REST API Video wall Manager

Reference guide

ENABLING BRIGHT OUTCOMES

BARCO

Table of contents

REST API Introduction

1

1.1 Getting started

REST API is supported for LCD, LED and RPC.

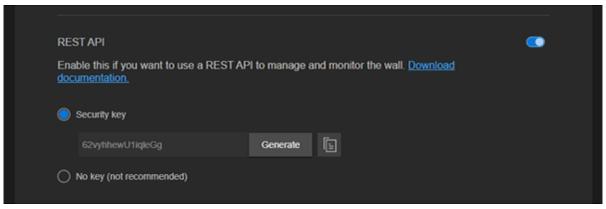
Overview

The Barco Video wall Manager REST API is provided to allow the video wall to be managed from external clients instead of always having to use the Video wall Manager user interface. It allows the client to query information from the wall and to change key settings.

The configuration needed to be correctly done to successfully integrate this API into a client application.

Configure the API endpoint via Video wall Manager UI

1. Open Video wall Manager and select **Menu** → *System settings* → *Security*.



REST API

Enable this if you want to use a REST API to manage and monitor the wall. Download documentation.

[✓ Security key]

62vyhhewU1iqleGg

No key (not recommended)

Image 1–1

2. Toggle the control to enable the API.
3. Select whether to secure the API by requiring the client to use a security key (recommended) or to not require any key
   1. The system provides an easy way to copy the key to the clipboard for use later. Click on the icon next to *Generate*.
   2. If no key is required, then any client that knows the IP address of the Video wall Manager Edge device can change settings on the device if the API is enabled.
4. Regardless of what client you are using to consume the API, it will need to be configured with the information of the REST API.
   1. IP address of the Video wall Manager Edge that is hosting the REST API you wish to integrate.
   2. The security key (see previous step) of the REST API.

Steps to perform to get/set information via the API

1. Make a call to authenticate, passing in the security key if needed (see "Authenticate", page 12).
2. Copy the contents of the "Set-Cookie" header out of the return.
3. Make subsequent gets/sets (e.g., get brightness) by adding a header named *Cookie* with the contents copied in the step above from the *Set-Cookie* header.

   See "Examples", page 61 for examples.

1.2 Responses

Return for a successful GET request

A response body with the following structure:

```
{
    “kind”: <internal address of the resource>,
    <returned data>
}
```

Example for get brightness

```
{
    "kind": "groups#wall#cbm",
    "brightness": 5,
    "maximum": 600,
    "minimum": 0
}
```

Return for a not successful GET request

A response body with the following structure:

```
{
    “error”: {
        “code”: <error code>,
        “errors”: [
        {
            “domain”: “API”,
            “location”:<the type of location>,
            "reason":<what was wrong>
        }],
        "message":<user frindly error message>
    }
}
```

Example for get brightness if the authentication cookie is missing, invalid or expired:

```
{
    "error": {
        "code": 401,
        "errors": [{
            "domain": "API",
            "location": "Cookie",
            "locationType": "header",
            "reason": "unauthorized"
        }],
        "message": "The user is not authorized to make the request.
                        Please check the security key with the administrator."
    }
}
```

Return for a successful SET request

A response body with the following structure:

```
{
    "actionId":<string id of the request>,
    "issuedBy":<internal address of the resource>,
    "kind":"action#response",
    "result":"accepted"
}
```

Example for SET brightness

```
{
    "actionId": "46001",
    "issuedBy": "groups#wall#cbm",
    "kind": "action#response",
    "result": "accepted"
}
```

Return for a not successful SET request

A response body with the following structure:

```
{
    “error”:{
        “code”:<error code>,
        "errors":[
        {
            "domain":"API",
            "location":<where the error occured>,
            "locationType>":<the type of location>,
            "reason":<what was wrong>
        }],
    "message":<user friendly error message>
    }
}
```

Example, return for set brightness if a negative brightness is sent.

```
{
    "error":{
        "code": 500,
        "errors":[
        {
            "domain": "groups#wall#cbm",
            "location": "brightness",
            "locationType": "parameter",
            "reason": "Not in range"
        }],
        "message": "The request failed because it is not a known request or is
                    lacking required parameters for that request. Review the
                    API documentation to determine the supported requests and
                    their required parameters."
        }
}
```

Processing time for a SET request

It can take some processing to serve the requested SET, the system will validate the request and will return if the request is accepted or if an error was found. The serving of the request if accepted will happen in the background.

