}
```

The destination session was created by Device A and the new destination session ID is `19366704770408`.

The same command can be sent to Device B but the name should be set to `"M5512-B-dst-session"` and the `"id"` in `"rem-sid"` should be set to `9281879910736` as that is the source session ID for Device A. Once this is completed there will be two streams on the network, one where Device A is the source and Device B is the destination, and the other where Device B is the source and Device A is the destination.

At this point it is a good idea to save the configuration of each device to ensure the session configurations remain the same.

3.1.6 Updating Session States

Once streams are established between sessions the sessions should be updated to ensure that audio is flowing. Typically, the source streams will be paused when they are intialized. In order to unpause the source sessions the `update_session_state` command should be sent to the devices. The source session IDs from each device will be needed.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "update_session_state",
     "json" :
     {
          "id" : 9281879910736,
          "pause" : false
     }
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "update_session_state",
     "error" : 0,
     "error_string" : "",
     "id" : 9281879910736,
     "success" : true
}
```

The source session on Device A will now be unpaused. The same command should now be sent to Device B using its source session ID.

3.1.7 Listing Local Sessions

It is good practice to check what sessions exist on a local device as well as seeing each session's status. To do this two different commands can be sent to each device to see the source and destination sessions. The `get_source_session_list` will list all source sessions on a specific device and `get_destination_session_list` will list all destination sessions on a specific device. The source session list will be requested from Device A first.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "get_source_session_list",
     "json" : ""
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "get_source_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "name" : "M5512-A-src-session",
               "remote-sid" : 0,
               "remote-sid-str" : "0",
               "session" : 9281879910736,
               "state" : "Enabled",
               "stream" : 0,
               "type" : "audio"
          }
     ],
     "success" : true
}
```

The `"list"` parameter is an object array that would list multiple source session objects if there were more than one source session initialized on the device. In this example however, there is only one source session and it is currently in the `"Enabled"` state as it was unpaused in the previous section.

The same command can be sent to Device B and the only differences will be the "name" and "session" parameters as they will reflect the settings for Device B.

The destination session list can now be requested from Device A. The command and response are very similar to that of requesting the source session list.

The following JSON message will be sent to http://192.168.0.99/cgi-bin/handleCommands using the HTTP POST method and the header should include a content-type of 'application/json':

```
{
     "command" : "get_destination_session_list",
     "json" : ""
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "get_destination_session_list",
     "error" : 0,
     "error_string" : "",
     "list" :
     [
          {
               "name" : "M5512-A-dst-session",
               "remote-sid" : 9600022684968,
               "remote-sid-str" : "9600022684968",
               "session" : 19366704770408,
               "state" : "Enabled",
               "stream" : 0,
               "type" : "audio"
          }
     ],
     "success" : true
}
```

It is important to note that with the response to the destination session list request the remote session ID is also provided. The command can now be sent to Device B.

3.1.8 Removing Local Sessions

Inevitably new sessions will need to be created. Because of this, sessions must be removed to make way for new ones. The `session_remove` command will remove the specified session from the device. Typically this will be used with the `get_source_session_list` or `get_destination_session_list` commands to identify which session should be removed. The Device A destination session will be removed in this example.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "session_remove",
     "json" :
     {
          "id" : 19366704770408
     }
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "session_remove",
     "error" : 0,
     "error_string" : "",
     "id" : 19366704770408,
     "success" : true
}
```

This response will remove the stream between Device B and Device A. The source session on Device B will still remain.

3.1.9 Configuring Device for Redundancy

In most professional applications there is the need for redundant streams to ensure a production continues if the primary feed degrades or is lost. **By default the device should already be configured for redundancy**, but in the case the device is not, this example can be used. In order to configure the device for redundancy the `sysmgr_set_system_parameter` command is used. Device A will be configured in this example.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "redundancy-enable" : true
     }
}
```

The response from Device A is:

