MIDRA™ 4K

REST API Programmer's Guide

For version v2.00.20 or higher

ANALOG WAY®

Pioneer in Analog, Leader in Digital

Table of contents

1. Presentation 4
1.1. Description 4
1.2. Server address 4
1.3. HTTP Requests 4
1.4. HTTP Statuses 4
1.5. HTTP Responses 5
1.6. HTTP Parameters 5
1.7. GET request diagram 5
1.8. POST request diagram 6
2. System commands 7
2.1. Reading system information 7
2.2. Rebooting the system 9
2.3. Shutting down the system 10
2.4. Waking up the system 11
3. Screen commands 12
3.1. Reading screen information 12
3.2. Recalling a preset from memory to a single screen 13
3.4. Recalling a master preset from memory 14
3.5. Reading a live layer status 15
3.6. Setting a live layer source 16
3.7. Reading background layer status 17
3.8. Setting a background layer source 18
3.9. Reading foreground layer status 19
3.10. Setting a foreground layer source 20
3.11. Single TAKE: Transitioning the Preview content to the Program (single screen) 21
3.12. Global TAKE: Transitioning the Preview content to the Program (multiple screens) 22
4. Auxiliary screen commands 23
4.1. Reading auxiliary screen information 23
4.2. Recalling a preset from memory to a single auxiliary screen 24
4.3. Reading the source of an auxiliary screen 25
4.4. Setting the source of an auxiliary screen 26
4.5. TAKE: Transitioning the Preview content to the Program (single auxiliary screen) 27
5. Multiviewer commands 28
5.1. Reading multiviewer output information 28

5.2. Recalling a preset from memory to a multiviewer output 29
5.3. Reading the source of a multiviewer output widget 30
5.4. Reading the status of a multiviewer output widget 31
5.5. Setting the source of a multiviewer output widget 32
6. Audio commands 33
6.1. Reading the output audio mode 33
6.2. Setting the output audio mode 34
6.3. Reading the output audio source 35
6.4. Reading the screen audio mode status 36
6.5. Setting the screen audio mode 37
6.6. Reading the screen audio preset status 38
6.7. Reading the source of the screen audio-layer preset 39
6.8. Setting the source of the screen audio-layer preset 40
6.9. Reading the auxiliary screen audio mode status 41
6.10. Setting the screen audio mode 42
6.11. Reading the auxiliary screen audio preset status 43
6.12. Reading the source of the auxiliary screen audio-layer preset 44
6.13. Setting the source of the auxiliary screen audio-layer preset 45
6.14. Reading the dante output audio mode 46
6.15. Setting the dante output audio mode 47
6.16. Reading the dante output audio source 48
6.17. Reading the line-out output audio mode 49
6.18. Setting the line-out output audio mode 50
6.19. Reading the line-out output audio source 51
7. Source commands 52
7.1. Reading input information 52
7.2. Reading foreground image information 53
7.3. Reading background image information 54
7.4. Reading background set information 55
8. Using thumbnails 56
8.1. Introduction 56
8.2. Live inputs thumbnails URL 56
8.3. Outputs thumbnails URL 56
8.4. Foreground Images thumbnails URL (per Screen) 56
8.5. Background Images thumbnails URL (per Screen) 56
8.6. Multiviewer thumbnails URL 56

9. Using Authentication 57
9.1. Introduction 57
9.2. Authenticating 57
9.3. Following REST API calls 58

1. Presentation

1.1. Description

The REST API for Midra™ 4K is a simple way for you to automate your interaction with the Midra™ 4K presentation systems. The REST API for Midra™ 4K is RESTful and HTTP-based. Basically, this means that the communication is made through normal HTTP requests.

1.2. Server address

The base server address is: http://<ipadress>/api/tpp/v1 where <ipadress> is the IP address of the Midra™ 4K presentation system.

1.3. HTTP Requests

HTTP requests can be made with tons of tools, each modern programming language has its own HTTP functions and libraries. The REST API for Midra™ 4K will handle each request in a meaningful manner, depending on the action required.

Method	Usage
GET	For simple retrieval of information about your screens, multiviewers, etc.., you should use the GET method. API will respond you with a JSON object. Using the returned information, you can eventually form additional requests. All the GET requests are made read-only, which means making a GET requests cannot change the state of any information stored on the Midra™ 4K presentation system.
POST	When you want to change an object property or trigger an action on the Midra™ 4K presentation system, you should choose POST method. The POST request includes all of the attributes necessary to change the desired object property or trigger the action.

1.4. HTTP Statuses

When you make a request to the API, you will get a response including the data you want with standard HTTP statuses, including error codes.

In case of an unusual event, such as trying to recall a preset memory index that does not exist on the Midra™ 4K presentation system, the status code will have an error code. Besides that, the body of the request will contain additional information about the event to provide you the most conventional way to fix the flow. To make it clear, status codes are usually in between 2XX-4XX range.

Code	Message	Description
200	OK	Successful operation (with content)
204	No Content	Successful operation (no content)
400	Bad Request	Syntax invalid
404	Not Found	The specified resource could not be found
405	Method Not Allowed	The specified request method is not allowed
409	Internal Server Error	Error reported by the server

1.5. HTTP Responses

For each successful and unsuccessful request, a JSON-formatted response body will be sent back. If you make a request for a single object, say, for a screen, the resource root will be a single object containing the data you requested. If you request a collection, say, a group of screens, response body will contain a collection.

