

---

# **Controller API**

***Release 5.0***

**Carallon Ltd**

**Dec 02, 2022**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Web API Authentication</b>	<b>5</b>
2.1	Authentication Methods . . . . .	5
<b>3</b>	<b>What's New</b>	<b>7</b>
3.1	v5.0 . . . . .	7
<b>4</b>	<b>HTTP API</b>	<b>9</b>
4.1	Authentication . . . . .	9
4.2	Beacon . . . . .	11
4.3	Channel / Park . . . . .	11
4.4	Command . . . . .	12
4.5	Config . . . . .	13
4.6	Content Targets . . . . .	16
4.7	Controller . . . . .	17
4.8	Group . . . . .	17
4.9	Input . . . . .	18
4.10	Log . . . . .	19
4.11	Lua Variable . . . . .	20
4.12	Output . . . . .	20
4.13	Override . . . . .	22
4.14	Project . . . . .	23
4.15	Protocol . . . . .	23
4.16	Remote Device . . . . .	25
4.17	Replication . . . . .	27
4.18	Hardware Reset . . . . .	27
4.19	Scene . . . . .	27
4.20	System . . . . .	29
4.21	Temperature . . . . .	30
4.22	Text Slots . . . . .	30
4.23	Time . . . . .	31
4.24	Timeline . . . . .	31
4.25	Trigger . . . . .	34
<b>5</b>	<b>JavaScript Query Library</b>	<b>37</b>
5.1	Beacon . . . . .	37
5.2	Channel / Park . . . . .	37
5.3	Command . . . . .	38
5.4	Config . . . . .	38

5.5	Content Targets . . . . .	39
5.6	Controller . . . . .	40
5.7	Group . . . . .	40
5.8	Input . . . . .	42
5.9	Log . . . . .	42
5.10	Lua Variable . . . . .	42
5.11	Output . . . . .	42
5.12	Override . . . . .	44
5.13	Project . . . . .	46
5.14	Protocol . . . . .	47
5.15	Remote Device . . . . .	47
5.16	Replication . . . . .	48
5.17	Scene . . . . .	48
5.18	System . . . . .	50
5.19	Temperature . . . . .	51
5.20	Text Slots . . . . .	51
5.21	Time . . . . .	52
5.22	Timeline . . . . .	52
5.23	Trigger . . . . .	56
<b>6</b>	<b>Websockets</b>	<b>57</b>
6.1	Websocket Subscriptions . . . . .	57
<b>7</b>	<b>Lua API</b>	<b>61</b>
7.1	Adjustment Target . . . . .	61
7.2	BPS . . . . .	63
7.3	Content Target . . . . .	64
7.4	Controller . . . . .	65
7.5	DateTime . . . . .	66
7.6	Group . . . . .	66
7.7	Location . . . . .	67
7.8	Override . . . . .	68
7.9	Project . . . . .	71
7.10	Network 2 . . . . .	71
7.11	Replication . . . . .	72
7.12	RIO . . . . .	72
7.13	Scene . . . . .	73
7.14	System . . . . .	74
7.15	Temperature . . . . .	75
7.16	Time . . . . .	76
7.17	Timeline . . . . .	76
7.18	Universe . . . . .	80
7.19	Variant . . . . .	81
7.20	Functions . . . . .	85

Web and Lua API documentation for Mosaic controllers.



## **INTRODUCTION**

Mosaic controllers offer web and Lua APIs providing access to system information, playback functions and trigger operations.

In addition, a small JavaScript library is hosted on the controller's web server, which wraps the HTTP requests of the web API and also provides a mechanism to subscribe to the controller's websocket channels via callbacks.





## WEB API AUTHENTICATION

If the controller has an admin password or the current project has users configured under Web Interface Access, then some endpoints of the HTTP API and some functions in the JavaScript library will require clients to authenticate in order to authorise the requests.

### 2.1 Authentication Methods

Two methods for authenticating users of the Web API are supported:

- *Cookie Authentication*: the default when using the API and/or query.js library in a custom web interface.
- *Token Authentication*: used with HTTP API requests, typically when the client is not a web browser.

With both methods, if the user is inactive for longer than 5 minutes then the cookie or token expires, requiring a username and password to be provided again.

#### 2.1.1 Cookie Authentication

Cookie authentication is typically used by the controller's web interface (either the default web interface or a custom web interface in a project). It works with both the HTTP API and the query.js library.

A cookie is returned by the controller in response to a *POST* request to the `/authorise` endpoint - this is where the default login page sends the user's username and password when they login. The cookie is stored by a web browser automatically, and the browser then sends this cookie with subsequent requests to authenticate the user. The cookie can be removed by making a *GET* request to the `/logout` endpoint, which can be done simply by navigating the browser to that endpoint.

#### Custom Login Page

Normally, a user will sign into the controller using the login page of the default web interface, which is shown if a user tries to visit a page that they don't have access to. In a custom web interface, uploaded as part of a project, a custom login page can be configured with the `AuthFormLoginRequiredLocation` directive in the main `.htaccess` file of the custom web interface. This custom login page is then shown instead of the default login page when a user tries to visit part of a custom web interface that they don't have access to.

Typically a login page will be an HTML page with a form element containing fields for the username and password. The HTML snippet below can be used to generate a form with these fields:

```
<form action="/authorise" method="POST">
  <input type="text" name="user">
  <input type="password" name="password">
```

(continues on next page)

(continued from previous page)

```
<button type="submit">Submit</button>  
</form>
```

The form's action is set to POST the form to the controller's `/authorise` endpoint. If login is successful, the response from the controller will contain a cookie, which the browser will store automatically.

### 2.1.2 Token Authentication

Token authentication is typically used by the HTTP API in cases where a web browser is not the client. The client requests a Bearer Token with a *POST* request to the controller's `/token` endpoint, providing the username and password, and this token is then used in future requests.

## WHAT'S NEW

### 3.1 v5.0

- Added controller propagation to certain HTTP API requests and query.js functions.
- `memory_free` changed to `memory_available` in the HTTP & JavaScript *System* information and in the Lua *System* namespace.
- `get_trigger_number` function added.
- `vlan_tag` property added to Lua *Controller*.
- `is_network_primary` property added to Lua *Controller*.
- `dns_servers` property added to the Lua *System* namespace.



## HTTP API

Reference for controller HTTP API.

### 4.1 Authentication

Authentication reference for the controller HTTP API.

#### 4.1.1 Authorise

##### Methods

##### POST

Accepts form data or JSON to authenticate a user's credentials, returning a cookie if successful.

POST /authorise

The payload, whether form data or JSON, should have the following attributes:

Attribute	Value Type	Description
user	string	The username of the user.
password	string	The user's password.

If the credentials are valid, a redirect response will be returned, pointing to the value of the `original_url` cookie sent with the original request, or / if this cookie wasn't present.

If the user cannot be authenticated because the username or password are incorrect then a redirect response will be returned, pointing to the value of the `Referer` header in the request.

The response will be a 400 error if either attribute is missing or a value is of an invalid type.

## 4.1.2 Logout

### Methods

#### GET

Ends the user's current session.

GET /logout

The request must be authenticated either with a cookie or by sending a valid Bearer token in the **Authorization** header.

If the request is made from a web browser using cookie authentication then the cookie will be deleted from the browser by the response. The web browser will reload the page from which the request was made if the **Referer** header is set.

## 4.1.3 Token

### Methods

#### POST

Accepts form data or JSON to authenticate a user's credentials, returning a bearer token if successful.

POST /token

The payload, whether form data or JSON, should have the following attributes:

Attribute	Value Type	Description
user	string	The username of the user.
password	string	The user's password.

If the credentials are valid, the response will have a 200 status code and the payload will be a JSON object with the following attributes:

Attribute	Value Type	Description
access_token	string	The access token to use in subsequent requests to authorise them.
expires_in	integer	The length of time in seconds before the token expires. Normally 300.
token_type	string	Type of the token. Normally "Bearer".

To use the token in a request, set the **Authorization** header value to **Bearer {your access token}**, where {your access token} should be replaced with the value of **access\_token** in the response.

The response will be a 401 error if the user cannot be authenticated because the username or password are incorrect.

The response will be a 400 error if either attribute is missing or a value is of an invalid type.

## 4.2 Beacon

### 4.2.1 Methods

#### POST

Toggle beacon mode on the controller.

POST /api/beacon

In beacon mode, a controller will flash its LEDs or it screen continuously.

## 4.3 Channel / Park

### 4.3.1 Methods

#### POST

Park an output channel or channels at a specified level.

POST /api/channel

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
universe	string	See <i>Universe Key String Format</i>	"dmx:1"
channels	string	Comma separated list of channel numbers.	"1-3,5"
level	integer	Level to set the channel(s) to: 0-255.	128

#### DELETE

Unpark an output channel or channels.

DELETE /api/channel

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
universe	string	See <i>Universe Key String Format</i>	"dmx:1"
channels	string	Comma separated list of channel numbers.	"1-3,5"

### 4.3.2 Universe Key String Format

A universe key string takes the form:

- `protocol:index` for protocols `dmx`, `pathport`, `sacn`, `art-net`;
- `protocol:kinetPowerSupplyNum:kinetPort` for protocol `kinet`;
- `protocol:remoteDeviceType:remoteDeviceNum` for protocol `rio-dmx`;
- `protocol:remoteDeviceType:remoteDeviceNum:port` for protocols `edn`, `edn-spi`.

Where:

- `kinetPowerSupplyNum` is an integer;
- `kinetPort` is an integer;
- `remoteDeviceType` can be `rio08`, `rio44` or `rio80`, `edn10` or `edn20`;
- `remoteDeviceNum` is an integer;
- `port` is an integer.

For example:

- `"dmx:1"`
- `"rio-dmx:rio44:1"`

## 4.4 Command

### 4.4.1 Methods

#### POST

Run a Lua script or pass a command to the command line parser on the controller.

---

**Note:** The Command Line Parser must be enabled in the web interface settings of the current project, else this endpoint will not be available.

---

POST `/api/cmdline`

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description
<code>input</code>	string	The script to parse or run.

For example:

```
{
  "input": "tl = 1 get_timeline(tl):start()"
}
```



## Response

If the Command Line Parser is enabled in the web interface settings of the current project then a 200 status code will be returned, along with the text **Executed** if the script was executed successfully. If an error occurred when attempting to run the script then the error string will be returned.

The response will be 501 Not Implemented if the Command Line Parser is not enabled, or 400 Bad Request if no project is loaded.

## 4.5 Config

### 4.5.1 Methods

#### POST

Edits the configuration of the controller.

POST /api/config

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
ip	string	Optional. Set the controller's IP address (management interface)	"192.168.1.3"
subnet_mask	string	Optional. Set the controller's subnet mask (management interface)	"255.255.255.0"
gateway	string	Optional. Set the controller's gateway address (management interface)	"192.168.1.1"
dhcp_enabled	boolean	Optional. Set whether the controller is assigned its IP address automatically by DHCP	true
name_server_1	string	Optional. Set the primary name server address	"192.168.1.1"
name_server_2	string	Optional. Set the secondary name server address	"8.8.8.8"
http_port	integer	Optional. Set the port opened for HTTP access to the controller's web server	80
https_port	integer	Optional. Set the port opened for HTTPS access to the controller's web server	443
year	integer	Optional. Set the year of the current date on the controller's clock	2021
month	integer	Optional. Set the month of the current date on the controller's clock (1-12)	4
day	integer	Optional. Set the day of the current date on the controller's clock (1-31)	25
hour	integer	Optional. Set the hour component of the current time on the controller's clock (0-23)	13
minute	integer	Optional. Set the minute component of the current time on the controller's clock (0-59)	21
second	integer	Optional. Set the second component of the current time on the controller's clock (0-59)	46
watchdog_enabled	boolean	Optional. Set whether the controller's hardware watchdog is enabled	true
log_level	integer	Optional. Set the level of verbosity of the controller's log (1-5)	3
syslog_enabled	boolean	Optional. Set whether the controller will send its log to a syslog server	false
syslog_ip	string	Optional. Set the IP address of a third party syslog server	"192.168.1.2"
ntp_enabled	boolean	Optional. Set whether the controller will fetch the current time automatically from an NTP server	true
ntp_ip	string	Optional. Set the IP address of a third party NTP server	"192.168.1.1"
password	string	Optional. Change the admin password on the controller.	"mySup3rS3cr3tP455w0rd!"

If the response status code is 200 (OK), the response body will be a JSON object with a `restart` attribute. The value of `restart` is boolean. If `true`, the controller will reset itself imminently in order to apply the changes.

**GET**

Returns information about the queried controller's configuration.

