OPEN CONNECTIVITY FOUNDATION®

ToolChain

Code generation

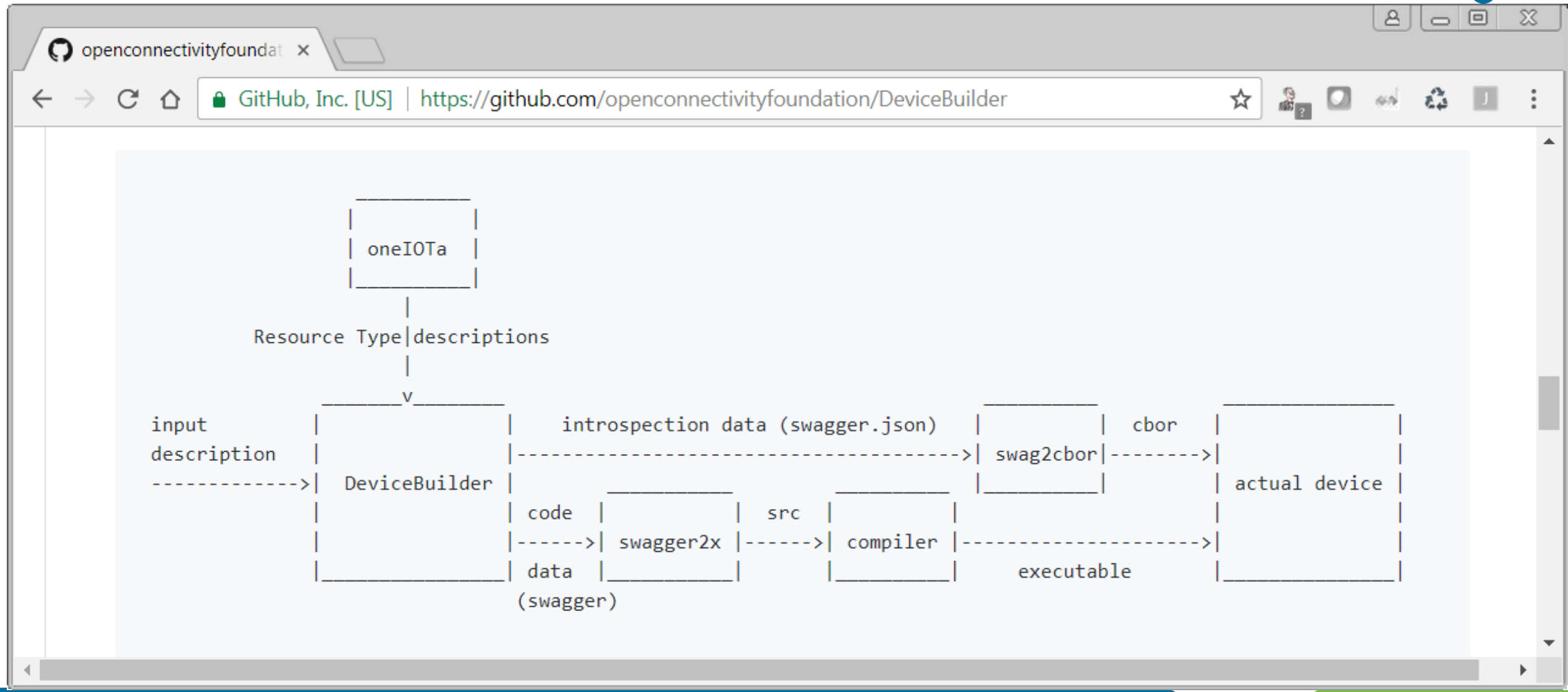
Code generation

Code generation is possible due to :

All Resources are defined in an machine readable format

- Using openAPI definition: Swagger2.0
- Defines resource by:
  - Which operations are supported (RETRIEVE, UPDATE, ..)
  - Schema definition of the payload per operation
  - An example of the payload
  - Define which query parameters are applicable

The DeviceBuilder tool chain



The DeviceBuilder toolchain

- The DeviceBuilder tool chain consist of a set of python scripts glued together with bash scripts.
- The script installs the necessary depended info it needs:
  - Git repos
  - Install python3 packages using pip3
- The chain works on Linux and on Window (using git bash).