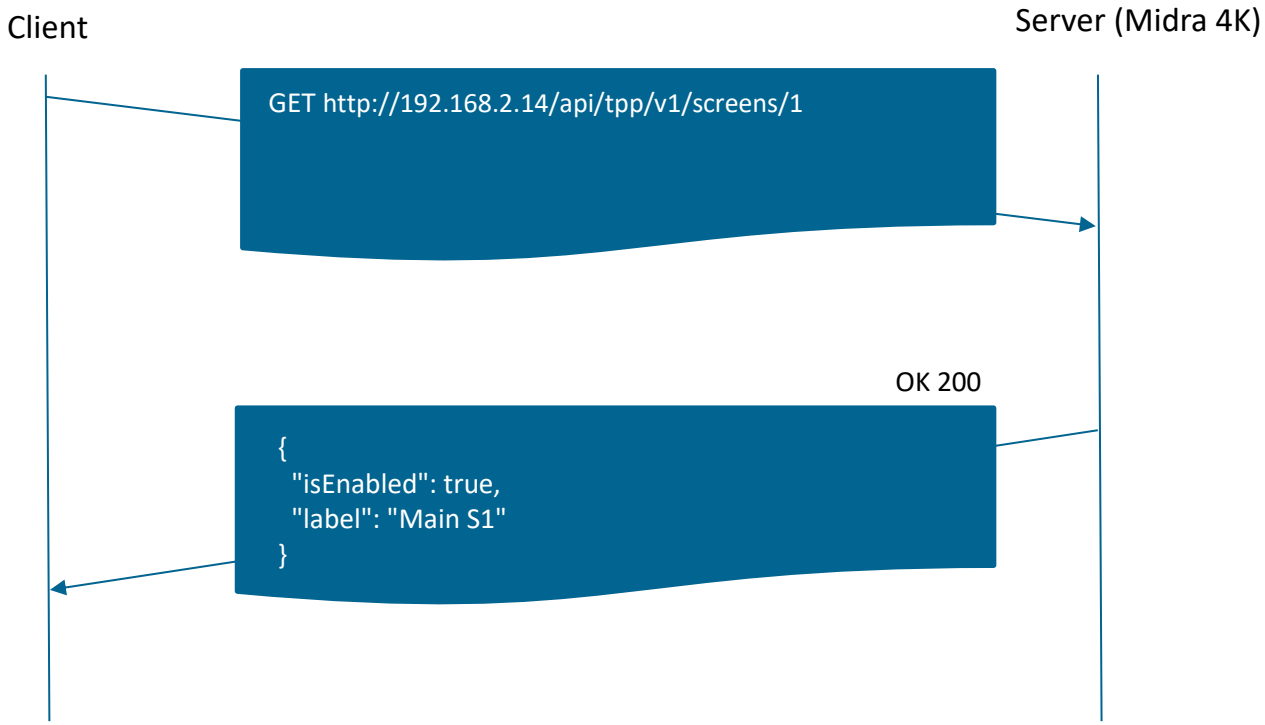
1.6. HTTP Parameters

Most of HTTP GET requests need query string parameters.

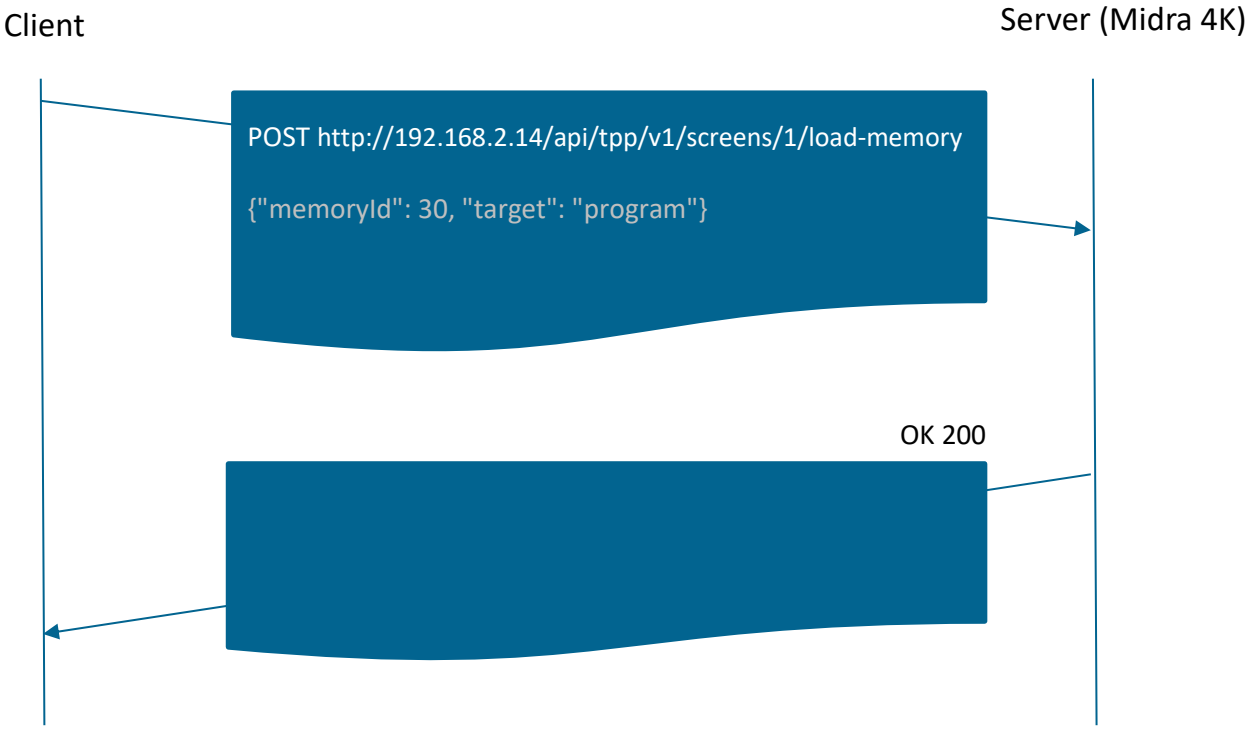
Most of HTTP POST requests need POST body parameters.

For all API requests, we provide examples calls with .NET C# command and raw TCP. With a single copy and paste, you can always try making a request and see the results.