GET /api/config

Returns a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
ip	string	Controller IP address (management interface)	"192.168.1.3"
subnet_mask	string	Controller subnet mask (management interface)	"255.255.255.0"
gateway	string	Gateway address (management interface)	"192.168.1.1"
dhcp_enabled	boolean	Whether the controller is assigned its IP address automatically by DHCP	true
name_server_1	string	Primary name server address	"192.168.1.1"
name_server_2	string	Secondary name server address	"8.8.8.8"
http_port	integer	Port opened for HTTP access to the controller's web server	80
https_port	integer	Port opened for HTTPS access to the controller's web server	443
year	integer	Year of the current date, according to the controller's clock	2021
month	integer	Month of the current date, according to the controller's clock (1-12)	4
day	integer	Day of the current date, according to the controller's clock (1-31)	25
hour	integer	Hour component of the current time, according to the controller's clock (0-23)	13
minute	integer	Minute component of the current time, according to the controller's clock (0-59)	21
second	integer	Second component of the current time, according to the controller's clock (0-59)	46
watchdog_enabled	boolean	Whether the controller's hardware watchdog is enabled	true
log_level	integer	Level of verbosity of the controller's log (1-5)	3
syslog_enabled	boolean	Whether the controller is sending its log to a syslog server	false
syslog_ip	string	IP address of a third party syslog server	"192.168.1.2"
ntp_enabled	boolean	Whether the controller is fetching current time automatically from an NTP server	true
ntp_ip	string	IP address of a third party NTP server	"192.168.1.1"

## 4.6 Content Targets

---

**Note:** Atlas/Atlas Pro only

---

### 4.6.1 Methods

#### POST

Control a content target; currently the only supported action is to master the intensity of a content target (applied as a multiplier to output levels).

POST /api/content\_target

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
action	string	The action to perform on the content target. Currently only <code>master_intensity</code> is supported.	"master_intensity"
type	string	Optional. Type of content target (only relevant on Atlas Pro): <code>primary</code> , <code>secondary</code> , <code>target_3</code> , <code>target_4</code> , <code>target_5</code> , <code>target_6</code> , <code>target_7</code> , <code>target_8</code> . Defaults to <code>primary</code> .	"secondary"
level	float or string containing a bounded integer	Master intensity level to set on the content target	0.5 or "50:100"
fade	float	Optional. Fade time to apply the intensity change, in seconds.	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds.	2.0

#### GET

Returns information about the current state of all Content Targets in the project.

GET /api/content\_target

Returns a JSON object with a single `content_targets` attribute, which has an array value. Each item in the array is a Content Target object with the following attributes:

Attribute	Value Type	Description	Value Example
name	string	Content target name	"Primary"
level	integer	Current intensity master level of the content target, 0-100	100

## 4.7 Controller

### 4.7.1 Methods

#### GET

Returns data about the controllers in the project.

GET /api/controller

Returns a JSON object with a single `controllers` attribute, which has an array value. Each item in the array is a Controller object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Controller number	1
<code>type</code>	string	Controller type, e.g. "MSC" or "MTPC"	"MSC"
<code>name</code>	string	Controller user name, or the default name if none is set	"Controller 1"
<code>serial</code>	string	Serial number of the controller	"009060"
<code>ip_address</code>	string	IP address of the controller if the controller is discovered; empty if the controller is not discovered or is the queried controller	"192.168.1.3" or ""
<code>online</code>	boolean	Whether the controller is detected as online on the local network	true
<code>is_network_primary</code>	boolean	Whether the controller is set as the network primary in the project	true

## 4.8 Group

---

**Note:** Not applicable to Atlas/Atlas Pro

---

### 4.8.1 Methods

#### POST

Control a group; currently the only supported action is to master the intensity of a group (applied as a multiplier to output levels). Action will propagate to all controllers in a project.

POST /api/group

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
action	string	The action to perform on the group. Currently only <code>master_intensity</code> is supported.	"master_intensity"
num	integer	Group number. Group 0 means the <i>All Fixtures</i> group.	1
level	float or string containing a bounded integer	Master level to set on the group	0.5 or "50:100"
fade	float	Optional. Fade time to apply the intensity change, in seconds.	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds.	2.0

## GET

Returns data about the fixture groups in the project.

GET /api/group[?num=groupNumbers]

num can be used to filter which groups are returned and is expected to be either a single number or a string expressing the required groups, e.g. "1, 2, 5-9".

---

**Note:** Group 0 will return data about the *All Fixtures* group.

---

Returns a JSON object with a single `groups` attribute, which has an array value. Each item in the array is a Group object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Group number (only included for user-created groups)	1
name	string	Group name	"Group 1"
level	integer	Group master level, 0-100	100

## 4.9 Input

### 4.9.1 Methods

#### GET

Returns the status of digital & analogue inputs on the queried controller.

GET /api/input

Returns a JSON object with the following attributes:

Attribute	Value Type	Description
gpio	array	Array of Input objects; returned when queried controller is MSC or MTPC + TPC-RIO
dmxIn	object	DMX Input object; returned when DMX input is configured on the queried controller

The Input object has the following properties:

Attribute	Value Type	Description	Value Example
input	integer	Input number	1
type	string	Analog, Digital, or Contact Closure	"Contact Closure"
value	integer or boolean	Value type depends on input type - Analog inputs return an integer, 0-255; other types return a bool.	true

The DMX Input object has the following properties:

Attribute	Value Type	Description	Value Example
error	string	If DMX input is configured but no DMX is received	"No DMX received"
dmxInFrame	array	Array of channel values	[0,0,0,0,0,0,0,0,0,255,255,255...255,0,255]
dmxInSourceCount	integer	The number of sources - will be 1 except for sACN.	1
dmxInProtocol	string	dmx, art-net or sacn	"dmx"

## 4.10 Log

### 4.10.1 Methods

#### GET

Returns the log from the controller.

GET /api/log

Returns a JSON object with the following attributes:

Attribute	Value Type	Description
log	string	The whole log from the controller

## 4.11 Lua Variable

### 4.11.1 Methods

#### GET

Returns the current value of specified Lua variables.

GET /api/lua?variables=luaVariables

luaVariables is expected to be a string or comma-separated list of strings, where each string is a Lua variable name.

Returns a JSON object with the Lua variables and their values as its key/value pairs - the Lua variable names are the keys.

For example, in a project that creates variables called bob and alice, GET /api/lua?variables=bob,alice could return a JSON object as follows:

```
{
  "alice": 1234,
  "bob": "a string variable"
}
```

## 4.12 Output

### 4.12.1 Methods

#### POST

Enable/disable the output of a selected protocol from the controller. Action will propagate to all controllers in a project.

POST /api/output

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
protocol	string	Protocol to disable. Options: dmx, pathport, sacn, art-net, kinet, rio-dmx, edn, edn-spi.	"parthport"
action	string	Whether to enable or disable output via the protocol.	"disable"



## GET

Returns the lighting levels being output by the queried controller.

GET /api/output?universe=universeKey

universeKey is a string; see *Universe Key String Format*.

For example: \* GET /api/output?universe=dmx:1 \* GET /api/output?universe=rio-dmx:rio44:1

If the queried controller is an MSC 1, the universe is DMX 2, DMX Proxy has been enabled for a MTPC in the project and the MTPC is offline then this request will return a JSON object with the following attributes:

Attribute	Value Type	Value Example
proxied_tpc_name	string	"Controller 2"

Otherwise a JSON object with the following attributes is returned:

Attribute	Value Type	Description	Value Example
channels	array	Array of integer (0-255) channel levels	[0,0,0,0,0,0,0,0,0,255,255,255...255,0,255]
disabled	bool	Whether the output has been disabled by a Trigger Action	false

### 4.12.2 Universe Key String Format

A universe key string takes the form:

- protocol:index for protocols dmx, pathport, sacn, art-net;
- protocol:kinetPowerSupplyNum:kinetPort for protocol kinet;
- protocol:remoteDeviceType:remoteDeviceNum for protocol rio-dmx;
- protocol:remoteDeviceType:remoteDeviceNum:port for protocols edn, edn-spi.

Where:

- kinetPowerSupplyNum is an integer;
- kinetPort is an integer;
- remoteDeviceType can be rio08, rio44 or rio80, edn10 or edn20;
- remoteDeviceNum is an integer;
- port is an integer.

For example:

- "dmx:1"
- "rio-dmx:rio44:1"

## 4.13 Override

### 4.13.1 Methods

#### PUT

Set the Intensity, Red, Green, Blue levels for a fixture or group. Action will propagate to all controllers in a project.

PUT /api/override

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
target	string	What the override should be applied to: group, fixture.	"group"
num	integer	Group or fixture number, depending on target. Group 0 means the <i>All Fixtures</i> group.	1
intensity	integer	Optional. Intensity to set as part of override: 0-255. Intensity override will not be changed if this attribute isn't provided.	128
red	integer	Optional. Red component to set as part of override: 0-255. Red override will not be changed if this attribute isn't provided.	255
green	integer	Optional. Green component to set as part of override: 0-255. Green override will not be changed if this attribute isn't provided.	255
blue	integer	Optional. Blue component to set as part of override: 0-255. Blue override will not be changed if this attribute isn't provided.	255
temperature	integer	Optional. Temperature component to set as part of override: 0-255. Temperature override will not be changed if this attribute isn't provided.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Braked"

#### DELETE

Release any overrides on fixtures or groups. Action will propagate to all controllers in a project.

DELETE /api/override

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
target	string	What the overrides should be cleared on: group, fixture.	"group"
num	integer	Optional. Group or fixture number, depending on target. If not provided, target is ignored and all overrides are cleared.	1
fade	float	Optional. Fade time in which to release overrides, in seconds.	2.0

## 4.14 Project

### 4.14.1 Methods

#### GET

Returns data about the current project.

GET /api/project

Returns a JSON object with the following attributes:

Attribute	Value Type	Value Example
name	string	"Help Project"
author	string	"Contoso"
filename	string	"help_project_v1.pd2"
unique_id	string	"{6b48627a-1d5e-4b2f-81e2-481e092a6a79}"
upload_date	string	"2017-01-30T15:19:08"

## 4.15 Protocol

### 4.15.1 Methods

#### GET

Returns all the universes in the project on the queried controller.

GET /api/protocol

Returns a JSON object with a single **outputs** attribute, which has an array value. Each item in the array is a Protocol object with the following attributes:

Attribute	Value Type	Description	Value Example
type	integer	Protocol type; possible types are: DMX (1), Pathport (2), Art-Net (4), KiNET (8), sACN (16), DVI (32), RIO DMX (64), EDN DMX (128), EDN SPI (256)	1
name	string	Protocol name	"DMX"
disabled	boolean	Whether the output has been disabled by a Trigger Action	false
universes	array	Array of Universe objects (see table below)	[{"key":{"index":1}, "name":"1"}, {"key":{"index":2}, "name":"2"}]
dmx_proxy	object	DMX Proxy object, if applicable (see table below)	{"ip_address":"192.168.1.17", "name":"Controller 1"}

Each Universe object has the following properties:

Attribute	Value Type	Description	Value Example
name	string	A simplistic version of the universe name, which for most protocols is simply the index number	"1"
key	object	Universe Key object (see table below)	{"index":1}

Each DMX Proxy object has the following properties:

Attribute	Value Type	Description	Value Example
name	string	Name of the controller that is outputting this universe by proxy	"Controller 1"
ip_address	string	IP address of the controller that is outputting this universe by proxy	"192.168.1.17"

The properties of the Universe Key object depend on the type.

For DMX, Pathport, sACN and Art-Net:

Attribute	Value Type	Value Example
index	integer	1

For KiNET:

Attribute	Value Type	Value Example
kinet_port	integer	1
kinet_power_supply_num	integer	1

For RIO DMX:

Attribute	Value Type	Description	Value Example
remote_device_num	integer	Remote device number (address)	1
remote_device_type	integer	Value can be 101 (RIO 80), 102 (RIO 44) or 103 (RIO 08)	101

For EDN:

Attribute	Value Type	Description	Value Example
remote_device_num	integer	EDN number (address)	1
remote_device_type	integer	Value can be 109 (EDN 20) or 110 (EDN 10)	110
port	integer	Number of EDN output port	1

## 4.16 Remote Device

### 4.16.1 Methods

#### GET

Returns data about all the remote devices in the project.