1.3 Error handling in API

Overview

HTTP Status Code	Reason	Description
401	unauthorized	The user is not authorized to make the request. Please check the security key with the administrator.
404	notFound	The requested operation failed because a resource associated with the request could not be found.
500	invalidStructure	The request failed because it was not properly structured. Review the API documentation on information on how to structure requests.
500	invalidRequest	The request failed because it is not a known request or is lacking required parameters for that request. Review the API documentation to determine the supported requests and their required parameters.
500	internalError	The request failed due to an internal error.
503	systemNotConfigured	The system is not configured to expose this API. Please check with the administator.

Example

```
{
    "domain": "API",
    "reason": "unauthorized",
    "code": 401,
    "message": "The user is not authorized to make the request.
                     Please check the security key with the administrator."
}
```

1.4 Authenticate

Remark

The client must authenticate regardless if the Video wall Manager REST API is configured to use a key or not. The only difference is if the authentication request will contain the key or have an empty string.

HTTP request

POST /api/v1/auth/key

Example

```
{
    POST https://<unit-IP>/api/v1/auth/key
}
```

Request body with key

```
{
    "type": "REST",
    "key": "jrYRtDchsg1xudxH"
}
```

Request body without key

```
{
    "type": "REST",
    "key": ""
}
```

Properties

Property	Description	Values	Optional
type	Type of API	REST. In future JSON etc.	No
key	Key generated by VwM UI	Can be optional if security key is “No Key”	Yes

Response

If successful, this method returns a response body with the following structure:

```
{
    "expiresIn": 1800,
    "kind": "auth#token",
    "scope": "API"
}
```

An authentication cookie is returned in the header *Set-Cookie*. The contents of this header will need to be passed in all subsequent calls in a header labeled *Cookie*. See “Examples”, page 61 for an example.

Rest API - General 2

2.1 GetVwMVersion

HTTP request

GET /api/v1/version

Cookie: sid

Example

```
{
    GET https://192.168.178.244/api/v1/version
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
version	Software version of VwM	

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#wall#version",
    "version" : "v3.4.0.0.1663958274"
}
```

2.2 GetAPIVersion

HTTP request

GET /api/version

Cookie: sid

Example

```
{
    GET https://192.168.178.244/api/version
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
version	Current version of the api	v1

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#api#version",
    "version": "v1"
}
```

Rest API - Wall

3

3.1 GetWallBrightness

HTTP request

GET /api/v1/wall/brightness

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/brightness
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
brightness	Will contain the brightness value in percentage	0 - 100
	Will be null/empty if peak brightness is enabled and boost factor is applied for LED wall	null/empty
minimum	Minimum value supported by panel/wall	Mostly 0
maximum	Maximum value supported by panel/wall	Integer value depending upon display e.g. 500, 600, 800 etc.
		100 - Maximum %

Response

If successful, this method returns a response body with the following structure LED:

```
{
    "kind": "groups#wall#cbm",
    "brightness": 22,
    "minimum": 0,
    "maximum": 100
}
```

If successful, this method returns a response body with the following structure for LCD/RPC:

```
{
    "kind": "groups#wall#cbm",
    "brightness": 22,
    "minimum": 0,
    "maximum": 600
}
```

3.2 SetWallBrightness

System will do the validation of data-type and range

HTTP request

POST /api/v1/wall/brightness

Cookie: sid

Example

```
{
    POST https://<unit-IP>/api/v1/wall/brightness
    Cookie: sid
}
```

Request body

```
{
    "brightness": 12
}
```

Properties

Property	Description	Values	Optional
brightness	Will contain the brightness value in percentage	0 - 100	No

Response

If successful, this method returns a response body with the following structure:

```
{
    kind: "action#respose",
    issuedBy: "groups#wall#cbm",
    actionId: string,
    result: "accepted";
    error: Error | null;
}
```

Example

```
HTTP/1.1 202 Accepted
{
    "kind": "action#response",
    "issuedBy": "groups#wall#cbm",
    "actionId": "8ec1242",
    "result": "accepted"
}
```

For error handling, see “Error handling in API”, page 11 topic “Standard Error Responses”.