1.7. GET request diagram



1.8. POST request diagram



2. System commands

2.1. Reading system information

GET	/api/tpp/v1/system

Response

produces: application/json

Name	Type	Description
type	string	the type of Midra 4K device: ‘QVU 4K’, ‘QMX 4K’, ‘PLS 4K’ or ‘EKS 4K’
label	string	the device label
version	json	a JSON object containing the current firmware version (see below)

Field ‘version’

produces: application/json

Name	Type	Description
major	integer	firmware major version number
minor	integer	firmware minor version number
patch	integer	firmware patch version number
beta	boolean	true is the firmware version is a beta version, false if not

Response example

```
{
  "type": "PLS 4K",
  "label": "Meeting Room",
  "version": {
    "major": 1,
    "minor": 0,
    "patch": 23,
    "beta": false
  }
}
```

Example: Read system information

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/system HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/system");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

2.2. Rebooting the system

POST	/api/tpp/v1/system/reboot

Example: Reboot the system

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/system/reboot HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/system/reboot");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   streamWriter.Write(“”);
   streamWriter.Flush();
}
```

2.3. Shutting down the system

POST /api/tpp/v1/system/shutdown

Body

consumes: application/json

Name	Type	Optional	Description
standby	boolean	No	true to activate the standby mode

Example: Shutdown the system

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/system/shutdown HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 17<CR><LF><CR><LF>{"standby": true}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"standby": true}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/system/shutdown");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { standby = "true" });
   streamWriter.Write(json);
}
```

2.4. Waking up the system

POST	/api/tpp/v1/system/wakeup

Example: Wake up the system

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/system/wakeup HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/system/wakeup");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   streamWriter.Write(“”);
   streamWriter.Flush();
}
```

3. Screen commands

3.1. Reading screen information

GET /api/tpp/v1/screens/{screenId}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)

Response

produces: application/json

Name	Type	Description
isEnabled	boolean	true if the screen is enabled, false if not
label	string	the screen label
layerMode	string	the layer mode (“mixing layers” or “split layers”)

Response example

```
{
  "isEnabled": true,
  "label": "Center"
  "layerMode": "mixing"
}
```

Example: Read screen 2 information

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/2 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

3.2. Recalling a preset from memory to a single screen

POST /api/tpp/v1/screens/{screenId}/load-memory

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)

Body

consumes: application/json

Name	Type	Optional	Description
memoryId	integer	No	the memory index (from 1 to 200)
target	string	Yes	the destination (“program” or “preview”). Default is “preview”

Example: Recall preset 30 to screen 2 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/2/load-memory HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 37<CR><LF><CR><LF>{"memoryId": 30, "target": "preview"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"memoryId": 30, "target": "preview"}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/load-memory");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { memoryId = 30, target = "preview" });
   streamWriter.Write(json);
}
```

3.4. Recalling a master preset from memory

POST /api/tpp/v1/load-master-memory

Body

consumes: application/json

Name	Type	Optional	Description
memoryId	integer	No	the memory index (from 1 to 50)
target	string	Yes	the destination (“program” or “preview”). Default is “preview”

Example: Recall master preset 10 to Preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/load-master-memory HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 37<CR><LF><CR><LF>{"memoryId": 10, "target":
"preview"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"memoryId": 10, "target": "preview"}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/load-master-
memory");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { memoryId = 10,  target = "preview" });
   streamWriter.Write(json);
}
```

3.5. Reading a live layer status

GET /api/tpp/v1/screens/{screenId}/live-layers/{layerId}/presets/{target}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
layerId	integer	the layer number (from 1 to 4)
target	string	the destination (“program” or “preview”)

Response

produces: application/json

Name	Type	Description
status	string	the layer status: "off", "open", "close", "cross", "flying", "flying depth"
sourceType	string	the type of source: "none", "color" or "input"
sourceId	integer	the source number

Response example

```
{
  "status": "open",
  "sourceType": "input",
  "sourceId": 8
}
```

Example: Read layer 4 current status on screen 1 preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/live-layers/4/presets/preview HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/live-layers/4/presets/preview");
var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

3.6. Setting a live layer source

POST /api/tpp/v1/screens/{screenId}/live-layers/{layerId}/presets/{target}/source

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
layerId	Integer	the layer number (from 1 to 4)
target	string	the destination (“program” or “preview”)

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the type of source: "none", "color", "input"
sourceId	integer	No	the source number

Example: Set screen 1 layer 3 source to live input 3 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/1/live-layers/3/presets/preview/source HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 38<CR><LF><CR><LF>{"sourceType": "input", "sourceId": 3}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "input", "sourceId": 3}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/live-layers/3/presets/preview/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { sourceType = "input", sourceId = 3 });
   streamWriter.Write(json);
}
```

3.7. Reading background layer status

GET /api/tpp/v1/screens/{screenId}/background-layer/presets/{target}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
target	string	the destination (“program” or “preview”)

Response

produces: application/json

Name	Type	Description
status	string	The layer status: "off", "open", "close"
sourceType	string	the type of source: "background-set" or "none"
sourceId	integer	the background set number (from 1 to 8)

Response example

```
{
  "status": "open",
  "sourceType": "background-set",
  "sourceId": 2
}
```

Example: Read background layer status on screen 1 preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/background-layer/presets/preview HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/background-
layer/presets/preview");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

3.8. Setting a background layer source

POST /api/tpp/v1/screens/{screenId}/background-layer/presets/{target}/source

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
target	string	the destination (“program” or “preview”)

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the type of source: "background-set" or "none"
sourceId	integer	No	the background source number (from 1 to 8)

Example: Set screen 2 background to background set 3 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/2/background-layer/presets/preview/source HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 47<CR><LF><CR><LF>{"sourceType": "background-set", "sourceId": 3}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "background-set", "sourceId": 3}