GET /api/remote\_device

Returns a JSON object with a single `remote_devices` attribute, which has an array value. Each item in the array is a Remote Device object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Remote device number (address)	1
type	string	RIO 08, RIO 44, RIO 80, BPS, BPI, RIO A, or RIO D	"RIO 44"
serial	array	Array of serial numbers (as strings) of all discovered devices matching the address and type	["001234", "005678"]
outputs	array	Array of Output objects (see table below); only returned for RIO 44 and RIO 08 on the queried controller	[{"output":1, "value":true}, {"output":2, "value":true}, {"output":3, "value":true}, {"output":4, "value":true}]
inputs	array	Array of Input objects (see table below); only returned for RIO 44 and RIO 80 on the queried controller	[{"input":1, "type":"Contact Closure", "value":true}, {"input":2, "type":"Contact Closure", "value":true}, {"input":3, "type":"Contact Closure", "value":true}, {"input":4, "type":"Contact Closure", "value":true}]
online	boolean	Whether the remote device is detected as being online on the local network	true

The Output JSON object has the following attributes:

Attribute	Value Type	Description	Value Example
output	integer	Number of the output, as labelled on the remote device	1
state	boolean	true means the output is on, false means it is off	true

The Input JSON object has the following attributes:

Attribute	Value Type	Description	Value Example
input	integer	Number of the input, as labelled on the remote device	1
type	string	Analog, Digital, or Contact Closure	"Digital"
value	integer or boolean	Value type depends on input type - Analog inputs return an integer, 0-255; other types return a bool.	true

## 4.17 Replication

### 4.17.1 Methods

#### GET

Returns data about the install replication.

GET /api/replication

Returns a JSON object with the following attributes:

Attribute	Value Type	Value Example
name	string	"Help Project"
unique_id	string	"{6b48627a-1d5e-4b2f-81e2-481e092a6a79}"

## 4.18 Hardware Reset

### 4.18.1 Methods

#### POST

Reboot the controller.

POST /api/reset

## 4.19 Scene

### 4.19.1 Methods

#### POST

Control a scene in the project. Action will propagate to all controllers in a project.

POST /api/scene

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
action	string	The action to perform on the scene(s): <b>start</b> , <b>release</b> , <b>toggle</b>	"start"
num	integer	The number of the scene to perform the action on. If not present, the action will be applied to all scenes in the project; omitting this attribute is valid for <b>release</b> .	1
fade	number	Optional. The fade time to apply to a <b>release</b> action, in seconds, or the scene release that results from a <b>toggle</b> action. If not provided, the default release fade time will be used.	2.0
group	string	Optional. Scene group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A. This attribute is valid for a <b>release</b> action without a specified <b>num</b> , meaning <i>release all scenes</i> .	"B"

For example, to start a scene 2, the request payload is:

```
{
  "action": "start",
  "num": 2
}
```

To release scene 2 in 3.5 seconds, the request payload would be:

```
{
  "action": "release",
  "num": 2,
  "fade": 3.5
}
```

To toggle scene 2, and release it in 2 seconds if it's already been started, the request payload would be:

```
{
  "action": "toggle",
  "num": 2,
  "fade": 2.0
}
```

To release all scenes in 2 seconds, the request payload would be:

```
{
  "action": "release",
  "fade": 2.0
}
```

To release all scenes except those in group B in 2 seconds, the request payload would be:

```
{
  "action": "release",
  "group": "!B",
  "fade": 2.0
}
```



## GET

Returns data about the scenes in the project and their state on the controller.

GET /api/scene[?num=sceneNumbers]

num can be used to filter which scenes are returned and is expected to be either a single number or a string expressing the required scenes, e.g. "1,2,5-9".

Returns a JSON object with a single scenes attribute, which has an array value. Each item in the array is a Scene object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Scene number	1
name	string	Scene name	"Scene 1"
state	string	none, started	"none"
onstage	boolean	Whether the scene is affecting output of any fixtures	true

## 4.20 System

### 4.20.1 Methods

#### GET

Returns data about the controller.

GET /api/system

Returns a JSON object with the following attributes:

Attribute	Value Type	Value Example
hardware_type	string	"MSC"
channel_capacity	integer	512
serial_number	string	"006321"
memory_total	string	"12790Kb"
memory_used	string	"24056Kb"
memory_available	string	"103884Kb"
storage_size	string	"1914MB"
bootloader_version	string	"0.9.0"
firmware_version	string	"2.8.0"
reset_reason	string	"Software Reset"
last_boot_time	string	"01 Jan 2017 09:09:38"
ip_address	string	"192.168.1.3"
subnet_mask	string	"255.255.255.0"
broadcast_address	string	"192.168.1.255"
default_gateway	string	"192.168.1.3"

## 4.21 Temperature

### 4.21.1 Methods

#### GET

Returns data about the controller's temperature.

GET /api/temperature

Returns a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
sys_temp	number	Only for MSC X and Atlas/Atlas Pro	40.2
core1_temp	number	Only for MSC X and Atlas/Atlas Pro	44
core2_temp	number	Only for MSC X rev 1	44.1
ambient_temp	number	Only for MTPC, MSC X rev 1	36.9
cc_temp	number	Only for MSC X rev 2 and Atlas/Atlas Pro	44.1
gpu_temp	number	Only for Atlas/Atlas Pro	38.2

## 4.22 Text Slots

### 4.22.1 Methods

#### PUT

Set the value of a text slot used in the project, which will propagate to all controllers in a project.

PUT /api/text\_slot

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
name	string	Text slot name	"myTextSlot"
value	string	New value for the text slot.	"Hello World!"

#### GET

Returns data about the text slots in the project and their current values.

GET /api/text\_slot[?names=slotNames]

slotNames can be used to filter which test slots are returned and is expected to be either a single string or an array of strings.

Returns a JSON object with a single text\_slots attribute, which has an array value. Each item in the array is a Text Slot object with the following attributes:

Attribute	Value Type	Value Example
name	string	"text"
value	string	"example"

## 4.23 Time

### 4.23.1 Methods

#### GET

Returns data about the time stored in the controller.

GET /api/time

Returns a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
datetime	string	Controller's local time as a string	"01 Feb 2017 13:44:42"
local_time	integer	Controller's local time in milliseconds	1485956682
uptime	integer	Milliseconds since last boot	493347

## 4.24 Timeline

### 4.24.1 Methods

#### POST

Control a timeline in the project. Action will propagate to all controllers in a project.

POST /api/timeline

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
action	string	The action to perform on the timeline(s): <code>start</code> , <code>release</code> , <code>toggle</code> , <code>pause</code> , <code>resume</code> , <code>set_rate</code> , <code>set_position</code>	"start"
num	integer	The number of the timeline to perform the action on. If not present, the action will be applied to all timelines in the project; omitting this attribute is valid for <code>release</code> , <code>pause</code> and <code>resume</code> .	1
fade	number	Optional. The fade time to apply to a <code>release</code> action, in seconds, or the timeline release that results from a <code>toggle</code> action. If not provided, the default release fade time will be used.	2.0
group	string	Optional. Timeline group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A. This attribute is valid for a <code>release</code> action without a specified <code>num</code> , meaning <i>release all timelines</i> .	"B"
rate	string	Required for a <code>set_rate</code> action; invalid otherwise. Value should be a string containing a floating point number or a bounded integer, where 1.0 means the timeline's default rate.	"0.1" or "10:100"
position	string	Required for a <code>set_position</code> action; invalid otherwise. Value should be a string containing a floating point number or a bounded integer, representing a fraction of the timeline length.	"0.1" or "10:100"

For example, to start a timeline 2, the request payload is:

```
{
  "action": "start",
  "num": 2
}
```

To release timeline 2 in 3.5 seconds, the request payload would be:

```
{
  "action": "release",
  "num": 2,
  "fade": 3.5
}
```

To toggle timeline 2, and release it in 2 seconds if it's running, the request payload would be:

```
{
  "action": "toggle",
  "num": 2,
  "fade": 2.0
}
```

To pause timeline 4, the request payload is:

```
{
  "action": "pause",
}
```

(continues on next page)

(continued from previous page)

```
{  
  "num": 4  
}
```

To resume timeline 4, the request payload is:

```
{  
  "action": "resume",  
  "num": 4  
}
```

To pause all timelines, the request payload is:

```
{  
  "action": "pause"  
}
```

To resume all timelines, the request payload is:

```
{  
  "action": "resume"  
}
```

To release all timelines in 2 seconds, the request payload would be:

```
{  
  "action": "release",  
  "fade": 2.0  
}
```

To release all timelines except those in group B in 2 seconds, the request payload would be:

```
{  
  "action": "release",  
  "group": "!B",  
  "fade": 2.0  
}
```

To set the rate of timeline 5 to half the default rate, the request payload would be:

```
{  
  "action": "set_rate",  
  "num": 5,  
  "rate": "0.5"  
}
```

To set the position of timeline 1 to a third of the way through, the request payload would be:

```
{  
  "action": "set_rate",  
  "num": 1,  
  "position": "1:3"  
}
```

## GET

Returns data about the timelines in the project and their state on the controller.

GET /api/timeline[?num=timelineNumbers]

num can be used to filter which timelines are returned and is expected to be either a single number or a string expressing the required timelines, e.g. "1,2,5-9".

Returns a JSON object with a single `timelines` attribute, which has an array value. Each item in the array is a Timeline object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Timeline number	1
name	string	Timeline name	"Timeline 1"
group	string	Timeline group name (A, B, C, D or empty string)	"A"
length	integer	Timeline length, in milliseconds	10000
source_bus	string	internal, timecode_1 ... timecode_6, audio_1 ... audio_4	"internal"
timecode_format	string	Timecode format	"SMPTE30"
audio_band	integer	0 is volume band	0
audio_channel	string	left, right or combined	"combined"
audio_peak	boolean	The Peak setting of the timeline, if set to an audio time source	false
time_offset	integer	1/1000 of a second	5000
state	string	none, running, paused, holding_at_end or released	"running"
onstage	boolean	Whether the timeline is affecting output of any fixtures	true
position	integer	1/1000 of a second	10000
priority	string	high, above_normal, normal, below_normal or low	"normal"
custom_properties	object	Object properties and property values correspond to custom property names and values	{}

## 4.25 Trigger

### 4.25.1 Methods

#### POST

Fire a trigger in the project.

POST /api/trigger

Payload is a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	User number of the trigger to fire.	2
var	string	Optional. Comma-separated to pass into the trigger.	e.g. a string "Foo"; integers 2,4,5; multiple strings '"string1", "string2", "string3"'
conditions	boolean	Optional. Whether to test the trigger's conditions before deciding to run its actions. Defaults to true.	true

## GET

Returns the triggers in the project.

GET /api/trigger?[type=triggerType]

triggerType is expected to be a string and can be used to filter the type of trigger returned. For example, "Timeline Started" would return only Timeline Started triggers in the project.

Returns a JSON object with a single `triggers` attribute, which has an array value. Each item in the array is a Trigger object with the following attributes:

Attribute	Value Type	Description	Value Example
type	string	Trigger type	"Startup"
num	integer	Trigger user number	1
name	string	User-defined trigger name	"Initialise"
group	string	Trigger group colour as a hex colour string	"#e18383"
description	string	User-defined description of trigger	""
trigger_text	string	Generated description of when the trigger will run, based on its properties	"At startup"
conditions	array	Array of Condition objects (see below)	[{"text": "Before 12:00:00 every day"}]
actions	array	Array of Action objects (see below)	[{"text": "Start Timeline 1"}]

The Condition and Action objects have the following properties:

Attribute	Value Type	Description	Value Example
text	string	Generated description of the condition or action, based on its properties	"Start Timeline 1"





## JAVASCRIPT QUERY LIBRARY

Reference for the query.js library.

### 5.1 Beacon

#### 5.1.1 Functions

##### **toggle\_beacon**

Toggle beacon mode on the controller.

`toggle_beacon(callback)`

In beacon mode, a controller will flash its LEDs or its screen continuously.

### 5.2 Channel / Park

#### 5.2.1 Functions

##### **park\_channel**

Park an output channel or channels at a specified level.

`park_channel(params, callback)`

`params` is expected to be an object with the same attributes as the HTTP *POST* request.

## unpark\_channel

Unpark an output channel or channels.

`unpark_channel(params, callback)`

`params` is expected to be an object with the same attributes as the HTTP *DELETE* request.

## 5.3 Command

### 5.3.1 Functions

#### run\_command

Run a Lua script or pass a command to the command line parser on the controller.

---

**Note:** The Command Line Parser must be enabled in the web interface settings of the current project, else this function will not be available.

---

`run_command(params, callback)`

`params` is expected to be an object with the same attributes as the HTTP *POST* request.

Returns Executed if the script was executed successfully or an error string if not.

## 5.4 Config

### 5.4.1 Functions

#### edit\_config

Edits the configuration of the controller.

`edit_config(params, callback)`

`params` is expected to be an object with the same attributes as the HTTP *POST* request.

The `callback` function will be passed the same object as is received from the HTTP *POST* request.