```
{
     "auxmsg" : null,
     "command" : "session_remove",
     "error" : 0,
     "error_string" : "",
     "id" : 19366704770408,
     "success" : true
}
```

Now that the device has been configured for redundancy the network interfaces and PTP settings for each port need to be configured.

3.1.10 Configuring Network Interfaces for Redundancy

When redundancy is enabled on the device the essence networks are split into three VLANs: `eth0`, `eth0.10`, and `eth0.20`. Only `eth0.10` and `eth0.20` are utilized but each VLAN needs to be configured. In order to do this the `sysmgr_set_system_parameter` command must be sent to the device three times, once for each VLAN.

For Device A the static Primary IP address (`eth0.10`) will be set to `192.168.0.99` with a subnet mask of `255.255.255.0`.

The following JSON message will be sent to `http://[DEVICE_IP]/cgi-bin/handleCommands` using the HTTP POST method (`[DEVICE_IP]` can be obtained from the front-panel display on the device under ST 2110 PRI IP Addr on Device A) and the header should include a content-type of `'application/json'`:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "add-eth-cfg" :
          {
               "eth" : "eth0.10",
               "mode" :
               {
                    "mode" : "static"
               },
               "static" :
               {
                    "ip" : "192.168.0.99",
                    "mask" : "255.255.255.0"
               }
          }
     }
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

The static Secondary IP address (`eth0.20`) will be set to `192.168.1.99` with a subnet mask of `255.255.255.0` for Device A.

The following JSON message will be sent to `http://[DEVICE_IP]/cgi-bin/handleCommands` using the HTTP POST method (`[DEVICE_IP]` can be obtained from the front-panel display on the device under ST 2110 PRI IP Addr on Device A) and the header should include a content-type of `'application/json'`:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "add-eth-cfg" :
          {
               "eth" : "eth0.20",
               "mode" :
               {
                    "mode" : "static"
               },
               "static" :
               {
                    "ip" : "192.168.1.99",
                    "mask" : "255.255.255.0"
               }
          }
     }
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

Finally, the `eth0` IP address needs to be set to `255.255.255.255` with a subnet mask of `255.255.255.255` for Device A.

The following JSON message will be sent to `http://[DEVICE_IP]/cgi-bin/handleCommands` using the HTTP POST method (`[DEVICE_IP]` can be obtained from the front-panel display on the device under ST 2110 PRI IP Addr on Device A) and the header should include a content-type of `'application/json'`:

```
{
     "command" : "sysmgr_set_system_parameter",
     "json" :
     {
          "add-eth-cfg" :
          {
               "eth" : "eth0",
               "mode" :
               {
                    "mode" : "static"
               },
               "static" :
               {
                    "ip" : "255.255.255.255",
                    "mask" : "255.255.255.255"
               }
          }
     }
}
```

The response from Device A is:

```
{
     "error" : 0,
     "error_string" : "",
     "success" : true
}
```

Be sure to reboot the device so IP address changes can take effect.

3.1.11 Configuring PTP for Redundancy

Since PTP is essential for a seamless transition between streams PTP must be configured on each of the networks. The PTP parameters will be changed on the devices using the `ptp_set_parameter` command. The command will be sent to Device A four times to configure a number of parameters.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "profile" : "smpte"
     }
}
```

This command message set the device to use the `SMPTE 2059-2` PTP profile to conform to the ST 2110 standard.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "slaveOnly" : true
     }
}
```

This command message sets the device to be slave only as these devices have not been designed to be dedicated master clocks (however, they can function as one as needed).

Finally, the PTP parameters need to be configured for each interface independently. First, the Primary Interface will be configured, followed by the Secondary.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "domain" : 0,
          "port" : 1,
          "announceInterval" : 0,
          "announceReceiptTimeout" : 3,
          "logMinDelayReqInterval": -3,
          "syncInterval" : -3
     }
}
```

The Primary Interface's PTP has been configured. Next is the Secondary Interface.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "ptp_set_parameter",
     "json" :
     {
          "domain" : 0,
          "port" : 2,
          "announceInterval" : 0,
          "announceReceiptTimeout" : 3,
          "logMinDelayReqInterval": -3,
          "syncInterval" : -3
     }
}
```

Be sure to save the running configuration so that the PTP parameters are retained.

3.1.12 Creating Redundant Source Sessions

This example is based around the example found in section 3.1.5 Creating Source and Destination Sessions. All the steps should be followed in the same order, however the command to create redundant streams is different. This example will create redundant source sessions on Device A.

The following JSON message will be sent to `http://192.168.0.99/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "stream_add_source",
     "json" :
     {
          "session-id" : 0,
          "name" : "M5512-A-src-session",
          "audio-sources" :
          [
               {
                    "channel-list" : [0, 1],
                    "id" : 0,
                    "type" : "audio",
                    "codec" : "L24",
                    "dscp" : 10,
                    "name" : "M5512-A-src-stream-pri",
                    "ptime" : 0,
                    "payload-type" : 98,
                    "transport" : "239.18.1.1",
                    "ttl" : 10,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "link-offset" : 2000,
                    "redundant-type" : "primary"
               },
               {
                    "channel-list" : [0, 1],
                    "id" : 0,
                    "type" : "audio",
                    "codec" : "L24",
                    "dscp" : 10,
                    "name" : "M5512-A-src-stream-sec",
                    "ptime" : 0,
                    "payload-type" : 98,
                    "transport" : "239.18.1.1",
```

```
                    "ttl" : 10,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "link-offset" : 2000,
                    "redundant-type" : "secondary"
                }
            ]
        }
}
```

Note that there is an additional `audio-sources` object. These objects are for primary and secondary sessions.