- The chain will be used on a raspberry pi

DeviceBuilder

- Python script to merge the resources to be used into a single file.
- Takes an set of resources that will be used in the implementation
  - Combines swagger resource files into 1 large swagger file
    - Renames paths and removes properties while processing.
  - Does this for optional core resources and all resources in oneIOTa.
    - Mandatory resources (core and security) are not processed.

- Input: file with resources to be implemented
- Output: swagger file that list all “application level” resources
  - All resources that the application needs to implement to function correctly.

swagger2x

- Create code by means of template technology
  - Can used jinja2 template technology because swagger2.0 = JSON
  - Add a bit of glue so that JSON constructs can be converted into correct code
  - Jinja2 constructs allows to loop over the DOM of the swagger file
    - Convert by means of instructions in template all the info of the swagger file into code
    - The "clever bit" is in the template, it is an mix of the target code and template language
    - An template is targeted for an specific code base
      - Multiple templates can co-exist, hence the tool can support all languages/APIs
      - Target: C++ IOTivity API.
- Input: generated swagger file from DeviceBuilder
- Output: code!

json2cbor

- Utility to create cbor files from json
  - Json -> cbor
  - Cbor -> json
- Only used to convert the json introspection file into cbor

- NOT for Security resource conversions,
  - The IOTivity json2bor tool converts the json in cbor correctly as being used in IOTivity security resources
  - Additionally https://github.com/alshafi/iotivity-tool can convert to the IOTivity format
    - Tool is installed but not used yet in the chain.

Single script

Single bash script uses the python scripts to create code

Input:

- Input file with the wanted resources
- Output directory
- Which device type is being implemented

For example:

```
sh DeviceBuilder_C++IotivityServer.sh ../input-lightdevice.json ../lightdevice "oic.d.light"
```

What do you get?

After running the DeviceBuilder script:

- Output Directory with:
  - Code currently based on C++ API of IOTivity 1.3.1
  - Introspection file that matches the implementation
    - Available in JSON and CBOR (converted from JSON)
  - Initial just works security file
    - Just a copy of an existing file

Problem: how to build this code

Build environment: IOTivity-setup

Github repo that uses IOTivity v1.3-rel as build environment and sets up the device builder tool chain to build an application

- This github repo sets up an “work area”
  - Installs tools/code/etc to build an IOTivity server application
    - Linux
    - raspberry pi
  - This includes installing a copy of the IOTivity code

Github resides at:

- https://github.com/openconnectivity/IOTivity-setup

IOTivity-setup github repo

The repo has setup scripts for:

- Setting up IOTivity with code and build environment
  - IOTivity version: 1.3-rel
  - Sets it up in local folder
- Setting up DeviceBuilder
  - Relative to IOTivity folder
  - Includes setup of iotivity-tool to generate the Security file.
- Setting up MRAA to interact with the hardware