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/background-layer/presets/preview/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { sourceType = "background-set", sourceId = 3 });
   streamWriter.Write(json);
}
```

3.9. Reading foreground layer status

GET /api/tpp/v1/screens/{screenId}/foreground-layer/presets/{target}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
target	string	the destination (“program” or “preview”)

Response

produces: application/json

Name	Type	Description
status	string	The layer status: "off", "open", "close"
sourceType	string	the type of source: "foreground-image" or "none"
sourceId	integer	the foreground preset number (from 1 to 4)

Response example

```
{
  "status": "open",
  "sourceType": "foreground-image",
  "sourceId": 2
}
```

Example: Read foreground layer status on screen 1 preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/foreground-layer/presets/preview HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/foreground-
layer/presets/preview");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

3.10. Setting a foreground layer source

POST /api/tpp/v1/screens/{screenId}/foreground-layer/presets/{target}/source

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
target	string	the destination (“program” or “preview”)

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the type of source: "foreground-image" or "none"
sourceId	integer	No	the foreground source number (from 1 to 4)

Example: Set screen 2 foreground to foreground preset 3 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/2/foreground-layer/presets/preview/source HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 49<CR><LF><CR><LF>{"sourceType": "foreground-image", "sourceId": 3}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "foreground-image", "sourceId": 3}

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/foreground-layer/presets/preview/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { sourceType = "foreground-image", sourceId = 3 });
   streamWriter.Write(json);
}
```

3.11. Single TAKE: Transitioning the Preview content to the Program (single screen)

POST /api/tpp/v1/screens/{screenId}/take

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)

Example: TAKE screen 2

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/2/take HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/take");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   streamWriter.Write(“”);
}
```

3.12. Global TAKE: Transitioning the Preview content to the Program (multiple screens)

POST /api/tpp/v1/take

Body

consumes: application/json

Name	Type	Optional	Description
screenIds	list	No	list of screen indexes that will be transitioned
auxiliaryScreenIds	list	No	list of auxiliary screen indexes that will be transitioned

Example: Take screen 1 and auxiliary screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/take HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 46<CR><LF><CR><LF>{"screenIds": [1], "auxiliaryScreenIds ": [1]}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"screenIds": [1], "auxiliaryScreenIds ": [1]}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/take");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   var screenIdList = new List<int>() { 1 };
   var auxIdList = new List< int>() { 1 };
   string json = new JavaScriptSerializer().Serialize(new { screenIds = screenIdList, auxiliaryScreenIds = auxIdList });
   streamWriter.Write(json);
}
```

4. Auxiliary screen commands

4.1. Reading auxiliary screen information

GET /api/tpp/v1/auxiliary-screens/{auxId}

Request

Name	Type	Description
auxId	integer	the auxiliary screen number (always 1, depending of the current mode)

Response

produces: application/json

Name	Type	Description
IsEnabled	boolean	true is the auxiliary screen is enabled, false if not
label	string	the auxiliary screen label

Response example

```
{
  "isEnabled": true,
  "label": "DSM"
}
```

Example: Read auxiliary screen 1 information

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/auxiliary-screens/1 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

4.2. Recalling a preset from memory to a single auxiliary screen

POST /api/tpp/v1/auxiliary-screens/{auxId}/load-memory

Request

Name	Type	Description
auxId	integer	the auxiliary screen number (always 1, depending of the current mode)

Body

consumes: application/json

Name	Type	Optional	Description
memoryId	integer	No	the memory index
target	string	Yes	the destination (“program” or “preview”). Default is “preview”

Example: Recall preset 5 to auxiliary screen 1 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/auxiliary-screens/1/load-memory HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 36<CR><LF><CR><LF>{"memoryId": 5, "target": "preview"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"memoryId": 5, “target”: “preview”}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/load-memory");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { memoryId = 5,  target = "preview" });
   streamWriter.Write(json);
}
```

4.3. Reading the source of an auxiliary screen

GET /api/tpp/v1/auxiliary-screens/{auxId}/background-layer/presets/{target}/source

Request

Name	Type	Description
auxId	integer	the auxiliary screen number (always 1, depending of the current mode)
target	string	the destination (“program” or “preview”)

Response

produces: application/json

Name	Type	Description
sourceType	string	the type of source: "none", "input" or "screen"
sourceId	integer	the source number

Response example

```
{
  "sourceType": "input",
  "sourceId": 5
}
```

Example: Read auxiliary screen 1 source (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET api/tpp/v1/auxiliary-screens/1/background-layer/presets/preview/source
HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/background-layer/presets/preview/source ");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

4.4. Setting the source of an auxiliary screen

POST /api/tpp/v1/auxiliary-screens/{auxId}/background-layer/presets/{target}/source

Request

Name	Type	Description
auxId	integer	the auxiliary screen number (always 1, depending of the current mode)
target	string	the destination (“program” or “preview”)

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the type of source: "none", "input" or "screen"
sourceId	integer	No	the source number (1 for screen, from 1 to 10 for inputs)

Example: Set the source of auxiliary screen 1 to input 5 (Preview)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/auxiliary-screens/1/background-layer/presets/preview/source
HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length:
38<CR><LF><CR><LF>{"sourceType": "input", "sourceId": 5}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "input", "sourceId": 5}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-
screens/1/background-layer/presets/preview/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { sourceType = "inputs", sourceId = 5 });
   streamWriter.Write(json);
}
```

4.5. TAKE: Transitioning the Preview content to the Program (single auxiliary screen)

POST /api/tpp/v1/auxiliary-screens/{auxId}/take

Request

Name	Type	Description
auxId	integer	the auxiliary screen number (from 1 to 1)

Example: Take auxiliary screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/auxiliary-screens/1/take HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/take");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   streamWriter.Write(“”);
}
```

5. Multiviewer commands

5.1. Reading multiviewer output information

GET /api/tpp/v1/multiviewer/

Response

produces: application/json

Name	Type	Description
isEnabled	boolean	true is the multiviewer output is enabled, false if not
label	string	the multiviewer output label

Response example

