pyDE1

Jeff Kletsky

Feb 13, 2023

CONTENTS:

CHAPTER

ONE

PYDE1 OVERVIEW

1.1 Description

A fully functional, API-first implementation of core software for control and use of the Decent Espresso DE1.

It provides core components, which can be run as an unattended service that automatically starts at boot, to supply stable, versioned APIs to provide all primary functions of use of a DE1 and data collection around it.

A web app has been able to demonstrate sufficiency of the APIs and functionality for the majority of day-to-day operations, including real-time graphing and history display. A running example is available, linked from DecentForum.com. It uses the supplied, stand-alone "replay" application to play back a shot in real time. This tool is also useful for development of companion applications.

Profiles and real-time data are captured into a SQLite3 database that allows multiple, concurrent access.

A stand-alone program is provided that can automatically upload "shots" to Visualizer as soon as they complete. It can also notify consumers of the URL returned.

An example program is provided that generates legacy-style, "shot files" that are compatible with Visualizer and John Weiss' shot-plotting programs.

The APIs are under semantic versioning. The REST-like, HTTP-transport versions can be retrieved from `version` at the document root, and also include the Python and package versions installed. Each of the JSON-formatted, MQTT packets contains a `version` key:value for that payload.

Consumers of these APIs should only need to understand high-level actions, such as "Here is a profile blob, please load it." The operations and choice of connectivity to the devices is "hidden" behind the APIs.

1.2 Documentation and Source

Documentation is available at https://pyde1.readthedocs.io/en/latest/

Source code is available at https://github.com/jeffsf/pyDE1.git

1.3 Requirements

Python 3.8 or later.

Current plans are to continue to support two versions prior to the latest Python version. PEP 664 shows a schedule of October, 2022 for Python 3.11.0 release. Users should plan on moving to at least Python 3.9 prior to that release. Python 3.9 is "standard" with the Debian Bullseye distro, as well as the current, Raspberry Pi OS release.

Available through `pip` and installed as dependencies:

- `aiosqlite`
- `bleak`
- `paho-mqtt`
- `PyYAML`
- `requests`

An MQTT broker compatible with MQTT 5 clients, such as `mosquitto 2.0`

A production-quality web server, such as `nginx` is highly recommended operating as a reverse proxy, rather than directly exposing the Python web server.

1.4 Status — Release

This code is used on a daily basis for operation of the author's DE1.

1.5 License

CHAPTER

TWO

WHAT'S NEW IN VERSION 2

2.1 Overview

pyDE1 Version 2 provides enhanced usability in several areas:

- Support of Acaia scales
- Support of Felicita Arc thanks to Mimoja
- Temporarily release control of a Bluetooth device
- Changing Bluetooth devices for a given role
- Synchronization across multiple clients
- Reworked Bluetooth scanning to simplify client logic
- Incorporation of Steam-To-Temperature functionality
- Script to convert profiles to JSON format, including from Visualizer
- Packaging and service definitions simplified

Although the DE1 and a Atomax Skale II mounted under the drip tray are "dedicated" devices, many other scales and Bluetooth peripherals are not. You might, for example, want to use your scale for pourover with a different app. Your scale might need to be disconnected from Bluetooth to allow its sleep timer to conserve battery. pyDE1 makes this easy by automatically *releasing* devices when the DE1 sleeps, then automatically *capturing* them when the DE1 wakes up. You can use your scale or thermometer with other apps or just let them go to sleep without having to explicitly disconnect them.

Another use case that has been requested is the ability to manage pyDE1 from multiple places. For example, you might have one UI on your phone and another on a tablet in the kitchen. Although this was possible with earlier versions, you'd have to refresh the display when you moved to another device to catch up with changes in the controls.

Note: The naming of some scripts and executables have been changed to simplify access to those scripts through packaging. You will need to update your service scripts to reflect the new install paths.

2.2 Managed Bluetooth Devices

pyDE1 Version 2 moves to a more sophisticated method for handling Bluetooth devices. Rather than considering the low-level connectivity alone, it now also considers if the device is "remembered". Rather than operating on the Bluetooth layer, commands are now on the logical level, including:

- Assign address or UUID
- Capture
- Release
- "Forget" (by assigning null/None to the address/UUID)

Note: The API endpoint has changed to `de1/availability`.

The `de1/connectivity` endpoint and `ConnectivityChange` notifications have been deprecated. They will be emulated through June, 2023 for backward compatibility with existing clients.

Initialization of the device and its instance in code are handled by pyDE1. pyDE1 will continue to function without a scale or thermometer, although limited in some operations as the data is not available.

"Ready" indicates that the device has been identified, initialized, and should be able to accept further commands. There are typically both API queries as well as broadcast messages around device availability, including the "role" of the device (DE1, scale, thermometer, other, ...) in controlling the system.

Multiple requests can be "stacked up", such a capture followed by release. This queue is only two deep, tracking the current action in progress as well as the desired end state, if different that that in progress. Requests that arrive with an operation in progress just change the desired end state without lengthening the queue. If the desired end state is different that that in progress, the in-flight operation will be cancelled, if possible. This allows, for example, releasing a device that isn't online, but has a pending capture request.

- Initial — The device exists, but no further actions have been taken. In the initial state the device typically does not have an assigned address/UUID (hereafter "address"). Assigning an address will typically make a release request. It may be a generic device. See further *Class-Changing Managed Bluetooth Devices*.
- Capturing — In the process of connecting to a known device, which may or may not be online and visible to the controller
- Captured – The device has been located and is connected at the Bluetooth level. There are no pending changes requested.
  - A *captured* device, once initialized and ready for commands, is marked as *ready*.
  - A device that is not *captured* is never *ready*.
- Releasing – The device is in the process of moving to a the released state.
- Released — The device is no longer in transition between states. If it has an address, a capture request can be made.

Changing the device's address will result in it being released, even if captured or ready. In the case of scales (and likely for thermometers, when implemented), the device is "generified". This is because the appropriate device class is not determined until Bluetooth advertisement packets are seen. It is a backlog item to consider if maintaining a registry of known address-to-class mappings is worthwhile.

2.2.1 Class-Changing Managed Bluetooth Devices

As soon as the device filling a given role can change, things get complicated. Either all internal connections to the device's implementation need to be torn down and reinitialized, or somehow the implementation needs to adapt to the new device. Complete tear-down and reinitialization is unattractive as it requires logic in all other components. Python does not officially support objects that change class. Several approaches to maintaining a consistent set of references as the class changed were explored. Although not supported, changing the class of an instance was selected. There is a set of automated tests to confirm that the expected behavior is present as newer versions of Python are released.

Generic Managed Bluetooth Device

The generic device, such as `GenericScale` is one that embodies most of the behavior all of the specific devices. It is not so much "least common denominator" but more of a "guaranteed stable plug-in point" for a device in the role, as that device comes and goes, or even changes type. If a consumer has a reference to "the scale", then that reference is good even if there is no physical device connected. If a consumer subscribed to an event, it remains subscribed even as the physical device comes and goes, or changes type.

The generic device provides a "device-changed" event to allow consumers that are taking advantage of device-specific features that the device may no longer supply those features or otherwise behave differently.

Often a generic device will be instantiated for a role without an address. Assigning an address to the device will involve releasing the device. It is valid to request capture of a generic device. When advertisement packets are received, code will transition the generic device to an appropriate specific device, if there is a match. Consumers interested in details of the device class can wait on the device-changed event. Those that utilize only the common functionality should only need to follow the ready status.

For the curious, the operation of generic => specific => generic transitions can be seen in

- `_initialize_after_connection()`
- `_adopt_class()`
- `_leave_class()`

2.3 Enhancements in Client Synchronization

There is nothing preventing multiple clients from accessing the pyDE1 APIs. It works quite well to, for example, turn on the DE1 from one device and control it from another. However, in previous versions, changes made on one device weren't automatically reflected on the other.

When changes are made to the pyDE1 controller or a DE1 connects, the resulting state of the impacted area this information is now sent over MQTT to its subscribers.

At this time the areas include the following topics:

- `update/de1/control`
- `update/de1/setting`
- `update/de1/calibration`
- `update/de1/profile/id`

Timestamps are available in the MQTT packets as well as in the HTTP response header `x-pyde1-timestamp` to assist in disambiguation of the two sources.

Additionally, availability and DE1 state have been added to the HTTP responses in a format compatible with clients retaining state as "mqtt".

2.4 Rework of Bluetooth Scanning

The approach to Bluetooth scanning was reworked to use changes in the Bleak Bluetooth library as well as to simplify client code.

Scanning is now by "role", one of

- DE1
- Scale
- Thermometer

The scan results are reported over MQTT and consist of a boolean `scanning` indicating if a scan is still underway, and `devices`, an array of accumulated information about all devices matching the requested role seen during the scan. The packet contains

- `address`
- `name`
- `rssi`

for each device, with updates as additional devices are found during the scan.

The results are no longer retained in the `DiscoveredDevices` structure and the APIs to access that structure are not available.

Note: MAPPING 7.0.0 — RESOURCE 5.0.0

- `Resource.SCAN` (`scan`) - Now takes a string representing the role - Can no longer be PATCH-ed (use PUT with the desired DeviceRole)
- `Resource.SCAN_DEVICES` (`scan_devices`) – has been removed

```
class DeviceRole (enum.Enum):
    DE1 = 'de1'
    SCALE = 'scale'
    THERMOMETER = 'thermometer'
    OTHER = 'other'
    UNKNOWN = 'unknown'
```

2.5 Steam-To-Temperature

Previously developed as a separate app, now integral with pyDE1

https://github.com/jeffsf/steam-to-temperature

2.5.1 Use

- Set the BlueDOT to either °C or °F, as desired.
- The program will attempt to connect to the BlueDOT when the DE1 is not sleeping. If not immediately found, it will retry, falling back to once every 30 seconds. The BlueDOT will beep briefly to indicate connection. If you're looking at the display, you'll see the high alarm displaying freezing (0°C or 32°F) while beeping.
- If the DE1 sleeps, it will disconnect from the BlueDOT, allowing it to be used elsewhere with the Thermoworks app. As long as it is disconnected from the Thermoworks app and is on and in range, it will reconnect when the DE1 wakes.
- Set the desired target temperature as the high alarm on the BlueDOT
- Put the probe in the steaming pitcher.
- Start steaming with the GHC (or app control for non-GHC machines)
- The steam will pause automatically, going into "puff mode". Remove the pitcher. (For tiny volumes, the puffs can be sufficient to raise the temperature slightly above target.)
- Stop the steaming with the GHC or app control. This will trigger the usual auto-purge sequence.

2.6 Utility to Convert Legacy Profiles to JSON format

`de1-profile-as-json` is now packaged and installed in the PATH of the venv used to install pyDE1, such as `/home/pyde1/venv/pyde1/bin/de1-profile-as-json`. If the venv is active for the user's shell, it will be directly available without specifying the full path. It can also be run using the full path without activating the venv.

This utility will accept a legacy profile, including downloading one from Visualizer, and output a JSON version.

> As most of the profiles distributed fail to properly attribute the author, the author can be overridden on the command line with the `-a` or `--author` flag.
>
> Without any arguments, the utility accepts input from STDIN and writes to STDOUT.
>
> The input can be specified from a file using the `-i` or `--input` flag followed by the filename.
>
> Alternately, a Visualizer URL for a profile or a Visualizer "share code" (four characters) can be entered after the `-v` or `--visualizer` flag.
>
> The output filename can optionally be specified using the `-o` or `--output` flag.
>
> If one specifies the `-d` or `--directory` flag followed by a directory, the output will be placed in that directory.
>
> If `-d` is specified, but not `-o` for a specific output name, a reasonable guess will be made based on the input file name and profile title.
>
> If the output file already exists, `-f` or `--force` can be used to overwrite.

```
usage: de1-profile-as-json [-h] [-a AUTHOR] [-i INPUT | -v REF] [-o OUTPUT] [-d DIR] [-f]

Executable to open a Tcl profile file and write as JSON v2.1. Input and output default␣
↪to STDIN and STDOUT

optional arguments:
  -h, --help                  show this help message and exit
  -a AUTHOR, --author AUTHOR  Replace author
  -i INPUT, --input INPUT     Input file
```

(continues on next page)

(continued from previous page)

```
  -v REF, --visualizer REF    Visualizer short code or profile URL
  -o OUTPUT, --output OUTPUT  Output file
  -d DIR, --dir DIR           Output directory
  -f, --force                 Overwrite if output exists
```

Although it is believed that the conversion is done accurately, it is always worthwhile to check the results prior to use.

2.7 Profile Specification JSON v2.1

This release includes a description of the JSON profile format, extended from Mimoja's original work. This document is structured as TypeScript for simplicity as well as potential reuse. However, it has not been validated in the context of a TypeScript app.

See *JSON Profile Spec*

2.8 Packaging / Service Definition Changes

The primary executables and scripts are now packaged in the `bin/` directory of the venv into which pyDE1 is installed. They are self-sufficient in that the venv does not need to be "activated" to run them in the context of that venv. This simplifies service scripts, as well as making utilities such as `pyde1-disconnect-btid.sh` easily available. These scripts include:

- `pyde1-run` – the main executable of the pyDE1 controller
- `pyde1-run-visualizer` – a companion executable that uploads shots to the Visualizer service
- `pyde1-disconnect-btid.sh` – a shell script to disconnect any Bluetooth devices that were recorded as being connected by pyDE1 in the event of a very ungraceful exit or during development
- `pyde1-replay` – a utility script to replay the packets from a previous shot over MQTT

New versions of the `.service` files are packaged. The new versions no longer require determining the "deep" path of the file (shown here as of v2.0):

```
[Unit]
Description=Main controller processes for pyDE1
Wants=mosquitto.service
After=syslog.target mosquitto.service

[Service]
# This needs to be the same user that "owns" the database
User=pyde1
Group=pyde1

ExecStartPre=/home/pyde1/venv/pyde1/bin/pyde1-disconnect-btid.sh
# The executable name can't be a variable
ExecStart=/home/pyde1/venv/pyde1/bin/pyde1-run
ExecStopPost=/home/pyde1/venv/pyde1/bin/pyde1-disconnect-btid.sh

Restart=always
StandardError=journal
# Sets the process name to that of the service
```

(continues on next page)

(continued from previous page)

```
SyslogIdentifier=%N

[Install]
WantedBy=multi-user.target
```

CHAPTER

THREE

API

There are three, versioned, supported APIs for pyDE1. These include an HTTP server for querying and setting of parameters, an MQTT service for obtaining real-time updates, and an SQLite3 database allowing concurrent access to history.

With these APIs and the configuration files, there should be no need to access code internals. If there is missing functionality, please file an enhancement request including the use case that you wish to satisfy.

The code internals are subject to change and should not be relied on as an API.

Semantic versioning is available for the HTTP and MQTT APIs. Changes are noted in the commit logs, as well as in the CHANGELOG. Breaking changes (those that are not backwards-compatible) will increment the major version. Details of how to obtain the running version are described in the detailed sections for each of the APIs.

The database schema is available through `PRAGMA user_version`. It is an incrementing integer. Although it is hoped that schema changes will be backwards compatible, checking the commit logs and CHANGELOG is suggested for authors of applications that directly access the database.

3.1 HTTP API

The HTTP API is used to query and change the state of the DE1, scale, and controller. This API provides a level of abstraction higher than that of the Bluetooth interface to the DE1. For example, a profile is uploaded through the API, directly from a JSON v2 file. The caller does not need to know how that is converted into frames and then loaded over Bluetooth to the DE1.