3.3 GetAbsoluteWallBrightness

HTTP request

GET /api/v1/wall/absoluteBrightness

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/absoluteBrightness
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
brightness	Will contain the brightness value in nits (cd/m²)	minimum - maximum[1]
		null - when brightness slider is in peak brightness area

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#wall#cbm",
    "brightness": 340,
    "minimum": 0,
    "maximum": 600
}
```

1. Maximum will depend upon the current mode (standard, high or peak) panel is running

3.4 SetAbsoluteWallBrightness

System will do the validation of data-type and range

HTTP request

POST /api/v1/wall/absoluteBrightness

Cookie: sid

Example

```
{
    POST https://<unit-IP>/api/v1/wall/absoluteBrightness
    Cookie: sid
}
```

Request body

```
{
    "brightness": 162
}
```

Properties

Property	Description	Values	Optional
brightness	Will contain the brightness value in nits (cd/m²). If value is integer but out or range then system will set the max if above or will set min if its below the range[2]	minimum - maximum	No

Response

If successful, this method returns a response body with the following structure:

```
{
    kind: "action#respose",
    issuedBy: "groups#wall#cbm",
    actionId: string,
    result: "accepted";
    error: Error | null;
}
```

Example

```
HTTP/1.1 202 Accepted
{
    "kind": "action#response",
    "issuedBy": "groups#wall#cbm",
    "actionId": "8ec1242",
    "result": "accepted"
}
```

For error handling, see "Error handling in API", page 11 .

2. If value is beyond what panel can currently support then max brightness supported by current mode (standard, high etc.) will be set.

3.5 GetWallPowerState

For more information on the supported power states, consult *“Barco Video wall Manager user guide”*, section *Switching the power state*. This manual can be downloaded from Barco’s website.

HTTP request

GET /api/v1/wall/power

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/power
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
power	Will contain the wall operation state	on, idle, standby

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#wall#state",
    "power": "on"
}
```

3.6 SetWallPowerState

For more information on the supported power states, consult *“Barco Video wall Manager user guide”*, section *Switching the power state*. This manual can be downloaded from Barco’s website.

System will do the validation of data-type and possible value that are case-sensitive (on, idle, standby)

About power state

Changing the power state takes some time. The command can be sent at any time, but if the system is processing a previous change to the power state, the result could be indeterminate as the commands are processed in parallel. Therefore it is recommended to give sufficient time for the wall to transition to its new power state before sending other commands.

The time needed to transition depends on various factors:

- State transitioning from/to
- Network topology
- Number of devices

It is recommended independently verify the required time per specific configuration.

HTTP request

POST /api/v1/wall/power

Cookie: sid

Example

```
{
    POST https://<unit-IP>/api/v1/wall/power
    Cookie: sid
}
```

Request body

```
{
    "power": "idle"
}
```

Properties

Property	Description	Values	Optional
power	Will contain the wall operation state	on, idle, standby	No

Response

If successful, this method returns a response body with the following structure:

```
{
    kind: "action#respose",
    issuedBy: "groups#wall#state",
    actionId: string;
    result: "accepted";
    error: Error | null
}
```

Example

```
HTTP/1.1 202 Accepted
```

```
{
    "kind": "action#response",
    "issuedBy": "groups#wall#state",
    "actionId": "8ec1242",
    "result": "accepted"
}
```

For error handling, see “Error handling in API”, page 11.

3.7 GetWallPresets

This request will be executed immediately.

HTTP request

GET /api/v1/wall/presets

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/presets
    Cookie: sid
}
```

Properties

Name	Type	Description
kind	string	Resource reference: "groups#wall#presetList"
presets[]	list	List of presets
preset.kind	string	Resource reference “groups#wall#preset"
preset.id	string	The id of the preset
preset.name	string	User friendly representation of the preset
preset.active	boolean	Flag representing the active state of the preset

Response

```
HTTP/1.1 200 OK
{
    "kind": "groups#wall#presetList",
    "presets": [
        preset resources
    ]
}
```

Example

```
{
    "kind": "groups#wall#presetList",
    "presets": [
        {
            "kind": "groups#wall#preset",
            "id": "source_1",
            "name": "I am preset 1",
            "active": true
        },
        {
            "kind": "groups#wall#preset",
            "id": "source_2",
            "name": "I am preset 2",
            "active": false
        }
    ]
}
```

3.8 SetWallPreset

For more information on the stored settings in a preset, consult *“Barco Video wall Manager user guide”*, section *Presets*. This manual can be downloaded from Barco’s website.

This method is executed asynchronously.

HTTP request

POST /api/v1/wall/preset

Cookie: sid

Example

```
POST https://<unit-IP>/api/v1/wall/preset
Cookie: sid
```

Request body

```
{
    "name": "ICMP-X"
}
```

Properties

Property	Description	Values	Optional
name	Specify the name of an existing preset	Should be a valid name that exists in the system. Will raise an error if not valid. Also name is case sensitive.	No

Response

```
{
    kind: "action#respose",
    issuedBy: "groups#wall#preset#activate",
    actionId: string;
    result: "accepted";
    error: Error | null
}
```

Example