```
{
  "isEnabled": true,
  "label": "MVW1"
}
```

Example: Read multiviewer output 1 information

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/multiviewer/ HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

5.2. Recalling a preset from memory to a multiviewer output

POST /api/tpp/v1/multiviewer/load-memory

Body

consumes: application/json

Name	Type	Optional	Description
memoryId	integer	No	the memory index (from 1 to 20)

Example: Recall preset 20 to multiviewer output

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/multiviewer/load-memory HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 16<CR><LF><CR><LF>{"memoryId": 20}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"memoryId": 20}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/load-memory");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { memoryId = 20 });
   streamWriter.Write(json);
}
```

5.3. Reading the source of a multiviewer output widget

GET /api/tpp/v1/multiviewer/widgets/{widgetId}/source

Request

Name	Type	Description
widgetId	integer	the widget number (from 1 to 16)

Response

produces: application/json

Name	Type	Description
sourceType	string	the type of source: "none", "input", "screen-program", "screen-preview" or "timer"
sourceId	integer	the source number

Response example

```
{
  "sourceType": "input",
  "sourceId": 2
}
```

Example: Read the source of the widget 10 on multiviewer output

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/multiviewer/widgets/10/source HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/widgets/10/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

5.4. Reading the status of a multiviewer output widget

GET /api/tpp/v1/multiviewer/widgets/{widgetId}

Request

Name	Type	Description
widgetId	integer	the widget number (from 1 to 16)

Response

produces: application/json

Name	Type	Description
isEnabled	boolean	true if the widget is enabled, false if not

Response example

```
{
  "isEnabled": true
}
```

Example: Read the status of the widget 10 on multiviewer output

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/multiviewer/widgets/10 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/widgets/10");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

5.5. Setting the source of a multiviewer output widget

POST /api/tpp/v1/multiviewer/widgets/{widgetId}/source

Request

Name	Type	Description
widgetId	integer	the widget number (from 1 to 16)

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the type of source: "none", "input", "screen-program", "screen-preview" or "timer"
sourceId	integer	No	the source number

Example: Set the source of the widget 10 to input 7 on multiviewer output

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/multiviewer/widgets/10/source HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 38<CR><LF><CR><LF>{"sourceType": "input", "sourceId": 7}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "input", "sourceId": 7}

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/widgets/10/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new { sourceType = "input", sourceId = 7 });
   streamWriter.Write(json);
}
```

6. Audio commands

6.1. Reading the output audio mode

GET /api/tpp/v1/outputs/{outputId}/audio/mode

Request

Name	Type	Description
outputId	integer	the output number: either '1' or '2'

Response

produces: application/json

Name	Type	Description
mode	string	the output audio mode: "none", "auto" or "direct routing"
sourceType	string	the audio source type: "none", "input", "custom", "dante in", "line in"
sourceId	integer	the audio source id associated to the sourceType

'sourceType' and 'sourceId' are only displayed for "direct routing" mode

Response example

```
{
  "mode": "auto",
}
```

Example: Read the status of audio for multiviewer

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/multiviewer/audio HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/multiviewer/audio ");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.2. Setting the output audio mode

POST /api/tpp/v1/outputs/{outputId}/audio/mode

Request

Name	Type	Description
outputId	integer	the output number: either '1' or '2'

Body

consumes: application/json

Name	Type	Optional	Description
mode	String	No	the selected mode: "none", "auto" or "direct routing"
sourceType	string	Yes	the audio source type: "none", "input", "custom", "dante in", "line in"
sourceId	integer	Yes	the audio source id associated to the sourceType

'sourceType' and 'sourceId' are mandatory for "direct routing" mode, and not required for other modes

Example: Set input 5 in direct routing mode as source of output 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/outputs/1/audio/mode HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 64<CR><LF><CR><LF>{"mode": "direct routing",
"sourceType": "input", "sourceId": 5}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"mode": "direct routing", "sourceType": "input", "sourceId": 5}

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/outputs/1/audio/mode");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {mode = "direct routing", sourceType = "input", sourceId = 5});
   streamWriter.Write(json);
}
```

6.3. Reading the output audio source

GET /api/tpp/v1/outputs/{outputId}/audio/source

Request

Name	Type	Description
outputId	integer	the output number: either '1' or '2'

Response

produces: application/json

Name	Type	Description
sourceType	string	the audio source type: "none", "input", "custom", "dante in", "line in"
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
  "sourceType": "input",
  "sourceId": 10
}
```

Example: Read the source status of audio output 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/outputs/1/audio/source HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/outputs/1/audio/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.4. Reading the screen audio mode status

GET /api/tpp/v1/screens/{screenId}/audio/mode

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)

Response

produces: application/json

Name	Type	Description
mode	string	the audio mode status: “direct routing”, "follow audio layer" or "follow live layer content"
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType
layerId	integer	the layer for the “follow live layer content” mode

Response example

```
{
  "mode": "follow live layer content",
  "layerId": 1
}
```

Example: Read the status of audio mode for screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/audio/mode HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/audio/mode");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.5. Setting the screen audio mode

POST /api/tpp/v1/screens/{screenId}/audio/mode

Request

Name	Type	Description
screenId	integer	the screen number: either '1' or '2'

Body

consumes: application/json

Name	Type	Optional	Description
mode	String	No	the selected mode: "direct routing", "follow audio layer" or "follow live layer content"
sourceType	string	Yes	the audio source type: "none", "input", "custom", "dante in", "line in"
sourceId	integer	Yes	the audio source id associated to the sourceType
layerId	integer	Yes	the audio layer for

'sourceType' and 'sourceId' are mandatory for "direct routing" mode,
'layerId' is mandatory for "follow live layer content"

Example: Set screen 1 audio mode as follow audio layer

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/1/audio/mode HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 30<CR><LF><CR><LF>{"mode": "follow audio layer"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"mode": "follow audio layer"}

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/audio/mode");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {mode = "follow audio layer"});
   streamWriter.Write(json);
}
```