Most payloads are JSON with status being returned in a REST-like manner with HTTP error codes. (Profiles and firmware are uploaded verbatim.)

The API has been organized into sections that align with user experience, rather than paralleling the DE1's low-level API. Users should not have to be overly concerned about if the functionality is provided by the DE1 firmware, or by the controller supplied by pyDE1. In some cases, certain firmware versions are able to control a feature directly with the DE1, yet others require the controller's interaction. From a caller perspective, the parameters are set up the same way for both.

3.1.1 API Overview

Principles of Operation

The API is REST-like in operation, with the operation being specified by the HTTP method. In general

- GET returns information about a resource without modifying it
- PATCH updates some information or state
- PUT sends a complete replacement

At this time, PATCH is used for virtually all changes. Profiles and firmware images are PUT as they completely replace the proir profile or firmware.

The *InboundAPI* process runs a single-threaded, HTTP server. It is intentionally sequential in operation as many operations assume that the initial state is that resulting from completion of all prior operations. On receiving a request, its URL is compared to the list of valid `Resource` values and confirmed that the requested method (GET, PATCH, PUT) is valid for that resource. If not, an error response is returned. If so, the request and request body are wrapped in an `APIRequest` object and queued for processing.

In the *Controller* process, a queue watcher retrieves the `APIRequest`. This code can be found in `pyDE1/dispatcher/`. It evaluates if connectivity to either or both the DE1 and scale is required. If connectivity requirements are not met, an exception is raised. Like other exceptions in processing these API requests, they are caught and an error `APIResponse` is queued back to the *InboundAPI* process, often including traceback information.

The `Resource` is used as a key into the `MAPPING` dict to determine where the various data values can be accessed, as well as how they are represented in JSON. Individual elements of the `MAPPING` are represented with an `IsAt` object. The `IsAt` instance identifies on which object the setter and getter can be found, the setter and getter, if it is read-only, write-only, or read/write, as well as the expected data type. The data type is checked before proceeding.

For most payloads, the JSON structure is then walked, getting or setting values as requested.

Note: With payloads containing multiple values, a PATCH operation may not be atomic if later elements fail after earlier ones have been set.

All operations have a timeout. These timeouts can be seen in the `Config`, either in the Bluetooth or HTML sections. Exceeding a timeout results in an exception being raised.

If no exception has been raised, a success `APIResponse` is queued back to the *InboundAPI* process. If an exception is raised, an error response is queued.

The *InboundAPI* process then retrieves the `APIResponse` and returns its contents as an HTTP response to the caller.

Note: API consumers should check the response headers to determine if the request was successful or not.

Note: When changes are made to the DE1's internal registers, an asynchronous read-back is usually requested. As it takes roughly 100 ms per transaction over Bluetooth with the current DE1 hardware and its Bluetooth firmware, a read over the API may not yet be updated when the setting API call returns.

While tempting to assume that the data in the DE1 is that which was written, there are certain registers that trim or reject values out of range, or round them slightly differently internally.

Example Responses

A successful setting change

```
$ curl -D - -X PATCH --data '{ "start_fill_level": 0 }' http://localhost:1234/de1/
↪setting/start_fill_level
HTTP/1.0 200 OK
Server: BaseHTTP/0.6 Python/3.9.2
Date: Tue, 16 Nov 2021 21:37:09 GMT
Content-type: application/json
Content-length: 3
Last-Modified: Tue, 16 Nov 2021 13:37:09 -0800

[]
```

An "early" error response due to an inappropriate method

```
$ curl -D - -X PUT --data '{ "start_fill_level": 0 }' http://localhost:1234/de1/setting/
↪start_fill_level
HTTP/1.0 501 Not Implemented
Server: BaseHTTP/0.6 Python/3.9.2
Date: Tue, 16 Nov 2021 21:37:56 GMT
Content-type: text/plain
Content-length: 63
Last-Modified: Tue, 16 Nov 2021 13:37:56 -0800

PUT not yet supported for Resource.DE1_SETTING_START_FILL_LEVEL
```

An error response due to a "bad" value

```
$ curl -D - -X PATCH --data '{ "start_fill_level": "0.0" }' http://localhost:1234/de1/
↪setting/start_fill_level
HTTP/1.0 400 Bad Request
Server: BaseHTTP/0.6 Python/3.9.2
Date: Tue, 16 Nov 2021 21:39:44 GMT
Content-type: text/plain
Content-length: 67
Last-Modified: Tue, 16 Nov 2021 13:39:44 -0800

DE1APITypeError('Expected int value at start_fill_level:, not 0.0')
```

An error response due to malformed JSON

```
$ curl -D - -X PATCH --data '{ start_fill_level: 0.0 }' http://localhost:1234/de1/
↪setting/start_fill_level
HTTP/1.0 400 Bad Request
Server: BaseHTTP/0.6 Python/3.9.2
Date: Tue, 16 Nov 2021 21:42:23 GMT
Content-type: text/plain
Content-length: 94
Last-Modified: Tue, 16 Nov 2021 13:42:23 -0800

JSONDecodeError('Expecting property name enclosed in double quotes: line 1 column 3␣
↪(char 2)'
```

An error response with traceback

```
$ curl -D - http://localhost:1234/de1
HTTP/1.0 409 Conflict
Server: BaseHTTP/0.6 Python/3.9.2
Date: Tue, 16 Nov 2021 21:44:46 GMT
Content-type: text/plain
Content-length: 395
Last-Modified: Tue, 16 Nov 2021 13:44:46 -0800

Traceback (most recent call last):
  File "/home/pyde1/deploy/pyde1-devel/src/pyDE1/dispatcher/dispatcher.py", line 120, in␣
↪_request_queue_processor
    _check_connectivity(got)
  File "/home/pyde1/deploy/pyde1-devel/src/pyDE1/dispatcher/dispatcher.py", line 96, in _
↪check_connectivity
    raise DE1NotConnectedError("DE1 not connected")
pyDE1.exceptions.DE1NotConnectedError: DE1 not connected
```

Client Synchronization

Multiple clients can access the pyDE1 API. It works quite well to, for example, turn on the DE1 from one device and control it from another. However, in previous versions, changes made on one device weren't easily reflected on the other.

Starting with pyDE1 v2.0, when changes are made to the pyDE1 controller or a DE1 connects, the resulting state of the impacted area this information is now sent over MQTT to its subscribers.

At this time the areas include the following topics:

- `update/de1/control`
- `update/de1/setting`
- `update/de1/calibration`
- `update/de1/profile/id`

Timestamps are available in the MQTT packets as well as in the HTTP response header `x-pyde1-timestamp` to assist in disambiguation of the two sources.

Versioning

The list of resources that can be accessed is defined in `pyDE1/dispatcher/resource.py` The list of resources is versioned, as is the mapping of those resources to data elements found in `pyDE1/dispatcher/mapping.py`.

The versions of the API (and other components) can be easily retrieved

```
$ curl http://localhost:1234/version
{
    "mapping_version": "7.0.0",
    "module_versions": {
        "PyYAML": "6.0",
        "aiosqlite": "0.18.0",
        "asyncio-mqtt": null,
        "bleak": "0.19.5",
```

(continues on next page)

(continued from previous page)


```
        "paho-mqtt": "1.6.1",
        "pyDE1": "2.0.0b2",
        "requests": "2.28.2"
    },
    "platform": "linux",
    "python": "3.9.2 (default, Feb 28 2021, 17:03:44) \n[GCC 10.2.1 20210110]",
    "python_info": {
        "major": 3,
        "micro": 2,
        "minor": 9,
        "releaselevel": "final",
        "serial": 0
    },
    "resource_version": "5.0.0",
    "source_data": null
}
```


Feature Availability

In addition to the software versions, the firmware version and hardware present on the DE1 can be important to clients.

Rather than requiring each client to keep a list of which firmware provides which features, *feature flags* are provided in an easily digested form.


```
$ curl http://localhost:1234/de1/feature_flags
{
    "feature_flags": {
        "allow_usb_charging": true,
        "app_feature_flag_user_present": true,
        "fw_version": 1333,
        "ghc_active": true,
        "hot_water_flow_control": false,
        "last_mmr0x80": 14432,
        "mmr_max_shot_press": false,
        "mmr_pref_ghc_mci": false,
        "refill_kit_present": true,
        "rinse_control": true,
        "safe_to_read_mmr_continuous": true,
        "sched_idle": true,
        "skip_to_next": true,
        "steam_purge_mode": true,
        "user_present": true
    }
}
```


`ghc_active` can be used to determine if the commands to start flow have been disabled or not.

Features introduced prior to firmware version 1250 (April 2020) are not captured.

3.1.2 Examples

Connect

To First-Found DE1

```
$ curl -X PATCH --data '{"id": "scan"}' http://localhost:1234/de1/id
[
    "D9:B2:48:AA:BB:CC"
]
```

To Specific DE1

```
$ curl -X PATCH --data '{"id": "D9:B2:48:AA:BB:CC"}' http://localhost:1234/de1/id
[]
```

Espresso Control

```
$ curl http://localhost:1234/de1/control/espresso
{
    "disable_auto_tare": false,
    "first_drops_threshold": 0.0,
    "last_drops_minimum_time": 3.0,
    "profile_can_override_stop_limits": false,
    "profile_can_override_tank_temperature": true,
    "stop_at_time": null,
    "stop_at_volume": null,
    "stop_at_weight": 46
}

$ curl -X PATCH --data '{ "stop_at_weight": 51 }' http://localhost:1234/de1/control/
↪espresso
[]
```

Query Current State

Although state *updates* are available through MQTT, the DE1 won't report state until it changes. A newly connected DE1 or client may need current state information to initialize.

```
$ curl http://localhost:1234/de1/state
{
  "state": {
      "state": "Sleep",
      "substate": "NoState"
  }
}
```

Change Profile

```
$ curl -X PUT --data '{"id": "3f8d1e22d77d860d53d011b4974720974d5380f2"}' http://
↪localhost:1234/de1/profile/id
[]
```

Upload Profile

Note that the profile's source file is delivered verbatim.

```
$ curl -X PUT --data @./defaultish_88.json http://localhost:1234/de1/profile
[]
```

List and Fetch Logs

```
$ curl http://localhost:1234/logs
[
    {
        "atime": 1632985202.3721957,
        "ctime": 1637049601.4134495,
        "id": "pyde1.log.46.gz",
        "mtime": 1633041788.1226397,
        "name": "pyde1.log.46.gz",
        "size": 9125
    },
    {
        "atime": 1636095601.4804242,
        "ctime": 1637049601.4454553,
        "id": "visualizer.log.11.gz",
        "mtime": 1636125920.634909,
        "name": "visualizer.log.11.gz",
        "size": 417
    },

    // similar entries omitted

    {
        "atime": 1636358402.1129558,
        "ctime": 1637049601.4134495,
        "id": "pyde1.log.8.gz",
        "mtime": 1636413162.628252,
        "name": "pyde1.log.8.gz",
        "size": 6525
    }
]
```

Fetch is by `id` (which presently is the file name, though this is not guaranteed)

```
$ curl http://localhost:1234/log/pyde1.log 2>/dev/null | tail
2021-11-16 10:44:55,153 INFO [Controller] DE1.CUUID.FrameWrite.Write: Frame #2 CtrlP,
↪DontCompare,DC_LT,DC_CompP,TBasketTemp,DontInterpolate,IgnoreLimit SetVal: 8.0 Temp:␣
```

(continues on next page)

(continued from previous page)

```
↪88.0 Len: 4.0 Trigger: 0 MaxVol: 0.0
2021-11-16 10:44:55,245 INFO [Controller] DE1.CUUID.FrameWrite.Write: Frame #3 CtrlP,
↪DontCompare,DC_LT,DC_CompP,TBasketTemp,Interpolate,IgnoreLimit SetVal: 4.0 Temp: 88.0␣
↪Len: 40.0 Trigger: 0 MaxVol: 0.0
2021-11-16 10:44:55,343 INFO [Controller] DE1.CUUID.FrameWrite.Write: Frame #4 Limit: 0␣
↪ignore_pi: True
2021-11-16 10:44:55,446 INFO [Controller] Database.Insert: Profile␣
↪68e02cd99418003806d8e5efdf711f078bdfcc22 already in profile table.
2021-11-16 10:44:55,455 INFO [Controller] DE1: Returned from db insert
2021-11-16 10:44:55,460 INFO [InboundAPI] Inbound.HTTP: 603 200 "OK" - PUT /de1/profile␣
↪HTTP/1.1 127.0.0.1
2021-11-16 10:46:49,993 INFO [InboundAPI] Inbound.HTTP: Request: GET /logs HTTP/1.1
2021-11-16 10:46:50,002 INFO [InboundAPI] Inbound.HTTP: 9 200 "OK" - GET /logs HTTP/1.1␣
↪127.0.0.1
2021-11-16 10:48:33,043 INFO [InboundAPI] Inbound.HTTP: Request: GET /log/pyde1.log HTTP/
↪1.1
2021-11-16 10:48:33,044 INFO [InboundAPI] Inbound.HTTP: 2 200 "OK" - GET /log/pyde1.log␣
↪HTTP/1.1 127.0.0.1
```

Search for a Thermometer

```
curl -X PUT --data 'thermometer' http://localhost:1234/scan
```

See also:

ScanResults

Get "Everything"

Note: Although this is possible and useful for reference, targeted requests are strongly suggested.

```
$ curl http://localhost:1234/de1
{
    "availability": {
        "mode": "ready",
        "mqtt": "{\"arrival_time\": 1675615807.6785038, \"create_time\": 1675615807.
↪6785834, \"state\": \"ready\", \"role\": \"de1\", \"id\": \"D9:B2:48:AA:BB:CC\", \
↪"name\": \"DE1\", \"version\": \"1.1.0\", \"event_time\": 1675615807.695822, \"sender\
↪": \"DE1\", \"class\": \"DeviceAvailability\"}"
    },
    "calibration": {
        "flow_multiplier": {
            "multiplier": 1.1
        },
        "line_frequency": {
            "hz": 60
        }
    },
    "connectivity": {
```

(continues on next page)

(continued from previous page)


```
        "mode": "ready"
    },
    "control": {
        "espresso": {
            "disable_auto_tare": false,
            "first_drops_threshold": 0.0,
            "last_drops_minimum_time": 3.0,
            "move_on_weight": [],
            "profile_can_override_stop_limits": false,
            "profile_can_override_tank_temperature": true,
            "stop_at_time": null,
            "stop_at_volume": null,
            "stop_at_weight": null
        },
        "hot_water": {
            "disable_auto_tare": false,
            "stop_at_time": 0,
            "stop_at_volume": 0,
            "stop_at_weight": null,
            "temperature": 0
        },
        "hot_water_rinse": {
            "disable_auto_tare": false,
            "flow": 6.0,
            "stop_at_time": 3.0,
            "temperature": 92.0
        },
        "steam": {
            "disable_auto_tare": false,
            "stop_at_time": 200
        },
        "tank_water_threshold": {
            "temperature": 0
        }
    },
    "id": {
        "id": "D9:B2:48:AA:BB:CC",
        "name": "DE1"
    },
    "read_once": {
        "cpu_board_model": 1.3,
        "firmware_build_number": 1333,
        "firmware_model": "UNSET",
        "ghc_info": "GHC_ACTIVE|TOUCH_CONTROLLER_PRESENT|LED_CONTROLLER_PRESENT",
        "heater_voltage": 120,
        "hw_config_hexstr": "ffffffff",
        "model_hexstr": "ffffffff",
        "serial_number_hexstr": "00000000",
        "version_ble": {
            "api": 4,
            "blesha_hexstr": 3319390896,
            "changes": 124,
```


