Switcher Webapi

Release 0.2.7

Tomer Figenblat

Aug 28, 2019

CONTENTS

1 What 1

2 Prerequisites 3

3 Notes 5

4 Install 7

5 Usage 9

6 Credits 15

7 License 17

8 Change Log 19

9 Code of Conduct 21

10 Contributing 23

11 Code Documentation 33

Python Module Index 43

Index 45

CHAPTER

ONE

WHAT

An asynchronous sanic webapp running inside a python docker image using uvloop as the event loop. Used as a rest api wrapper for aioswitcher.

If you're using the Switcher water heater and you want to wrap a rest api around it... you came to the right place!

CHAPTER

TWO

PREREQUISITES

- Install and configure your Switcher device.
- **Collect the following information from the device's following NightRang3r instructions in the Switcher-V2-Python reposit**

  ip_address
  phone_id
  device_id
  device_pass
- Install docker

CHAPTER

THREE

NOTES

- If you don't want to be forced to restart the container if the device's ip address changes, please consider assigning the device with a static ip address.
- The Switcher-V2-Python repository is build with python 2.7.
- The aioswitcher was tested with the Switcher V2 device by myself and with the Switcher Touch device by the community.
- This project was intended for local usage, it's ok if you want to use it remotely, just make sure to take the proper security measures such as reverse proxy and ssl.
- The WebAPI has a throttle mechanism to prevent overfloating the device with frequent requests, it defaults to 5 seconds throttle time.
- Some users have been reporting lately about failures using the Switcher-V2-Python script after upgrading the device firmware to 3.0, please follow the relevant issues in the script repository before doing the same.

CHAPTER

FOUR

INSTALL

```
docker run -d -p 8000:8000 \
-e CONF_DEVICE_IP_ADDR=192.168.100.157 \
-e CONF_PHONE_ID=1234 \
-e CONF_DEVICE_ID=ab1c2d \
-e CONF_DEVICE_PASSWORD=12345678 \
--name switcher_webapi tomerfi/switcher_webapi:latest"
```

You can also add another optional environment variable:

```
-e CONF_THROTTLE=5.0
```

for setting the throttle time between consecutive requests, this is optional and the default value is **5.0**.

Here's an example of running the container using *docker-compose* setting the environment variables in a designated file.

```
# docker-compose.yml
version: "3.7"

services:
  switcher_api:
    image: tomerfi/switcher_webapi:latest
    container_name: "switcher_webapi"
    env_file:
      - .env_vars
    ports:
      - 8000:8000
    restart: unless-stopped
```

```
# .env_vars
CONF_DEVICE_IP_ADDR=192.168.100.157
CONF_PHONE_ID=1234
CONF_DEVICE_ID=ab1c2d
CONF_DEVICE_PASSWORD=12345678
CONF_THROTTLE=5.0
```

CHAPTER

FIVE

USAGE

Once running, you can send REST requests towards the container. With the exception of the *create_schedule* requests, all the requests requiring input accepts it as a json body or in the form of query parameters.

5.1 get_state

URL: */switcher/get_state*

Method: *GET*

Request parameters: *None*

Response body example:

```
{
  "successful": true,
  "state": "on",
  "time_left": "00:47:25",
  "auto_off": "01:30:00",
  "power_consumption": 2669,
  "electric_current": 12.3
}
```

5.2 turn_on

URL: */switcher/turn_on*

Method: *POST*

Request parameters:

Key	Required	Description
minutes	*Optional*	turn on the device with an off timer of 1-180 minutes.

Request body example:

```
{
  "minutes": "30"
}
```

Response body example:

```
{
  "successful": true
}
```

5.3 turn_off

URL: */switcher/turn_off*

Method: *POST*

Request parameters: *None*

Response body example:

```
{
  "successful": true
}
```

5.4 set_auto_shutdown

URL: */switcher/set_auto_shutdown*

Method: *POST*

Request parameters:

Key	Required	Description
hours	*Mandatory*	hours value 1-3.
minutes	*Mandatory*	minutes value 0-59.

Note: The auto shutdown configuration value accept any total value of hours and minutes between 1 and 3 hours.

Request body example:

```
{
  "hours": "1",
  "minutes": "30"
}
```

Response body example:

```
{
  "successful": true
}
```

5.5 set_device_name

URL: */switcher/set_device_name*

Method: *POST*

Request parameters:

Key	Required	Description
name	*Mandatory*	device name, accepts length of 2-32 characters.