6.6. Reading the screen audio preset status

GET /api/tpp/v1/screens/{screenId}/audio/presets/{presetId}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
presetId	string	the preset: “program” or “preview”

Response

produces: application/json

Name	Type	Description
target	string	the target preset: “program” or "preview"
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
"target": "preview",
"sourceType": “input”,
 "sourceId": 4
}
```

Example: Read the status of audio preset for screen 1 preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/audio/presets/preview HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/audio/presets/preview");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
var responseText = streamReader.ReadToEnd();
}
```

6.7. Reading the source of the screen audio-layer preset

GET /api/tpp/v1/screens/{screenId}/audio-layer/presets/{presetId}/source

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
presetId	string	the preset: “program” or “preview”

Response
produces: application/json

Name	Type	Description
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
"sourceType": “input”,
"sourceId": 4
}
```

Example: Read the source of the audio-layer preview for screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/audio-layer/presets/preview/source HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/audio-layer/presets/preview/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
var responseText = streamReader.ReadToEnd();
}
```

6.8. Setting the source of the screen audio-layer preset

POST /api/tpp/v1/screens/{screenId}/audio-layer/presets/{presetId}/source

Request

Name	Type	Description
screenId	integer	the screen number: either '1' or '2'
presetId	string	the screen preset: either "program" or "preview"

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the audio source type: "none", "input", "custom", "dante in", "line in"
sourceId	integer	No	the audio source id associated to the sourceType

Example: Set line in 2 as source of audio layer of screen 2 program

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/screens/2/audio-layer/presets/program/source HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length: 40<CR><LF><CR><LF>{"sourceType": "line in", "sourceId": 2}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "line in", "sourceId": 2}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/audio-layer/program/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {sourceType = "line in", sourceId = 2});
   streamWriter.Write(json);
}
```

6.9. Reading the auxiliary screen audio mode status

GET /api/tpp/v1/auxiliary-screens/{auxScreenId}/audio/mode

Request

Name	Type	Description
screenId	integer	the screen number (always 1)

Response

produces: application/json

Name	Type	Description
mode	string	the audio mode status: “direct routing”, "follow audio layer" or "follow content"
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
  "mode": "follow content"
}
```

Example: Read the status of audio mode for auxiliary screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/auxiliary-screens/1/audio/mode HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/audio/mode");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.10. Setting the screen audio mode

POST /api/tpp/v1/auxiliary-screens/{auxScreenId}/audio/mode

Request

Name	Type	Description
auxScreenId	integer	the screen number (always 1)

Body

consumes: application/json

Name	Type	Optional	Description
mode	String	No	the selected mode: “direct routing”, "follow audio layer" or "follow content"
sourceType	string	Yes	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	Yes	the audio source id associated to the sourceType

‘sourceType’ and ‘sourceId’ are mandatory for “direct routing” mode,

Example: Set auxiliary screen 1 audio mode as follow audio layer

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/auxiliary-screens/1/audio/mode HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 30<CR><LF><CR><LF>{"mode": "follow audio layer"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"mode": "follow audio layer"}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-
screens/1/audio/mode");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {mode = "follow audio layer"});
   streamWriter.Write(json);
}
```

6.11. Reading the auxiliary screen audio preset status

GET /api/tpp/v1/auxiliary-screens/{auxScreenId}/audio/presets/{presetId}

Request

Name	Type	Description
AuxScreenId	integer	the screen number (always 1)
presetId	string	the preset: “program” or “preview”

Response

produces: application/json

Name	Type	Description
target	string	the target preset: “program” or "preview"
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
"target": "preview",
"sourceType": “custom”,
"sourceId": 7
}
```

Example: Read the status of audio preset for auxiliary screen 1 preview

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/auxiliary-screens/1/audio/presets/preview HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/audio/presets/preview");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
var responseText = streamReader.ReadToEnd();
}
```

6.12. Reading the source of the auxiliary screen audio-layer preset

GET /api/tpp/v1/auxiliary-screens/{auxScreenId}/audio-layer/presets/{presetId}/source

Request

Name	Type	Description
auxScreenId	integer	the screen number (always 1)
presetId	string	the preset: “program” or “preview”

Response
produces: application/json

Name	Type	Description
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
"sourceType": “input”,
"sourceId": 4
}
```

Example: Read the source of the audio-layer preview for auxiliary screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/auxiliary-screens/1/audio-layer/presets/preview/source
HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-screens/1/audio-layer/presets/preview/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
var responseText = streamReader.ReadToEnd();
}
```

6.13. Setting the source of the auxiliary screen audio-layer preset

POST /api/tpp/v1/auxiliary-screens/{auxScreenId}/audio-layer/presets/{presetId}/source

Request

Name	Type	Description
auxScreenId	integer	the screen number (always 1)
presetId	string	the screen preset: either “program” or “preview”

Body

consumes: application/json

Name	Type	Optional	Description
sourceType	string	No	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	No	the audio source id associated to the sourceType

Example: Set line in 2 as source of audio layer of auxiliary screen 1 program

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/auxiliary-screens/1/audio-layer/presets/program/source
HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Content-Length:
40<CR><LF><CR><LF>{"sourceType": "line in", "sourceId": 2}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"sourceType": "line in", "sourceId": 2}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/auxiliary-
screens/1/audio-layer/program/source");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {sourceType = "line in", sourceId = 2});
   streamWriter.Write(json);
}
```

6.14. Reading the dante output audio mode

GET /api/tpp/v1/dante/output-groups/{groupId}/mode

Request

Name	Type	Description
groupId	integer	the output number (from 1 to 4)

Response

produces: application/json

Name	Type	Description
mode	string	the output audio mode: “direct routing” or “follow screen”
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType
screenId	integer	the screen id