(continues on next page)

(continued from previous page)


```
            "commits": 559,
            "release": 1.5
        },
        "version_lv": {
            "api": 0,
            "blesha_hexstr": 0,
            "changes": 0,
            "commits": 0,
            "release": 0.0
        }
    },
    "setting": {
        "auto_off_time": {
            "time": 30.0
        },
        "before_flow": {
            "heater_idle_temperature": 85.0,
            "heater_phase1_flow": 2.0,
            "heater_phase2_flow": 4.0,
            "heater_phase2_timeout": 5.0
        },
        "fan_threshold": {
            "temperature": 40
        },
        "refill_kit": {
            "present": false
        },
        "start_fill_level": {
            "start_fill_level": 1.0
        },
        "steam": {
            "flow": 0.7,
            "high_flow_time": 2.0,
            "purge_deferred": true,
            "temperature": 160
        },
        "target_group_temp": {
            "temperature": 0.0
        },
        "time": {
            "timestamp": 0
        },
        "usb_outlet": {
            "enabled": true
        }
    },
    "state": {
        "mqtt": "{\"arrival_time\": 1675615805.8662703, \"create_time\": 1675615805.
↪8665967, \"state\": \"Sleep\", \"substate\": \"NoState\", \"previous_state\": \
↪"NoRequest\", \"previous_substate\": \"NoState\", \"is_error_state\": false, \"version\
↪": \"1.0.0\", \"event_time\": 1675615805.866659, \"sender\": \"DE1\", \"class\": \
↪"StateUpdate\"}",
```


(continues on next page)

(continued from previous page)

```
        "state": {
            "last_updated": 1675615805.874492,
            "state": "Sleep",
            "substate": "NoState"
        }
    }
}
```

```
$ curl http://localhost:1234/scale
{
    "availability": {
        "mode": "ready",
        "mqtt": "{\"arrival_time\": 1675615815.4239364, \"create_time\": 1675615815.
↪4240103, \"state\": \"ready\", \"role\": \"scale\", \"id\": \"FF:06:AF:AA:BB:CC\", \
↪"name\": \"AtomaxSkaleII: Skale\", \"version\": \"1.1.0\", \"event_time\": 1675615815.
↪4284487, \"sender\": \"AtomaxSkaleII\", \"class\": \"DeviceAvailability\"}"
    },
    "connectivity": {
        "mode": "ready"
    },
    "id": {
        "id": "FF:06:AF:AA:BB:CC",
        "name": "AtomaxSkaleII: Skale"
    }
}
```

```
$ curl http://localhost:1234/thermometer
{
    "availability": {
        "mode": "ready",
        "mqtt": "{\"arrival_time\": 1675615824.7947285, \"create_time\": 1675615824.
↪7948647, \"state\": \"ready\", \"role\": \"thermometer\", \"id\": \"00:A0:50:AA:BB:CC\
↪", \"name\": \"BlueDOT\", \"version\": \"1.1.0\", \"event_time\": 1675615824.8039956, \
↪"sender\": \"BlueDOT\", \"class\": \"DeviceAvailability\"}"
    },
    "id": {
        "id": "00:A0:50:AA:BB:CC",
        "name": "BlueDOT"
    }
}
```

3.2 MQTT API

The MQTT API provides updates of the state of the DE1, scale, and controller. An MQTT broker allows subscribing to the various notifications without putting additional load on the controller.

3.2.1 Versioning

The MQTT API's JSON payloads are under semantic versioning. At this time, there is not discoverability that the list of topics have changes.

The payload version can be found with the `version` key. For example:

```
{
  "arrival_time": 1637014022.035329,
  "create_time": 1637014022.0355482,
  "state": "Idle",
  "substate": "HeatWaterTank",
  "previous_state": "Sleep",
  "previous_substate": "NoState",
  "is_error_state": false,
  "version": "1.0.0",
  "event_time": 1637014022.0356083,
  "sender": "DE1",
  "class": "StateUpdate"
}
```

The non-JSON payloads, such as the human-readable log messages and MQTT will, are not versioned at this time.

3.2.2 Will (Client Status)

The pyDE1 clients all register a will that is sent by the broker to subscribers in the event of a non-graceful disconnect. The code also tries to indicate if the client is present.

For pyDE1, the wills are set up in `pyDE1/api/outbound/mqtt`. The two topics are:

- `<config.mqtt.TOPIC_ROOT>/status/mqtt/logging`
- `<config.mqtt.TOPIC_ROOT>/status/mqtt/notification`

The values presently published are

```
class MQTTStatusText (enum.Enum):
  on_connection = 'Here'
  on_graceful_disconnect = 'Gone'
  on_will = 'Died'
```

3.2.3 Logging Feeds

As configured, there are two MQTT feeds of logging data from pyDE1, one with most fields from the `LogRecord` and the other already formatted. The log level is ERROR, by default. Both the level and the formatter can be set through the config file.

- `<config.mqtt.TOPIC_ROOT>/log`
- `<config.mqtt.TOPIC_ROOT>/log/record`

For version 1.0.0, the attributes from the `LogRecord` in the JSON include `created`, `levelname`, `levelno`, `message`, `name`, `process`, `processName`, `thread`, `threadName`.

3.2.4 Client Synchronization

Starting with pyDE1 v2.0, when changes are made to the pyDE1 controller or a DE1 connects, the resulting state of the impacted area this information is now sent over MQTT to its subscribers.

At this time the areas include the following topics:

- `update/de1/control`
- `update/de1/setting`
- `update/de1/calibration`
- `update/de1/profile/id`

The packets contain a JSON version similar to what a GET of the resource following the `update/` prefix would provide. For example, changing the stop-at-weight level over the HTTP API results in an MQTT packet like

On `update/de1/control`


```
{"espresso": {"stop_at_time": null, "stop_at_volume": null,
              "stop_at_weight": 34, "move_on_weight": [],
              "disable_auto_tare": false,
              "profile_can_override_stop_limits": false,
              "profile_can_override_tank_temperature": true,
              "first_drops_threshold": 0.0, "last_drops_minimum_time": 3.0},
 "steam": {"stop_at_time": 200, "disable_auto_tare": false},
 "hot_water": {"stop_at_time": 0, "stop_at_volume": 0,
               "stop_at_weight": null, "disable_auto_tare": false,
               "temperature": 0},
 "hot_water_rinse": {"stop_at_time": 3.0, "disable_auto_tare": false,
                     "temperature": 92.0, "flow": 6.0},
 "tank_water_threshold": {"temperature": 0},
 "timestamp": 1675628682.0626736}
```


Timestamps are available in the MQTT packets as well as in the HTTP response header `x-pyde1-timestamp` to assist in disambiguation of the two sources.

3.2.5 Event Payloads

The majority of MQTT updates come from subclasses of the `EventPayload` class, defined in `pyDE1/event_manager/payloads.py`. These will always include:

- `arrival_time` – When the "trigger" occurred
- `create_time` – When the payload was created
- `sender` – A string indicating the "source" of the message (typically the class name)
- `event_time` – When the event was published
- `version` – A string of the semantic version of the payloads

All times are those returned by `time.time()`

Those generated by an `EventWithNotification` will also include an `action` of either `set` or `clear`.

3.2.6 Overview of Notifications

This section lists and briefly describes the intent of the main notifications provided by pyde1.

StateUpdate

Contains the state and substate reported by the DE1.

The DE1 reports this only on changes. Current state, if needed, can be queried through the HTTP API, such as for app initialization after contacting to an already connected controller. (The state can't be queried over Bluetooth from the DE1.)

ShotSampleWithVolumesUpdate

Contains the information reported by the DE1 in the ShotSample packet. This includes various pressures, flow rates, and temperatures.

It is augmented with estimated volumes for preinfuse, flow, and total, as well as an array of by-frame volumes.

Use of `de1_time` is preferred. At some time in the future, `de1_time` may represent a best-estimate of the DE1's notion of reporting time, rather than the packet arrival time.

The DE1 reports this every 25 half-cycles of the AC while not in Sleep. This is 4 per second for 50 Hz and 4.8 per second for 60 Hz. While in Sleep, the rate appears to drop by a factor of three (these and other DE1-generated rates with firmware 1283).

WaterLevelUpdate

Contains the current water level in the tank and the refill level setting of the DE1.

Sent roughly 2.5 times a second when not in Sleep, 1/3 of that during Sleep (on 60 Hz).

WeightAndFlowUpdate

Contains the current weight along with estimates of weight and mass flow and their corresponding times.

Use of the corresponding times is preferred as they incorporate estimation delays.

```
current_weight
current_weight_time
average_flow
average_flow_time
median_weight
median_weight_time
median_flow
median_flow_time
```

Time is as would be reported by `time.time()`. Weight is as-set on the scale, typically in grams. Mass-flow is in weight units per second, typically grams/second.

`scale_time` may, in the future, represent a corrected time base for the scale, rather than just using the packet-arrival time.

Sent at the reporting rate of the scale, often 10 per second.

ScaleButtonPress

Sent when a button press is reported by the scale.

Includes an integer to identify the button pressed. Encoding is specific to each scale.

ScaleTareSeen

Sent after a tare request when the scale reports a value "close enough" to zero, within the timeout to respond to the tare request.

SequencerGateNotification

The `FlowSequencer` is responsible for managing and tracking flow during any of the flow phases, espresso, steam, hot water, and flush (hot water rinse). It assigns a sequence ID at the start of a sequence, that is used to associate the various records in the database with each other. There are then several "gates" that a sequence goes through. All gates are cleared (they are implemented as `Event` objects and adopt that object's notion of `.set()`, .``.clear()``, and `.wait()`) when a DE1 state change indicates a new sequence beginning. As each gate is passed, it is set. Notifications are sent over MQTT for both clear and set.

```
class SequencerGateName (EventNotificationName):
    GATE_SEQUENCE_START = "sequence_start"
    GATE_FLOW_BEGIN = "sequence_flow_begin"
    GATE_EXPECT_DROPS = "sequence_expect_drops"
    GATE_EXIT_PREINFUSE = "sequence_exit_preinfuse"
    GATE_FLOW_END = "sequence_flow_end"
    GATE_FLOW_STATE_EXIT = "sequence_flow_state_exit"
    GATE_LAST_DROPS = "sequence_last_drops"
    GATE_SEQUENCE_COMPLETE = "sequence_complete"
```

The `sequence_id` is included in all packets, along with the `action` of either `clear` or `set`.

StopAtNotification

When the `FlowSequencer` is managing termination, a StopAtNotification is sent at termination that includes the `stop_at` type (time, volume, weight), `target_value`, `current_value`, as well as the `active_state`.

An `action` is sent to indicate if and when stop-at is active near the start of a sequence.

```
class StopAtNotificationAction (enum.Enum):
    ENABLED = 'enabled'
    TRIGGERED = 'triggered'
    DISABLED = 'disabled'
    DE1CONTROLLED = 'de1 controlled'
```

When the stop-at action is controlled by the DE1, no `triggered` notification is sent.

AutoTareNotification

Sent to indicate when auto-tare is enabled and disabled by the `FlowSequencer`

```
class AutoTareNotificationAction (enum.Enum):
    ENABLED = 'enabled'
    DISABLED = 'disabled'
```

ScannerNotification

Warning: Removed in pyDE1 v2.0 see *ScanResults*

ScanResults

Starting with pyDE1 v2.0, accumulated scan results are provided during the scan, as well as an indication if the scan has completed. Scanning is done by role, such as DE1, scale, or thermometer. Only devices matching the Bluetooth advertisement filter are returned. Updates are provided as new devices are discovered, facilitating dynamic updating of a picker widget.

In response to `curl -X PUT --data 'thermometer' http://localhost:1234/scan`

```
{"arrival_time": 1675618617.1314912, "create_time": 1675618617.1314912,
    "role": "thermometer", "scanning": true,
    "devices": [],
    "version": "1.0.0", "event_time": 1675618617.1315718,
    "sender": "BluetoothScanner", "class": "ScanResults"}

{"arrival_time": 1675618617.3791888, "create_time": 1675618617.3791888,
    "role": "thermometer", "scanning": true,
    "devices": [{"address": "00:A0:50:AA:BB:CC", "name": "BlueDOT", "rssi": -72}],
    "version": "1.0.0", "event_time": 1675618617.3816643,
    "sender": "BluetoothScanner", "class": "ScanResults"}

{"arrival_time": 1675618617.8187141, "create_time": 1675618617.8187141,
    "role": "thermometer", "scanning": true,
    "devices": [{"address": "00:A0:50:AA:BB:CC", "name": "BlueDOT", "rssi": -72}],
```

(continues on next page)

(continued from previous page)

```
    "version": "1.0.0", "event_time": 1675618617.8210998,
    "sender": "BluetoothScanner", "class": "ScanResults"}

{"arrival_time": 1675618622.35119, "create_time": 1675618622.35119,
    "role": "thermometer", "scanning": false,
    "devices": [{"address": "00:A0:50:AA:BB:CC", "name": "BlueDOT", "rssi": -72}],
    "version": "1.0.0", "event_time": 1675618622.3513,
    "sender": "BluetoothScanner", "class": "ScanResults"}
```

If multiple devices had been found, they would have been added to the array of devices. `"scanning": false` indicates that the scan has completed

```
class DeviceRole (enum.Enum):
    DE1 = 'de1'
    SCALE = 'scale'
    THERMOMETER = 'thermometer'
    OTHER = 'other'
    UNKNOWN = 'unknown'
```

ConnectivityChangeNotification

Deprecated since version v2.0: Use *DeviceAvailability*

As connectivity to a DE1 or scale progresses through various states, it is reported so that an app can take action when the device is "ready", as well as change state if connectivity has degraded or been lost. (The pyDE1 core will try to reconnect, without intervention, on an unexpected disconnection.)

```
class ConnectivityState (enum.Enum):
    UNKNOWN = 'unknown'
    CONNECTING = 'connecting'
    CONNECTED = 'connected'
    READY = 'ready'  # "Ready for use"
    NOT_READY = 'not_ready'  # Was READY, but is no longer
    DISCONNECTING = 'disconnecting'
    DISCONNECTED = 'disconnected'
```

Not all states are passed through by all paths.

DeviceAvailability

In pyDE1 v2.0, the way that Bluetooth devices are handles was changed to permit a device to be "released" for other uses, then subsequently "captured". Additionally, scales change class between a generic scale and a device-specific one when captured. Watching the `role` suggests which device is changing, especially when not associated with a physical device at that moment.

```
class DeviceAvailabilityState (enum.Enum):
    INITIAL = 'initial'
    UNKNOWN = 'unknown'
    CAPTURING = 'capturing'
    CAPTURED = 'captured'
    READY = 'ready'  # "Ready for use"
```

(continues on next page)

(continued from previous page)

```
    NOT_READY = 'not ready'  # Was READY, but is no longer
    RELEASING = 'releasing'
    RELEASED = 'released'
```

Not all states are passed through by all paths.