Request body example:

```
{
  "name": "my new device name"
}
```

Response body example:

```
{
  "successful": true
}
```

5.6 get_schedules

URL: */switcher/get_schedules*

Method: *GET*

Request parameters: *None*

Response body example:

```
{
 "successful": true,
  "found_schedules": true,
  "schedules": [
    {
      "schedule_id": "0",
      "enabled": true,
      "recurring": true,
      "days": [
        "Tuesday",
        "Wednesday",
        "Thursday",
        "Friday",
        "Saturday",
        "Sunday"
      ],
      "start_time": "17:30",
      "end_time": "18:30",
      "duration": "1:00:00",
      "schedule_data": "0001fc01e871a35cf87fa35c",
      "next_run": "Due next Tuesday at 17:30"
    },
    {
      "schedule_id": "1",
      "enabled": true,
      "recurring": true,
      "days": ["Monday"],
```

(continues on next page)

(continued from previous page)


```
      "start_time": "17:00",
      "end_time": "18:00",
      "duration": "1:00:00",
      "schedule_data": "0101020160a6c95c70b4c95c",
      "next_run": "Due tommorow at 17:00"
    }
  ]
}
```


Note: The *schedules* list can contain up to 8 schedules with the identifiers of 0-7 representing the actual schedule slots on the device.

5.7 enable_schedule

URL: */switcher/enable_schedule*

Method: *PATCH*

Request parameters:

Key	Required	Description
schedule_data	*Mandatory*	the *schedule_data* associated with the chosen schedule. retrieved with */switcher/get_schedules*.

Request body example:


```
{
  "schedule_data": "0101020160a6c95c70b4c95c"
}
```


Response body example:


```
{
  "successful": true
}
```


5.8 disable_schedule

URL: */switcher/disable_schedule*

Method: *PATCH*

Request parameters:

Key	Required	Description
schedule_data	*Mandatory*	the *schedule_data* associated with the chosen schedule. retrieved with */switcher/get_schedules*.

Request body example:

```
{
  "schedule_data": "0101020160a6c95c70b4c95c"
}
```

Response body example:

```
{
  "successful": true
}
```

5.9 delete_schedule

URL: */switcher/delete_schedule*

Method: *DELETE*

Request parameters:

Key	Required	Description
schedule_id	*Mandatory*	the *schedule_id* associated with the chosen schedule. retrieved with */switcher/get_schedules*.

Request body example:

```
{
  "schedule_id": "2"
}
```

Response body example:

```
{
  "successful": true
}
```

5.10 create_schedule

URL: */switcher/create_schedule*

Method: *PUT*

Request parameters:

Key	Required	Description
days	*Mandatory*	list of days for the schedule to run in. (empty for non-recurring schedules).
start_hours	*Mandatory*	start time hours value 0-23.
start_minutes	*Mandatory*	start minutes value 0-59.
stop_hours	*Mandatory*	stop time hours value 0-23.
stop_minutes	*Mandatory*	stop minutes value 0-59.

Request body example:

```
{
  "days": ["Monday", "Wednesday", "Friday"],
  "start_hours": "20",
  "start_minutes": "30",
  "stop_hours": "21",
  "stop_minutes": "0"
}
```

Response body example:

```
{
  "successful": true
}
```

Possible values for the *days* list:

- Sunday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday

Note: Due to its complexity, the *create_schedule* request accepts its arguments in the form of a json body only, query parameters will not be accepted.

5.11 Exceptions

Unless unhandled, all exceptions will return a json object in response body:

```
{
  "successful": false,
  "message": "the error description"
}
```

CHAPTER

SIX

CREDITS

This project was enabled by creating the aioswitcher pypi module, initially created for use with the home assistant component.

Not this nor the aioswitcher project would have been able to happen without the amazing work preformed by NightRang3r and AviadGolan in the Switcher-V2-Python project.

So… Thanks!

CHAPTER

SEVEN

LICENSE

CHAPTER

EIGHT

CHANGE LOG

8.1 0.2.7 (2019-06-25)

0.2.7 Full Changelog

- Updated dependencies and requirements
- Documentation fix-ups

8.2 0.2.6 (2019-06-24)

0.2.6 Full Changelog

- Better documentation site including a code documentation

8.3 0.2.5 (2019-06-22)

0.2.5 Full Changelog

- Better testing and ci workflows
- Better documentation site including a contributing section
- Added github templates

8.4 0.2.4 (2019-06-16)

0.2.4 Full Changelog

- Added usage of the container-structure-test tool with Tox and CircleCi
- Fixed a bunch of minor fixes and typos

8.5 0.2.3 (2019-06-15)

0.2.3 Full Changelog

- Documentation finishing touches

8.6 0.2.2 (2019-06-15)

0.2.2 Full Changelog

- Added docs
- Code cleanup
- CI rearrangement

8.7 0.2.1 (2019-05-05)

0.2.1 Full Changelog

- Fixed VERSION file with the correct version value

8.8 0.2 (2019-05-05)

0.2 Full Changelog

- Initial stable release

8.9 0.01 (2019-05-05)

0.01 Full Changelog

- Pre-release

8.10 0.0.2 (2019-05-05)

0.0.2 Full Changelog

- Pre-release

8.11 0.0.1 (2019-05-05)

0.0.1 Full Changelog

- Pre-release

CHAPTER

NINE

CODE OF CONDUCT

9.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socioeconomic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

9.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

9.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

9.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

9.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at tomer.figenblat@gmail.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

9.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available here.