```
HTTP/1.1 202 Accepted
{
    "kind" : "action#response",
    "issuedBy" : "groups#wall#preset#activate",
    "actionId" : "8ec1242",
    "result" : "accepted"
}
```

For error handling, see “Error handling in API”, page 11.

3.9 GetWallTemperature

Deprecated.

Temperatures are always reported in degrees Celsius.

HTTP request

GET /api/v1/wall/temperature

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/temperature
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
[]	Will contain the array of temperature for devices.	
	Processor array will not be present in case of LCD/RPC	
	Display array will always be available for all wall type	

Response - 1 : LED wall

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device#temperature",
    "processors": [{
        "id": "processor_1961864157574103942",
        "refNumber": 12,
        "temperatures": {
            "board": 41.0,
            "fpga": 74.0
        }
    }],
    "displays": [{
        "id": "d0616fc3599f",
        "position": {
            "column": 1,
            "row": 1
        },
        "temperatures": {
            "interface": 37.79,
            "left": 29.5,
            "main": 60.79,
            "right": 29.5
        }
    }]
}
```

Response - 2 : LCD/RPC wall

If successful, this method returns a response body with the following structure:

```
{
    "kind" : "group#wall#device#temperature",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position":
        {
            "column":1,
            "row":1
        },
        "temperatures": {
            "lcm": 37.79,
            "inputBoard": 29.5
            }
        }
    ]
}
```

3.10 GetWallAlert

HTTP request

GET /api/v1/wall/alert

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/alert
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
[]	Will contain the array of alerts for devices.	
	Processor array will not be present in case of LCD/RPC	
	Display array will always be available for all wall type	
code	Error code raised by a device itself. For more info, see *Video wall Manager* user guide	Range as defined in the *Video wall Manager* user guide
type	Type of alert	OK | warning | error
Id	Unique id of a device	

Response - 1 : LED wall

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device#alert",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "alerts": {
                "code": "ACE9066304.INP1005005",
                "type": "warning"
            }
        }
    ],
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "alerts": {
                "code": "3xsadasdas4.TP0.96034",
                "type": "warning"
            }
        }
    ]
}
```

Response - 2 : LCD/RPC wall

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device#alert",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "alerts": {
                "code": "3xsadasdas4.TP0.96034",
                "type": "warning"
            }
        }
    ]
}
```

3.11 GetWallSize

HTTP request

GET /api/v1/wall/size

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/size
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
width	Width of the wall i.e. Number of columns	Number greater than 0
height	Height of the wall i.e. Number of rows	Number greater than 0

Response

If successful, this method returns a response body with the following structure:

```
{
     "kind": "groups#wall#size",
     "width": 20,
     "height": 10
}
```

3.12 GetWallDevice

HTTP request

GET /api/v1/wall/device

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/device
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
[]	Will contain the array of devices	
	Processor array will not be present in case of LCD/RPC	
	Display array will always be available for all wall type	

Response - 1 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12
        }
    ],
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            }
        }
    ]
}
```

Response - 2 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                    "row": 1
                }
            }
        ]
}
```

3.13 GetWallHealth

HTTP request

GET /api/v1/wall/health

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/health
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
health	Indicate the health aka deviation status of the wall.[3]	ok | warning | error

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#wall#health",
    "health": "warning"
}
```

3. Will return the highest health severity (aka healthSummary) among all the devices

3.14 GetWallOSD

This request will be executed immediately.

HTTP request

GET https://<unit-IP>/api/v1/wall/osd

Cookie: sid

Example

```
{
    "kind": "groups#wall#osdList",
    "osd":
        {
            "luminance",
            "currentgain",
            "rec boardserialnr",
            "rec moduleserialnr",
            "rec boardarticlenr",
            "rec modulearticlenr",
            "ledb boardserialnr",
            "ledb moduleserialnr",
            "ledb boardarticlenr",
            "ledb modulearticlenr",
            "main temperature",
            "hub temperature",
            "Dehumidify",
            "version",
            "framerate",
            "runtime",
            "uptime",
            "coordinates",
            "front access",
            "diag",
            "linkcom count",
            "pll",
            "crc",
            "routing",
            "INP info",
            "Bandwidth",
            "calitarget",
            "typeID",
            "Power supply",
            "Upstream dropped",
            "MotorIC programmed",
            "Dehumlog"
        }
}
```

Request body

-

Properties

Property	Description	Values
kind	Resource reference: "groups#wall#osdList"	string
osd[]	List of osd names.	list

Response

```
HTTP/1.1 200 OK
{
    "kind": "groups#wall#osdList",
    "osd": {
        osd names
    }
}
```

3.15 SetWallOSD

This method is executed asynchronously.

HTTP request

GET /api/v1/wall/osd

Cookie: sid

Example

