tvONE REST API

Document version:	1.1.0
Firmware version:	M406 and above
Supported products:	C3-540, C3-510 and C3-503

Table of Contents

REST API

Getting started

Base Request URI

The format of the base request URI is:

```
http(s)://{device_ip}/api/{api_version}/{resource_path}
```

- `{device_ip}` is the IP address of the device
- `{api_version}` is the version of the API you wish to use
  - In this format: v{x} (where x is the number)
  - For example: v1
- `{resource_path}` is the path to the resource you wish to interact with
  - For example: .../Windows/Window1

An example of a fully formatted URI is: `https://192.168.0.1/api/v1/Slots/Slot1`

HTTP version

The only supported HTML version is HTTP 1.1

HTTP request message header

- Required: Specify a HTTP Method. The methods supported by the API are GET, POST and PUT
- Required: Specify the Authorisation header to support the basic authentication required by the service
- POST and PUT must specify the media type `Content-type: application/json`

HTTP response message header

- Required: HTTP status codes include 2xx success codes and the 4xx and 5xx error codes. API will document which status codes are returned for each request
- Media type will always be defined as `Content-type: application/json`

HTTPS session management

When using HTTPS connections, it is important to allow sessions in your client code.
This will significantly speed up communication with the device

HTTP authentication

The API uses HTTP Basic Authentication. This is sent to the device using the Authorization HTTP header.

For example, `Authorization: Basic YWRtaW46dGVzdA==`

GET method

This method is used for all requests where data is to be returned to the caller

Example

Get a Window:

Request:

```
GET https://192.168.0.1/api/v1/routing/windows/Window1
```

Response:
Status code: 200
Content-type: application/json

```
{
	Alias: Window1
	Fullname: Window1
	Input: Slot1.In1
	…
}
```

Response status codes

See the HTTP response status codes for details

PUT method

This method is used for all cases where data is being sent to the device. As all API resources are already existing on the device (and may just be empty) you are always updating an existing resource using a PUT method.

Example

Update a Window

Request:

```
PUT https://192.168.0.1/api/v1/routing/windows/Window1
```

Request body:
Content-type: application/json

```
{
      Alias: NewName
}
```

Response:
Status code: 200
Body: empty

Response status codes

See the HTTP response status codes for details

POST method

This method is used only when you are calling an RPC command. This is not strictly REST but there are some commands we use that do not fit into the REST resource definition.

Example

Play a play queue

Request:

```
POST https://192.168.0.1/api/v1/Slots/Slot1/In1/ActiveQueue/Play
```

Request body:
Empty

Response:
Status code: 200

Response status codes

See the HTTP response status codes for details

HTTP Response status codes

Response codes

Every request to the device will return with one of the responses status codes listed below.

Below is a list of the HTTP response codes

Response status code	Description
200 (OK)	Request was successful
400 (Bad Request)	Parameter and/or body is missing, not formatted correctly or is invalid in some way (e.g. string too long, or number out of range). For PUT, and incorrect JSON body will cause this error (e.g. malformed properties)
401 (Unauthorized)	The user credentials supplied are not correct or the Basic Auth header is not sent
404 (Not Found)	The request path does not exist.
405 (Method Not Allowed)	The method is not allowed for the path. For example, using a POST or PUT on a command that only supports GET
413 (Payload too large)	Message-body is larger than the allowed buffer, which is 16KB. PUT only
500 (Internal Server Error)	The server failed to execute the request in some way. Defined by a message-body shorter than what is given by the Content-Length header
501 (Not Implemented)	The request method has not been implemented. I.e. it is not one of GET, HEAD, POST, PUT
503 (Service Unavailable)	Insufficient resources (e.g. concurrent connection limit reached)
505 (HTTP version no supported)	HTTP-version is not "HTTP/1.1"

Most error response codes do not return any additional data as they as self-explanatory, however the 401 Bad Resquest error always returns additional data that includes an error code and a message.

HTTP 400 (Bad Request) codes and messages:

List of all error codes and messages for the 401 Bad Request response

Error code	Error message	Description
105	Identifier too long	The string you are trying to set is not within the limits specified for that property. For example, an Alias that is more than 19 characters
103	Syntax Error	General syntax error mostly likely caused by setting an invalid numeric value. For example, setting IP_Address to “fred” where an IP address is expected.
113	Unrecognised Field name	The property you are trying to set does not exist. For example, Window.Name Note: This only occurs when using the PUT method. When using the GET method and a property cant be found, you will get a 404 Not Found response
121	Field is read only	The property is read only and cannot be set
122	Value given is above allowable range	The number you are trying to set is not within the valid range for that property. For example, a Brightness value of 200
123	Value given is above allowable range	The number you are trying to set is not within the valid range for that property. For example, a SCurver value of 0.1
124	Value given is not supported	Setting a string value to unsupported value. For example, setting “true” instead of “Yes” for a YesNo property
125	Error writing to field	Unexpected value (or null) or internal error when setting field. For example, setting string when a specific type is expected (Output DefaultLoRes = “fred”)
127	This name is already in use	Setting an Alias to a name that is already in use
128	Unrecognised Object name	The value specified is not the correct object type. For example, setting Storyboard Canvas property to an input (Slot1.In1)
152	The system is in standby, resume from standby and try again	Only certain commands are allowed when CORIOview is in standby. to run

		all commands you need to leave standby mode.
158	FP Custom Preset Editing in progress	Occurs when you try and save settings using System\SaveAllSettings or System\SaveLiveConfig while you are editing presets from the front panel

HTTP 400 (Bad Request) Invalid characters:

An invalid character is considered to be any character which is not an unreserved character (per RFC 3986 Section 2.3.).

These unreserved characters are:
ALPHA / DIGIT / "-" / "." / "_" / "~" or the reserved character forward slash "/".

HTTP Content-length mismatch

If the content length specified by the HTTP header Content-length does not match the actual size of the message body, your request will fail.

Content-length is larger than body	Content-length is smaller than body
The server will wait for all the data to arrive but because the content-length is larger, there is not enough data. In this case the request will "hang" until cancelled. At which point you will get a HTTP 500 error response	The server will read up to the Content-length. The remainder of the data will then be interpreted as a new packet which will cause a failure on the server. The most likely HTTP error is the 400 (Bad Request) as the data will be "malformed". But there is a possibility for any other error to be returned in this case