For answers to common questions about this code of conduct, see faq.

CHAPTER

TEN

CONTRIBUTING

First off, thank you for taking the time to contribute.

Note: This repository is a docker image wrapping a python Rest API around the aioswitcher pypi module.

If your contribution is a more of a *core contribution*, please consider maybe it belongs to the integrating module and not the wrapping docker. You can find the module repository here.

Contributing is pretty straight-forward:

- Fork the repository
- Commit your changes
- Create a pull request against the **dev** branch

Please feel free to contribute, even to this contributing guideline file, if you see fit.

TOC

- *Items description*
  - *Configuration files*
  - *Docker*
  - *Python*
  - *Shell*
  - *Ignore files*
  - *Requirement files*
  - *Package management*
  - *Documentation*
- *Continuous Integration*
  - *CircleCi*
  - *CodeCov*
  - *Codacy*
  - *Requires-io*
  - *David-DM*

    - *Snyk*
- *Continuous Deployment*
    - *Docker Hub*
    - *Read the Docs*
    - *Metadata*
- *Environments and Tools*
- *Testing*
- *Guidelines*
    - *NPM Scripts*
    - *Shell Scripts*
    - *Makefile*
- *Chat*
- *Best Practices*

10.1 Items description

10.1.1 Configuration files

- `.circle/config.yml` is the configuration file for CircleCi Continuous Integration and Deployment Services.
- `.codecov.yml` is the configuration file for CodeCov Code Coverage.
- `.coveragerc` is the configuration file for Coverage.py creating coverage reports with the pytest-cov plugin.
- `.yamllint` is the configuration for yamllint A Linter for YAML Files linting yml files.
- `.remarkrc` is the configuration file for remark-lint plugin for Remark linting md files.
- `bandit.yml` is the configuration file for Bandit common security issues finder checking python scripts.
- `container_structure.yml` is the configuration file for GoogleContainerTools container-structure-test validating the container content.
- `doc8.ini` is the configuration file for Doc8 Style Checker checking rst file types.
- `mypy.ini` is the configuration file for MyPy Static Type Checker for checking static types in python scripts.
- `package.json` is npm's package manager file for managing dependencies, scripts and etc.
- `pyproject.toml` is designated to be the main configuration file for python based on PEP518, not fully operative in this project yet.
- `.spelling` is the dictionary/ignore file used by both markdown-spellcheck and vale. Case-insensitive words in this file will not raise a spelling mistake error.
- `.vale.ini` is the configuration for vale.
- `tox.ini` is the configuration file for Tox Testing Automation testing the python code.

10.1.2 Docker

- `Dockerfile` is the instruction file for building the *docker image*.

10.1.3 Python

- `pyscripts` is where *python* scripts are stored.

10.1.4 Shell

- `shellscripts` is where *shell* scripts are stored.

10.1.5 Ignore files

- `.dockerignore` used for controlling what goes in the *docker image*.
- `.gitignore` used for controlling what will not be pushed to *github*.
- `.remarkignore` used for ignoring specific files or folders from *remark-lint*.

10.1.6 Requirement files

- `requirements.txt` is a list of python requirements constraints.
- `requirements_prod.txt` is a list of python requirements for running the solution.
- `requirements_docs.txt` is a list of python requirements for testing and building the documentation.
- `requirements_test.txt` is a list of python requirements for testing the solution.

10.1.7 Package management

The `package.json` file specified by npm manages our dependencies, scripts and some metadata.

10.1.8 Documentation

- `docs/sources` is where the *rst files* for creating the Sphinx Documentation are stored for build, deployment and hosting by Read the Docs.
- `docs/sources/vale_styles` is where the vale styles for vale linter are stored. Out of the box, vale comes with three pre-configured plugins: proselint, write-good and joblint. The `docs/sources/vale_styles/18F` and `docs/sources/vale_styles/docs` plugins were manually added from the vale repository.
- `docs/Makefile` the basic *Makefile* for Sphinx documentation generator. From the `docs` path, type `make html` and sphinx will create the documentation site locally in `docs/build`.

10.2 Continuous Integration

10.2.1 CircleCi

By hook configuration, for every pull request, CircleCi will execute the workflows described in `.circleci/config.yml` and update the PR conversation with the results.

As a final step, CircleCi will push the Coverage.py XML Report to both CodeCov for code coverage analysis and Codacy for code quality analysis. Both will of course push their results into the PR conversation.

Some of the steps are considered required and may prevent the PR from being merged. But no worries, everything is fixable.