```
POST  https://<unit-IP>/api/v1/wall/osd
Cookie: sid
```

Request body

```
{
    "name": "ledb boardserialnr"
}
```

Properties

Property	Description	Values
name[4]	Specify the name of supported osd	Should be a valid name supported by panel. Will raise an error if not valid. Also name is case sensitive
	To turn off the OSD on the wall	Off value will turn off the osd on all the panels in the wall i.e. will make osd enabled = false

Response

```
{
    kind: "action#respose",
    issuedBy: "groups#wall#osd",
    actionId: string;
    result: "accepted";
    error: Error | null
}
```

Example

```
HTTP/1.1 202 Accepted
{
    "kind": "action#response",
    "issuedBy": "groups#wall#osd",
    "actionId": "8ec1242",
    "result": "accepted"
}
```

For error handling, see "Error handling in API", page 11.

4. Name can have space in between e.g. "front access"

3.16 GetWallName

HTTP request

GET /api/v1/wall/name

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/wall/name
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
name	Return the array of names of walls driven by VwM	Name of the wall(s). Mostly single name.

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#wall#name",
    "name":
        {
            "Center Stage",
            "WME-2398765"
        }
}
```

4 Rest API - Device

4.1 GetDeviceTemperature

HTTP request

GET /api/v1/device/:deviceId/temperature

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/:deviceId/temperature
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
id	Id of the device	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
{}	Will contain the object of temperature for devices	

Response - 1 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#temperature",
    "id": "processor_1961864157574103942",
    "refNumber": 12,
    "temperatures": {
        "board": 41.0,
        "fpga": 74.0
    }
}
```

Reponse - 2 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#temperature",
    "id": "d0616fc3599f",
    "position": {
        "column": 1,
        "row": 1
    },
    "temperatures": {
        "interface": 37.79,
        "left": 29.5,
        "main": 60.79,
        "right": 29.5
    }
}
```

Reponse - 3 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#temperature",
    "id": "d0616fc3599f",
    "position": {
        "column": 1,
        "row": 1
    },
    "temperatures": {
        "lcm": 37.79,
        "inputBoard": 29.5
    }
}
```

4.2 GetDevicesTemperature

HTTP request

GET /api/v1/device/temperature

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/temperature
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
{}	Will contain the object temperature for devices	
	Processor array will not be present in case of LCD/RPC	
	Display array will always be available for all wall type	

Response - 1 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#temperature",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "temperatures": {
                "board": 41.0,
                "fpga": 74.0
            }
        }
    ],
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
        },
            "temperatures": {
                "interface": 37.79,
                "left": 29.5,
                "main": 60.79,
                "right": 29.5
            }
        }
    ]
}
```

Reponse - 2 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#temperature",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "temperatures": {
                "lcm": 37.79,
                "inputBoard": 29.5
            }
        }
    ]
}
```

4.3 GetDeviceFanSpeed

HTTP request

GET /api/v1/device/:deviceId/fanSpeed

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/:deviceId/fanSpeed
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
id	Id of the device.	
refNumber	Reference/Front-panel number of the processor.	
position	Position of the display in the wall Display array will always be available for all wall type.	
speed	Fan speed of a device/processor	

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#fanSpeed",
    "id": "processor_1961864157574103942",
    "refNumber": 12,
    "speed": 3670
}
```

4.4 GetDevicesFanSpeed

HTTP request

GET /api/v1/device/fanSpeed

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/fanSpeed
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
speed	Fan speed of all devices/processors in the wall	
	Processor array	

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#fanSpeed",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
             "speed": 3775
        },
        {
            "id": "processor_1961864157574106789",
            "refNumber": 11,
             "speed": 3600
        }
    ]
}
```

4.5 GetDeviceRuntime

HTTP request

GET /api/v1/device/:deviceId/runtime

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/deviceId/runtime
    Cookie: sid
}
```

Properties

Property	Description	Values
id	Id of the device	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
[]	Will contain the array of runtimes for devices	

Response - 1 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#runtime",
    "id": "processor_1961864157574103942",
    "refNumber": 12,
    "runtimes": {
        "runtime": 40600,
        "uptime": 7800
    }
}
```

Reponse - 2 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#runtime",
    "id": "d0616fc3599f",
    "position": {
        "column": 1,
        "row": 1
    },
    "runtimes": {
        "runtime": 40600,
        "uptime": 7800
    }
}
```

Reponse - 3 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#runtime",
    "id": "d0616fc3599f",
    "position": {
        "column": 1,
        "row": 1
    },
    "runtimes": {
        "inputBoard": 9600,
        "displayInUse": 7800,
        "display": 9500
    }
}
```

4.6 GetDevicesRuntime

HTTP request

GET /api/v1/device/runtime

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/runtime
    Cookie: sid
}
```

Properties

Property	Description	Values
id	Id of the device.	
refNumber	Reference/Front-panel number of the processor.	
position	Position of the display in the wall Display array will always be available for all wall type.	
[]	Will contain the array of runtimes for devices	
	Processor array will not be present in case of LCS/RPC	
	Display array will always be available for all wall types.	