Here is a find/capture sequence that illustrates both the availability states, as well as how the details of the scale change as it moves from a generic to a scale that is ready for use.


```
{"arrival_time": 1675619181.4547968, "create_time": 1675619181.4813273,
"state": "releasing", "role": "scale",
"id": "", "name": "GenericScale: (unknown)",
"version": "1.1.0", "event_time": 1675619181.4886014,
"sender": "GenericScale", "class": "DeviceAvailability"}

{"arrival_time": 1675619181.5112107, "create_time": 1675619181.523558,
"state": "released", "role": "scale",
"id": "", "name": "GenericScale: (unknown)",
"version": "1.1.0", "event_time": 1675619181.5262911,
"sender": "GenericScale", "class": "DeviceAvailability"}

{"arrival_time": 1675619181.5937052, "create_time": 1675619181.6589596,
"state": "initial", "role": "scale",
"id": "FF:06:AF:AA:BB:CC", "name": "AtomaxSkaleII: (unknown)",
"version": "1.1.0", "event_time": 1675619181.6753747,
"sender": "AtomaxSkaleII", "class": "DeviceAvailability"}

{"arrival_time": 1675619181.632159, "create_time": 1675619181.6696842,
"state": "capturing", "role": "scale",
"id": "FF:06:AF:AA:BB:CC", "name": "AtomaxSkaleII: (unknown)",
"version": "1.1.0", "event_time": 1675619181.6768405,
"sender": "AtomaxSkaleII", "class": "DeviceAvailability"}

{"arrival_time": 1675619183.9847703, "create_time": 1675619184.0039356,
"state": "captured", "role": "scale",
"id": "FF:06:AF:AA:BB:CC", "name": "AtomaxSkaleII: (unknown)",
"version": "1.1.0", "event_time": 1675619184.0151615,
"sender": "AtomaxSkaleII", "class": "DeviceAvailability"}

{"arrival_time": 1675619184.844756, "create_time": 1675619184.8448336,
"state": "ready", "role": "scale",
"id": "FF:06:AF:AA:BB:CC", "name": "AtomaxSkaleII: Skale",
"version": "1.1.0", "event_time": 1675619184.8514688,
"sender": "AtomaxSkaleII", "class": "DeviceAvailability"}
```

BlueDOTUpdate

Sent when a report is received from a BloeDOT thermometer.

```
self.temperature: Optional[float] = None
self.high_alarm: Optional[float] = None
self.units: str = "C"
self.alarm_byte: Optional[Union[bytearray, int]] = None
self.name: Optional[str] = None
```

The units are determined by user setting. The temperatures reported in those units, either `C` or `F`.

ScaleButtonPress

Sent when a button is pressed on scales that report such events.

ScaleTareSeen

Sent when a tare requested appears to have been fulfilled. This is usually if the weight is "close enough" to zero.

ScaleChange

In pyDE1 v2.0 and later, the scale can be changed. As different scales may have different capabilities, this provides a packet similar to the *DeviceAvailability* packet.

FirmwareUpload

Firmware upload to the DE1 is done asynchronously by pyDE1. These updates provide feedback on the progress of the upload with the `uploaded` and `total` bytes, along with the `state` of the upload.

```
class FirmwareUploadState (enum.Enum):
    STARTING = 'starting'
    UPLOADING = 'uploading'
    COMPLETED = 'completed'
    FAILED = 'failed'
    CANCELED = 'canceled'
```

Internal-Only Notifications

The following notifications are only used internally. They are not available over MQTT:

- ScaleWeightUpdate *(precursor of WeightAndFlowUpdate)*
- ShotSampleUpdate *(precursor of ShotSampleWithVolumesUpdate)*

3.3 SQLite3 Database

The SQLite3 database primarily saves profiles and data related to sequences (shots) with high fidelity. It also saves a few bits of persistent data, such as the scale-period estimates. It is not intended to be a generic settings store.

SQLite3 allows concurrent, multi-user access with WAL mode. Additional SQLite3 databases can be opened by other apps, allowing joins and other operations between them.

Note: When opening the database from another app or the command line, only use the *pyde1* user.

Although other users have read access, failing to use the *pyde1* user may result in files being created that the *pyde1* user does not have write access to. This can cause "read-only" errors, even though the *pyde1* user has write access to the main database file.

3.3.1 Backing Up

For any database, file-system backups may not result in a self-consistent snapshot. Using the database's backup utility is highly recommended.

One approach is shown in this script

```
#!/bin/sh

filename=$(date +'/home/pyde1/db_backup/pyde1.%Y-%m-%d_%H%M.sqlite3')
sqlite3 /var/lib/pyde1/pyde1.sqlite3 ".backup $filename"
xz $filename
```

This can be run periodically by *pyde1* by editing the crontab with

```
sudo -u pyde1 crontab -e
```

to add a line like

```
00 03 * * * /home/pyde1/bin/pyde1-backup.sh
```

(Every day at 0300, local time)

3.3.2 Schema

The schema version is available as `PRAGMA user_version`. The schema itself is distributed at `pyDE1/database/schema`.

Times are real values, such as would be returned by Python `time.time()`

Sequences

Most of the database is dedicated to capturing the notifications that are generated immediately before and during a `FlowSequencer` sequence.

There is a rolling buffer that captures the most recent notifications that then gets written to the database shortly after the start of a sequence.

Each sequence has a "master record" that is created at the start of the sequence. Some fields are updated, such as times, are updated as the sequence progresses. As this is done asynchronously, this record may not be complete the instant the sequence ends.

```
CREATE TABLE sequence (
    id              TEXT NOT NULL PRIMARY KEY,
    active_state    TEXT,
    start_sequence  REAL,
    start_flow      REAL,
    end_flow        REAL,
    end_sequence    REAL,
    profile_id      TEXT NOT NULL REFERENCES profile(id),
    -- https://www.sqlite.org/quirks.html#no_separate_boolean_datatype
    profile_assumed INTEGER, -- will match TRUE and FALSE keywords
    resource_version                            TEXT,
    resource_de1_id                             TEXT,
    resource_de1_read_once                      TEXT,
    resource_de1_calibration_flow_multiplier    TEXT,
    resource_de1_control_mode                   TEXT,
    resource_de1_control_tank_water_threshold   TEXT,
    resource_de1_setting_before_flow            TEXT,
    resource_de1_setting_steam                  TEXT,
    resource_de1_setting_target_group_temp      TEXT,
    resource_scale_id                           TEXT
);
```

Virtually all of the MQTT notifications are captured and associated with the `sequence.id` to allow for recreation of the data during the shot. As an example

```
CREATE TABLE shot_sample_with_volume_update (
    sequence_id         TEXT NOT NULL REFERENCES sequence(id),
    version             TEXT,
    sender              TEXT,
    arrival_time        REAL,
    create_time         REAL,
    event_time          REAL,
    --
    de1_time            REAL,
    --
    sample_time         INTEGER,
    group_pressure      REAL,
    group_flow          REAL,
    mix_temp            REAL,
    head_temp           REAL,
    set_mix_temp        REAL,
    set_head_temp       REAL,
    set_group_pressure  REAL,
```

(continues on next page)

(continued from previous page)

```
    set_group_flow      REAL,
    frame_number        INTEGER,
    steam_temp          REAL,
    --
    volume_preinfuse    REAL,
    volume_pour         REAL,
    volume_total        REAL,
    volume_by_frames    TEXT    -- Python list, default formatting
);
```

Profiles

Profiles, uploaded through the HTTP API, get stored in the database, along with their metadata. They are referenced by a unique ID over the uploaded content, as well as indexed by a `fingerprint` of the frames that would be delivered to the DE1. The same `fingerprint` is the same for the DE1, but would be from different source data or have different metadata.

```
CREATE TABLE profile (
    id              TEXT NOT NULL PRIMARY KEY,
    source          BLOB NOT NULL,
    source_format   TEXT NOT NULL,
    fingerprint     TEXT NOT NULL,
    date_added      REAL,
    title           TEXT,
    author          TEXT,
    notes           TEXT,
    beverage_type   TEXT
);
```

persist_hkv

This is a small table used internally to persist time-varying data across restarts of pyDE1 or connection and disconnection of devices. It should be considered opaque and not part of the supported API.

3.4 JSON Profile Spec

`profile_json.d.ts`

```
/** Copyright © 2023 Jeff Kletsky. All Rights Reserved.
 *
 * License for this software, part of the pyDE1 package, is granted under
 * GNU General Public License v3.0 only
 * SPDX-License-Identifier: GPL-3.0-only
 */

export const VERSION = '2.1';
export const SPEC_REVISION = '2.1';
```

(continues on next page)

(continued from previous page)

```
/** This file has a primary purpose of documenting the pyDE1 profile format
 *
 *  Credit and appreciation to Mimoja for the initial implementation
 *  of JSON profiles in the de1app (TCL).
 *
 *  This definition removes most redundant and contextually meaningless
 *  information in the TCL representation of Buckman. Buckman's implementation
 *  tied the data structure and naming to the UI implementation in de1app.
 *  This implementation retains some legacy fields that de1app apparently needs
 *  to be able to select an editor for a profile. It also extends
 *  Mimoja's implementation with additional metadata related to
 *  the source of the profile and how and when it was created.
 *
 *  Implementations MAY be tolerant of the presence of extra fields
 *  that may have been written by other tools.
 *
 *  TypeScript was selected as a reasonable documentation format.
 *  This document has not been validated in a TypeScript project.
 *
 *  Limitations on the range of numeric values is not specified here.
 *
 *  Please report any issues or points that need clarification.
 */

/** NB: TCL does not discriminate between the number 5.3 and the string "5.3".
 *      As a result, JSON written by de1app writes a string rather than a number.
 *
 *      Implementations that read de1app-generated JSON profiles MAY be robust
 *      to this representation. However, any JSON written SHOULD be compliant
 *      with JSON standards and represent numbers as numbers, not strings.
 */

export type NonEmptyArray<T> = [T, ...T[]];

export type PumpType = 'pressure' | 'flow';
export type MoveOnType = 'seconds' | 'volume' | 'weight';
export type ExitType = 'pressure' | 'flow';
export type ExitCondition = 'over' | 'under';
export type TransitionType = 'fast' | 'smooth';
export type TemperatureSensor = 'coffee' | 'water';

/** A loose labeling of the general category of the intent of the profile.
 *  Potentially can be extended by users. Primarily used for skipping upload
 *  to Visualizer and for categorization in DYE, Visualizer, and others. */
export type BeverageType =
    'espresso'
    | 'calibrate'
    | 'cleaning'
    | 'manual'
    | 'pourover'
    | 'tea_portafilter';
```

(continues on next page)

(continued from previous page)

```
/** Buckman tied profile type to the name of the UI screen
 *  on which it was edited. See also type ProfileEditor */
export type LegacyProfileType =
    'settings_2a'
    | 'settings_2b'
    | 'settings_2c'
    | 'settings_2c2';

/** A normalization of LegacyProfileType.
 *  Does not impact how the DE1 operates. */
export type ProfileEditor = 'advanced' | 'flow' | 'pressure';

export type ISOTimestamp = string
export type SemanticVersion = string

export interface ProfileJSON {
    /** Identifies the semantic version of the JSON format, presently 2.1 */
    version: SemanticVersion;
    /** A one-line string that can identify the profile to a user */
    title: string;
    /** A longer description of the profile and/or its use
     *  that can be multi-line */
    notes: string;
    /** The author of the profile.
     *  NB: "Decent" is likely inappropriately present
     *  for user-generated profiles */
    author: string;
    /** A general category for the beverage or function.
     *  Used by DYE, Visualizer uploaders, and others */
    beverage_type: BeverageType;
    /** An array of one or more steps or frames describing the actions
     *  the DE1 should take */
    steps: NonEmptyArray<ProfileStep>
    /** If non-zero and non-null, the estimated volume
     *  at which the DE1 should stop */
    target_volume: number | null;
    /** If non-zero and non-null, the weight
     *  at which the DE1 should be stopped */
    target_weight: number | null;
    /** An integer indicating the zero-based frame number after which to start
     *  the "pour" accounting of time and volume */
    target_volume_count_start: number;
    /** If non-zero, the target temperature in °C to which the tank
     *  should be heated prior to starting the frames. */
    tank_temperature: number;
    /** Legacy field from de1app profiles, seemingly all contain "en" */
    lang: string;
    /** Legacy identifier in the de1app */
    legacy_profile_type: LegacyProfileType;
    /** Legacy style of de1app editor to display or edit the profile */
    type: ProfileEditor;
    /** A reference to the source of the profile.
```

(continues on next page)

(continued from previous page)

```
     *  The field is present but often the empty string from de1app */
    reference_file: string;
    /** An optional descriptor of how and when this version was generated */
    creator?: CreatorData;
}

export type ProfileStep = ProfileStepPressure | ProfileStepFlow;

export interface ProfileStepBase {
    /** A label suitable for rendering in a UI */
    name: string;
    /** How the target over time should move from the controlled variable
     *  at the start of the frame (at run time) to the target for this frame */
    transition: TransitionType;
    /** An optional exit condition based on flow or pressure */
    exit?: StepExitConditition;
    /** The volume in mL dispensed in this fram over which
     *  the frame would be exited, */
    volume: number;
    /** The duration of this frame over which the frame would be exited */
    seconds: number;
    /** If present, the weight in g over which the frame would be exited */
    weight?: number;
    /** The target temperature in °C for this frame. See also sensor: */
    temperature: number;
    /** The sensor to use to measure temperature for this frame */
    sensor: TemperatureSensor;
}

export interface ProfileStepPressure extends ProfileStepBase {
    /** Defines this as a pressure-driven step */
    pump: 'pressure';
    /** The target pressure in bar */
    pressure: number;
}

export interface ProfileStepFlow extends ProfileStepBase {
    /** Defines this as a flow-driven step */
    pump: 'flow';
    /** The target flow in mL/s */
    flow: number;
}

export interface StepExitConditition {
    /** Exit based on flow or pressure. Omit StepExitCondition otherwise */
    type: ExitType;
    /** Is the exit to occur when the measured value crosses the threshold
     *  from above or from below. */
    condition: ExitCondition;
    /** The numeric threshold */
    value: number;
}
```

(continues on next page)

(continued from previous page)

```
/** See current DE1 documentation on how the Limiter parameters impact operation.
 *  At least at this time, pressure-driven and flow-driven profiles
 *  behave differently. The description is the same in the profile for both. */
export interface Limiter {
    value: number;
    range: number
}

export interface CreatorData {
    /** A reference to the product name or utility */
    name: string;
    /** An identifier string of the version of the product or utility
     *  Although semantic versioning is preferred, it is not required. */
    version: string;
    /** An ISO timestamp of when the conversion was performed.
     *  Should include full date, time with at least seconds, and timezone */
    timestamp: ISOTimestamp;
}
```

CHAPTER

FOUR

SECURITY

4.1 General

You are responsible for your own security. This document does not warrant any specific level of security, nor does it cover all possible security-related issues. At best, it highlights some of the things you should be considering for any software and OS.

Some general best practices:

- Keep your OS patched for all security issues, and update packages promptly.
- Run only the minimal set of programs and services
- Perform all operations with the lowest level of privilege for the task
- Blocking all and permitting selected is often stronger than permitting all and blocking known threats
- Maintain strong authorization credentials

4.2 TLS Certificates

TLS certificates, in the context of pyDE1 and allied services, generally provide two functions, to confirm the identity of the service to a caller and to set up an encrypted connection. What is now called TLS was previously known as SSL and many sites and programs still refer to it as such.

Warning: The private portion of these certificates should be considered as very sensitive data. Should they be used in an unauthorized setting, any device that "trusts" the certificate, directly or through its signature, would believe the imposter.

4.2.1 Verifiable Certificates Strongly Suggested

If you have your own domain, using a recognized Certificate Authority (CA) with a revocation list is the preferred way to obtain certificates. Let's Encrypt is one service that can provide host-specific certificates at no cost. These certificates can both be used to set up TLS connections, as well as verify the identity of the host to which you're connecting.