10.2.2 CodeCov

CodeCov is keeping tabs on our code coverage. When a report is uploaded (by CircleCi), CodeCov will check our code coverage and push its conclusions to *github*.

10.2.3 Codacy

Codacy is here to check the quality of our code. When a report is uploaded (by CircleCi), Codacy will check our code quality and push its conclusions to *github*.

10.2.4 Requires-io

Requires.io is keeping an eye for versions updates upon the python requirements listed in the various `requirements files` and in `tox.ini` file.

10.2.5 David-DM

David-DM is keeping an eye for versions updates upon the npm requirements listed in the *package.json* file.

10.2.6 Snyk

Snyk is keeping an eye out for vulnerabilities in our npm dependencies, our pypi requirements and our *docker image dependencies*.

10.3 Continuous Deployment

10.3.1 Docker Hub

When a **git-tag** with the regex of `/^[0-9.]+$/` is set, Docker Hub Cloud will build the image based on the `Dockerfile` instructions file and tag it twice: - `<git-tag>` - latest

10.3.2 Read the Docs

By hook configuration, for every `git-release-tag` and push to the `dev` branch Read the Docs will build the documentation site based on `docs/source` and host it with the following tags: * stable tag is for the release snapshot. * latest tag is for the dev branch.

By hook configuration, Read the Docs will build the documentation site based on `docs/source` and host it: * stable tag will be built for every release snapshot. * latest tag will be built for every push the dev branch, so it'll reflect unreleased changes.

10.3.3 Metadata

By hook configuration, for every *docker image* build by Docker Hub, MicroBadger will receive a notification and publish the image metadata.

10.4 Environments and Tools

Note: The following (Python, virtualenv, nodeenv and Tox) needs to be pre-installed before local testing with `tox`.

- The Python scripts in `pyscripts` was written with Python 3.7 in mind, which added a few tweaks and adjustments, especially in regards to asyncio.
- Python's virtualenv, a tool for segregating Python environments.
- Python's nodeenv, a tool that enables us to create Node.js virtual environment in resemblance to virtualenv, the tool also allows combining nodeenv inside virtualenv, which is exactly what we're doing with `tox`.
- Docker, as some of the testing automation are performed within a run-once docker container.
- Tox for automating unit testing in your local environment. * Tox utilizes Python's virtualenv.
  - Tox is configured with `tox.ini`.
  - To run tox, simply execute `tox` from `tox.ini`'s path. It is recommended that you also run `tox --help` to get familiar with the various options such as `-e` and `-r` that will help you perform faster and better tests.)

Note: **Please note**: the rest of the steps require no installation on your behalf, but knowing them is important seeing they are key elements for testing with `Tox` and/or `CircleCi`.

- *NPM Package*: package-json-validator for validating the `package.json` file.
- *Python Module*: doc8 for checking restructured text (rst) files residing in `docs/source` and used to create the documentation site.
  - doc8 is configured with `doc8.ini`.
- *Docker Image*: jdkato/vale for linting restructured text files residing in `docs/source` for spelling/syntax mistakes.
  - jdkato/vale ignore file is `.spelling`.
  - jdkato/vale is configured with `.vale.ini`.
- *Python Module*: sphinx for building the documentation site from the restructured Text (rst) files residing in `docs/source`.

  - It's worth mentioning that the documentation site hosted with Read the Docs is based upon the theme sphinx-rtd-theme

- *NPM Package*: remark-lint which is a plugin for Remark and the remark-cli command line tool for linting *markdown* files residing at the base path and in `.github`.
  - remark-lint uses a couple of presets and tools, all can be found under the dependencies key in `package.json`.
  - remark-lint ignore list is the file `.remarkignore`.
  - remark-lint is configured with `.remarkrc`.
- *NPM Package*: markdown-spellcheck for checking the project *markdown* files for spelling errors.
  - markdown-spellcheck dictionary file is `.spelling`.
- *Python Package*: yamllint for linting the project yml files. * yamllint is configured with `.yamllint`.
- *Docker Image*: koalaman/shellcheck is used for checking shell script residing in `shellscripts`.
- *Docker Image*: hadolint/hadolint is used for linting the instruction file `Dockerfile`.
- *Linux Tool*: container-structure-test for verifying the docker image content.
  - The tool runs with the helper script `shellscripts/container-structure-test-verify.sh`, it will not fail if the tool is not present when running `tox` locally. But this will probably come up with CircleCi so please consider installing the tool manually.
  - container-structure-test is configured with `container_structure.yml`.
- *Python Package*: isort for sorting imports. isort runs automatically with tox in `no-fail` mode for recommendations only.
- *Python Package*: bandit for finding common security issues with against the scripts residing in `pyscripts`. * bandit is configured with `bandit.yml`.
- *Python Package*: isort for sorting python imports. * isort is configured with `pyproject.toml`.
- *Python Package*: flake8 for checking python scripts residing in `pyscripts`.
- *Python Package*: black for formatting python scripts residing in `pyscripts`. * black is configured with `pyproject.toml`.
- *Python Package*: mypy for checking static typing tests against python scripts residing in `pyscripts`. * mypy is configured with `mypy.ini`.
- *Python Package*: pytest as testing framework for running test-cases written in `pyscripts/test_server.py`. * pytest uses a bunch of awesome plugins listed in `requirements_test.txt`.
- *Docker Image*: circleci/circleci-cli for validating the `.circleci/config.yml` file.