Response - 1 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#runtime",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "runtimes": {
                "runtime": 40600,
                "uptime": 7800
            }
        }
    ],
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "runtimes": {
                "runtime": 40600,
                "uptime": 7800
            }
        }
    ]
}
```

Reponse - 2 // Depending upon wall type

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#runtime",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "runtimes": {
                "inputBoard": 9600,
                "displayInUse": 7800,
                "display": 9500
            }
        }
    ]
}
```

4.7 GetDeviceHealth

HTTP request

GET /api/v1/device/:deviceId/health

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/:deviceId/health
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
health	Indicate the health aka deviation status of the specific/single device	ok | warning | error

Response - 1

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#device#health",
    "id": "processor_1961864157574103942",
    "refNumber": 12,
    "health": "ok"
}
```

Response - 2

If successful, this method returns a response body with the following structure:

```
{
    "kind": "groups#device#health",
    "id": "d0616fc3599f",
    "position": {
        "column": 1,
        "row": 1
    },
    "health": "ok"
}
```

4.8 GetDevicesHealth

HTTP request

GET /api/v1/device/health

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/health
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
id	Id of the device	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
[]	Will contain the array of devices with health status for devices[5]	
health	Indicate the health aka deviation status of the all the devices in the wall (display and/ or processor)	ok | warning | error

Response

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#health",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "health": "warning"
        },
        {
            "id": "processor_1961864157574102345",
            "refNumber": 11,
            "health": "error"
        },
        {
            "id": "processor_1961864157574423545",
            "refNumber": 9,
            "health": "ok"
        }
    ],
    "displays": [
```

5. If no health then the list will be empty

```
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "health": "ok"
        }
    ]
}
```

4.9 GetDeviceAlert

HTTP request

GET /api/v1/device/:deviceId/alert

Cookie: sid

Example

```
{
    GET https://192.168.178.244/api/v1/device/:deviceId/alert
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
id	Id of the device	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
[]	Will contain the array of alerts for devices[6]	
code	Error code raised by a device itself. For more info, see *Video wall Manager* user guide	Range as defined in the *Video wall Manager* user guide
type	Type of alert	ok | warning | error

Response - 1

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#alert",
    "processor":
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "alerts": [
                {
                    "code": "ACE9066304.INP1005005",
                    "type": "warning"
                },
                {
                    "code": "ACE9066304.INP1005006",
                    "type": "error"
                }
            ]
        }
}
```

6. If no alert/deviation then the list will be empty

Response - 2

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#alert",
    "display":
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            }
            "alerts": [
                {
                    "code": "ACE9066304.MVL01005",
                    "type": "warning"
                },
                {
                    "code": "ACE9066304.MVL01008",
                    "type": "error"
                }
            ]
        }
}
```

4.10 GetDevicesAlert

HTTP request

GET /api/v1/device/alert

Cookie: sid

Example

```
{
    GET https://<unit-IP>/api/v1/device/alert
    Cookie: sid
}
```

Request body

-

Properties

Property	Description	Values
id	Id of the device	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
[]	Will contain the array of alerts for devices[7]	
code	Error code raised by a device itself. For more info, see *Video wall Manager* user guide	Range as defined in the *Video wall Manager* user guide
type	Type of alert	ok | warning | error

Response - 1

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device#alert",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "alerts": {
                "code": "ACE9066304.INP1005005",
                "type": "warning"
            }
        },
        {
            "id": "processor_1961864157574103942",
            "refNumber": 11,
            "alerts": {}
        }
    ],
    "displays": [
        {
```

7. If no alert/deviation then the list will be empty

```
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "alerts": {
                "code": "3xsadasdas4.TP096034",
                "type": "warning"
            }
        }
    ]
}
```

Response - 2

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#wall#device#alert",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "alerts": {
                "code": "3xsadasdas4.MVL6034",
                "type": "warning"
            }
        }
    ]
}
```

4.11 GetDevicesPowerState

HTTP request

GET /api/v1/device/power

Cookie: sid

Example