4.2.2 Self-Signed Certificates

Note: I do not advocate use of self-signed certificates over verifiable certificates

If you do not have your own domain, many people use "self-signed" certificates to set up TLS connections, but are dicey, at best, to verify identity. Most modern browsers will raise a security warning with a self-signed certificate. How you and those around you respond to those dialogs will impact the level of risk involved with "accepting" the certificate. Most browsers will let you examine the certificate before taking action. I strongly suggest doing so and confirming that the certificate's details are what you expect. I can't comment further on the best ways to handle these certificates. The risks likely vary by OS and the level of trust you grant your system and apps for each certificate.

There are at least two ways of generating self-signed certificates. One is to create your own, "trusted" CA, and have it sign certificates for various uses. One can distribute the public key of the CA to all of your client devices and "trust" that certificate. When one of your signed certificates is presented by a service, the trust of the CA extends to the cert of the service. For devices on which you haven't installed the CA public key, it will appear as invalid, as there is no trust. Overviews of this approach can be found in many places. One can be found at MariaDB.

Stand-alone, self-signed certificates are another option. With these, there is nothing to "trust" other than the certificate presented by the service using it. They seem very popular with stand-alone IoT devices and networking hardware. One source of a command line to generate a self-signed certificate is adapted[1] from StackOverflow

```
openssl req -x509 -newkey rsa:4096 -keyout privkey.pem -out cert.pem -sha256 -days 365 -
↪nodes
```

which will shortly prompt you for information that will be part of the certificate, and generally visible in the "details" when examined in a browser.

I suggest making them recognizable to yourself, as well as distinct for each certificate you create.

One suggestion is to set:

- *Organization Name* – your name
- *Organizational Unit Name* – something related to the service
- *Common Name* – The fully-qualified name of the host, or the IP address, if you're unable to set up local DNS
- *Email Address* – I often leave it blank

Many home routers will let you reserve an IP address for a specific host. Often a hostname can be assigned for the DNS that the router provides.

4.2.3 Certificates and `mosquitto`

Many services that employ sensitive data, such as TLS certificates, read that data when they first start (as *root)* and then drop privilege before starting the service. This allows the sensitive data to be readable only by *root* and not readable by the unprivileged user that the service runs as.

For some reason, current versions of mosquitto no longer take this approach. The certificates, including the private portions, need to be readable by the *mosquitto* user.

[1] The private key has been named to be consistent with Let's Encrypt naming.

4.3 Firewalls and Networking

If you're reading this to determine how to access pyDE1 from the open Internet, you'll have to find other resources on that. To be very clear, even with `nginx` reverse proxying the Python HTTP server and running `mosquitto`, it is not secure. This is one of those "If you have to ask …" things.

One piece of firewall worth noting is that `mosquitto` apparently can't restrict the *websockets* listener to the localhost interface.

CHAPTER

FIVE

QUICK START ON RASPBERRY PI

5.1 Overview

Although pyDE1 can be used on most any, current Linux-based OS and potentially macOS, the Raspberry Pi provides a compact, budget-friendly platform for pyDE1 and the related services. This page describes some of the options for hardware and outlines and links to the various steps generally needed to bring up a ready-to-pull system.

5.2 Hardware

A multi-core Raspberry Pi is recommended. The ones generally available at the start of 2022 include:

- Raspberry Pi 3B+
- Raspberry Pi Zero 2
- Raspberry Pi 4B

Any of these should be sufficient to run pyDE1, the web server, the MQTT server, serve pages for a web-based UI, and ancillary programs, such as uploading to Visualizer or steam-to-temperature.

Any of the distributors listed on the Raspberry Pi site are likely reputable and less expensive than going through third-party sales, such as eBay. If in the US, pishop.ca may be worth checking as they ship to the US and seem to be restocked on a different schedule than pishop.us.

5.2.1 Accessories

Often worth ordering with the Pi are any special adapters you might need to connect a keyboard or display.

- 3B+ (standard HDMI, standard USB A)
- Pi Zero 2
  - Mini-HDMI to something you have
  - USB-OTG (micro B to A female)
- Pi 4B (standard USB A)
  - Micro-HDMI to something you have

The "official" Raspberry Pi cables and power supplies aren't a lot more than the generic ones. They *might* be better quality.

The 3B+ and Zero 2 can be run off a good-quality, USB "phone" charger that can reliably supply 2.4 A. The 4B needs a USB-C supply of at least 15 W capacity.

If the case you choose doesn't come with heat sinks, they're usually not very expensive and hopefully won't bump up the shipping costs.

5.2.2 microSD Cards

I usually buy my cards somewhere other than where I buy my hardware. A while back, they introduced "A" ratings for microSD cards, for "Application" (phone) use. Cameras tend to occasionally write and read big things, where applications tend to have much smaller and more frequent reads and writes. I'd suggest at least an A1 rating (in addition to Class, U, and V ratings). Speed isn't too much of an issue, but longevity is, at least for me, worth a couple extra dollars. A 32 GB card should be plenty. 64 GB cards are probably not much more expensive.

5.2.3 Cases

I've been using C4Labs cases for many years. I think they're some of the best out there. They are laser-cut acrylic or wood, not the cheap moulded junk. He's a small business in Washington State, US, and I see that he has distribution through at least Pi Hut-UK. I believe they all include heat sinks.

For the Pi Zero 2, I like the Zebra Zero Heatsink Case (get the Zero 2 version) or, if you want access to the pin header, the Zebra Zero 2 Heatsink and GPIO Access Case.

The Pi 3B+ I have is in a solid-top Zebra Classic Case

For the Pi 4B I have, I went with the Zebra Bagel Fan Case, which includes a fan at that price. The fan is quiet enough on 3.3 V to *not* buy a Noctua, from someone who trades out fans on just about everything. There are several other options, but I trusted the advice I got that the fan was a good idea for the power dissipation of the 4B.

5.3 Installation Outline

This section outlines the installation steps described on other pages, either here, or with the documentation for the UI or other software.

- *Raspberry Pi OS*
  - Download installer, install image, preconfigured for WiFi and ssh access
  - Boot image and update OS
  - Change from `pi` user to a name of your choice, secure `sudo`
  - Add yourself to the `bluetooth` group
  - Install `python3-venv` and some utility packages
  - Potentially tweak WiFi for stability
- *Web Server*, `nginx`
  - Install using `apt`
  - Configure for reverse-proxying of pyDE1
  - Configure for reverse-proxying of websockets
  - Configure for GUI
  - Potentially configure TLS certificates
- *MQTT Broker*, `mosquitto`
  - Install using `apt`

  - Configure for MQTT (for applications) and websockets (for browser)
  - Potentially configure TLS certificates
  - Modify firewall to block off-host access to websockets
  - Select and configure passwords and access control lists
  - Configure logging (so it's more human-readable)
- *Install pyDE1 and Visualizer uploader* along with it
  - Script to create the `pyde1` user
  - Script to create the expected directories
  - Script to create and populate a Python virtual environment ("venv")
  - Script to move the config files into `/usr/local/etc/pyde1/`
  - Edit the config files to suit (remember your user names and passwords)
  - Script to enable pyDE1 and Visualizer uploader at boot
  - Configure log rotation
- Install your choice of UIs
- Enjoy a coffee

CHAPTER

SIX

RASPBERRY OS INSTALL TIPS

Raspberry OS images (formerly "Raspbian") can be obtained from the Raspberry Pi OS page. These are still 32-bit images, so should run on all models. For the purposes of these hints, the "lite" image is used. Those with a graphical desktop and with additional software should configure in the same way.

In the past one would have had to jump through some hoops to enable SSH and WiFi on first boot. Now, the Raspberry Pi Imager makes this a lot easier, as well as "fixing" puzzling password problems because it came configured for a GB keyboard and you then changed it to your own.

On macOS, you can run it directly from the mounted DMG file, without dragging it to your Applications folder.

Before you write the image, open the hidden advanced-settings screen with cmd+shift+x for macOS or ctrl+shift+x other OSes. You can now preconfigure networking with relative ease.

Make sure you set the WiFi country, or it may not be able to connect.

- Enable SSH
  - Use password authentication
    - Set password for 'pi' user: <enter your new password here>
- Configure wifi
  - SSID: <the name of your access point>
  - Password: <the password for your access point>
  - Wifi country <select your country code>
- Set locale settings
  - Time zone: <select yours>
  - Keyboard layout: <select yours>

Make other changes you might want, then click `SAVE`

As I don't use the Raspberry Pi with a graphical desktop, the Lite image is sufficient

> Operating System > Raspberry Pi OS (other) > Raspberry Pi OS Lite

As it will be writing to a "raw" device (the microSD), your OS may ask for permissions to write to removable media or similar.

6.1 Update OS

Note: See *Raspberry Pi 4 and BLE*, below, before updating if you have a Pi 4

Do this periodically

```
pi@raspberrypi:~ $ sudo apt update
pi@raspberrypi:~ $ sudo apt upgrade
pi@raspberrypi:~ $ sudo reboot
```

6.2 Require a Password for sudo

Requiring a password probably makes things a bit more secure.

Confirm if pi (or your user) is in the *sudo* group first.

```
pi@raspberrypi:~ $ id
uid=1000(pi) gid=1000(pi) groups=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),
↪29(audio),44(video),46(plugdev),60(games),100(users),105(input),109(netdev),997(gpio),
↪998(i2c),999(spi)
```

If not, `sudo usermod -a -G sudo pi` and recheck.

Confirm that the *sudo* group is configured for access with `%sudo ALL=(ALL:ALL) ALL`

```
pi@raspberrypi:~ $ sudo fgrep sudo /etc/sudoers
# This file MUST be edited with the 'visudo' command as root.
# Please consider adding local content in /etc/sudoers.d/ instead of
# See the man page for details on how to write a sudoers file.
# Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@includedir /etc/sudoers.d
```

Then cross your fingers and comment out the only line in `sudo visudo /etc/sudoers.d/010_pi-nopasswd`

```
# pi ALL=(ALL) NOPASSWD: ALL
```

6.3 Change pi to Something Better

I can make a security argument here for changing the "main" user name, but I'll admit that convenience for me is a bigger driver. (This is an optional, though recommended step.)

There are four files that map user and group names to numbers and then the name of the "home" directory in one of those. It's definitely possible to botch this and get locked out. That's why I do it before there is much time invested. Easier to re-image than to try juggling things.

Warning: This needs to be done in one `sudo` session, as there are times when things are inconsistent and you may not be able to authenticate.

```
pi@raspberrypi:~ $ sudo bash
[sudo] password for pi:
root@raspberrypi:/home/pi# cd /home
root@raspberrypi:/home# ln -s pi jeff
root@raspberrypi:/home# ls -l
total 4
lrwxrwxrwx 1 root root    2 Nov 18 20:51 jeff -> pi
drwxr-xr-x 2 pi   pi   4096 Nov 18 21:09 pi
```

`vipw` and change `pi` to `jeff` (or whatever) in the two places in the line

```
pi:x:1000:1000:,,,:/home/pi:/bin/bash
```

`vipw -s` and change `pi` to `jeff` in the one place in the line

```
pi:$5$a_bunch_of_apparently_random_characters:18930:0:99999:7:::
```

`vigr` and change `pi` to `jeff` in lines like

```
sudo:x:27:pi
pi:x:1000:
```

(Don't change `spi` or `gpio` or similar)

`vigr -s` and change `pi` to `jeff` similarly

Then complete things by renaming the home directory

```
root@raspberrypi:/home# rm jeff
root@raspberrypi:/home# mv pi jeff
root@raspberrypi:/home# exit
exit
pi@raspberrypi:~ $ whoami
jeff
pi@raspberrypi:~ $ exit
logout
```

Next time you log in, log in as your new user name (with the same password)

6.4 Add Yourself to *bluetooth* Group

```
jeff@pi-walnut:~ $ sudo usermod -a -G bluetooth jeff
```

That way you can run `bluetoothctl` without elevated privilege.

6.5 Install python3-venv

The Python module to create virtual environments is not installed in the base image. If `$ dpkg --get-selections | fgrep python3` does not list `python3-venv`, install it with

```
sudo apt install python3-venv
```

There's no "harm" in installing it if it is already there. `apt` would mark it as "manually installed" if it was previously installed as a dependency.

6.6 Utilities and Packages I Often Use

```
sudo apt install git ldnsutils locate sqlite3
```

`ldnsutils` provides `drill`, which I find useful to query DNS

`locate` is a quick, file-name search across the entire system that is helpful for *"Where are the .service files again?"* and the like.

`htop` is already installed with the Raspberry OS image and is a more fully featured monitoring tool than `top` without getting into huge number of packages that something like `glances` brings in.

6.7 Timekeeping

TL;DR

Unless you're concerned about tens of milliseconds, skip installing `ntp`, stick with the default, but disable `dhcpcd` from restarting it on every DHCP renewal.

The default DHCP client, `dhcpcd` is configured to restart the timekeeping utility on every lease renewal. Depending on how your router or DHCP server is set up, this might be every few minutes. This can limit the ability to get a good estimate of time, as well as causing log spam.

For most people that aren't moving their computer from network to network without rebooting it, there is little reason to restart timekeeping with each DHCP renewal. The "hooks" that do this can be disabled by adding to the end of `/etc/dhcpcd.conf`

```
# Additions start here

nohook hostname ntp-common.conf chrony.conf timesyncd.conf ntp.conf openntpd.conf
```

The above list comes from examining the hooks in `/lib/dhcpcd/dhcpcd-hooks`. Setting of hostname was also disabled, as it is often "permanently" configured in `/etc/hostname` and reflected in `/etc/hosts`.

`ntp` installs a more sophisticated time-keeping package than the default. I believe it is more accurate than the default `systemd-timesyncd`. `systemd-timesyncd` apparently has the advantage of persisting the last-known time to disk and restoring it at boot. This is helpful for machines that do not have a real-time clock (RTC) that survives without power, such as on the Raspberry Pi boards. It has a disadvantage of only using a single time server, without the set of algorithms of NTP to estimate and stabilize the clock from multiple sources. The accuracy you get with `systemd-timesyncd` will depend on which server gets randomly selected and "Internet weather".

6.8 WiFi Dropouts

My Pi Zero 2 seems to randomly drop off WiFi, even with an ssh session open. There are suggestions that WMM or Fast Roaming are problematic, as well as power control. WMM is primarily related to QoS, but there is a *Power Save Certification* as well. Reflecting on it, the 3B+ may also have some issues.

Guessing that power control in the Pi is at the core of the problem, especially as it is within a couple meters of the AP and it doesn't seem to impact other devices on the network

```
$ iwlist wlan0 power
wlan0     Current mode:on

$ sudo iwconfig wlan0 power off

$ iwlist wlan0 power
wlan0     Current mode:off
```

seems to have resolved it. One way to make the change permanent is to create `/etc/systemd/system/wlan0_power_mgmt_off.service` containing

```
[Unit]
Description=Disable power-save on wlan0
After=sys-subsystem-net-devices-wlan0.device

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/sbin/iwconfig wlan0 power off

[Install]
WantedBy=sys-subsystem-net-devices-wlan0.device
```

and enable it with `sudo systemctl enable wlan0_power_mgmt_off.service`

Unit file after https://raspberrypi.stackexchange.com/questions/96606/make-iw-wlan0-set-power-save-off-permanent

6.9 Raspberry Pi 4 and BLE

Important: This applies to the Raspberry Pi 4B only

It appears that the current (January 2022) Raspberry Pi OS updates break the Bluetooth LE functionality on a Raspberry Pi 4B. So far, it has not been seen on a 3B+ or Zero 2, only the 4B. The symptoms are that a device connects, then immediately disconnects. This can be seen using *bluetoothctl* so is at the OS level, long before bleak or pyDE1 come into play.