10.5 Testing

Testing is performed with Pytest, Full-featured Python testing tool. The various Rest Http request test-cases is in `pyscripts/test_server.py`.

For automated local tests, use `tox`.

10.6 Guidelines

Note: The project's semvar is being handled in both `VERSION` file for creating the docker image with `Makefile` and in `package.json` for packaging handling.

Here are some guidelines (recommendations) for contributing to the `switcher_webapi` project: * Code docstrings documentation is here.

- If you add a python dependency, for order keeping and for Snyk's sake, Please add the dependency with the fixed version to `requirements.txt`, And add with no version statement in any or all of the other requirements file based on the dependency use case.
- If you add a new file, please consider is it should be listed within any or all of the `ignore files`.
- If you change something inside the `docker image` it is strongly recommended verifying it with the container-structure-test
- While not all the test steps in `CircleCi` and in `Tox` are parallel to each other, most of them are, so tests failing with `Tox` will probably also fail with `CircleCi`.
- If you're writing python code, please remember to static type your code or else it will probably fail `mypy` tests.
- You can run npm's script `spell-md-interactive` for handling all spelling mistakes before testing. You can also choose to run `spell-md-report` to print a full report instead of handling the spelling mistakes one-by-one. * markdown-spellcheck dictionary is the file `.spelling`.

10.6.1 NPM Scripts

Before using the scrips, you need to install the dependencies. From the `package.json` file path, run `npm install`, Then you can execute the scripts from the same path.

- `npm run lint-md` will run remark against *markdown* files.
- `npm run validate-pkg` will run package-json-validator against the `package.json` file.
- `npm run spell-md-interactive` will run markdown-spellcheck against *markdown* files in an interactive manner allowing us to select the appropriate action.
- `npm run spell-md-report` will run markdown-spellcheck against *markdown* files and print the report to stdout.

10.6.2 Shell Scripts

Note: The shell scripts in `shellscripts` were written for `bash` and not for `sh`.

- `bash shellscripts/container-structure-test-verify.sh` will verify the existence of container-structure-test and execute it. The script will `exit 0` if the tool doesn't exists so it will not fail `tox`.
- `bash shellscripts/push-docker-description.sh` allows the deployment of the local `README.md` file as a docker image description in Docker Hub. Please use it with `Makefile` as arguments are required.
- `bash shellscripts/run-once-docker-operations.sh <add-argument-here>` will verify the existence of Docker before executing various *run-once docker operations* based on the following arguments. If the script find that Docker is not installed, it will `exit 0` so it will not fail `tox`:

- **argument**: `lint-dockerfile` will execute the docker image hadolint/hadolint linting the local `Dockerfile`.
- **argument**: `check-shellscripts` will execute the docker image koalaman/shellcheck for checking the shell scripts residing in `shellscripts`.
- **argument**: `circleci-validate` will execute the docker image circleci/circleci-cli for validating the `.circleci/config.yml` file.
- argument `vale-rstdocs` will execute the docker image jdkato/vale checking for spelling or syntax mistakes in restructured text file residing in `docs/source`.

10.6.3 Makefile

Using the `Makefile` is highly recommended, especially in regards to docker operations. Try `make help` to list all the available tasks: * `make docker-build` will build image from relative `Dockerfile`.

- `make docker-build-testing-image` will build image from relative `Dockerfile` using a testing tag.
- `make docker-remove-testing-image` will remove the testing image (must be build first).
- `make docker-build-no-cache` will build image from `Dockerfile` with no caching.
- `make structure-test` will run the container-structure-test tool against the built image (must be build first) using the relative `container_structure.yml` file.
- `make docker-build-structure-test` will build the image and test the container structure.
- `make docker-build-no-cache-structure-test` will build the image and test the container structure.
- `make docker-full-structure-testing` will build the image with the testing tag and remove after structure test.
- `make docker-tag-latest` will add latest tag before pushing the latest version.
- `make docker-run` will run the built image as a container (must be built first).
- `make docker-build-and-run` will build image from `Dockerfile` and run as container.
- `make docker-build-no-cache-and-run` will build image from `Dockerfile` with no caching and run as container.
- `make push-description` will push the relative `README.md` file as full description to docker hub, requires username and password arguments.
- `make verify-environment-file` will verify the existence of the required environment variables file and its content.