```
{
    GET https://192.168.178.244/api/v1/device/power
    Cookie: sid
}
```

Properties

Property	Description	Values
id	Id of the device	
[]	Will contain the array of alerts for devices[8]	
refNumber	Reference/Front-panel number of the processor	
position	Position of the display in the wall Display array will always be available for all wall type	
power	Will contain the device operation state	on | idle | standby

Response - 1

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#power",
    "processors": [
        {
            "id": "processor_1961864157574103942",
            "refNumber": 12,
            "power": "on"
        },
        {
            "id": "processor_1961864157574104343",
            "refNumber": 11,
            "power": "on"
        }
    ],
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "power": "on"
        },
        {
            "id": "d0616fc3598a",
            "position": {
```

8. If no alert/deviation then the list will be empty

```
                "column": 1,
                "row": 2
            },
            "power": "on"
        }
    ]
}
```

Response - 2

If successful, this method returns a response body with the following structure:

```
{
    "kind": "group#device#power",
    "displays": [
        {
            "id": "d0616fc3599f",
            "position": {
                "column": 1,
                "row": 1
            },
            "power": "on"
        },
        {
            "id": "d0616fc3598a",
            "position": {
                "column": 1,
                "row": 2
            },
            "power": "on"
        }
    ]
}
```

A Examples

A.1 Postman example

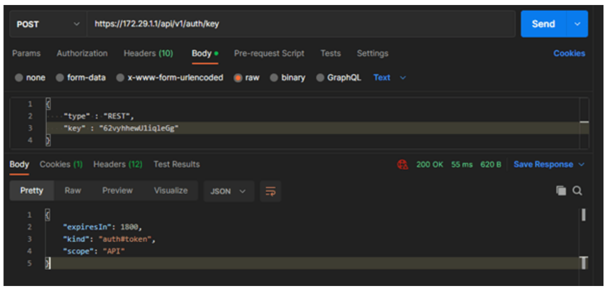
This requires the Postman utility to be downloaded and installed on your computer.

Authenticate

1. Use a POST command.
2. Set the url to be *https://<wme_ip_address>/api/v1/auth/key*
3. In the body set raw data.

```
{
    "type" : "REST",
    "key" : "62vyhhewU1iqleGg"
}
```

4. Click **Send**.



POST https://172.29.1.1/api/v1/auth/key

Body: raw, Text

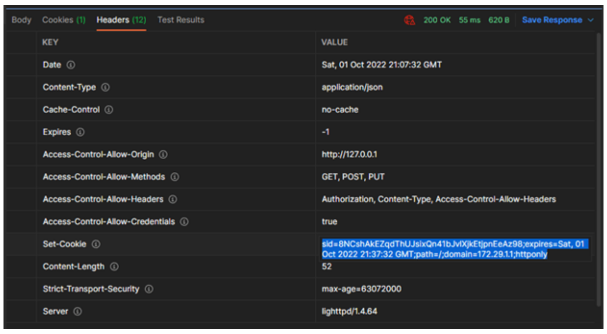
```
{
    "type" : "REST",
    "key" : "62vyhhewU1iqleGg"
}
```

Response Body: 200 OK 55 ms 620 B, Pretty, JSON

```
{
    "expiresIn": 1800,
    "kind": "auth#token",
    "scope": "API"
}
```

Image A–1

5. In the response Headers copy the contents of the header *Set-Cookie*.



Headers (12) — 200 OK 55 ms 620 B

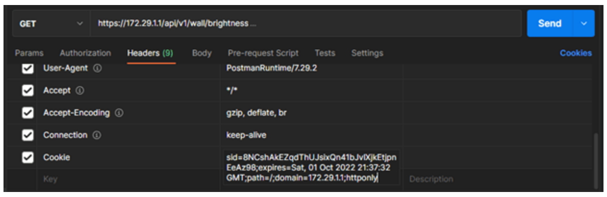
KEY	VALUE
Date	Sat, 01 Oct 2022 21:07:32 GMT
Content-Type	application/json
Cache-Control	no-cache
Expires	-1
Access-Control-Allow-Origin	http://127.0.0.1
Access-Control-Allow-Methods	GET, POST, PUT
Access-Control-Allow-Headers	Authorization, Content-Type, Access-Control-Allow-Headers
Access-Control-Allow-Credentials	true
Set-Cookie	sid=8NCshAkEZqdThUJsixQn41bJvlXjkEtjpnEeAz98;expires=Sat, 01 Oct 2022 21:37:32 GMT;path=/;domain=172.29.1.1;httponly
Content-Length	52
Strict-Transport-Security	max-age=63072000
Server	lighttpd/1.4.64

Image A–2

ii. Make a subsequent GET call. E.g., get brightness

1. Use a GET command.
2. Set the url to be *https://<wme_ip_address>/api/v1/wall/brightness*
3. Add a header named *Cookie* and paste in the data copied from the return header *Set-Cookie* in the above procedure.