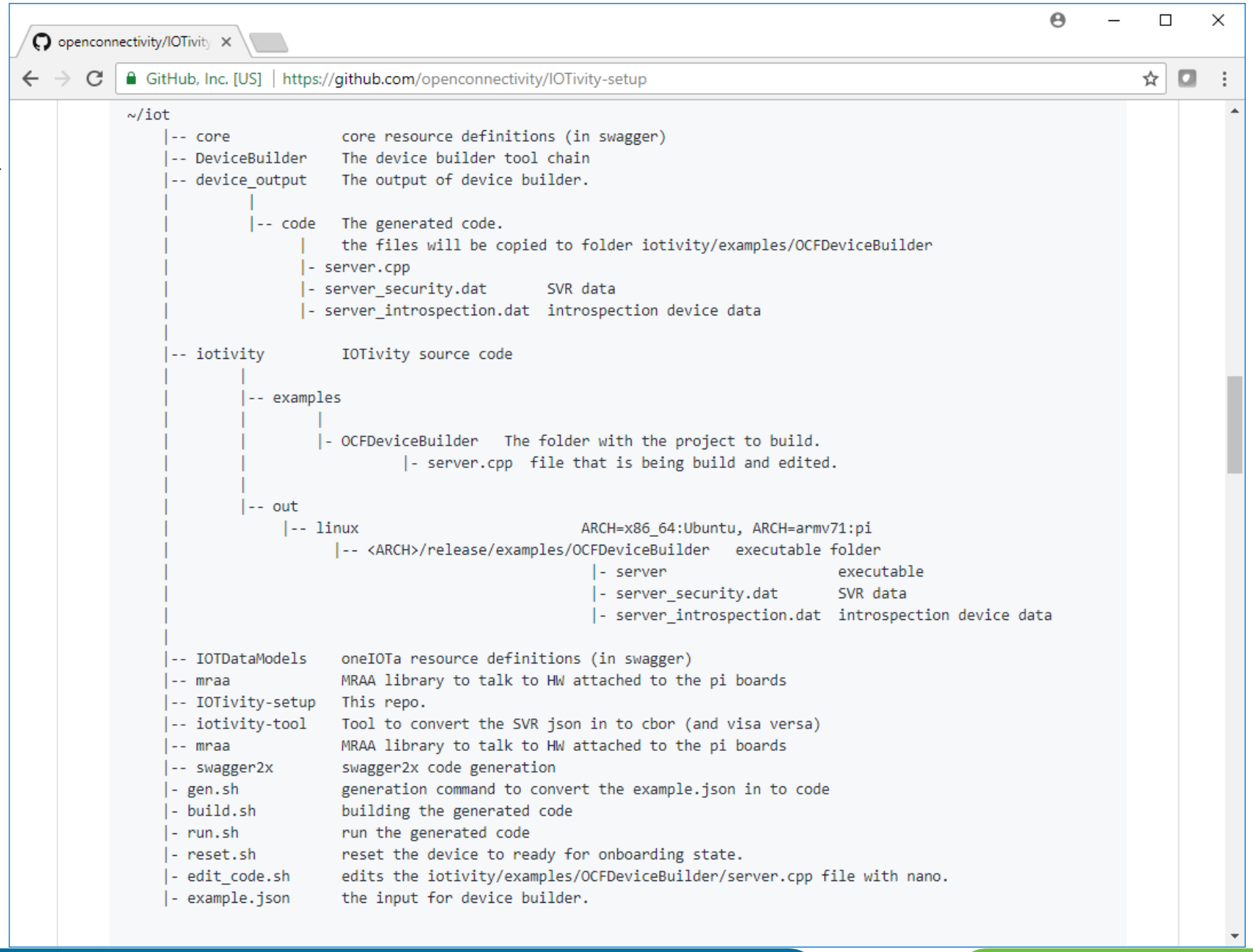
Everything is set up by executing:

curl https://openconnectivity.github.io/IOTivity-setup/install.sh | bash

Folder structure

All folders are placed in the top folder

In this case ~/iot



Preconfigured commands (in iot folder)

- Generate code with gen.sh
- Compile code with build.sh
- Edit code with edit_code.sh (using nano, editing in the iotivity tree)
- Run code with run.sh
- Reset to onboarding state with reset.sh
  - Using just works, file is a copy from the IOTivity tree.

The scripts that are convenience wrappers around command line tools.

- All paths, etc. are filled in
- Code is generated from example.json
- Code is generated in iotivity/examples/OCFDeviceBuilder

Examples of the DeviceBuilder input files

Example for the pimoroni pi hat boards:

- Automation-phat
- Environ-phat

See local folder IOTivity-setup/pi-boards

Other examples in the DeviceBuilder tree:

- /DeviceBuilderInputFormat-file-examples
  - Light device
    - Only with an binary switch
    - Binary switch + dimming
    - Binary switch + dimming + colour chroma

Example input file: binary switch

```
[   {
     "path" : "/binaryswitch",
     "rt"   : [ "oic.r.switch.binary" ],
     "if"   : ["oic.if.a", "oic.if.baseline" ],
     "remove_properties" : [ "range", "step" , "id", "precision" ]
   },  {
     "path" : "/oic/p",
     "rt"   : [ "oic.wk.p" ],
     "if"   : ["oic.if.baseline", "oic.if.r" ],
     "remove_properties" : [ "n", "range", "value", "step", "precision", "vid"  ]
   } ]
```

Path to be used

rt as look up value

Which interfaces are supported

Which properties you do not want

Needed for introspection

OPEN CONNECTIVITY
FOUNDATION®

Where to find IOTivity-setup

OCF github: project IOTivity-setup :

https://github.com/openconnectivity/IOTivity-setup

This project contains on the top level the script:

install.sh

Which can be used by issuing on the (bash) command line:

curl https://openconnectivity.github.io/IOTivity-setup/install.sh | bash

Where to find DeviceBuilder

OCF github: project DeviceBuilder :

https://github.com/openconnectivityfoundation/DeviceBuilder

This project contains on the top level the script:

DeviceBuilder_C++IotivityServer.sh