10.7 Chat

Feel free to join the project's public Slack Channel. GitHub, Codacy Docker Hub and Snyk are integrated with the channel and keeping its members updated.

10.8 Best Practices

This project tries to follow the CII Best Practices guidelines. That's not an easy task and I'm not sure achieving 100% is even possible for this specific project. At the time writing this, the project has achieved 42%. (The writing of this file was actually according one to those guidelines).

Any contribution bumping up this percentage will be gladly embraced.

CHAPTER

ELEVEN

CODE DOCUMENTATION

11.1 Application run scripts

11.1.1 pyscripts/request_handlers.py

Request handlers for the Switcher WebAPI.

async request_handlers.**_create_raw_schedule_data**(*schedule_days: List[int], start_hours: int, start_minutes: int, stop_hours: int, stop_minutes: int*) → str

Use as helper creating raw schedule data for creating schedules.

Parameters

- **schedule_days** – selected days for the schedule to run in.
- **start_hours** – hour to start the device at.
- **start_minutes** – minutes to start the device at.
- **stop_hours** – hour to stop the device at.
- **stop_minutes** – minutes to stop the device at.

Returns Raw schedule data needed for creating the requested schedule.

async request_handlers.**_parse_schedule_body**(*body: Dict*) → str

Use as helper parsing body of create schedule requests.

Parameters **body** – json body of the create schedule requests.

Raises **sanic.exceptions.InvalidUsage** – when missing a mandatory argument.

Returns Schedule data object needed for creating the new schedules.

request_handlers.**_validate_day_to_int**(*day: str*) → int

Use as helper converting string weekday to int for creating schedules.

Parameters **day** – string represntation of the weekday.

Raises **sanic.exceptions.InvalidUsage** – when encounterd unknown weekday string.

Returns

The in represntation of the weekday.

More information is available in the Usage section.

request_handlers.**_validate_time_integers**(*start_hours: int*, *start_minutes: int*, *stop_hours: int*, *stop_minutes: int*) → None

Use as helper validating time arguments of creating schedule requests.

Parameters

- **start_hours** – hour to start the device at (0-23).
- **start_minutes** – minutes to start the device at (0-59).
- **stop_hours** – hour to stop the device at (0-23).
- **stop_minutes** – minutes to stop the device at (0-59).

Raises **sanic.exceptions.InvalidUsage** – when the validation failes.

async request_handlers.**create_schedule_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/create_schedule.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises **sanic.exceptions.ServerError** – when encounterd an error.

Returns

Json object represnting the request status.

More information is available in the Usage section.

Warning: Accepts json body only, no query parameters allowed.

async request_handlers.**delete_schedule_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/delete_schedule.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when encounterd unknown weekday.

- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the `Usage` section.

Note: Accepts arguments as json body or query parameters.

---

async request_handlers.**disable_schedule_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/disable_schedule.

Parameters

- **request** – `sanic`'s request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when encounterd faulty data.
- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the `Usage` section.

Note: Accepts arguments as json body or query parameters.

---

async request_handlers.**enable_schedule_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/enable_schedule.

Parameters

- **request** – `sanic`'s request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when encounterd faulty data.

- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the `Usage` section.

Note: Accepts arguments as json body or query parameters.

---

async request_handlers.**get_schedules_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/get_schedules.

Parameters

- **request** – `sanic`'s request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the configured schedules on the device.

More information is available in the `Usage` section.

Note: Accepts arguments as json body or query parameters.

---

async request_handlers.**get_state_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/get_state.

Parameters

- **request** – `sanic`'s request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the current state of the device.

More information is available in the `Usage` section.

Note: Accepts arguments as json body or query parameters.

async request_handlers.**set_auto_shutdown_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/set_auto_shutdown.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when requested is not 59-180 minutes.
- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the Usage section.

Note: Accepts arguments as json body or query parameters.

async request_handlers.**set_device_name_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/set_device_name.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when name length is no 2-32 characters.
- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the Usage section.

Note: Accepts arguments as json body or query parameters.

async request_handlers.**turn_off_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/turn_off.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the Usage section.

Note: Accepts arguments as json body or query parameters.

async request_handlers.**turn_on_handler**(*request: sanic.request.Request*, *ip_address: str*, *phone_id: str*, *device_id: str*, *device_password: str*) → sanic.response.HTTPResponse

Use for handling requests to /switcher/turn_on.

Parameters

- **request** – sanic's request object.
- **ip_address** – the local ip address.
- **phone_id** – the extracted phone id.
- **device_id** – the extracted device id.
- **device_password** – the extracted device password.

Raises

- **sanic.exceptions.InvalidUsage** – when timer is no 1-180 minutes.
- **sanic.exceptions.ServerError** – when encounterd any error.

Returns

Json object represnting the request status.

More information is available in the Usage section.

Note: Accepts arguments as json body or query parameters.