‘sourceType’ and ‘sourceId’ are only displayed for “direct routing” mode; ‘screenId’ is only displayed for “follow screen” mode”

Response example

```
{
  "mode": "follow screen",
  "screenId": 1
}
```

Example: Read the mode status of the audio dante group 2 (channels 9 to 16)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/dante/output-groups/2/mode HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/dante/output-groups/2/mode");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.15. Setting the dante output audio mode

POST /api/tpp/v1/dante/output-groups/{groupId}/mode

Request

Name	Type	Description
groupId	integer	the output number (from 1 to 4)

Body

consumes: application/json

Name	Type	Optional	Description
mode	String	No	the selected mode: “follow screen” or “direct routing”
sourceType	string	Yes	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	Yes	the audio source id associated to the sourceType
screenId	integer	Yes	the screen id

‘sourceType’ and ‘sourceId’ are mandatory for “direct routing” mode,
‘screenId’ is mandatory for “follow screen” mode

Example: Set input 5 in direct routing mode as source of dante group 2 (channels 9 to 16)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/dante/output-groups/2/mode HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 64<CR><LF><CR><LF>{"mode": "direct routing",
"sourceType": "input", "sourceId": 5}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"mode": "direct routing", "sourceType": "input", "sourceId": 5}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/dante/output-groups/2/mode");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {mode = "direct routing", sourceType = "input", sourceId = 5});
   streamWriter.Write(json);
}
```

6.16. Reading the dante output audio source

GET /api/tpp/v1/dante/output-groups/{groupId}/source

Request

Name	Type	Description
groupId	integer	the output number (from 1 to 4)

Response
produces: application/json

Name	Type	Description
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
  "sourceType": "input",
  "sourceId": 10
}
```

Example: Read the source status of dante audio output group 2 (channels 9 to 16)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/dante/output-groups/2/source HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/dante/output-groups/2/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.17. Reading the line-out output audio mode

GET /api/tpp/v1/line-outs/{outputId}/mode

Request

Name	Type	Description
outputId	integer	the line-out output number (from 1 to 2)

Response
produces: application/json

Name	Type	Description
mode	string	the output audio mode: “direct routing” or “follow screen”
channelPair	String	The pair of audio channels: “channels 1&2”, “channels 3&4”, “channels 5&6”, “channels 7&8”
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType
screenId	integer	the screen id

‘sourceType’ and ‘sourceId’ are only displayed for “direct routing” mode; ‘screenId’ is only displayed for “follow screen” mode”

Response example

```
{
  "mode": "follow screen",
  "channelPair": "channels 3&4",
  "screenId": 2
}
```

Example: Read the mode status of the audio line-out 2

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/line-outs/2/mode HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/line-outs /2/mode");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

6.18. Setting the line-out output audio mode

POST /api/tpp/v1/line-outs/{outputId}/mode

Request

Name	Type	Description
outputId	integer	the line-out output number (from 1 to 2)

Body

consumes: application/json

Name	Type	Optional	Description
mode	String	No	the selected mode: “follow screen” or “direct routing”
sourceType	string	Yes	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	Yes	the audio source id associated to the sourceType
screenId	integer	Yes	the screen id
channelPair	string	No	the pair of audio channel

‘sourceType’ and ‘sourceId’ are mandatory for “direct routing” mode,
‘screenId’ is mandatory for “follow screen” mode

Example: Set input 5 channels 1&2 in direct routing mode as source of line-out 2

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /api/tpp/v1/line-outs/2/mode HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 95<CR><LF><CR><LF>{"mode": "direct routing",
"sourceType": "input", "sourceId": 5, "channelPair": "channels 1&2"}
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Important: the *Content-Length* header field value must contain a decimal number representing the number of bytes found in the payload of the message {"mode": "direct routing", "sourceType": "input", "sourceId": 5, "channelPair": "channels 1&2"}

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/line-outs/2/mode");
httpWebRequest.ContentType = "application/json";
httpWebRequest.Method = "POST";

using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
{
   string json = new JavaScriptSerializer().Serialize(new {mode = "direct routing", sourceType = "input", sourceId = 5,
channelPair = "channels 1&2"});
   streamWriter.Write(json);
}
```

6.19. Reading the line-out output audio source

GET /api/tpp/v1/line-outs/{outputId}/source

Request

Name	Type	Description
outputId	integer	the output number (from 1 to 2)

Response
produces: application/json

Name	Type	Description
sourceType	string	the audio source type: “none”, “input”, “custom”, “dante in”, “line in”
sourceId	integer	the audio source id associated to the sourceType

Response example

```
{
  "sourceType": "input",
  "sourceId": 10
}
```

Example: Read the source of line-out audio output 2

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/line-outs/2/source HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/line-out/2/source");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

7. Source commands

7.1. Reading input information

GET /api/tpp/v1/inputs/{inputId}

Request

Name	Type	Description
inputId	integer	the input number (from 1 to 10)

Response
produces: application/json

Name	Type	Description
isEnabled	boolean	true if the input is enabled, false if not
label	string	the input label
plugType	string	The type of active plug: "HDMI", "DP", "SDI"
isValid	boolean	true if the input signal is valid, false if not

Response example

```
{
  "isEnabled": true,
  "label":  "Cam PTZ",
  "plugType": "HDMI”,
  "isValid": true,
}
```

Example: Read input 2 information

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/inputs/2 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/inputs/2");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

7.2. Reading foreground image information

GET /api/tpp/v1/screens/{screenId}/foreground-images/{foregroundImageId}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
foregroundImageId	integer	the image number (from 1 to 4)

Response

produces: application/json

Name	Type	Description
isEmpty	boolean	true if the foreground image is empty, false if not
isValid	boolean	true if the foreground image is valid, false if not
label	string	the foreground image label
isEnabled	boolean	Tue if the foreground image is enabled, false if not

Response example