## get\_config

Returns information about the queried controller's configuration.

`get_config(callback)`

Returns an object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_config(config => {
  let year = config.year
})
```

## 5.5 Content Targets

---

**Note:** Atlas/Atlas Pro only

---

### 5.5.1 Functions

#### master\_content\_target\_intensity

`master_content_target_intensity(params, callback)`

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
type	string	Optional. Type of content target (only relevant on Atlas Pro): <code>primary</code> , <code>secondary</code> , <code>target_3</code> , <code>target_4</code> , <code>target_5</code> , <code>target_6</code> , <code>target_7</code> , <code>target_8</code> . Defaults to <code>primary</code> .	"secondary"
level	float or string containing a bounded integer	Master level to set on the group	0.5 or "50:100"
fade	float	Optional. Fade time to apply the intensity change, in seconds.	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds.	2.0

### **get\_content\_target\_info**

get\_content\_target\_info(callback)

Returns an object with a single `content_targets` attribute, which has an array value. Each item in the array is a Content Target object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_content_target_info(c => {  
  let level = c.content_targets[0].level // level of primary content target  
})
```

## **5.6 Controller**

### **5.6.1 Functions**

#### **get\_controller\_info**

get\_controller\_info(callback)

Returns an object with a single `controllers` attribute, which has an array value. Each item in the array is a Controller object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_controller_info(controller => {  
  let name = controller[0].name // name of the first controller  
})
```

## **5.7 Group**

---

**Note:** Not applicable to Atlas/Atlas Pro

---

### **5.7.1 Functions**

#### **master\_intensity**

master\_intensity(params, callback)

Propagates to all controllers in a project.

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Group number. Group 0 means the <i>All Fixtures</i> group.	1
level	float or string containing a bounded integer	Master level to set on the group	0.5 or "50:100"
fade	float	Optional. Fade time to apply the intensity change, in seconds.	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds.	2.0

For example:

```
// Master group 1 to 50% in 3 seconds
Query.master_intensity({
  "num": 1,
  "level": "50:100",
  "fade": 3
}, result => {
  // Check for error
})
```

## get\_group\_info

Returns data about the fixture groups in the project.

`get_group_info(callback[, num])`

Returns an object with a single `groups` attribute, which has an array value. Each item in the array is a `Group` object with the same attributes as in the HTTP *GET* response.

`num` can be used to filter which groups are returned and is expected to be a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
num	string <b>or</b> integer	Define the numbers of the group that should be returned	"1,2,5-9" <b>or</b> 5

**Note:** Group 0 will return data about the *All Fixtures* group.

For example:

```
Query.get_group_info(g => {
  let name = g.groups[0].name // name of the first group returned
}, {"num": "2-4"})
```

## 5.8 Input

There's no function in the JavaScript Query library to get the digital & analogue inputs at the moment.

## 5.9 Log

There's no function in the JavaScript Query library to get the log at the moment.

## 5.10 Lua Variable

### 5.10.1 Functions

#### **get\_lua\_variables**

Returns the current value of specified Lua variables.

`get_lua_variables(luaVariables, callback)`

Returns an object with the requested Lua variables and their values as key/value pairs, in the same manner as the HTTP *GET* request.

`luaVariables` can be a string or an array of strings, where each string is a Lua variable name.

For example:

```
Query.get_lua_variables(["foo", "bar"], v => {  
  let foo = v.foo  
  let bar = v.bar  
})
```

## 5.11 Output

### 5.11.1 Functions

#### **disable\_output**

Disable the output of a specified protocol from the controller. Propagates to all controllers in a project.

`disable_output(params, callback)`

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
protocol	string	Protocol to disable. Options: dmx, pathport, sacn, art-net, kinet, rio-dmx, edn, edn-spi.	"parthport"

## enable\_output

Enable the output of a specified protocol from the controller. Propagates to all controllers in a project.

`enable_output(params, callback)`

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
protocol	string	Protocol to enable. Options: <code>dmx</code> , <code>pathport</code> , <code>sacn</code> , <code>art-net</code> , <code>kinet</code> , <code>rio-dmx</code> , <code>edn</code> , <code>edn-spi</code> .	"parthport"

## get\_output

Returns the lighting levels being output by the queried controller.

`get_output(universeKey, callback)`

Returns an object with the same attributes as in the HTTP *GET* response.

`universeKey` can be a string (see *Universe Key String Format*) or it can be an object with the following attributes:

Attribute	Value Type	Description
protocol	integer	Constants defined in <code>query.js</code> are: <code>DMX</code> (1), <code>PATHPORT</code> (2), <code>ARTNET</code> (4), <code>KINET</code> (8), <code>SACN</code> (16), <code>DVI</code> (32), <code>RIO_DMX</code> (64), <code>EDN</code> (128)
index	integer	Required unless <code>protocol</code> is <code>KINET</code> , <code>RIO_DMX</code> or <code>EDN</code>
kinet_power_supply_num	integer	Only required if <code>protocol</code> is <code>KINET</code>
kinet_port	integer	Only required if <code>protocol</code> is <code>KINET</code>
remote_device_type	integer	Only required if <code>protocol</code> is <code>RIO_DMX</code> or <code>EDN</code>
remote_device_num	integer	Only required if <code>protocol</code> is <code>RIO_DMX</code> or <code>EDN</code>
port	integer	Only required if <code>protocol</code> is <code>EDN</code>

For example:

```

Query.get_output({
  protocol: KINET,
  kinet_port: 1,
  kinet_power_supply_num: 1
}, u => {
  console.log(u)
})

Query.get_output({
  protocol: DMX,
  index: 1
}, u => {
  console.log(u)
})

Query.get_output("dmx:1", u => {

```

(continues on next page)

(continued from previous page)

```
    console.log(u)
  })
```

## 5.11.2 Universe Key String Format

A universe key string takes the form:

- `protocol:index` for protocols `dmx`, `pathport`, `sacn`, `art-net`;
- `protocol:kinetPowerSupplyNum:kinetPort` for protocol `kinet`;
- `protocol:remoteDeviceType:remoteDeviceNum` for protocol `rio-dmx`;
- `protocol:remoteDeviceType:remoteDeviceNum:port` for protocols `edn`, `edn-spi`.

Where:

- `kinetPowerSupplyNum` is an integer;
- `kinetPort` is an integer;
- `remoteDeviceType` can be `rio08`, `rio44` or `rio80`, `edn10` or `edn20`;
- `remoteDeviceNum` is an integer;
- `port` is an integer.

For example:

- `"dmx:1"`
- `"rio-dmx:rio44:1"`

## 5.12 Override

### 5.12.1 Functions

#### **set\_group\_override**

Set the Intensity, Red, Green, Blue levels for a group. Propagates to all controllers in a project.

`set_group_override(params, callback)`

`params` is expected to be an object with the following attributes:



Attribute	Value Type	Description	Value Example
num	integer	Group or fixture number, depending on target. Group 0 means the <i>All Fixtures</i> group.	1
intensity	integer	Optional. Intensity to set as part of override: 0-255. Intensity override will not be changed if this attribute isn't provided.	128
red	integer	Optional. Red component to set as part of override: 0-255. Red override will not be changed if this attribute isn't provided.	255
green	integer	Optional. Green component to set as part of override: 0-255. Green override will not be changed if this attribute isn't provided.	255
blue	integer	Optional. Blue component to set as part of override: 0-255. Blue override will not be changed if this attribute isn't provided.	255
temperature	integer	Optional. Temperature component to set as part of override: 0-255. Temperature override will not be changed if this attribute isn't provided.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Braked"

### clear\_group\_overrides

Release any overrides on a group, or all groups. Propagates to all controllers in a project.

`clear_group_overrides(params, callback)`

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Optional. Group number. If not provided, all overrides are cleared.	1
fade	float	Optional. Fade time in which to release overrides, in seconds.	2.0

### set\_fixture\_override

Set the Intensity, Red, Green, Blue levels for a fixture. Propagates to all controllers in a project.

```
set_fixture_override(params, callback)
```

params is expected to be an object with the same attributes as for *set\_group\_override*.

### clear\_fixture\_overrides

Release any overrides on a fixture, or all fixtures. Propagates to all controllers in a project.

```
clear_fixture_overrides(params, callback)
```

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Optional. Fixture number. If not provided, all overrides are cleared.	1
fade	float	Optional. Fade time in which to release overrides, in seconds.	2.0

### clear\_all\_overrides

Release all overrides. Propagates to all controllers in a project.

```
clear_all_overrides(params, callback)
```

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
fade	float	Optional. Fade time in which to release overrides, in seconds.	2.0

## 5.13 Project

### 5.13.1 Functions

#### get\_project\_info

Returns data about the current project.

```
get_project_info(callback)
```

Returns an object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_project_info(project => {
  const author = project.author
})
```

## 5.14 Protocol

### 5.14.1 Functions

#### get\_protocols

Returns all the universes in the project on the queried controller.

get\_protocols(callback)

Returns an object with a single `outputs` attribute, which has an array value. Each item in the array is a Protocol object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_protocols(p => {
  const protocol_name = p.outputs[0].name // name of the first protocol
})
```

## 5.15 Remote Device

### 5.15.1 Functions

#### get\_remote\_device\_info

Returns data about all the remote devices in the project.

get\_remote\_device\_info(callback)

Returns an object with a single `remote_devices` attribute, which has an array value. Each item in the array is a Remote Device object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_remote_device_info(r => {
  const type = r.remote_devices[0].type // type of the first remote device
})
```

## 5.16 Replication

### 5.16.1 Functions

#### **get\_replication**

Returns data about the install replication.

`get_replication(callback)`

Returns an object with the same attributes as in the HTTP *GET* response.

## 5.17 Scene

### 5.17.1 Functions

#### **start\_scene**

`start_scene(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Scene number	5

For callback please see *JavaScript Command Callback*.

#### **release\_scene**

`release_scene(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Scene number	5
<code>fade</code>	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0

For callback please see *JavaScript Command Callback*.

**toggle\_scene**

`toggle_scene(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Scene number	5
<code>fade</code>	float	Optional. The release fade time in seconds to apply if the toggle action results in the scene being released. If not provided, the default fade time will be used.	2.0

For callback please see *JavaScript Command Callback*.

**release\_all\_scenes**

`release_all_scenes(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>fade</code>	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
<code>group</code>	string	Optional. Scene group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

For callback please see *JavaScript Command Callback*.

**release\_all**

Release all timelines and scenes. Propagates to all controllers in a project.

`release_all(params, callback)`

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>fade</code>	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
<code>group</code>	string	Optional. Timeline/Scene group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

For callback please see *JavaScript Command Callback*.

### get\_scene\_info

Returns data about the scenes in the project and their state on the controller.

`get_scene_info(callback[, num])`

Returns an object with a single `scenes` attribute, which has an array value. Each item in the array is a Scene object with the same attributes as in the HTTP GET response.

`num` can be used to filter which scenes are returned and is expected to be a JSON object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	string <b>or</b> integer	Define the numbers of the scene that should be returned	<code>"1,2,5-9"</code> <b>or</b> <code>5</code>

For example:

```
Query.get_scene_info(s => {  
  let name = s.scenes[0].name // name of the first scene returned  
}, {"num": "1,2-5"})
```

## 5.17.2 JavaScript Command Callback

Functions in the JavaScript API that perform actions on the controller, e.g. `start_timeline`, have an optional `callback` argument. This expects a function, which is called when a response to the underlying HTTP API request is received. Its argument, if non-null, is the response body. If the content type of the response was `"application/json"` then the argument will be an object - the result of parsing the body as JSON.