11.1.2 pyscripts/start_server.py

Sanic server for the Switcher WebAPI.

start_server.**before_start**(*app: sanic.Sanic*, *loop: asyncio.events.AbstractEventLoop*) → None
Use for preparing data and register mappings before start.

This function is annotated with `sanic.Sanic.listener("before_server_start")`.

It is called by `Sanic` just before the server starts. Its job is:

- Gather the initial data for running the server.
- Register a middleware for aquiring a throttler for all requests.
- Add routes using the `mappings module` and the `request_handlers module`.

Parameters

- **app** – the running `sanic` app.
- **loop** – the main event loop.

start_server.**timeout**(*request: sanic.request.Request*, *exception: sanic.exceptions.SanicException*) → sanic.response.HTTPResponse
Use as custom handler for logging internal service errors.

This function is annotated with `` `sanic.Sanic.exception(ServerError)`.

It is called by `Sanic` for every `ServerError` exception.

Its job is to log the exception and return a code 500 response.

Parameters

- **request** – the incoming request object.
- **exception** – the exception thrown.

11.2 Unit testing scripts

11.2.1 pyscripts/conftest.py

Fixtures and mockings for unit testing the Switcher WebAPI.

conftest.**mock_control_response**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the control response.

Yields Mocked `SwitcherV2ControlResponseMSG` object.

conftest.**mock_create_schedule_request**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the create_schedule response.

Yields Mocked `SwitcherV2CreateScheduleResponseMSG` object.

conftest.**mock_delete_schedule_request**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the delete_schedule response.

Yields Mocked `SwitcherV2DeleteScheduleResponseMSG` object.

conftest.**mock_disable_enable_schedule_request**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the disable_enable_schedule response.

Yields Mocked `SwitcherV2DisableEnableScheduleResponseMSG` object.

conftest.**mock_get_schedules_response**(*schedule_object*) → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the get_schedules response.

Parameters **schedule_object** – Fixture of mocked `SwitcherV2Schedule` object.

Yields Mocked `SwitcherV2GetScheduleResponseMSG` object.

conftest.**mock_get_state_response**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the get_state response.

Yields Mocked `SwitcherV2StateResponseMSG` object.

conftest.**mock_loop**() → Generator[asyncio.events.AbstractEventLoop, Any, None]
Fixture for running an event loop.

Yields Test event loop for running test server.

conftest.**mock_sanic_test_app**() → Generator[sanic.Sanic, Any, None]
Fixture for creating a test instance on the sanic app.

Yields Test sanic application for mocking testing server.

conftest.**mock_schedule_object**() → Generator[None, None, aioswitcher.schedules.SwitcherV2Schedule]
Fixture for the aioswitcher.schedules.SwitcherV2Schedule object.

Returns Mocked `SwitcherV2Schedule` object.

conftest.**mock_set_auto_shutdown_response**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the set_auto_shutdown response.

Yields Mocked `SwitcherV2SetAutoOffResponseMSG` object.

conftest.**mock_set_device_name_response**() → Generator[asynctest.MagicMock, Any, None]
Fixture for mocking the set_device_name response.

Yields Mocked `SwitcherV2UpdateNameResponseMSG` object.

conftest.**mock_switcher_api_context_manager**() → Generator[None, Any, None]
Fixture for mocking the SwitcherV2Api context manager.

conftest.**mock_tcp_connection**() → Generator[None, Any, None]
Fixture for mocking asyncio.open_connection.

conftest.**mock_test_client**(*loop: asyncio.events.AbstractEventLoop*, *sanic_test_app: sanic.Sanic*) → Generator[None, None, asyncio.events.AbstractEventLoop]
Fixture for starting server in the event loop.

Parameters

- **loop** – Fixture of mocked `AbstractEventLoop` object.
- **sanic_test_app** – Fixture of mocked `Sanic` object.

Returns An event loop with a running server.

11.2.2 pyscripts/helpers.py

Helper functions for unit testing the Switcher WebAPI.

helpers.**get_local_ip_address**() → str
Use for getting the local host's ip address.

Returns The local ip address.

helpers.**get_next_weekday**(*is_iso: bool = False*) → int
Use for getting next day weekday.

Parameters **is_iso** – If true, Monday=1 and Sunday=7. Else Monday=0 and Sunday=6.

Returns The int value represnting the the next weekday (tommorow).

11.2.3 pyscripts/test_server.py

Unit tests for the Switcher WebAPI.

async test_server.**test_create_schedule_request**(*create_schedule_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/create_schedule request.

Parameters **create_schedule_response** – fixture of mocked SwitcherV2CreateScheduleResponseMSG object.

async test_server.**test_delete_schedule_request**(*delete_schedule_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/delete_schedule request.

Parameters **delete_schedule_response** – fixture of mocked SwitcherV2DeleteScheduleResponseMSG object.

async test_server.**test_disable_schedule_request**(*disable_enable_schedule_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/disable_schedule request.