Though not confirmed or resolved by the OS maintainers, it seems to be a conflict between the Broadcom WiFi and Bluetooth firmware.

If you have already updated, downgrading the WiFi firmware seems to resolve the issue. Generally available should be the same version presently being used in Debian

```
sudo apt install firmware-brcm80211=20210315-3
```

This version can be "held" so that `apt upgrade` will not automatically replace it with

```
sudo apt-mark hold firmware-brcm80211
```

It is possible that the version presently being installed by the image flasher is operational as well. Version `1:20210315-3+rpt2` does not appear to be readily available from the Raspberry Pi repos, but could be held after installation as indicated above. It has not been investigated if there are any differences in the `+rpt4` version that apply to the specific chips used for the 4B.

6.10 Developers' Sidebar – Using pip and VCS

To be able to test out the sufficiency of the package and the installation instructions, I didn't want to "publish" a package to PyPi that was either incomplete or broken.

There is VCS Support for pip that allows an install to be done "on the fly" from various VCS systems, or a file system.

For my configuration, the following worked

```
(test-pip-vcs) jeff@pi-walnut:~ $ pip install git+ssh://jeff@my.example.com/full/path/to/
↪pyDE1.git@test#egg=pyDE1
(test-pip-vcs) jeff@pi-walnut:~ $ pip list
Package            Version
------------------ ---------
aiosqlite          0.17.0
bleak              0.13.0
certifi            2021.10.8
charset-normalizer 2.0.7
dbus-next          0.2.3
idna               3.3
paho-mqtt          1.6.1
pip                20.3.4
pkg-resources      0.0.0
pyDE1              0.9.1
PyYAML             6.0
requests           2.26.0
setuptools         44.1.1
typing-extensions  4.0.0
urllib3            1.26.7
```

Other approaches are outlined at https://packaging.python.org/tutorials/installing-packages/#installing-from-a-local-src-tree

CHAPTER

SEVEN

CONFIGURING NGINX

7.1 Overview

`nginx` is a web server that is widely used in production environments. An unfortunate effect of serving this market is that it is non-trivial to configure, due to the great flexibility.

In this example configuration, it is used as a reverse proxy for the Python HTTP server and the websockets interface provided by `mosquitto`. It also demonstrates the kind of configuration that a web app might use for static files and dynamic content with uWSGI.

> **Warning:** Security should not be taken lightly. Even when "only" exposed on the loopback interface, any service poses a vector for attack.
>
> Although this example shows configuration for TLS, this secures the data while in flight and not the service. Users must determine and implement appropriate security for their needs.

7.2 Installing nginx

```
apt install nginx
```

7.3 Example nginx Configuration

The default `/etc/nginx/nginx.conf` includes a directive to read configuration from `/etc/nginx/conf.d/`. However, a couple of lines in the Debian Bullseye package's version can't be easily overridden. To minimize and localize the changes, making future, nginx upgrades easier, these lines are commented out as shown in this `diff` output.

```
--- nginx.conf.orig 2021-09-06 09:16:12.021343622 -0700
+++ nginx.conf      2021-08-30 21:30:36.611817810 -0700
@@ -29,15 +29,15 @@
    # SSL Settings
    ##

-   ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
-   ssl_prefer_server_ciphers on;
+#  ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
+#  ssl_prefer_server_ciphers on;
```

(continues on next page)

(continued from previous page)

```
    ##
    # Logging Settings
    ##

-   access_log /var/log/nginx/access.log;
-   error_log /var/log/nginx/error.log;
+#  access_log /var/log/nginx/access.log;
+#  error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
```

7.3.1 General Configuration

The first set of configuration is read into the `http` block of `nginx.conf`, before "sites" are read. It applies to all instances. From the end of the `http` block in `nginx.conf`:

```
include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
```

It is shown here as snippets in multiple files within `/etc/nginx/conf.d/` which may be easier to manage using `git` or another VCS. The snippets of the general configuration could be combined into a single file, such as `/etc/nginx/conf.d/local.conf` or otherwise arranged as makes sense to you.

Enable On-the-Fly Compression

Modern browsers can decompress content as it receives it. Compression can save transmission time, improving overall response time on lower bandwidth connections. This section enables on-the-fly compression at the server. This includes, for example, large data sets for plotting of history.

`conf.d/gzip.conf`

```
# gzip on;  # Declared on in nginx.conf
gzip_vary on;
gzip_proxied any;
gzip_comp_level 6;
gzip_buffers 16 8k;
gzip_http_version 1.1;
gzip_types text/plain text/css application/json application/javascript
           text/xml application/xml application/xml+rss text/javascript;
```

Adjust Logging

These changes modify the logging format from the "CLF" to one with a bit more information. Note that the error log's format can't be overridden.

conf.d/logging.conf

```
# http://nginx.org/en/docs/http/ngx_http_log_module.html#log_format

log_format  main_rt  '$remote_addr - $remote_user [$time_local] '
       '"$scheme://$host" "$request" '
       '$status $body_bytes_sent "$http_referer" '
       '"$http_user_agent" "$http_x_forwarded_for" '
       '${request_time}s $sent_http_content_type';

# http://nginx.org/en/docs/http/ngx_http_log_module.html#access_log
# http://nginx.org/en/docs/ngx_core_module.html#error_log

access_log  /var/log/nginx/access.log  main_rt;
error_log  /var/log/nginx/error.log;   # Can't set format, see above
```

Set DNS Resolvers

For *nginx* to be able to locate the servers that it is proxying, it needs DNS resolvers. It does not use the OS's notion of resolvers. These should be set to your *local* resolvers or other resolvers that are always available.

conf.d/resolvers.conf

```
# replace with the IP address of your resolver(s)
resolver 192.168.1.1;
# resolver 192.168.1.1 192.168.1.2;
```

Reverse Proxying, General Configuration

Some of the headers added here may only be of interest if you have another instance of *nginx* running behind your first-contact instance. They inform the proxyed server of where the original request came from, which otherwise would appear to come from this `nginx` instance, rather than the remote browser.

Websockets need some special configuration, as described at http://nginx.org/en/docs/http/websocket.html

conf.d/reverse_proxy.conf

```
#
# Setup for reverse proxy
#

# https://www.nginx.com/resources/wiki/start/topics/examples/forwarded/
# talks about the RFC 7239 Forwarded header, but there's no built-in yet
# also warnings about https://trac.nginx.org/nginx/ticket/1316

proxy_http_version 1.1;

# http://nginx.org/en/docs/http/ngx_http_realip_module.html
```

(continues on next page)

(continued from previous page)

```
# "Should an upstream server be able to set the IP?"
# Here, no. This is the first point of contact

map $http_x_forwarded_proto $xfp_set_if_unset {
    ''      $scheme;
    default $http_x_forwarded_proto;
}

# http://nginx.org/en/docs/http/websocket.html

map $http_upgrade $connection_upgrade {
    ''      close;
    default upgrade;
}

proxy_set_header        Host $host;
proxy_set_header        X-Real-IP $remote_addr;
proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header        X-Forwarded-Proto $xfp_set_if_unset;

proxy_set_header      Upgrade $http_upgrade;
proxy_set_header      Connection $connection_upgrade;
```

Enable Keep-Alive

Current defaults of 75 seconds and `tcp_nodelay on` seem sufficient. No change appears to be required.

http://nginx.org/en/docs/http/ngx_http_core_module.html#keepalive_timeout http://nginx.org/en/docs/http/ngx_http_core_module.html#tcp_nodelay

Redirect HTTP to HTTP-S

As `conf.d/tls.conf`, described later, enables HSTS for all sites, the redirect should be for all sites as well.

`conf.d/redirect_tls.conf`

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    location / {
            return 301 https://$host$request_uri;
    }
}
```

Configure TLS

> **Warning:** Users should evaluate and make their own decisions around TLS and other security questions.
>
> These configurations are provided as examples.

Mozilla maintains and has been periodically updating configuration suggestions at https://wiki.mozilla.org/Security/Server_Side_TLS

The configurations below are based on the generator at https://ssl-config.mozilla.org/

These were last examined January, 2022.

Note: Recommendations change with time, sometimes quickly when vulnerabilities are discovered.

It is strongly suggested that users periodically check for and implement changes in security recommendations.

Using Let's Encrypt Certificates, TLSv1.3 Only, OCSP Stapling, and HSTS

This configuration requires control over a public domain name to be able to generate the Let's Encrypt certificates.

TLSv1.3 should be supported by up-to-date devices, but may not be supported by older OS installs, such as Android without an updated browser. Anything older than TLSv1.2 should be considered as insecure. For an application like this with clients typically under your control, if TLSV1.3 isn't supported, you probably should update the device's software.

> Supports Firefox 63 *(2018)*, Android 10.0 *(2019)*, Chrome 70 *(2018)*, Edge 75 *(2019)*, … , and Safari 12.1 *(2019)*

OCSP (Online Certificate Status Protocol) only makes sense for publicly verifiable certificates.

HSTS (HTTP Strict Transport Security) informs browsers that the site should only be accessed using HTTP-S. As it is cached by the browser, you may want to confirm that the redirect from HTTP to HTTP-S is working properly before enabling it.

`tls.conf`

```
# https://wiki.mozilla.org/Security/Server_Side_TLS
# "nginx 1.18.0, modern config, OpenSSL 1.1.1k
#  Supports Firefox 63, Android 10.0, Chrome 70, Edge 75, Java 11,
#  OpenSSL 1.1.1, Opera 57, and Safari 12.1"
# generated 2022-01-22, Mozilla Guideline v5.6, nginx 1.18.0, OpenSSL 1.1.1k, modern␣
↪configuration
# https://ssl-config.mozilla.org/#server=nginx&version=1.18.0&config=modern&openssl=1.1.
↪1k&guideline=5.6

ssl_certificate      certs/fullchain.pem;
ssl_certificate_key  certs/privkey.pem;
ssl_session_timeout  1d;
ssl_session_cache    shared:MozSSL:10m;  # about 40000 sessions
ssl_session_tickets  off;

ssl_protocols TLSv1.3;
```

(continues on next page)

(continued from previous page)

```
ssl_prefer_server_ciphers off;

# HSTS (ngx_http_headers_module is required) (63072000 seconds, two years)
add_header Strict-Transport-Security "max-age=63072000" always;

# OCSP stapling
ssl_stapling on;
ssl_stapling_verify on;

# verify chain of trust of OCSP response using Root CA and Intermediate certs
# https://community.letsencrypt.org/t/howto-ocsp-stapling-for-nginx/13611/5
#   "You need to set the ssl_trusted_certificate to chain.pem
#    for OCSP stapling to work.
ssl_trusted_certificate certs/chain.pem;
```

Using Let's Encrypt Certificates, TLSv1.3 and TLSv1.2, OCSP Stapling, and HSTS

Warning: TLSv1.2 is not considered as secure as TLSv1.3.

Unless you've got some "ancient" software that can't be upgraded, TLSv1.2 is not recommended for use where you've got control over important clients.

This configuration relaxes the requirement for TLSv1.3 and permits TLSv1.2. At this time, anything older than TLSv1.2 is no longer recommended. Should you have a need for older protocols, please consult the Mozilla references or other sources directly.

Note: Configuration for TLSv1.2 requires Diffie-Hellman parameters

```
curl https://ssl-config.mozilla.org/ffdhe2048.txt > /etc/nginx/ffdhe2048.txt
```

tls.conf

```
# https://wiki.mozilla.org/Security/Server_Side_TLS
# "nginx 1.18.0, intermediate config, OpenSSL 1.1.1k
#  Supports Firefox 27, Android 4.4.2, Chrome 31, Edge, IE 11 on Windows 7,
#  Java 8u31, OpenSSL 1.0.1, Opera 20, and Safari 9"
# generated 2022-01-22, Mozilla Guideline v5.6, nginx 1.18.0, OpenSSL 1.1.1k,␣
↪intermediate configuration
# https://ssl-config.mozilla.org/#server=nginx&version=1.18.0&config=intermediate&
↪openssl=1.1.1k&guideline=5.6

ssl_certificate      certs/fullchain.pem;
ssl_certificate_key  certs/privkey.pem;
ssl_session_timeout  1d;
ssl_session_cache    shared:MozSSL:10m;  # about 40000 sessions
ssl_session_tickets  off;

# curl https://ssl-config.mozilla.org/ffdhe2048.txt > /etc/nginx/ffdhe2048.txt
```

(continues on next page)

(continued from previous page)

```
ssl_dhparam /etc/nginx/ffdhe2048.txt;

# intermediate configuration
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
↪GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-
↪CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
ssl_prefer_server_ciphers off;

# HSTS (ngx_http_headers_module is required) (63072000 seconds, two years)
add_header Strict-Transport-Security "max-age=63072000" always;

# OCSP stapling
ssl_stapling on;
ssl_stapling_verify on;

# verify chain of trust of OCSP response using Root CA and Intermediate certs
# ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;
## verify chain of trust of OCSP response using Root CA and Intermediate certs
# https://community.letsencrypt.org/t/howto-ocsp-stapling-for-nginx/13611/5
#   "You need to set the ssl_trusted_certificate to chain.pem
#    for OCSP stapling to work.
ssl_trusted_certificate certs/chain.pem;
```

Using Self-Signed Certificates, TLSv1.3 Only, and HSTS

Although self-signed certificates present challenges including getting modern browsers to accept them, users without control over a public domain name aren't able to obtain publicly verifiable certificates. See *Security > Self-Signed Certificates* for more information. As previously noted, OCSP (Online Certificate Status Protocol) doesn't make sense for self-signed certificates.

`tls.conf`

```
# https://wiki.mozilla.org/Security/Server_Side_TLS
# "nginx 1.18.0, modern config, OpenSSL 1.1.1k
#  Supports Firefox 63, Android 10.0, Chrome 70, Edge 75, Java 11,
#  OpenSSL 1.1.1, Opera 57, and Safari 12.1"
# generated 2022-01-22, Mozilla Guideline v5.6, nginx 1.18.0, OpenSSL 1.1.1k, modern␣
↪configuration
# https://ssl-config.mozilla.org/#server=nginx&version=1.18.0&config=modern&openssl=1.1.
↪1k&guideline=5.6

ssl_certificate      certs/cert.pem;
ssl_certificate_key  certs/privkey.pem;
ssl_session_timeout  1d;
ssl_session_cache    shared:MozSSL:10m;  # about 40000 sessions
ssl_session_tickets  off;

ssl_protocols TLSv1.3;
ssl_prefer_server_ciphers off;
```

(continues on next page)

(continued from previous page)