```
{
  "isEmpty": false,
  "isValid": true,
  "label": “FRG4”,
  "isEnabled": true
}
```

Example: Read foreground image 2 information for screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/foreground-images/2 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/foreground-images/2");
var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

7.3. Reading background image information

GET /api/tpp/v1/screens/{screenId}/background-images/{backgroundImageId}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
backgroundImageId	integer	the image number (from 1 to 4)

Response

produces: application/json

Name	Type	Description
IsEmpty	boolean	true if the background image is empty, false if not
isValid	boolean	true if the background image is valid, false if not
label	string	the background image label
isEnabled	boolean	true if the background image is enabled, false if not

Response example

```
{
  "isEmpty": false,
  "isValid": true,
  "label": “FRG4”,
  "isEnabled": true
}
```

Example: Read background image 2 information for screen 1

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/1/background-images/2 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest = (HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/1/background-images/2");
var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

7.4. Reading background set information

GET /api/tpp/v1/screens/{screenId}/background-sets/{backgroundSetId}

Request

Name	Type	Description
screenId	integer	the screen number (from 1 to 2)
backgroundSetId	integer	the background set number (from 1 to 8)

Response

produces: application/json

Name	Type	Description
IsEmpty	boolean	true is the background set empty, false if not

Response example

```
{
  "isEmpty": false
}
```

Example: Read background set 5 information for screen 2

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
GET /api/tpp/v1/screens/2/background-sets/5 HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

C#

```
var httpWebRequest =
(HttpWebRequest)WebRequest.Create("http://192.168.2.140/api/tpp/v1/screens/2/background-sets/5");

var httpResponse = (HttpWebResponse) httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
{
   var responseText = streamReader.ReadToEnd();
}
```

8. Using thumbnails

8.1. Introduction

Thumbnails of live inputs, still images, outputs and multiviewer outputs are available. These thumbnails are regularly refreshed (except still images thumbnails which are refreshed only on change).
Snapshot request rate must not be more than 1 per second.
Picture size is 256 pixels (width) by up to 256 pixels (height). Black borders are automatically added, depending on aspect ratio. Picture type is PNG.

8.2. Live inputs thumbnails URL

http://<ipadress>/api/device/snapshots/inputs/1

up to

http://<ipadress>/api/device/snapshots/inputs/10

8.3. Outputs thumbnails URL

http://< ipadress>/api/device/snapshots/outputs/1

up to

http://<ipadress>/api/device/snapshots/outputs/2

8.4. Foreground Images thumbnails URL (per Screen)

http://< ipadress>/api/device/snapshots/screens/{screenId}/top/1

up to

http://<ipadress>/api/device/snapshots/screens/{screenId}/top/4

8.5. Background Images thumbnails URL (per Screen)

http://< ipadress>/api/device/snapshots/screens/{screenId}/back/1

up to

http://<ipadress>/api/device/snapshots/screens/{screenId}/back/4

8.6. Multiviewer thumbnails URL

http://< ipadress>/api/device/snapshots/multiviewer

9. Using Authentication

9.1. Introduction

If the access to the Web RCS has been password protected, an attempt to access any of the API endpoints over HTTP will result in a 401 Unauthorized error.

9.2. Authenticating

POST /auth/login?identifier={id}&password={pwd}

Request

Name	Type	Description
id	string	the user identifier (“Admin”)
pwd	string	the user password

Response

produces: text/html

Name	Type	Description
Set-Cookie	string	The JSON authentication cookie

The JSON authentication must be copied and stored for the following REST API calls.

Example: Authentication request (identifier = “Admin” and password = “pass”)

Raw TCP socket (connected on port 80 of 192.168.2.140)

```
POST /auth/login?identifier=Admin&password=pass HTTP/1.1<CR><LF><CR><LF>
```

<CR> replace with Carriage Return ASCII code: 13 (0x0D) <LF> replace with Line Feed ASCII code: 10 (0x0A)

Response example

```
…
Set-Cookie: auth-jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoid2ViUkNTIiwiaWF0IjoxNjE0OTUyNjY0MzA3LCJleHAiOjE2MTQ5NTQ2NjQzMDd9.1cGfJJ25PA7GGdu2ZWWQerkQmzx7FsebuEZqraBD0T8
…
```

9.3. Following REST API calls

The following REST API calls must include the JSON authentication cookie into the HTTP header.

For example, the **GET** request:

```
GET /api/tpp/v1/system HTTP/1.1<CR><LF><CR><LF>
```

becomes:

```
GET /api/tpp/v1/system HTTP/1.1<CR><LF>Content-Type: application/json<CR><LF>Cookie: auth-jwt=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoid2ViUkNTIiwiaWF0IjoxNjE0ODY2MTI5MDI5LCJleH
AiOjE2MTQ4NjgxMjkwMjl9.upnIqpw7VGEZOqdNB3H908VRR4mkjdpW4stHxT5-c0<CR><LF><CR><LF>
```

where the value of the Cookie field must match the JSON authentication cookie retrieved during the authentication call.

And the **POST** request:

```
POST /api/tpp/v1/screens/1/load-memory HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 36<CR><LF><CR><LF>{"memoryId": 2, "target": "preview"}
```

becomes:

```
POST /api/tpp/v1/screens/1/load-memory HTTP/1.1<CR><LF>Content-Type:
application/json<CR><LF>Content-Length: 36<CR><LF>Cookie: auth-jwt=eyJhbGciOiJIUzI1NiIsInR5cC
I6IkpXVCJ9.eyJ1c2VyIjoid2ViUkNTIiwiaWF0IjoxNjE0ODY2MTI5MDI5LCJleHAiOjE2MTQ4NjgxMjkwMjl
9.upnIqpw7VGEZOqdNB3H908VRR4mkjdpW4stHxT5-Hc0<CR><LF><CR><LF>{"memoryId": 2, "target":
"preview"}
```

where the value of the Cookie field must match the JSON authentication cookie retrieved during the authentication call.

December 2022
Version 2.0

Connect with us on