Parameters **disable_enable_schedule_response** – fixture of mocked SwitcherV2DisableEnableScheduleResponseMSG object.

async test_server.**test_enable_schedule_request**(*disable_enable_schedule_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/enable_schedule request.

Parameters **disable_enable_schedule_response** – fixture of mocked SwitcherV2DisableEnableScheduleResponseMSG object.

async test_server.**test_get_schedules_request**(*get_schedules_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/get_schedules request.

Parameters **get_schedules_response** – fixture of mocked SwitcherV2GetScheduleResponseMSG object.

async test_server.**test_get_state_request**(*get_state_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/get_state request.

Parameters **get_state_response** – fixture of mocked SwitcherV2StateResponseMSG object.

async test_server.**test_set_auto_shutdown_request**(*set_auto_shutdown_response: asynctest.MagicMock*) → None
Unit test-cases for /switcher/set_auto_shutdown request.

Parameters **set_auto_shutdown_response** – fixture of mocked SwitcherV2SetAutoOffResponseMSG object.

async test_server.**test_set_device_name_request**(*set_device_name_response: asynctest.MagicMock*) → None

Unit test-cases for /switcher/set_device_name request.

Parameters **set_device_name_response** – fixture of mocked SwitcherV2UpdateNameResponseMSG object.

async test_server.**test_turn_off_request**(*control_response: asynctest.MagicMock*) → None

Unit test-cases for /switcher/turn_off request.

Parameters **control_response** – fixture of mocked SwitcherV2ControlResponseMSG object.

async test_server.**test_turn_on_request**(*control_response: asynctest.MagicMock*) → None

Unit test-cases for /switcher/turn_on request.

Parameters **control_response** – fixture of mocked SwitcherV2ControlResponseMSG object.

11.3 Shared scripts

11.3.1 pyscripts/consts.py

Various constants and test values for the Switcher WebAPI project.

11.3.2 pyscripts/mappings.py

Url mappings for the Switcher WebAPI project are located here.

PYTHON MODULE INDEX

c

conftest, 39
consts, 42

h

helpers, 40

m

mappings, 42

r

request_handlers, 33

s

start_server, 39

t

test_server, 41

INDEX

Symbols

_create_raw_schedule_data() (*in module request_handlers*), 33
_parse_schedule_body() (*in module request_handlers*), 33
_validate_day_to_int() (*in module request_handlers*), 33
_validate_time_integers() (*in module request_handlers*), 33

B

before_start() (*in module start_server*), 39

C

conftest (*module*), 39
consts (*module*), 42
create_schedule_handler() (*in module request_handlers*), 34

D

delete_schedule_handler() (*in module request_handlers*), 34
disable_schedule_handler() (*in module request_handlers*), 35

E

enable_schedule_handler() (*in module request_handlers*), 35

G

get_local_ip_address() (*in module helpers*), 40
get_next_weekday() (*in module helpers*), 41
get_schedules_handler() (*in module request_handlers*), 36
get_state_handler() (*in module request_handlers*), 36

H

helpers (*module*), 40

M

mappings (*module*), 42
mock_control_response() (*in module conftest*), 39
mock_create_schedule_request() (*in module conftest*), 39
mock_delete_schedule_request() (*in module conftest*), 39
mock_disable_enable_schedule_request() (*in module conftest*), 39
mock_get_schedules_response() (*in module conftest*), 40
mock_get_state_response() (*in module conftest*), 40
mock_loop() (*in module conftest*), 40
mock_sanic_test_app() (*in module conftest*), 40
mock_schedule_object() (*in module conftest*), 40
mock_set_auto_shutdown_response() (*in module conftest*), 40
mock_set_device_name_response() (*in module conftest*), 40
mock_switcher_api_context_manager() (*in module conftest*), 40
mock_tcp_connection() (*in module conftest*), 40
mock_test_client() (*in module conftest*), 40

R

request_handlers (*module*), 33

S

set_auto_shutdown_handler() (*in module request_handlers*), 37
set_device_name_handler() (*in module request_handlers*), 37
start_server (*module*), 39

T

test_create_schedule_request() (*in module test_server*), 41
test_delete_schedule_request() (*in module test_server*), 41
test_disable_schedule_request() (*in module test_server*), 41

test_enable_schedule_request() (*in module test_server*), 41
test_get_schedules_request() (*in module test_server*), 41
test_get_state_request() (*in module test_server*), 41
test_server (*module*), 41
test_set_auto_shutdown_request() (*in module test_server*), 41
test_set_device_name_request() (*in module test_server*), 42
test_turn_off_request() (*in module test_server*), 42
test_turn_on_request() (*in module test_server*), 42
timeout() (*in module start_server*), 39
turn_off_handler() (*in module request_handlers*), 38
turn_on_handler() (*in module request_handlers*), 38