How to use the AquaControl REST API

The AquaControl REST API is available on mXa and AQM devices and is built using Swagger.

Some terminology

- API (Application Programming Interface)
- A *resource* is some abstract data that's accessible through the API. It is usually uniquely identified by a path, for example: /system/info.
- URL (Uniform Resource Locator) is an address of a resource
- HTTP methods (or verbs) indicate the actions that can be performed on the resources.

HTTP Methods (verbs)

This API uses 3 of the HTTP methods:

- GET: Use GET when you want to retrieve a resource.
- POST: Use POST when you want to create or update a resource.
- DELETE: Use DELETE when you want to delete a resource.

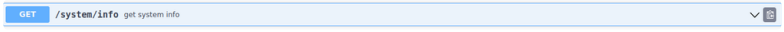
How to interact with the API

The first option: the Swagger web interface

The API documentation is provided by the Swagger web interface. Open it in a browser and look at it:

http://<device_IP_address>:8000/documentation.

Replace <device_IP_address> with the host name or IP address of your device (for example: 192.168.11.12)

The Swagger page lists all the resources that are accessible via the API. They are listed in pairs: (METHOD PATH), for example: (GET /system/info) 

The POST and DELETE methods always require additional parameters. Sometimes the GET method requires additional parameters, for example GET /preset/{id} requires the identifier of the preset we want to retrieve.

Click on the down pointing arrow to the right of each resource to see if additional parameters are required, and what the expected parameters are.

So, for each resource you're interested in, use the Swagger interface to detemine:

- the resource path
- the available methods
- the parameters required by each method-resource pair.

The second option: Python

This section will show how to use the 3 methods (GET, POST, DELETE) on some resources.

Before we begin, you'll need a Python interpreter with the "requests" module installed. I assume that you know how to install Python and the "requests" module on your computer. (If not, maybe I'll cover that later).

You'll also need to know:

- the host name or IP address of the mXa/AQM device you're working with
- the user name and password of a user on that device

First, log into the API and get a session key:

```
import requests

protocol = "http"
target_host_name = "192.168.11.12"
port = 8000
api_version = "v1.0-beta"
user_name = "admin"
password = "open_sesame"

api_url_prefix = f"{protocol}://{target_host_name}:{port}/{api_version}/"

# Create a session, which will store your authentication cookie
session = requests.session()

# You need to log in before you can change things on the device
login_url = api_url_prefix + "session/login"
user_credentials = {"username": user_name, "password": password, "keepLoggedIn": True}
response = session.post(login_url, json=user_credentials)

print(response.status_code)
print(response.json())
print(session.cookies.get_dict())
```

After you run the code above, I expect that you see: "Status code: 200".

If you got a 200, that's great.

If you got a status code different than 200, then look at the next line, (the HTTP response) and try to figure out from the response what went wrong.

The code above also prints the session cookie. You don't need it for anything, at least not in the following examples. You may want to save it, and use it later to avoid logging in again.

Now, let's see how the GET method works. (You need to run the following code in the same process you ran the code above (you need that session).

```
# Use GET to request the state of a resource
# Get a list of general purpose digital inputs
print("Getting GPI list:")

response = session.get(api_url_prefix + "generalPurposeInput")

print(f"Status code: {response.status_code}")
print(f'Response as JSON: "{response.json()}"')
print(f'Response as bytes: "{response.content}"')
```

So, GET is pretty simple, just append a path to the URL prefix, and call session.get().

After running this I expect the status code to be 200 as well.

Next, let's change the state of a GPIO by using the POST method:

```
# Use POST to change or update the state of a resource
# Change the state of a general purpose digital putput pin from low to high
print("Setting GPO1 high")

response = session.post(api_url_prefix +
"workingsettings/generalPurposeOutputConfiguration/General%20Purpose%20Output%20Pin.1",
                        json={"value": "high"})

print(f"Status code: {response.status_code}")
print(f"Response: {response.json()}")
```

Like with GET, build a URL by appending the correct path to the api_url_prefix string.

Then, call session.post(). This time, you'll need to pass additional parameters to the post call in a "json" parameter.

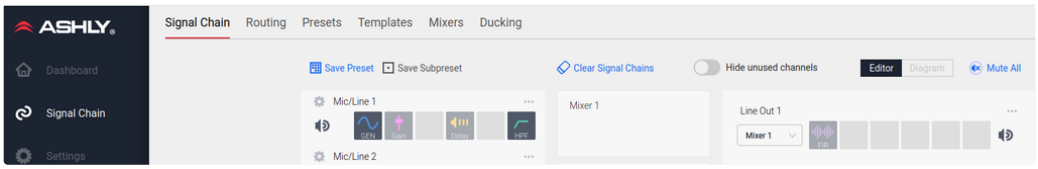
Look at the Swagger web interface to find out what parameters you need for each resource.

Again, a status code of 200 means success.

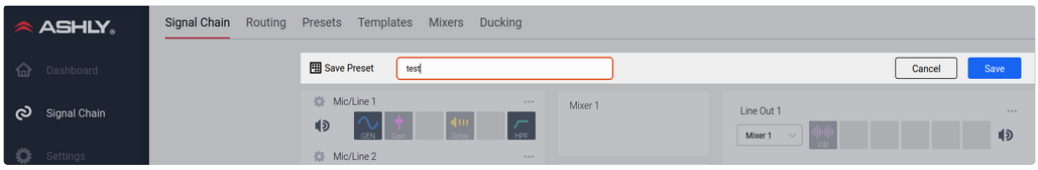
And the last thing for now, let's try to delete something. A preset, for example.

But before we can successfully delete a preset, we need to create one.

Go into the AquaControl UI, click "Signal Chain" on the left bar, then click "Save Preset":



Name it "test", then click the "Save" button.



Now, back to the Python interpreter:

```
# Use DELETE to remove a resource
# This will try to remove a preset named "test"
preset_name = "test"

print(f"Removing preset: {preset_name}")

response = session.delete(api_url_prefix + f"preset/{preset_name}")

print(f"Status code: {response.status_code}")
print(f"Response: {response.json()}")
```

Execute the code above, and look for a 200 status code:

```
Removing preset: test
Status code: 200
Response: {'success': True, 'data': [{'name': 'test', 'type': 'Preset'}]}
```

Now, run the delete code again. Since the preset does not exist anymore, it should return an error:

```
Removing preset: test
Status code: 422
Response: {'statusCode': 422, 'error': 'Unprocessable Entity', 'message': 'Error, Could not a find a Preset with Name: test'}
```

OK, this is the end of today's lesson.

I hope you're a little more comfortable with Swagger and Python requests.