```
# HSTS (ngx_http_headers_module is required) (63072000 seconds, two years)
add_header Strict-Transport-Security "max-age=63072000" always;
```

7.3.2 Site Configuration

Linux-based OSes seem to use a `sites-available` / `sites-enabled` configuration approach. With this approach, the configuration is kept in `sites-available` and a symlink is placed in `sites-enabled` for those that should be used for the starting or reloading instance.

Once you have confirmed that `nginx` is running properly, remove the symlink in `sites-enabled/` to `default`. Once configured, a symlink to `../sites-available/pyde1` in `sites-enabled/` will use the new configuration on the next restart of `nginx`.

Main Server Block

This is the body of the configuration. The `server_name` must be one that corresponds to that of the TLS certificate. TLS generally "won't work" with a numeric IP address in the address bar. Configuration of local DNS is outside the scope of these instructions. Please consult your "router" instructions.

This block does the following:

- Sets up a listener on port 443 for HTTP-S connections to `www.example.com`
- Sets the cache expiration to be immediate. This can be removed when your development phase is complete and you are not changing content files.
- `location ~ /\.` – Prohibit access to `.git` or the like
- `location /favicon.ico` – Don't log its absence
- `location /pyde1/` – Proxy to the Python, HTTP server
- `location /de1-plot/ws` – Proxy to the `mosquitto` WebSocket port (location specific to external web-app config)
- `location /de1-plot/db` – Proxy to the uWSGI server socket (location specific to external web-app config)

Note: The location of the UI relative to the server root can usually be a personal choice.

The location of resources relative to that location will be determined by the UI and where it expects to find them. This configuration is for the KEpyDE1 test UI installed at `/de1-plot/`

`sites-available/pyde1`

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name www.example.com;

    root /var/www/html;

    # Do not cache while doing development
    # http://nginx.org/en/docs/http/ngx_http_headers_module.html#expires
    # https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control
```

(continues on next page)

(continued from previous page)

```
    expires   0;

    # This seems to work, but not ^~
    location ~ /\. {
        return 404;
    }

    location / {
        index  index.html index.htm;
    }

    location /favicon.ico {
        log_not_found off;
    }

    location /pyde1/ {
        proxy_pass http://127.0.0.1:1234/ ;
    }

    location /de1-plot/ws {
        proxy_pass http://127.0.0.1:1884/ ;
    }

    # http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#set

    location /de1-plot/db/ {

        # The "obvious" doesn't work
        # rewrite /de1-plot/db/(.*) /$1 break;

        include uwsgi_params;
        set $rewritten_uri $request_uri;
        if ($request_uri ~ /de1-plot/db/(.*)) {
            set $rewritten_uri /$1;
        }
        uwsgi_param REQUEST_URI $rewritten_uri;
        uwsgi_pass unix:///tmp/uwsgi-pyde1-db.sock;
    }
}
```

Change Site From "default" to "pyde1"

To enable the "pyde1" site definition, remove the *symlink* to `default` in `sites-enabled` and link in the new `pyde1` (or whatever you've called it).

```
jeff@pi-walnut:/etc/nginx/sites-enabled $ ls -l
total 0
lrwxrwxrwx 1 root root 26 Nov 20 14:13 default -> ../sites-available/default
jeff@pi-walnut:/etc/nginx/sites-enabled $ sudo rm default
jeff@pi-walnut:/etc/nginx/sites-enabled $ sudo ln -s ../sites-available/pyde1 .
jeff@pi-walnut:/etc/nginx/sites-enabled $ ls -l
```

(continues on next page)

(continued from previous page)

```
total 0
lrwxrwxrwx 1 root root 24 Nov 20 14:14 pyde1 -> ../sites-available/pyde1
```

Note: `sudo nginx -t -c /etc/nginx/nginx.conf` can be used to test configuration

Remember to `sudo systemctl restart nginx.service` to have the changes take effect.

CHAPTER

EIGHT

CONFIGURING MOSQUITTO

8.1 Overview

`mosquitto` is an MQTT broker. When used with the pyDE1 suite, it receives requests to publish MQTT notifications from the pyDE1 core, as well as the Visualizer uploader. Clients can subscribe to one or more of these and the broker will send them to the client. These notifications are what can be used to update a client in effectively real time.

MQTT needs care in configuration as there is no distinction between a publisher and a subscriber. Any client, if allowed by the broker, can publish. This includes connections over WebSockets.

The approach taken with this example configuration is to at least reduce the exposure within reasonable bounds. For the coffee connoisseur, this is "Starbucks-quality" security. Not completely unpalatable, accepted unknowingly by millions, yet completely lacking in so many things. Using Unix domain sockets for same-host connections would be preferable, but the "standard" MQTT library, paho, doesn't support them. So, *localhost* is has to be. For simplicity, as these are same-host connections, TLS is not used.

> **Warning:** Security should not be taken lightly. Even when "only" exposed on the loopback interface, any service poses a vector for attack. If on an accessible network segment, and even if not, use of TLS is highly recommended, along with more secure authorization than the username/password supplied by MQTT.

8.2 Installing mosquitto

```
apt install mosquitto mosquitto-clients
```

`mosquitto-clients` is optional. It provides the `mosquitto_pub` and `mosquitto_sub` utilities that are useful for debugging and monitoring.

8.3 Example mosquitto Configuration

The default `/etc/mosquitto/mosquitto.conf` includes a directive to read configuration from `/etc/mosquitto/conf.d/`.

As discussed in the overview, this example configuration does not use TLS and, as such, is not suitable for use on anything but the loopback interface ("localhost") and then only on hosts where you are confident that the loopback interface can't be monitored.

> **Warning:** It appears that a WebSocket listener can't be restricted to the loopback interface, exposing it on the network.

The (mis-)behavior seen at https://www.eclipse.org/lists/mosquitto-dev/msg00799.html still appears to be the current behavior at least as of 2.0.11

8.3.1 Listeners Without TLS

Without TLS, the configuration of listeners is straightforward.

The ports here are not not "set in stone" the way that HTTP and HTTP-S are specified.

`conf.d/listeners.conf`

```
listener 1883 localhost

listener 1884
protocol websockets
```

8.3.2 Listeners Adding TLS

Here, port 8883 was selected for MQTT over TLS. The WebSocket listener is reverse-proxied by `nginx`, so it is not enabled here.

`conf.d/listeners.conf`

```
listener 1883 localhost

listener 8883
cafile /etc/ssl/certs/ca-certificates.crt
# For verifiable certs (e.g, Let's Encrypt)
# certfile /etc/mosquitto/certs/fullchain.pem
# For stand-alone, self-signed certs
certfile /etc/mosquitto/certs/cert.pem
# For all types of certs
keyfile /etc/mosquitto/certs/privkey.pem
tls_version tlsv1.3

listener 1884
protocol websockets
```

As previously noted, `mosquitto` needs to be able to read the private key as the `mosquitto` user, not `root`. This is somewhat ugly, as reading as `root` then dropping privelege helps protect the key from compromise. For now, we note and live with the risks. At least until I can find a better approach, `privkey.pem` needs to be (only) `mosquitto`-readable.

```
$ sudo chown mosquitto:root /etc/mosquitto/certs/privkey.pem
$ sudo chmod 600 /etc/mosquitto/certs/privkey.pem
$ ls -l /etc/mosquitto/certs/privkey.pem
-r-------- 1 mosquitto root 3272 Jan  4 15:19 /etc/mosquitto/certs/privkey.pem
```

Current versions of `pyDE1` allow configuration of TLS for MQTT through the config files. For details of the parameters, see paho's `Client.set_tls()`. With a verifiable certificate, setting `mqtt.TLS: true` should be sufficient. With self-

signed certificates, `mqtt.TLS_CA_CERTS` likely would also need to be set to the path to the corresponding CA or public certificate in use.

8.3.3 Blocking Off-Host Access to WebSockets

As the listener for WebSockets is on all interfaces, it presents enough of a security risk to block the port from off-host access. There are several firewall tools for Linux, many of which are very outdated and now deprecated. Here are some simple rules using `nftables` that should block access to port 1884 from other hosts. For more information on `nftables`, see, for example, https://wiki.nftables.org/wiki-nftables/index.php/Simple_rule_management

> **Warning:** This is not a complete firewall. It will need to be integrated with your existing firewall.

```
nft add inet filter input iifname != 'lo' tcp dport 1884 drop
```

On a "fresh" Debian Bullseye system, the resulting ruleset may look something like the following:

```
$ sudo nft -a list ruleset
table inet filter { # handle 1
    chain input { # handle 1
        type filter hook input priority filter; policy accept;
        iifname != "lo" tcp dport 1884 drop # handle 4
    }

    chain forward { # handle 2
        type filter hook forward priority filter; policy accept;
    }

    chain output { # handle 3
        type filter hook output priority filter; policy accept;
    }
}
```

You may need to enable and start `nftables.service` if it is not yet running. `systemctl status nftables.service` will show if it is enabled and/or running. The `enable`, `start`, ... actions require root privilege (`sudo`). The default configuration file is `/etc/nftables.conf`.

8.3.4 Access Control and Authorization

As there is no inherent concept of a listen-only MQTT client, it is important to restrict to which topics clients can publish, or you risk running an "open" MQTT server.

`/etc/mosquitto/conf.d/auth.conf`

```
# allow_anonymous false requires changing the JavaScript
# to include username and password. Ideally, this could be
# dynamically generated for some minor security.
# For now, just allow anonymous read access.
allow_anonymous true

# include_dir dir
#    [...] All files that end in '.conf' will be loaded
```

(continues on next page)

(continued from previous page)

```
# (so files not ending in .conf are "safe" here)

password_file /etc/mosquitto/conf.d/passwords
acl_file /etc/mosquitto/conf.d/acls
```

Note: The user names and passwords you pick here will need to be edited into `pyde1.conf` and `pyde1-visualizer.conf`, which will be installed later.

`/etc/mosquitto/conf.d/acls`

```
# Here it is assumed that the topic root is pyDE1

# "The first set of topics are applied to anonymous clients,
#  assuming allow_anonymous is true."
topic read pyDE1/#

# The main executable
user pyde1
topic readwrite pyDE1/#

# The visualizer uploader
user pyde1-visualizer
topic read pyDE1/#
topic write pyDE1/VisualizerUpload
```

`/etc/mosquitto/conf.d/passwords`

Create the passwords with `mosquitto_passwd`. The first time you probably need to add the `-c` (create new password file) option.

Warning: Do *not* use the login password here.

You do not need an OS-level user for these.

```
mosquitto_passwd -c /etc/mosquitto/conf.d/passwords pyde1
mosquitto_passwd /etc/mosquitto/conf.d/passwords pyde1-visualizer
```

These passwords and user names need to agree with those in `pyde1.conf` and `pyde1-visualizer.conf`

Confirm that the file is readable by *mosquitto* but not writable by other than *root*.

8.3.5 Logging

`/etc/mosquitto/conf.d/logging.conf`

The change here is to use human-readable timestamps in the log files, rather than Unix timestamps.

```
log_timestamp_format %Y-%m-%dT%H:%M:%S
```

Note: Remember to `sudo systemctl restart mosquitto.service` to have the changes take effect.

CHAPTER

NINE

INSTALLING AND ENABLING PYDE1

9.1 Overview

To enhance security, the `pyDE1` executables run as a dedicated user, one with limited privilege. When possible, only read access is granted to the files. None of the executables or configuration files should be writable by *pyde1*. In this guide, they are set to *root* ownership.

> **Warning:** Accessing the database as any user other than *pyde1* may cause files to be written by that user, preventing access by *pyde1*[1]

There are `sh` scripts provided that should make the process relatively straightforward. As they are often run as *root* with `sudo`, read them and convince yourself that they are going to do what you expect, *before* running them blindly.

Unless specifically noted, all scripts are to be run with *root* privilege using `sudo`.

9.2 Walk-Through

The scripts have been broken down into relatively small operations. This allows them to be easily re-run should an error occur.

The scripts are *not* distributed in the `pip` package. They are present in the pyDE1 git repo, in the `install/` directory. The repo can be cloned, or individual files downloaded. Make sure that the `_config` file is in the same directory as the shell scripts. Generally no changes need to be made to the `_config` file.

```
# Copyright © 2021 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

# sourced by install scripts

PYDE1_USER=pyde1
PYDE1_GROUP="${PYDE1_USER}"
VENV_PATH="/home/${PYDE1_USER}/venv/pyde1"
```

[1] One way to access the database is with `sudo -u pyde1 sqlite3 /var/lib/pyde1/pyde1.sqlite3`

9.2.1 10-create-user.sh

This script will create the *pyde1* user if it does not exist and give it access to the *bluetooth* group.

```
#!/usr/bin/sh -e

# Copyright © 2021 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

. "$(dirname $0)"/_config

if [ -z "$SUDO_USER" ] ; then
  >&2 echo "Script must be run with sudo"
elif [ "$SUDO_UID" =  0 ] ; then
  >&2 echo "Script must be run with sudo by a normal user"
fi

# Create the pyde1 user if they do not yet exist.

if getent passwd "$PYDE1_USER" ; then
  echo "User $PYDE1_USER already exists"
else
  echo "Creating user $PYDE1_USER"
  adduser --system --group "$PYDE1_USER"
fi
usermod -a -G bluetooth "$PYDE1_USER"
id "$PYDE1_USER"
```

9.2.2 20-create-dirs.sh

This script will create the following directories and set their ownership and permissions:

- /var/log/pyde1
- /var/lib/pyde1

```
#!/usr/bin/sh -e

# Copyright © 2021, 2022 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

. "$(dirname $0)"/_config

echo "Creating target directories"

mkdir -p /var/log/pyde1
chown $PYDE1_USER /var/log/pyde1
```

(continues on next page)

(continued from previous page)

```
mkdir -p /var/lib/pyde1
chown $PYDE1_USER /var/lib/pyde1

ls -ld /var/log/pyde1
ls -ld /var/lib/pyde1
```

9.2.3 30-populate-venv

This creates a *root-owned*, Python virtual environment ("venv"). It then updates `pip` and `setuptools` and adds the `pyDE1` package and its dependencies to the venv.

Note: If you installed a non-default version of Python, such as 3.9 or 3.10 on an install of Raspberry Pi OS based on "Buster", you will need to explicitly reference that version when creating the venv.

`python -m venv $VENV_PATH` would need to be edited to explicitly to refer to your chosen version. References after `. $VENV_PATH/bin/activate` should not need modification, as that sets `python` to refer to the one in the venv.

```
#!/usr/bin/sh -e

# Copyright © 2021 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

. "$(dirname $0)"/_config

echo "Creating Python venv at $VENV_PATH"

if ! dpkg --get-selections | egrep '^python3-venv\s+install$' ; then
  apt install python3-venv
fi

mkdir -p $VENV_PATH

python -m venv $VENV_PATH

. $VENV_PATH/bin/activate

pip install -U pip
pip install -U setuptools
pip install pyDE1
pip list

# "Where is pyDE1?"
python -c \
  'import importlib.resources ; print(importlib.resources.files("pyDE1"))'
```

9.2.4 40-config-files.sh

This copies the config files from the location where `pip` installed them in the venv and into `/usr/local/etc/pyde1`. It will make a timestamped backup of any file that would be overwritten.

Note: Some of the configuration files may contain sensitive credentials, such as MQTT and Visualizer usernames and passwords. These files are set to *root:pyde1* ownership with no other read access.

It also copies the `pyde1.service` and `pyde1-visualizer.service` files, similarly making backups. These files are edited in place to adjust for the specifics of the local install from the previous steps. The editor (`sed`) backs up the original version with a `.bak` suffix.