## 5.18 System

### 5.18.1 Functions

#### get\_system\_info

`get_system_info(callback)`

Returns an object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_system_info(system => {  
  const capacity = system.channel_capacity  
})
```

## 5.19 Temperature

### 5.19.1 Functions

#### get\_temperature

get\_temperature(callback)

Returns an object with the same attributes as in the HTTP *GET* response.

For example:

```
Query.get_temperature(temp => {
  const ambient = temp.ambient_temp
})
```

## 5.20 Text Slots

### 5.20.1 Functions

#### set\_text\_slot

Set the value of a text slot used in the project, which will propagate to all controllers in a project.

set\_text\_slot(params, callback)

params is expected to be an object with the same attributes as the HTTP *PUT* request.

#### get\_text\_slot

Returns data about the text slots in the project and their current values.

get\_text\_slot(callback[, filter])

Returns an object with a single `text_slots` attribute, which has an array value. Each item in the array is a Text Slot object with the same attributes as in the HTTP *GET* response.

`filter` can be used to filter which text slots are returned and is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
names	string <b>or</b> array	Define the names of the text slots that should be returned, either as a single string or an array of strings	["test_slot1", "anotherSlot"] <b>or</b> "test_slot1"

For example:

```
Query.get_text_slot(t => {
  let value = t.text_slots[0].value // value of the first text slot returned
}, {"names":["test_slot1","test_slot2"]})
```

## 5.21 Time

### 5.21.1 Functions

#### **get\_current\_time**

`get_current_time(callback)`

Returns an object with the same attributes as in the *GET* GET response.

For example:

```
Query.get_current_time(time => {  
  const uptime = time.uptime  
})
```

## 5.22 Timeline

### 5.22.1 Functions

#### **start\_timeline**

`start_timeline(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	5

For callback please see *JavaScript Command Callback*.

#### **release\_timeline**

`release_timeline(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	5
<code>fade</code>	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0

For callback please see *JavaScript Command Callback*.



**toggle\_timeline**

`toggle_timeline(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	5
<code>fade</code>	float	Optional. The release fade time in seconds to apply if the toggle action results in the timeline being re-released. If not provided, the default fade time will be used.	2.0

For callback please see *JavaScript Command Callback*.

**pause\_timeline**

`pause_timeline(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	5

For callback please see *JavaScript Command Callback*.

**resume\_timeline**

`resume_timeline(params, callback)`

Propagates to all controllers in a project.

`params` is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	5

For callback please see *JavaScript Command Callback*.

**pause\_all**

Pause all timelines in the project which are currently running. Propagates to all controllers in a project.

`pause_all(callback)`

For callback please see *JavaScript Command Callback*.

### **resume\_all**

Resume all timelines in the project which are currently paused. Propagates to all controllers in a project.

`resume_all(callback)`

For callback please see *JavaScript Command Callback*.

### **release\_all\_timelines**

`release_all_timelines(params, callback)`

Propagates to all controllers in a project.

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
fade	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
group	string	Optional. Timeline group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

For callback please see *JavaScript Command Callback*.

### **release\_all**

Release all timelines and scenes. Propagates to all controllers in a project.

`release_all(params, callback)`

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
fade	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
group	string	Optional. Timeline/Scene group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

For callback please see *JavaScript Command Callback*.

### **set\_timeline\_rate**

`set_timeline_rate(params, callback)`

Propagates to all controllers in a project.

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Timeline number	5
rate	string	A string containing a floating point number or a bounded integer, where 1.0 means the timeline's default rate.	"0.1" or "10:100"

For callback please see *JavaScript Command Callback*.

### set\_timeline\_position

`set_timeline_position(params, callback)`

Propagates to all controllers in a project.

params is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Timeline number	5
position	string	A string containing a floating point number or a bounded integer, representing a fraction of the timeline length.	"0.1" or "10:100"

For callback please see *JavaScript Command Callback*.

### get\_timeline\_info

`get_timeline_info(callback[, num])`

Returns data about the timelines in the project and their state on the controller.

Returns an object with a single `timelines` attribute, which has an array value. Each item in the array is a Timeline object with the same attributes as in the HTTP GET response.

num can be used to filter which timelines are returned and is expected to be an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	string <b>or</b> integer	Define the numbers of the timeline that should be returned	"1, 2, 5-9" <b>or</b> 5

For example:

```
Query.get_timeline_info(t => {
  let name = t.timelines[0].name // name of the first timeline returned
}, {"num": "1-4"})
```

## 5.22.2 JavaScript Command Callback

Functions in the JavaScript API that perform actions on the controller, e.g. `start_timeline`, have an optional `callback` argument. This expects a function, which is called when a response to the underlying HTTP API request is received. Its argument, if non-null, is the response body. If the content type of the response was "application/json" then the argument will be an object - the result of parsing the body as JSON.

## 5.23 Trigger

### 5.23.1 Functions

#### **fire\_trigger**

`fire_trigger(params, callback)`

`params` is expected to be an object with the same attributes as the HTTP *POST* request.

#### **get\_trigger\_info**

`get_trigger_info(callback[, type])`

Returns an object with a single `triggers` attribute, which has an array value. Each item in the array is a Trigger object with the same attributes as in the HTTP *GET* response.

`type` is expected to be a string and can be used to filter the type of trigger returned. For example, "Timeline Started" would return only Timeline Started triggers in the project.

For example:

```
Query.get_trigger_info(t => {  
  let name = t.triggers[0].name // name of first startup trigger returned  
}, "Startup")
```

## WEBSOCKETS

Reference for controller websocket clients.

### 6.1 Websocket Subscriptions

Subscriptions allow data to be pushed to the web client whenever there is a change within the project.

#### 6.1.1 JavaScript

##### **subscribe\_timeline\_status**

Subscribe to changes in timeline status.

`subscribe_timeline_status(callback)`

The `callback` is called each time a timeline changes state on the controller. Each time it is passed an object with the following attributes:

Attribute	Value Type	Description	Value Example
<code>num</code>	integer	Timeline number	1
<code>state</code>	string	The new state of the timeline: <code>none</code> , <code>running</code> , <code>paused</code> , <code>holding_at_end</code> , <code>released</code>	"running"
<code>onstage</code>	boolean	Whether the timeline is currently affecting the output of any fixtures in the project.	true
<code>position</code>	integer	Current time position of the timeline playback, in milliseconds	5000

For example:

```
Query.subscribe_timeline_status(t => {  
  alert(t.num + ": " + t.state)  
})
```

### **subscribe\_scene\_status**

Subscribe to changes in scene status.

`subscribe_scene_status(callback)`

The `callback` is called each time a scene changes state on the controller. Each time it is passed an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Scene number	1
state	string	The new state of the scene: <code>none</code> , <code>started</code> , <code>released</code>	"started"
onstage	boolean	Whether the scene is currently affecting the output of any fixtures in the project.	true

For example:

```
Query.subscribe_scene_status(s => {  
  alert(s.num + ": " + s.state)  
})
```

### **subscribe\_group\_status**

Subscribe to changes in group level, as set by the Master Intensity action.

`subscribe_group_status(callback)`

The `callback` is called each time the group master level changes on the controller. Each time it is passed an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Group number	1
name	string	Group name	"Group 1"
level	integer	New master intensity level of the group: 0-255	128

For example:

```
Query.subscribe_group_status(g => {  
  alert(g.num + ": " + g.level)  
})
```

### **subscribe\_remote\_device\_status**

Subscribe to changes in remote device online/offline status.

`subscribe_remote_device_status(callback)`

The `callback` is called each time the remote device online/offline status changes. Each time it is passed an object with the following attributes:

Attribute	Value Type	Description	Value Example
num	integer	Remote device number	1
type	string	Type of remote device: RIO 80, RIO 44, RIO 08, BPS, RIO A, RIO D, EDN 20, EDN 10	"RIO 80"
online	boolean	New online state of the remote device	true
serial	string	Remove device serial number	"001001"

For example:

```
Query.subscribe_remote_device_status(r => {
  alert(r.num + ": " + (r.online ? "online" : "offline"))
})
```

### subscribe\_beacon

Subscribe to changes in the device beacon.

subscribe\_beacon(callback)

The callback is called each time the controller beacon status changes. Each time it is passed an object with the following attributes:

Attribute	Value Type	Description	Value Example
on	boolean	New beacon status	true

For example:

```
Query.subscribe_beacon(b => {
  alert(b.on ? "Beacon turned on" : "Beacon turned off")
})
```

### subscribe\_lua

The receiver for the push\_to\_web() Lua function.

subscribe\_lua(callback)

The callback is called each time a script on the controller calls the push\_to\_web() function. Each time it is passed an object with a single attribute - the name or key string passed as the first argument to push\_to\_web(). The value of this attribute is the second argument passed to push\_to\_web(), converted to a string.

For example, if a project needs to send a touch slider level to the web interface, it might have the following in a trigger Lua script:

```
level = getMySliderLevel() -- user-defined function to get the current slider level
push_to_web("slider_level", level) -- invoke callbacks on subscribers
```

If level is equal to e.g. 56 then the object passed the JavaScript callback will be:

```
{
  "slider_level": "56"
}
```

And the subscription could be setup as follows:

```
Query.subscribe_lua(l => {  
    key = Object.keys(l)[0] // "slider_level" in the above example  
    value = l.key           // "56" in the above example  
    alert(key + ": " + value)  
})
```



## LUA API

Mosaic controllers offer a Lua API providing access to system information, playback functions and trigger operations.

### 7.1 Adjustment Target

---

**Note:** Only supported on Atlas Pro.

---

An `Adjustment` object is returned from `get_adjustment`.

#### 7.1.1 Properties

Property	Value Type
<code>rotation_offset</code>	float
<code>x_position_offset</code>	float
<code>y_position_offset</code>	float

For example:

```
target = get_adjustment(1)
r_offset = target.rotation_offset
```

#### 7.1.2 Member functions

The following are member functions of `Adjustment` objects.

### transition\_rotation

```
transition_rotation([angle[, count[, period[, delay[, useShortestPath]]]]])
```

Applies a rotation to the adjustment target according to the parameters:

Parameter	Value Type	Description	Value	Example
angle	float	Optional. Angle of rotation to transition to, in degrees. Defaults to zero.	90.0	
count	integer	Number of times to repeat the rotation transformation.	1	
period	integer	The period of the rotation, in seconds - the time to perform one count of the transformation.	2	
delay	integer	Time to wait before starting the rotation, in seconds.	0	

### transition\_x\_position

```
transition_x_position([x_offset[, count[, period[, delay]]]])
```

Moves the adjustment target along the x axis according to the parameters:

Parameter	Value Type	Description	Value	Example
x_offset	float	Optional. Offset to apply to the x position. Defaults to 0.	25.0	
count	integer	Number of times to repeat the x translation.	1	
period	integer	The period of the translation, in seconds - the time to perform one count of the transformation.	2	
delay	integer	Time to wait before starting the translation, in seconds.	0	

### transition\_y\_position

```
transition_y_position([x_offset[, count[, period[, delay]]]])
```

Moves the adjustment target along the y axis according to the parameters:

Parameter	Value Type	Description	Value	Example
y_offset	float	Optional. Offset to apply to the y position. Defaults to 0.	25.0	
count	integer	Number of times to repeat the y translation.	1	
period	integer	The period of the translation, in seconds - the time to perform one count of the transformation.	2	
delay	integer	Time to wait before starting the translation, in seconds.	0	

## 7.2 BPS

A BPS object is returned from *get\_bps*.

### 7.2.1 Member functions

The following are member functions of BPS objects.

#### get\_state

`get_state(buttonNum)`

Returns the state of the button with integer number `buttonNum`, which can be one of the constants `RELEASED`, `PRESSED`, `HELD` or `REPEAT`.

For example:

```
bps = get_bps(1)
btn = bps.get_state(1)
```

#### set\_led

`set_led(button, effect[, intensity[, fade]])`

Set the effect and intensity of a BPS button LED according to the parameters:

Parameter	Value Type	Description	Value Example
<code>button</code>	integer (1-8)	Number of the BPS button to set an effect on	1
<code>effect</code>	integer	Integer value of constants: <code>OFF</code> , <code>ON</code> , <code>SLOW_FLASH</code> , <code>FAST_FLASH</code> , <code>DOUBLE_FLASH</code> , <code>BLINK</code> , <code>PULSE</code> , <code>SINGLE</code> , <code>RAMP_ON</code> , <code>RAMP_OFF</code>	<code>SLOW_FLASH</code>
<code>intensity</code>	integer (0-255)	Optional. Intensity level to set on the LED. If this parameter is not specified, full intensity will be set on the LED.	255
<code>fade</code>	float	Optional. Fade time to apply the override change, in seconds.	2.0

For example:

```
-- Set button 1 on BPS 1 to Fast Flash at full intensity
get_bps(1).set_led(1,FAST_FLASH,255)
```

## 7.3 Content Target

---

**Note:** Only supported on Atlas and Atlas Pro.

---

A ContentTarget object is returned from *get\_content\_target*.

### 7.3.1 Properties

Property	Value Type	Description
master_intensity_level	<i>Variant</i>	
rotation_offset	float	Atlas Pro only
x_position_offset	float	Atlas Pro only
y_position_offset	float	Atlas Pro only

For example, on a Atlas:

```
target = get_content_target(1)
current_level = target.master_intensity_level
```

And on a Atlas Pro:

```
target = get_content_target(1, PRIMARY)
current_angle = target.rotation_offset
```

### 7.3.2 Member functions

The following are member functions of ContentTarget objects.

#### set\_master\_intensity

`set_master_intensity(level[, fade[, delay]])`

Masters the intensity of the content target according to the parameters:

Parameter	Value Type	Description	Value Example
level	float (0.0-1.0) or integer (0-255)	Master level to set on the content target.	0.5 or 128
fade	float	Optional. Fade time to apply the intensity change, in seconds.	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds.	3.0

For example, on a Atlas:

```
-- Master the primary content target in composition 1 to 50% (128/255 = 0.5) in 3 seconds
get_content_target(1):set_master_intensity(128,3)
```

Or on a Atlas Pro:

```
-- Master the secondary content target in composition 2 to 100% in 2.5 seconds
get_content_target(2, SECONDARY):set_master_intensity(255,2.5)
```

### transition\_rotation

**Note:** Only supported on Atlas Pro.