   All that is absolutely required is the sid (e.g., “sid=8NCshAkEZqdThUJsixQn41bJvlXjkEtjpnEeAz98”) but the entire header can be used
4. Click **Send**.

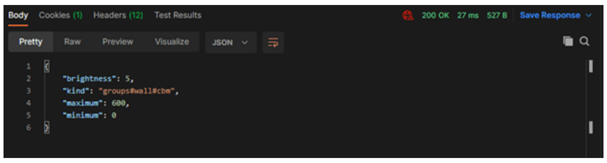


GET https://172.29.1.1/api/v1/wall/brightness ...

✓ User-Agent	PostmanRuntime/7.29.2
✓ Accept	*/*
✓ Accept-Encoding	gzip, deflate, br
✓ Connection	keep-alive
✓ Cookie	sid=8NCshAkEZqdThUJsixQn41bJvlXjkEtjpnEeAz98;expires=Sat, 01 Oct 2022 21:37:32 GMT;path=/;domain=172.29.1.1;httponly

Image A–3

5. The results are returned in the response body.



200 OK 27 ms 527 B

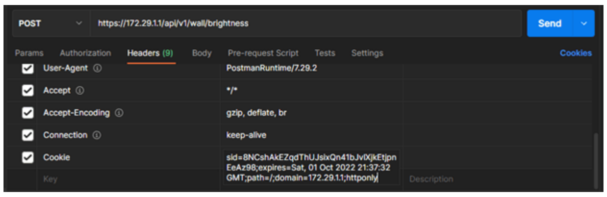
```
{
    "brightness": 5,
    "kind": "groups#wall#cbm",
    "maximum": 600,
    "minimum": 0
}
```

Image A–4

iii. Make a subsequent SET call. E.g., get brightness

1. Use a POST command.
2. Set the url to be https://<wme_ip_address>/api/v1/wall/brightness
3. Add a header named *Cookie* and paste in the data copied from the return header *Set-Cookie* in the above procedure.

   All that is absolutely required is the sid (e.g., “sid=8NCshAkEZqdThUJsixQn41bJvlXjkEtjpnEeAz98”) but the entire header can be used



POST https://172.29.1.1/api/v1/wall/brightness

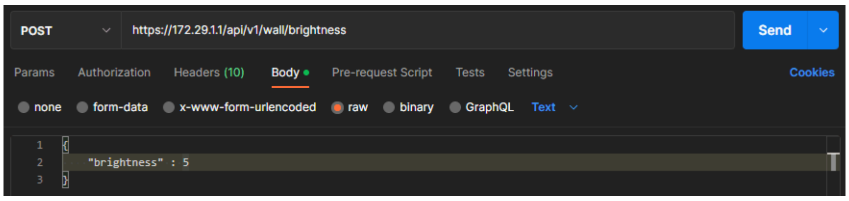
✓ User-Agent	PostmanRuntime/7.29.2
✓ Accept	*/*
✓ Accept-Encoding	gzip, deflate, br
✓ Connection	keep-alive
✓ Cookie	sid=8NCshAkEZqdThUJsixQn41bJvlXjkEtjpnEeAz98;expires=Sat, 01 Oct 2022 21:37:32 GMT;path=/;domain=172.29.1.1;httponly

Image A–5

4. In the body set raw data with the request parameters.

```
{
    "brightness" : 5
}
```

5. Click **Send**.



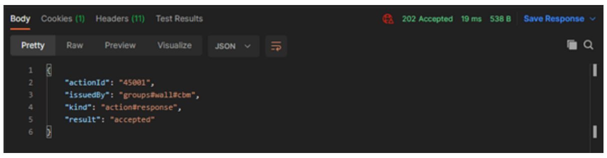
POST https://172.29.1.1/api/v1/wall/brightness

Body: ● raw

```
{
    "brightness" : 5
}
```

Image A–6

6. The results are returned in the response body.



202 Accepted 19 ms 538 B

```
{
    "actionId": "45001",
    "issuedBy": "groups#wall#cbm",
    "kind": "action#response",
    "result": "accepted"
}
```

Image A–7

A.2 Other examples

Overview

Some reference examples are provided as a starting point on how to authenticate and then use the authentication to consume the REST API

Extra info can be found on Barco’s website, *https://www.barco.com/en/support/docs/TDE12150*.

Index

R5916038 /03 | 2024-01-10