The transport address is defined in this example and **must be the same for both sessions**. It is good practice to define `transport` as it helps keep an organized network.

All settings for each object must be the same aside from `name` and `redundant-type`.

3.1.13 Creating Redundant Stream Destinations

This example is based around the example found in section 3.1.5 Creating Source and Destination Sessions. All the steps should be followed in the same order, however the command to create redundant streams is different. In this example, the more favorable `stream_add_destination_manually` command will be used, as the source `transport` can be defined, and there is no need to find the remote session ID. This example will create redundant destination sessions on Device B to receive the stream from Device A.

The following JSON message will be sent to `http://192.168.0.100/cgi-bin/handleCommands` using the HTTP POST method and the header should include a content-type of `'application/json'`:

```
{
     "command" : "stream_add_destination_manually",
     "json" :
     {
          "session-id" : 0,
          "name" : "",
          "audio-sources" :
          [
               {
                    "channel-list" : [0, 1],
                    "id" : 0,
                    "codec" : "L24",
                    "dscp" : 10,
                    "name" : "M5512-B-dst-stream-pri",
                    "ptime" : 0,
                    "payload-type" : 98,
                    "transport" : "239.18.1.1",
                    "ttl" : 10,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "link-offset" : 2000,
                    "redundant-type" : "primary"
               },
               {
                    "channel-list" : [0, 1],
                    "id" : 0,
                    "codec" : "L24",
                    "dscp" : 10,
                    "name" : "M5512-B-dst-stream-sec",
                    "ptime" : 0,
                    "payload-type" : 98,
                    "transport" : "239.18.1.1",
```

```
                    "ttl" : 10,
                    "udp" : -1,
                    "media-clk-offset" : 0,
                    "link-offset" : 2000,
                    "redundant-type" : "secondary"
                }
            ]
        }
}
```

Note that there is an additional `audio-sources` object. These objects are for primary and secondary sessions.

All settings for each object must be the same aside from `name` and `redundant-type`.

4 Appendix B

4.1 Command Notes

The following commands may be a part of the web user interface on ST 2110 capable products produced by Studio Technologies, Inc.:

To be determined.

5 Appendix C

5.1 Postman

Any software capable of transmitting and receiving JSON is acceptable for sending the commands in this document. However, if there is a need for software capable of performing these tasks Postman API Development Environment can be downloaded for free from https://www.getpostman.com/apps.

The Studio Technologies team has created example collections (groups of commands) that can be imported into Postman so that the user does not have to rewrite the commands from scratch. Please feel free to contact support at www.studio-tech.com and request the JSON collections.

6 Appendix D

6.1 Change Log

v0.4 | 06 Jan 2019

- `codec`: All instances of this description have been changed to state that the `"L24"` format should be selected.
- 3.1.9 Configuring Device for Redundancy: Added documentation on configuring devices for redundancy.
- 3.1.10 Configuring Network Interfaces for Redundancy: Added documentation on configuring the network interfaces for redundancy.
- 3.1.11 Configuring PTP for Redundancy: Added documentation on configuring PTP for redundancy.
- 3.1.12 Creating Redundant Source Sessions: Added documentation on adding redundant source sessions.
- 3.1.13 Creating Redundant Destination Sessions: Added documentation on adding redundant destination sessions.
- Document Formatting: 5 Appendix C, 5.1 Change Log changed to 6 Appendix D, 6.1 Change Log.
- Document Formatting: 5 Appendix C, 5.1 Postman added to document the use of the Postman JSON API Environment.
- Document Information: JSON API based on minuet_aes67_release-v.2.2.0-104.

v0.3 | 03 Aug 2018

- 2.6.1 filecfg_save_running: `"pos"` cannot be `10`. This position is reserved for the factory default configuration.
- 3.1.3 Requesting and Configuring PTP Parameters: When setting the PTP profile the profile must be set in one command and any other parameters must be sent in another command.
- Document Formatting: ″ changed to `""`.

v0.2 | 11 Jul 2018

- 2.3.2 ptp_set_parameter: `"ptp-profile"` changed to `"profile"`.
- 2.4.2 stream_add_destination: Added `"user-sdp"` string parameter.
- 2.4.3 stream_add_destination_manually: Combined `"sources"` and `"audio-sources"` object array parameters.
- 2.4.4 stream_add_source: Combined `"sources"` and `"audio-sources"` object array parameters.

v0.1 | 25 Jun 2018

- Initial Release.

This page has been intentionally left blank.