```
transition_rotation([angle[, count[, period[, delay[, useShortestPath]]]])
```

Applies a rotation to the content target according to the parameters:

Parameter	Value Type	Description	Value	Example
angle	float	Optional. Angle of rotation to transition to, in degrees. Defaults to zero.	90.0	
count	integer	Number of times to repeat the rotation transformation.	1	
period	integer	The period of the rotation, in seconds - the time to perform one count of the transformation.	2	
delay	integer	Time to wait before starting the rotation, in seconds.	0	

### transition\_y\_position

```
transition_y_position([y_offset[, count[, period[, delay]]]])
```

Moves the content target along the y axis according to the parameters:

Parameter	Value Type	Description	Value	Example
y_offset	float	Optional. Offset to apply to the y position. Defaults to 0.	25.0	
count	integer	Number of times to repeat the y translation.	1	
period	integer	The period of the translation, in seconds - the time to perform one count of the transformation.	2	
delay	integer	Time to wait before starting the translation, in seconds.	0	

## 7.4 Controller

A Controller object is returned from e.g. `get_current_controller`.

## 7.4.1 Properties

Property	Value Type	Description	Value Example
number	integer	Controller number	1
name	string	Controller name	"Controller 1"
vlan_tag	string	VLAN tag number as a string. "None" if there is no tag set	"65535"
is_network_primary	boolean	Whether this controller is set as the Network Primary in the project	true

For example:

```
cont = get_current_controller()
name = cont.name
```

## 7.5 DateTime

A DateTime object is returned from e.g. *System* properties.

### 7.5.1 Properties

Property	Value Type	Value Example
year	integer	2022
month	integer	12
monthday	integer	3
time_string	string	"11:35:32"
date_string	string	"03 Dec 2022"
weekday	integer (0 => Sunday)	0
hour	integer	11
minute	integer	35
second	integer	32
utc_timestamp	integer	1670045912

## 7.6 Group

A Group object is returned from *get\_group*.

## 7.6.1 Properties

Property	Value Type	Description	Value Example
name	string	Group name	"Group 1"
master_intensity_level	<i>Variant</i>	The intensity level that this group is currently being mastered to	

For example:

```
grp = get_group(1)
name = grp.name
```

## 7.6.2 Member functions

The following are member functions of `Group` objects.

### set\_master\_intensity

`set_master_intensity(level[, fade[, delay]])`

Masters the intensity of the group according to the parameters:

Parameter	Value Type	Description	Value Example
level	float (0.0-1.0) or integer (0-255)	Master level to set on the group	0.5 or 128
fade	float	Optional. Fade time to apply the intensity change, in seconds	2.0
delay	float	Optional. Time to wait before applying the intensity change, in seconds	3.0

For example:

```
-- Master group 1 to 50% (128/255 = 0.5) in 3 seconds
get_group(1):set_master_intensity(128,3)
```

## 7.7 Location

A `Location` object is returned from `get_location`.

## 7.7.1 Properties

Property	Value Type	Value Example
lat	float	51.512
long	float	-0.303

For example:

```
lat = get_location().lat
```

## 7.8 Override

An Override object is returned from *get\_fixture\_override* and *get\_group\_override*.

### 7.8.1 Member functions

The following are member functions of Override objects.

#### set\_irgb

`set_irgb(intensity, red, green, blue, [fade, [path]])`

Overrides the intensity, red, green and blue levels for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
intensity	integer (0-255)	Intensity level to set as an override.	128
red	integer (0-255)	Red level to set as an override.	128
green	integer (0-255)	Green level to set as an override.	128
blue	integer (0-255)	Blue level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

For example:

```
-- Get override for fixture 22
override = get_fixture_override(22)
-- Set the override colour to red (and full intensity)
override:set_irgb(255, 255, 0, 0)
```



### set\_intensity

set\_intensity(intensity, [fade, [path]])

Overrides the intensity level for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
intensity	integer (0-255)	Intensity level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

For example:

```
-- Get override for group 3
override = get_group_override(3)
-- Set the intensity to 50% in 2 seconds
override.set_intensity(128, 2.0)
```

### set\_red

set\_red(red, [fade, [path]])

Overrides the red level for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
red	integer (0-255)	Red level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

### set\_green

set\_green(green, [fade, [path]])

Overrides the green level for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
green	integer (0-255)	Green level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

### **set\_blue**

```
set_blue(blue, [fade, [path]])
```

Overrides the blue level for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
blue	integer (0-255)	Blue level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

### **set\_temperature**

```
set_temperature(temperature, [fade, [path]])
```

Overrides the temperature level for the fixture or group according to the parameters:

Parameter	Value Type	Description	Value Example
temperature	integer (0-255)	Temperature level to set as an override.	128
fade	float	Optional. Fade time to apply the override change, in seconds.	2.0
path	string	Optional. Crossfade path to use when applying the override: Default, Linear, Start, End, Braked, Accelerated, Damped, Overshoot, Col At Start, Col At End, Int At Start, Int At End, Colour First, Intensity First	"Linear"

**clear**

```
clear([fade])
```

Removes any override on the fixture or group. Optionally specify a fade time in seconds as a float, e.g. 2.0.

For example:

```
-- Clear the override on fixture 1
get_fixture_override(1):clear()
```

See also: [clear\\_all\\_overrides](#).

## 7.9 Project

A Project object is returned from [get\\_current\\_project](#).

### 7.9.1 Properties

Property	Value Type	Value Example
name	string	"Help Project"
author	string	"Contoso"
filename	string	"help_project_v1.pd2"
unique_id	string	"{6b48627a-1d5e-4b2f-81e2-481e092a6a79}"

For example:

```
project_name = get_current_project().name
```

## 7.10 Network 2

Information about the controller's second network interface is available in the `protocol_interface` namespace. In trigger action scripts the `protocol_interface` namespace is added directly to the environment; in IO modules it is in the controller namespace, i.e. `controller.protocol_interface`.

### 7.10.1 Properties

The `protocol_interface` namespace has the following properties:

Property	Value Type	Value Example
has_interface	boolean	true
is_up	boolean	true
ip_address	string	"192.168.1.12"
subnet_mask	string	"255.255.255.0"
gateway	string	"192.168.1.1"

For example:

```
if protocol_interface.has_interface == true then
  ip = protocol_interface.ip_address
end
```

## 7.11 Replication

A Replication object is returned from *get\_current\_replication*.

### 7.11.1 Properties

Property	Value Type	Value Example
name	string	"Help Project"
unique_id	string	"{6b48627a-1d5e-4b2f-81e2-481e092a6a79}"

For example:

```
rep_name = get_current_replication().name
```

## 7.12 RIO

A RIO object is returned from *get\_rio*.

For example:

```
rio = get_rio(RIO44, 1)
input = rio:get_input(1)
output_state = rio:get_output(1)
```

### 7.12.1 Member functions

The following are member functions of RIO objects.

#### **get\_input**

`get_input(inputNum)`

Returns the state of the input with integer number `inputNum` as a boolean if the input is set to Digital or Contact Closure, or an integer if the input is set to Analog.

For example:

```
rio = get_rio(RIO44, 3)
input = rio:get_input(1)
```

## get\_output

`get_output(outputNum)`

Returns the state of the output with integer number `outputNum` as a boolean.

For example:

```

rio = get_rio(RIO44, 2)
output_state = rio.get_output(1)

```

## set\_output

`set_output(outputNum, state)`

Sets the output of a RIO to on or off according to the parameters:

Parameter	Value Type	Description	Value Example
<code>outputNum</code>	integer (1-8)	Number of the RIO output to change the state of. Range depends on type of RIO.	1
<code>state</code>	boolean or integer	State to set the output to. Can be any of: 0, 1, true, false, ON or OFF	OFF

## 7.13 Scene

A Scene object is returned from `get_scene`.

### 7.13.1 Properties

Property	Value Type	Description	Value Example
<code>name</code>	string	Scene name	"Scene 1"
<code>group</code>	string	Scene group name (A, B, C, D or empty string)	"A"
<code>state</code>	integer	Integer value of constants: <code>Scene.NONE</code> , <code>Scene.STARTED</code> or <code>Scene.RELEASED</code>	1
<code>onstage</code>	boolean	Whether the scene is affecting output of any fixtures	false
<code>custom_properties</code>	table	Table keys and values correspond to custom property names and values	

For example:

```

scn = get_scene(1)
name = scn.name
state = scn.state

```

### 7.13.2 Member functions

The following are member functions of Scene objects.

#### **start**

`start()`

Starts the scene. For example:

```
-- start scene 1
get_scene(1):start()
```

#### **release**

`release([fade])`

Releases the scene. Optionally specify a fade time in seconds as a float, e.g. 2.0.

For example:

```
-- release scene 3 with a fade of 1 second
get_scene(3):release(1.0)
```

#### **toggle**

`toggle([fade])`

Toggles the playback of the scene - if it's running, release it; if it's not running, start it. Optionally specify a release fade time in seconds as a float, e.g. 2.0.

For example:

```
-- toggle scene 2, releasing in time 3 secs if it's running
get_scene(2):release(3.0)
```

## 7.14 System

In trigger action scripts the `system` namespace is added directly to the environment; in IO modules it is in the controller namespace, i.e. `controller.system`.

### 7.14.1 Properties

The system namespace has the following properties:

Property	Value Type	Value Example
hardware_type	string	"lpc"
channel_capacity	integer	512
serial_number	string	"006321"
memory_total	string	"12790Kb"
memory_used	string	"24056Kb"
memory_available	string	"103884Kb"
storage_size	string	"1914MB"
bootloader_version	string	"0.9.0"
firmware_version	string	"2.8.0"
reset_reason	string	"Software Reset"
last_boot_time	<i>DateTime</i>	
ip_address	string	"192.168.1.3"
subnet_mask	string	"255.255.255.0"
broadcast_address	string	"192.168.1.255"
default_gateway	string	"192.168.1.3"
dns_servers	table of strings	"1.1.1.1", "1.0.0.1"

For example:

```
capacity = system.channel_capacity

boot_time = system.last_boot_time.time_string
```

## 7.15 Temperature

A Temperature object is returned from *get\_temperature*.

### 7.15.1 Properties

Property	Value Type	Description	Value Example
sys_temp	number	Only for MSC X and Atlas/Atlas Pro	40.2
core1_temp	number	Only for MSC X and Atlas/Atlas Pro	44
core2_temp	number	Only for MSC X rev 1	44.1
ambient_temp	number	Only for MTPC, MSC X rev 1	36.9
cc_temp	number	Only for MSC X rev 2 and Atlas/Atlas Pro	44.1
gpu_temp	number	Only for Atlas/Atlas Pro	38.2

For example:

```
temp = get_temperature()
log(temp.ambient_temp)
```

## 7.16 Time

Information about the controller's clock is available in the `time` namespace. In trigger action scripts the `time` namespace is added directly to the environment; in IO modules it is in the `controller` namespace, i.e. `controller.time`.

### 7.16.1 Properties

The `time` namespace has the following properties:

Property	Value Type	Value Example
<code>is_dst</code>	boolean	<code>true</code>
<code>gmt_offset</code>	integer	<code>1</code>

### 7.16.2 Functions

The `time` namespace has the following functions, which each return a *DateTime* object:

- `get_current_time()`
- `get_sunrise()`
- `get_sunset()`
- `get_civil_dawn()`
- `get_civil_dusk()`
- `get_nautical_dawn()`
- `get_nautical_dusk()`
- `get_new_moon()`
- `get_first_quarter()`
- `get_full_moon()`
- `get_third_quarter()`

For example:

```
current_hour = time.get_current_time().hour
```

## 7.17 Timeline

A `Timeline` object is returned from *get\_timeline*.



### 7.17.1 Properties

Property	Value Type	Description	Value Example
name	string	Timeline name	"Timeline 1"
group	string	Timeline group name (A, B, C, D or empty string)	"A"
length	integer	Timeline length, in milliseconds	10000
source_bus	integer	Integer value of constants: DEFAULT, TCODE_1 ... TCODE_6, AUDIO_1 ... AUDIO_4	1
timecode_format	string	Timecode format	"SMPTE30"
audio_band	integer	0 is equivalent to the constant: VOLUME	0
audio_channel	integer	Integer value of constants: LEFT, RIGHT or COMBINED	1
audio_peak	boolean	The Peak setting of the timeline, if set to an audio time source	false
time_offset	integer	Milliseconds	5000
state	integer	Integer value of constants: Timeline.NONE, Timeline.RUNNING, Timeline.PAUSED, Timeline.HOLDING_AT_END or Timeline.RELEASED	1
onstage	boolean	Whether the timeline is affecting output of any fixtures	true
position	integer	Milliseconds	5000
priority	integer	Integer value of constants: HIGH_PRIORITY, ABOVE_NORMAL_PRIORITY, NORMAL_PRIORITY, BELOW_NORMAL_PRIORITY or LOW_PRIORITY	0
custom_properties	table	Table keys and values correspond to custom property names and values	

For example:

```

tl = get_timeline(1)
name = tl.name
state = tl.state

if (tl.source_bus == TCODE_1) then
  -- do something
end

```

### 7.17.2 Member functions

The following are member functions of Timeline objects.

#### **start**

`start()`

Starts the timeline. For example:

```
-- start timeline 1
get_timeline(1):start()
```

#### **release**

`release([fade])`

Releases the timeline. Optionally specify a fade time in seconds as a float, e.g. 2.0.

For example:

```
-- release timeline 3
get_timeline(3):release(1.0)
```

#### **toggle**

`toggle([fade])`

Toggles the playback of the timeline - if it's running, release it; if it's not running, start it. Optionally specify a release fade time in seconds as a float, e.g. 2.0.

For example:

```
-- toggle timeline 2, releasing in time 3 secs if it's running
get_timeline(2):release(3.0)
```

#### **pause**

`pause()`

Pauses the timeline.

#### **resume**

`resume()`

Resumes the timeline.

### set\_rate

set\_rate(rate)

Sets the rate of playback of the timeline. Set the rate as a float or an integer with range, e.g. 0.1 or Variant(10, 100) would set the rate to 10% of normal speed.

For example:

```

-- set the rate of timeline 1 to 20% of normal speed
get_timeline(1):set_rate(0.2)
-- set the rate of timeline 2 to 30% of normal speed
get_timeline(2):set_rate(Variant(30,100))

```

### set\_position

set\_position(position)

Jumps the position of playback of the timeline. Set the position as a float or an integer with range, e.g. 0.1 or Variant(10, 100) would set the position to 10% of the timeline length.

For example:

```

-- set the position of timeline 1 to 50% of timeline length
get_timeline(1):set_position(0.5)
-- set the position of timeline 2 to 20% of timeline length
get_timeline(2):set_position(Variant(2,10))

```

### set\_default\_source

Set the time source for the timeline to the default.

For example:

```

get_timeline(1):set_default_source()

```

### set\_timecode\_source

set\_timecode\_source(timecodeBus[, offset])

Set a timecode source for the timeline according to the parameters:

Parameter	Value Type	Description	Value Example
timecodeBus	integer	Integer value of constants: TCODE_1 ... TCODE_6	TCODE_1
offset	integer	Optional offset to apply to the timecode, in milliseconds	1000

### set\_audio\_source

```
set_audio_source(audioBus, band, channel[, peak])
```

Set a audio band as the time source for the timeline according to the parameters:

Parameter	Value Type	Description	Value Example
audioBus	integer	Integer value of constants: AUDIO_1 ... AUDIO_4	AUDIO_1
band	integer	The audio band to sample (number of bands depends on audio source configuration; 0 => volume)	0
channel	integer	Integer value of constants: LEFT, RIGHT or COMBINED	LEFT
peak	boolean	Optional. Whether to use the peak levels from the audio band as the time source input (default false)	false

## 7.18 Universe

A Universe object is returned from e.g. `get_dmx_universe`.

### 7.18.1 Member functions

The following are member functions of Universe objects.

#### get\_channel\_value

```
get_channel_value(channel)
```

Gets the current level of a channel in the universe, where `channel` is the integer channel number (1-512).

For example:

```
uni = get_dmx_universe(1) -- get DMX Universe 1
level = uni:get_channel_value(1) -- get channel 1 from the returned universe
```

#### park

```
park(channel, value)
```

Parks an output channel at a given value according to the parameters:

Parameter	Value Type	Description	Value Example
channel	integer (1-512)	Number of the output channel	1
value	integer (0-255)	Level to set the channel to	128

For example:

```
-- Park channel 4 of DMX universe 1 at 128 (50%)
get_dmx_universe(1):park(4,128)
```

## unpark

unpark(channel)

Clears the parked value on an output channel, where `channel` is the integer channel number (1-512).

For example:

```
-- Unpark channel 4 of DMX universe 1
-- (it will go back to normal output levels)
get_dmx_universe(1):unpark(4)
```

## 7.19 Variant

### 7.19.1 Introduction

Within Lua Scripting (as with other scripting languages) it is possible to store data within a named location (variable).

Lua typically doesn't differentiate between the contents of a variable (unlike some programming languages) and the type (integer, string, boolean) of the variable can change at any time.

Mosaic has added an object to the scripting environment called a **Variant**, which can be used to contain the data with an assignment as to the type of data that is contained. This means that a single **Variant** can be utilised and handled differently depending on the data that is contained and how it is being used.

### 7.19.2 Definition

#### Properties

A **Variant** object has the following properties:

Property	Description
<code>integer</code>	Get or set an integer data type
<code>range</code>	Get or set the range of an integer data type
<code>real</code>	Get or set a real data type (number with decimal point)
<code>string</code>	Get or set a string data type
<code>ip_address</code>	Get or set an IP address data type

## Member functions

### Constructor

`Variant()`

Create new variant.

### `is_integer`

Returns `true` or `false` to show whether the stored data has an integer representation.

### `is_string`

Returns `true` or `false` to show whether the stored data has a string representation.

### `is_ip_address`

Returns `true` or `false` to show whether the stored data has an IP address representation.

## 7.19.3 Usage

`Variant(value, range)`

### Defining a variant

Within your Lua script you can create a `Variant` with the following syntax:

```
var = Variant() -- where var is the name of the variant.
```

### Variant types

#### Integer

An integer variant can be used to store a whole number:

```
var = Variant() -- where var is the name of the variant
var.integer = 123 -- set var to an integer value of 123
log(var.integer) -- get the integer value stored in var
log(var.real) -- get the integer value stored in var and convert it to a float
log(var.string) -- get the integer value stored in var and convert it to a string
```

As shown in the example code, above, the `integer` property of a `Variant` can be used to either get or set the value of the `Variant` as an integer (whole number).

```
var:is_integer() -- returns a boolean if the variant contains an integer
```

## Range

An integer can be stored with an optional range parameter:

```
var = Variant() -- where var is the name of the variant
var.integer = 123 -- set var to an integer value of 123
var.range = 255 -- set the range of var to be 255
```

This can be used to calculate fractions and/or to define that a **Variant** is a 0-1, 0-100 or 0-255 value.

The range of a **Variant** should be set if you intend to use the **Variant** to set an intensity or colour value.

Some captured variables have a range attribute, and this is indicated in the log like this:

```
Trigger 7 (Ethernet Input): Captured 3 variables
Captured variables
  1 - Integer: 100 of 255
```

## Real

A real **Variant** can be used to store a floating point (decimal) number.

```
var = Variant() -- where var is the name of the variant.
var.real = 12.3 -- set var to an integer value of 12.3
log(var.real) -- get the integer value stored in var
```

As shown in the example code, above, the **real** property of a **Variant** can be used to either get or set the value of the **Variant** as a real number.

## String

A string **Variant** can be used to store a string of ASCII characters.

```
var = Variant() -- where var is the name of the variant
var.string = "example" -- set var to a string value of "example"
log(var.string) -- get the string value stored in var
```

As shown in the example code, above, the **string** property of a **Variant** can be used to either get or set the value of the **Variant** as a string.

```
var:is_string() -- returns a boolean if the variant contains a string
```

## IP address

```
var = Variant() -- where var is the name of the variant  
  
var.ip_address = "192.168.1.23" -- set var to the IP Address 192.168.1.23 or -1062731497  
  
log(var) -- get the stored data ("192.168.1.23")  
  
log(var.ip_address) -- get the stored IP Address (-1062731497)  
  
log(var.string) -- get the stored IP Address and convert it to a string ("192.168.1.23")  
  
log(var.integer) -- get the stored IP Address and convert it to an integer (-1062731497)
```

As shown in the example code, above, the `ip_address` property of a `Variant` can be used to either get or set the value of the `Variant` as an IP Address.

As a setter, you can pass a dotted decimal string (e.g. “192.168.1.23” or the integer representation -1062731497).

```
var:is_ip_address() -- returns a boolean if the variant contains a IP Address
```

## Shorthand

A `Variant` can also be defined using a shorthand:

```
var = Variant(128,255) -- create variable var as an integer (128) with range 0-255  
  
var = Variant(128) -- create variable var as a real number (128.0)  
  
var = Variant(12.3) -- create variable var as a real number (12.3)  
  
var = Variant("text") -- create variable var as a string ("text")
```

---

**Note:** There isn't a shorthand for IP Addresses.

---

### 7.19.4 Default variants

Some script functions return a `Variant`, including *get\_trigger\_variable*. For example:

```
get_trigger_variable(1).integer
```

The `master_intensity_level` properties of *Group* and *Content Target* are both `Variants`:

```
get_group(1).master_intensity_level.integer  
  
get_group(1).master_intensity_level.range  
  
get_content_target(1).master_intensity_level.integer  
  
get_content_target(1).master_intensity_level.range
```



## 7.20 Functions

The following functions are available in trigger action scripts and in IO modules. In trigger action scripts they are added directly to the environment; in IO modules they are available in the `controller` namespace.

### 7.20.1 Queries

#### `get_current_project`

Returns a *Project* object.

For example:

```
project_name = get_current_project().name
```

#### `get_current_replication`

Returns a *Replication* object.

For example:

```
rep_name = get_current_replication().name
```

#### `get_location`

Returns a *Location* object.

For example:

```
lat = get_location().lat
```

#### `get_timeline`

`get_timeline(timelineNum)`

Returns a single *Timeline* object for the timeline with user number `timelineNum`.

For example:

```
tl = get_timeline(1)
name = tl.name
state = tl.state

if (tl.source_bus == TCODE_1) then
  -- do something
end
```

### get\_scene

get\_scene(sceneNum)

Returns a single *Scene* object for the scene with user number sceneNum.

For example:

```
scn = get_scene(1)
name = scn.name
state = scn.state
```

### get\_group

get\_group(groupNum)

Returns a single *Group* object for the group with user number groupNum.

For example:

```
grp = get_group(1)
name = grp.name
```

---

**Note:** Passing 0 as groupNum will return *Group* for the *All Fixtures* group. This can also be used on Atlas family projects to master the intensity of the entire unit.

---

### get\_fixture\_override

get\_fixture\_override(fixtureNum)

Returns an *Override* object for the fixture with user number fixtureNum.

For example:

```
-- Get override for fixture 22
override = get_fixture_override(22)
-- Set the override colour to red (and full intensity)
override:set_irgb(255, 255, 0, 0)
```

### get\_group\_override

get\_group\_override(groupNum)

Returns an *Override* object for the group with user number groupNum.

---

**Note:** Passing 0 as groupNum will return an *Override* for the *All Fixtures* group.

---

For example:

```
-- Get override for group 3
override = get_group_override(3)
-- Set the intensity to 50% in 2 seconds
override:set_intensity(128, 2.0)
```

### get\_current\_controller

Returns the *Controller* that the script is being executed on.

For example:

```
cont = get_current_controller()
name = cont.name
```

### get\_network\_primary

Returns the *Controller* in the project that is set as the *network primary*.

### is\_controller\_online

is\_controller\_online(controllerNum)

Returns true if the controller with user number controllerNum has been discovered, or false otherwise.

For example:

```
if (is_controller_online(2)) then
    log("Controller 2 is online")
else
    log("Controller 2 is offline")
end
```

### get\_temperature

Returns a *Temperature* object with measurements from the controller's temperature sensors.

For example:

```
temp = get_temperature()
log(temp.ambient_temp)
```

### get\_rio

get\_rio(type, num)

Returns a *RIO* object representing a RIO matching the parameters:

- type can be one of the constants RIO80, RIO44 or RIO80.
- num is the remote device number within the Designer project.

For example:

```
rio = get_rio(RI044, 1)
input = rio:get_input(1)
output_state = rio:get_output(1)
```

---

**Note:** The constants for type are in the `controller` namespace within IO modules, e.g. `controller.RI044`.

---

### get\_bps

`get_bps(num)`

Returns a *BPS* object with remote device number `num`.

For example:

```
bps = get_bps(1)
btn = bps:get_state(1)
```

### get\_text\_slot

`get_text_slot(slotName)`

Returns the value of the text slot with name `slotName`. If no such text slot exists in the project then an empty string will be returned.

For example:

```
log(get_text_slot("my text slot"))
```

### get\_dmx\_universe

`get_dmx_universe(idx)`

Returns a *Universe* object for the DMX universe with number `idx`.

For example:

```
uni = get_dmx_universe(1) -- get DMX Universe 1
level = uni:get_channel_value(1) -- get channel 1 from the returned universe
```

### get\_artnet\_universe

`get_artnet_universe(idx)`

Returns a *Universe* object for the Art-Net universe with number `idx`.

### get\_pathport\_universe

get\_pathport\_universe(idx)

Returns a *Universe* object for the Pathport universe with number idx.

### get\_sacn\_universe

get\_sacn\_universe(idx)

Returns a *Universe* object for the sACN universe with number idx.

### get\_kinet\_universe

get\_kinet\_universe(power\_supply\_num, port\_num)

Returns a *Universe* object for the KiNET power supply port matching the parameters:

- power\_supply\_num is the KiNET power supply number in the project.
- port\_num is the port number of the KiNET power supply.

### get\_edn\_universe

get\_edn\_universe(remote\_device\_type, remote\_device\_num, port\_num)

Returns a *Universe* object for the EDN output matching the parameter:

- remote\_device\_type is be one of the constants EDN10 or EDN 20.
- remote\_device\_num is the remote device number of the EDN in the project.
- port\_num is the DMX output port number of the EDN.

---

**Note:** The constants for remote\_device\_type are in the controller namespace within IO modules, e.g. controller.EDN20.

---

### get\_input

get\_input(idx)

Returns the state of the controller's input numbered idx as a boolean (for digital inputs) or an integer (for analog inputs).

For example:

```
in1 = get_input(1)

if in1 == true then
  log("Input 1 is digital and high")
elseif in1 == false then
  log("Input 1 is digital and low")
else
  log("Input 1 is analog at " .. in1)
end
```

### **get\_dmx\_input**

`get_dmx_input(channel)`

Returns the value of the DMX channel number as an integer. If no DXM input is detected then `nil` will be returned.

### **get\_trigger\_variable**

`get_trigger_variable(idx)`

Returns the trigger variable at index `idx` as a *Variant*.

For example:

```
-- Use with a Touch Colour Move Trigger
red = get_trigger_variable(1).integer
green = get_trigger_variable(2).integer
blue = get_trigger_variable(3).integer

-- Use with Serial Input "<s>\r\n"
input = get_trigger_variable(1).string
```

### **get\_trigger\_number**

`get_trigger_number()`

Returns the number of the trigger that ran this script. Will return `nil` if called from another context.

### **get\_resource\_path**

`get_resource_path(filename)`

Returns the path to the resource file, where `filename` is the name of a file on the controller's internal storage.

For example:

```
dofile(get_resource_path("my_lua_file.lua"))
```

### **get\_content\_target**

---

**Note:** Only supported on Atlas and Atlas Pro.

---

On a Atlas: `get_content_target(compositionNum)`

On a Atlas Pro: `get_content_target(compositionNum, type)`

Returns a *Content Target* object representing the Content Target in the project that matches the parameters:

- `compositionNum` is the user number of the composition containing the desired Content Target.
- `type` describes the Content Target type and can be one of the constants `PRIMARY`, `SECONDARY` or `TARGET_3 ... TARGET_8`.

---

**Note:** The constants for `type` are in the `controller` namespace within IO modules, e.g. `controller.TARGET_5`.

---

Will return `nil` if no matching Content Target exists in the project.

For example, on a Atlas:

```
target = get_content_target(1)
current_level = target.master_intensity_level
```

And on a Atlas Pro:

```
target = get_content_target(1, PRIMARY)
current_angle = target.rotation_offset
```

## get\_adjustment

---

**Note:** Only supported on Atlas Pro.

---

`get_adjustment(num)`

Returns an *Adjustment Target* object representing the Adjustment Target in the project with the integer user number `num`:

Will return `nil` if no matching Adjustment Target exists in the project.

For example:

```
target = get_adjustment(1)
target:transition_x_position(10,1,5) -- Move 10 pixels right in 5 seconds
target:transition_y_position(10,1,5) -- Move 10 pixels down in 5 seconds
target:transition_rotation(90,1,5) -- Rotate by 90 degrees in 5 seconds
```

## get\_log\_level

Returns the current log level of the controller, which can be one of the following constants:

- `LOG_DEBUG`
- `LOG_TERSE`
- `LOG_NORMAL`
- `LOG_EXTENDED`
- `LOG_VERBOSE`
- `LOG_CRITICAL`

---

**Note:** These constants are in the `controller` namespace within IO modules, e.g. `controller.LOG_NORMAL`.

---

**get\_syslog\_enabled**

Returns true if Syslog is enabled, or false otherwise.

**get\_syslog\_ip\_address**

Returns the IP address of the Syslog server as a string.

**get\_ntp\_enabled**

Returns true if NTP is enabled.

**get\_ntp\_ip\_address**

Returns the IP address of the NTP server as a string.

## 7.20.2 Actions

**log**

`log([level, ]message)`

Write a message to the controller's log according to the parameters:

Parameter	Value Type	Description	Value Example
level	Integer value of constants: LOG_DEBUG, LOG_TERSE, LOG_NORMAL, LOG_EXTENDED, LOG_VERBOSE, LOG_CRITICAL; defaults to LOG_NORMAL	Optional. The log level to apply to the message.	LOG_VERBOSE
message	string	The message to add to the log.	"Your log message"

For example:

```
log(LOG_CRITICAL, "This is a critical message!") -- logs a message at Critical log level
log("This is a normal message.") -- logs a message at Normal log level.
```



## set\_log\_level

`set_log_level(log_level)`

Changes the log level of the controller, showing more or less detailed information, where `log_level` is an integer value of the constants:

- LOG\_DEBUG (5)
- LOG\_TERSE (4)
- LOG\_NORMAL (3)
- LOG\_EXTENDED (2)
- LOG\_VERBOSE (1)
- LOG\_CRITICAL (0)

## pause\_all

Pause all timelines in the project.

## resume\_all

Resume all timelines in the project.

## release\_all

`release_all([fade,] [group])`

Release all timelines and scenes in the project.

Parameter	Value Type	Description	Value Example
fade	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
group	string	Optional. Group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

## release\_all\_timelines

`release_all_timelines([fade,] [group])`

Release all timelines in the project.

Parameter	Value Type	Description	Value Example
fade	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
group	string	Optional. Group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

### release\_all\_scenes

```
release_all_scenes([fade,] [group])
```

Release all scenes in the project.

Parameter	Value Type	Description	Value Example
fade	float	Optional. Release fade time in seconds. If not provided, the default fade time will be used.	2.0
group	string	Optional. Group name: A, B, C or D. Prepend the group name with ! to apply the action to all groups <i>except</i> the specified group, e.g. !A.	"B"

### clear\_all\_overrides

```
clear_all_overrides([fade])
```

Removes all overrides from all fixtures and groups. Optionally specify a fade time in seconds as a float, e.g. 2.0.

### enqueue\_trigger

```
enqueue_trigger(num[,var...])
```

Queue trigger number `num` to be fired on the next controller playback refresh. The trigger's conditions will be tested. Optional variables `var` can be passed in as additional arguments.

For example:

```
-- enqueue trigger 2, passing in three variables: 255, 4.0 and "string"
enqueue_trigger(2,255,4.0,"string")
```

### enqueue\_local\_trigger

```
enqueue_local_trigger(num[,var...])
```

Same behaviour as for *enqueue\_trigger* but the trigger `num` will only be queued on the controller that ran the function - the trigger will not propagate to other controllers in the project.

### force\_trigger

```
force_trigger(num[,var...])
```

Queue trigger number `num` to be fired on the next controller playback refresh without testing the trigger's conditions - the trigger actions will always run. Optional variables `var` can be passed in as additional arguments.

For example:

```
-- force the execution of trigger 2's actions
-- pass in three variables: 255, 4.0 and "string"
force_trigger(2,255,4.0,"string")
```

### force\_local\_trigger

force\_local\_trigger(num[,var...])

Same behaviour as for *force\_trigger* but the trigger num will only be queued on the controller that ran the function - the trigger will not propagate to other controllers in the project.

### set\_text\_slot

set\_text\_slot(name, value)

Set the value of the text slot named name in the project to value, for example:

```
-- Set "My slot" to value "Hello world!"
set_text_slot("My slot", "Hello world!")
```

### set\_control\_value

set\_control\_value(name, [index,] value[, emitChange])

Set the value on a Touch Slider or Colour Picker according to the parameters:

Parameter	Value Type	Description	Value Example
name	string	The Key of the Touch Control.	slider001
index	integer (1-3)	Optional. Axis of movement - a slider has 1; a colour picker has 3. Will default to 1 if this parameter isn't specified.	1
value	integer (0-255)	New value to set.	128
emitChange	boolean	Optional. Whether to fire associated triggers as a result of the control value change. Defaults to false.	true

For example:

```
-- Set slider001 to half (and don't fire any associated triggers)
set_control_value("slider001", 128)
-- Set the second axis (green) to full on colour020
set_control_value("colour020", 2, 255)
```

### set\_control\_state

set\_control\_state(name, state)

Set the state on a Touch control according to the parameters:

Parameter	Value Type	Description	Value Example
name	string	The Key of the Touch Control.	slider001
state	string	The name of the state as defined in the Touch theme.	Green

For example:

```
-- Set slider001 to a state called "Green"
set_control_state("slider001", "Green")
```

### **set\_control\_caption**

set\_control\_caption(name, caption)

Set the caption on a Touch control according to the parameters:

Parameter	Value Type	Description	Value Example
name	string	The Key of the Touch Control.	button001
caption	string	The text to display as the control's caption.	On

For example:

```
-- Set button001's caption to "On"
set_control_caption("button001", "On")
```

### **set\_interface\_page**

set\_interface\_page(number[, transition])

Change the current page on the Touch interface according to the parameters:

Parameter	Value Type	Description	Value Example
number	integer	Touch interface page to change to.	2
transition	integer	Optional page transition. Integer value of constants: SNAP, PAN_LEFT, PAN_RIGHT	PAN_LEFT

---

**Note:** Must be executed on the MTPC that hosts the interface.

---

For example:

```
-- Change the touch screen interface to page 4 with a snap transition
set_interface_page(4, SNAP)
```

### **set\_interface\_enabled**

set\_interface\_enabled([enabled])

Enable/disable the touchscreen, according to the optional boolean parameter `enabled` (default: `true`).

---

**Note:** Must be executed on the MTPC that hosts the interface.

---

For example:

```
-- Disable the touchscreen
set_interface_enabled(false)
```

### set\_interface\_locked

set\_interface\_locked([lock])

Lock/unlock the touchscreen, according to the optional boolean parameter lock (default: true).

**Note:** Must be executed on the MTPC that hosts the interface.

For example:

```
-- Lock the touchscreen
set_interface_locked()
-- Unlock the touchscreen
set_interface_locked(false)
```

### push\_to\_web

push\_to\_web(name, value)

Sends data as JSON to clients who are subscribed to the relevant websocket channel, e.g. custom web interfaces using *subscribe\_lua* in the `query.js` library. The parameters are as follows:

Parameter	Value Type	Description	Value Example
name	string	JSON attribute name	"myVar"
value	<i>Variant</i>	Value for the JSON, which will be sent as a string.	"String value" or 1234

For example:

```
myData = 1234
-- Will push JSON object {"my_data": "1234"}
push_to_web("my_data", myData)
```

### disable\_output

disable\_output(protocol)

Disables the output of a single protocol from the controller, where `protocol` is the integer value of the constants:

- DMX
- PATHPORT
- ARTNET
- KINET
- SACN
- DVI
- RIO\_DMX

For example:

```
-- Disable the DMX output from the controller
disable_output(DMX)
```

### **enable\_output**

`enable_output(protocol)`

Enables the output of a single protocol from the controller, where `protocol` is the integer value of the constants defined for *disable\_output*.

For example:

```
-- Enable the DMX output from the controller
enable_output(DMX)
```

### **set\_timecode\_bus\_enabled**

`set_timecode_bus_enabled(bus[, enable])`

Enable or disable a timecode bus, where `bus` is the integer value of the constants `TCODE_1` ... `TCODE_6` and `enable` is a boolean, determining whether the bus is enabled (default `true`) or not.