Rather than run `disconnect-btid.sh` directly from the install, it is copied to `/usr/local/bin/pyde1-disconnect-btid.sh`. This script is run by `pyde1.service` to help clean up any "stale" Bluetooth connections related to a prior run that may have terminated ungracefully.

```
#!/usr/bin/sh -e

# Copyright © 2021, 2023 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

. "$(dirname $0)"/_config

mkdir -p /usr/local/bin/pyde1
mkdir -p /usr/local/etc/pyde1

CP_BACKUP="cp -v --backup --suffix=$(date +'.%Y%m%d_%H%M')"

$CP_BACKUP ${PYDE1_ROOT}/services/config/* /usr/local/etc/pyde1/

# As the config files may contain credentials,
# make them unreadable to anyone but root and pyde1

chown root:${PYDE1_GROUP} /usr/local/etc/pyde1/*
chmod 640 /usr/local/etc/pyde1/*

$CP_BACKUP ${PYDE1_ROOT}/services/unit-files/* /usr/local/etc/pyde1/
chown root:root /usr/local/etc/pyde1/*.service
chmod 644 /usr/local/etc/pyde1/*.service

for f in /usr/local/etc/pyde1/*.service ; do
  # This is rather fragile. TODO: Consider a template
  sed -i'.bak'  \
    -e "s|^User=.*|User=${PYDE1_USER}|" \
    -e "s|^Group=.*|Group=${PYDE1_GROUP}|" \
    -e "s|/home/pyde1/venv/pyde1|${VENV_PATH}"
    $f
done

ls -l /usr/local/etc/*
```

9.2.5 Adjust Config Files to Suit

Examine the various config files `/usr/local/etc/pyde1/*.conf` and edit as needed.

Changes that are commonly needed include:

In `pyde1.conf`

- `mqtt:`
  - `USERNAME`
  - `PASSWORD`
- `de1:`
  - `LINE_FREQUENCY`

In `pyde1-visualizer.conf`

- `mqtt:`
  - `USERNAME`
  - `PASSWORD`
- `visualizer:`
  - `USERNAME`
  - `PASSWORD`

After completing the edits, ensure that they are readable by the *pyde1* group and not by anyone else, other than *root*.

```
ls -l *.conf
-rw-r----- 1 root pyde1 3555 Nov  3 18:18 pyde1.conf
-rw-r----- 1 root pyde1 1789 Nov  3 18:18 pyde1-replay.conf
-rw-r----- 1 root pyde1 2308 Nov  3 18:18 pyde1-visualizer.conf
```

If needed, the ownership and permissions can be corrected with

```
sudo chown root:pyde1 *.conf
sudo chmod 640 *.conf
```

9.2.6 50-enable-services.sh

This links the service definitions in `/usr/local/etc/pyde1` for the `pyde1.service` and the `pyde1-visualizer.service` to where `systemd` (the "startup manager" for Debian) knows about them, enables the services, and restarts them. Unless they are explicitly disabled, they will start on every boot.

Further information on service management can be found with

```
man systemctl
man journalctl
```

```
#!/usr/bin/sh -e

# Copyright © 2021 Jeff Kletsky. All Rights Reserved.
#
# License for this software, part of the pyDE1 package, is granted under
```

(continues on next page)

(continued from previous page)

```
# GNU General Public License v3.0 only
# SPDX-License-Identifier: GPL-3.0-only

for service in /usr/local/etc/pyde1/pyde1.service \
           /usr/local/etc/pyde1/pyde1-visualizer.service ; do

  systemctl link -f $service
  systemctl daemon-reload
  service_name=$(basename $service)
  systemctl enable $service_name
  systemctl restart $service_name

done
```

9.3 Log Rotation

The standard log-rotation utility on Debian is `logrotate` with configuration in `/etc/logrotate.d/`

One configuration that rotates daily, compresses, and retains 60 days' of logs is

```
/var/log/pyde1/pyde1.log {
    daily
    missingok
    rotate 60
    compress
    delaycompress
    notifempty
    create
}

/var/log/pyde1/visualizer.log {
    daily
    missingok
    rotate 60
    compress
    delaycompress
    notifempty
    create
}
```

Both the `mosquitto` and `nginx` packages install self-named config into `/etc/logrotate.d/`

CHAPTER

TEN

UPDATING PYDE1

Updates to pyDE1 can be applied with `pip` when they become available.

The most-current, stable version can be checked at https://pypi.org/project/pyDE1/

Unless you are working with the author on a new feature or resolving a bug, the Stable releases are recommended.

10.1 Updating Your venv, Including pyDE1

To update the packages used by your virtual environment (venv), `pip` can check to see if any are outdated.

First, activate the venv so you are "inside" of it, rather than using the OS' versions. Then you can check for outdated packages.

```
jeff@pi-walnut:~ $ . ~pyde1/venv/pyde1/bin/activate
(pyde1) jeff@pi-walnut:~ $ pip list --outdated
Package Version Latest Type
------- ------- ------ -----
bleak   0.14.1  0.14.2 wheel
```

Unfortunately, `pip` doesn't have an "update all" command, so I often end up just going through the list, using the `-U` flag for updating.

```
(pyde1) jeff@pi-walnut:~ $ pip install -U bleak
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: bleak in /home/pyde1/venv/pyde1/lib/python3.9/site-
↪packages (0.14.1)
Collecting bleak
  Downloading https://www.piwheels.org/simple/bleak/bleak-0.14.2-py2.py3-none-any.whl␣
↪(114 kB)
     || 114 kB 190 kB/s
Requirement already satisfied: dbus-next in /home/pyde1/venv/pyde1/lib/python3.9/site-
↪packages (from bleak) (0.2.3)
Installing collected packages: bleak
  Attempting uninstall: bleak
    Found existing installation: bleak 0.14.1
    Uninstalling bleak-0.14.1:
      Successfully uninstalled bleak-0.14.1
Successfully installed bleak-0.14.2
```

If you're a command-line wrangler, you can figure out how to use `xargs` for multiples, but I usually don't have enough to update to find my notes on that.

10.1.1 Restart Services

After updating, it is generally a good idea to restart the services that depend on it and check that things are running smoothly. (You don't need to have the venv activated for this.)

To exit the pager from the `status` command, use `q`

```
(pyde1) jeff@pi-walnut:~ $ sudo systemctl restart pyde1.service
(pyde1) jeff@pi-walnut:~ $ sudo systemctl restart pyde1-visualizer.service
(pyde1) jeff@pi-walnut:~ $ systemctl status pyde1.service
 pyde1.service - Main controller processes for pyDE1
   Loaded: loaded (/usr/local/etc/pyde1/pyde1.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-01-28 09:11:46 PST; 20s ago
  Process: 30816 ExecStartPre=sh ${PYDE1_PATH}/services/runnable/disconnect-btid.sh␣
↪(code=exited, status=0/SUCCESS)
 Main PID: 30818 (python3)
    Tasks: 26 (limit: 1597)
      CPU: 7.505s
   CGroup: /system.slice/pyde1.service
           ├─30818 /home/pyde1/venv/pyde1/bin/python3 /home/pyde1/venv/pyde1/lib/python3.
↪9/site-packages/pyDE1/run.py
           ├─30819 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.resource_
↪tracker import main;main(3)
           ├─30823 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.spawn␣
↪import spawn_main; spawn_main(tracker_fd=4, pi>
           ├─30824 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.spawn␣
↪import spawn_main; spawn_main(tracker_fd=4, pi>
           ├─30825 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.spawn␣
↪import spawn_main; spawn_main(tracker_fd=4, pi>
           ├─30826 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.spawn␣
↪import spawn_main; spawn_main(tracker_fd=4, pi>
           └─30827 /home/pyde1/venv/pyde1/bin/python3 -c from multiprocessing.spawn␣
↪import spawn_main; spawn_main(tracker_fd=4, pi>

Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,519 DEBUG [MainProcess]␣
↪Config.YAML: Setting database.FILENAME
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,519 DEBUG [MainProcess]␣
↪Config.YAML: Setting de1.LINE_FREQUENCY
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,520 DEBUG [MainProcess]␣
↪Config.YAML: Setting de1.DEFAULT_AUTO_OFF_TIME
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,520 DEBUG [MainProcess]␣
↪Config.YAML: Setting de1.STOP_AT_WEIGHT_ADJUST
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,521 INFO [MainProcess]␣
↪Config.YAML: Config overrides loaded from /usr/lo>
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,700 INFO [MainProcess] root:␣
↪Configured stderr_handler: <StreamHandler <>
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,701 INFO [MainProcess] root:␣
↪Configured mqtt_handler: <PipeHandler (ERRO>
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,702 INFO [MainProcess] root:␣
↪Configured logfile_handler: <WatchedFileHan>
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,704 INFO [MainProcess] root:␣
↪Started <logging.handlers.QueueListener obj>
Jan 28 09:11:47 pi-walnut pyde1[30818]: 2022-01-28 09:11:47,705 DEBUG [MainProcess]␣
↪root: log_queue_listener handlers: (<StreamHandl>
```

(continues on next page)

(continued from previous page)

```
(pyde1) jeff@pi-walnut:~ $ systemctl status pyde1-visualizer.service
 pyde1-visualizer.service - Auto-upload to Visualizer
   Loaded: loaded (/usr/local/etc/pyde1/pyde1-visualizer.service; enabled; vendor␣
↪preset: enabled)
   Active: active (running) since Fri 2022-01-28 09:11:57 PST; 52s ago
 Main PID: 30907 (python3)
    Tasks: 5 (limit: 1597)
      CPU: 1.633s
   CGroup: /system.slice/pyde1-visualizer.service
           └─30907 /home/pyde1/venv/pyde1/bin/python3 /home/pyde1/venv/pyde1/lib/python3.
↪9/site-packages/pyDE1/services/runnable/p>

Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,798 DEBUG␣
↪[MainProcess] Config.YAML: Setting logging.handlers>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,799 DEBUG␣
↪[MainProcess] Config.YAML: Setting logging.handlers>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,799 DEBUG␣
↪[MainProcess] Config.YAML: Setting logging.LOGGERS
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,799 DEBUG␣
↪[MainProcess] Config.YAML: Setting mqtt.USERNAME
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,800 DEBUG␣
↪[MainProcess] Config.YAML: Setting mqtt.PASSWORD
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,800 WARNING␣
↪[MainProcess] Config.YAML: No entries found for d>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,801 INFO␣
↪[MainProcess] Config.YAML: Config overrides loaded f>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,802 INFO␣
↪[MainProcess] root: Configured stderr_handler: <Stre>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,802 INFO␣
↪[MainProcess] root: Configured mqtt_handler: <NullHa>
Jan 28 09:11:58 pi-walnut pyde1-visualizer[30907]: 2022-01-28 09:11:58,803 INFO␣
↪[MainProcess] root: Configured logfile_handler: <Wat>
```

10.1.2 Exiting the venv

Though usually it doesn't do any harm to stay in the venv, it can be exited with

```
(pyde1) jeff@pi-walnut:~ $ deactivate
jeff@pi-walnut:~ $
```

10.2 Updating UI Components

It is likely that UI components can be updated and they will be recognized as soon as a request is made to the webserver. Check the documentation for your UI on this.

Some components *might* need uWSGI (or other execution gateway) restarted. This will depend on the configuration file. For example, KEpyDE1's config file uses the `touch-reload` feature that automatically updates the code to be run as soon as it is changed on disk.

```
jeff@pi-walnut:~ $ fgrep touch-reload /etc/uwsgi-emperor/vassals/pyde1-db.ini
touch-reload = dbget.py
touch-reload = database_access.py
```

CHAPTER

ELEVEN

BACKING UP THE PYDE1 DATABASE

11.1 When and Why

Backup strategies are always a subject of debate. If you're reading this, I'm assuming you have already made some decision around your strategy. This page discusses some of the options for database compression as well as some hints about how to automate backups.

Even if you aren't scheduling periodic backups, when a schema update is automatically performed from schema 2 or later, the database is backed up. This backup is in the same folder as the "live" database. It may be deleted when deemed appropriate.

11.2 Manual Backup

A manual backup of the pyDE1 database is possible using `sqlite3` commands. As with any connection to the database, it should be done using the user that owns the database, `pyde1` by default.

In an interactive session, one way to accomplish this is with

```
sudo -u pyde1 sqlite3 /var/lib/pyde1/pyde1.sqlite3
```

The `backup` command can be executed from the prompt

```
sqlite> .help backup
.backup ?DB? FILE        Backup DB (default "main") to FILE
     --append            Use the appendvfs
     --async             Write to FILE without journal and fsync()
```

11.3 Compression Options

The pyDE1 database is highly compressible. Selection of a compression tool will depend on what has been installed on the system and the value of the time vs. compression (disk space) tradeoff. Most Linux-based systems already have `gzip` installed. It is a reasonable compressor in terms of speed and ratio. Mainly as it is already installed, it is the default for the *post-facto* compression of backups made during the schema-upgrade process.

Here are some example compression times and results. They were tested on a Raspberry Pi 3B+ with a 64 GB microSD, probably an upper-grade SanDisk. The database has 3196 rows in the `sequence` table. Uncompressed it is 511.5 MB. Default compression was used for each program, unless noted in the table. Time is "real" from the `time` utility.

Program	Time	Compressed	Fraction
`lz4`	0:15	143.1 MB	28.0 %
`zstd -1`	0:23	101.2 MB	19.8 %
`zstd`	0:43	100.2 MB	19.6 %
`gzip`	1:26	98.6 MB	19.3 %
`zstd -10`	1:59	86.8 MB	17.0 %
`zstd -19`	32:36	76.6 MB	15.0 %
`xz`	15:54	65.6 MB	12.8 %

For scheduled backups, `xz` at the default compression level provides the greatest disk-space savings of those tested. It may be "too slow" even for overnight runs with very large databases.

Compression speed seems significantly faster on the Raspberry Pi 4, with xz compression taking about half the time (7:24).

11.4 Scheduled Backups

As pyDE1 configures its database to be multi-user, it is possible to schedule backups with standard OS tools, such as `cron`. One approach is to use a script that performs the backup to a unique name and then compresses it when complete.

```
#!/bin/sh

filename=$(date +'/home/pyde1/db_backup/pyde1.%Y-%m-%d_%H%M.sqlite3')
sqlite3 /var/lib/pyde1/pyde1.sqlite3 ".backup $filename"
xz $filename
```

and then scheduling that script for execution by `pyde1`.

One way to do this is to edit the `crontab` for the `pyde1` user. Do not run as `root` as doing so may cause database files to be owned by `root`, preventing access by `pyde1`.

```
sudo -u pyde1 crontab -e
```

CHAPTER

TWELVE

CHANGELOG

12.1 2.0.0 — Pending

See also:

What's New in Version 2

12.2 1.5.2 - 2022-11-11

12.2.1 Overview

Adds a five-second timeout to Visualizer connections that might resolve an intermittent stoppage of uploads. If the upload times out, the shot is re-queued.

Should this seem to cause local issues, rollback to v1.5.1 can be done through `pip install pyDE1==1.5.1` or similar

12.2.2 Fixed

- Visualizer uploads now have a five-second timeout – `1ac011e5`

12.2.3 New

- Notes on firmware versions 1328 and 1330 added – `4362254a`

12.3 1.5.1 - 2022-11-11

12.3.1 Overview

Resolves issue with changes in bleak v0.19.0 – removes version restriction

12.3.2 Removed

- Unused `BLEAK_AFTER_0_17` check – `085394f`

12.4 1.5.0 - 2022-10-24

Note: Requires bleak v0.18.1 — Use v1.5.1 or later for bleak v0.19

Warning: Now requires Python 3.9 or later

12.4.1 Overview

MAPPING 5.0.0

Add move-on by weight functionality, including parsing from JSON profile.

"Bump resist" to help prevent triggering SAW/MOW too early

Provide a utility to convert Tcl profile files to JSON, including directly from a Visualizer URL. Author names can be corrected in the process.

Tested with Python 3.11.0 and more extensively with 3.11.0rc2

Changes to accommodate breaking changes in bleak v0.18.1

Significant refactoring of FlowSequencer

Improvements to logging

12.4.2 New

- Move-on by weight functionality – `75a52e3`, `674f06c`
- "Bump resist" for SAW / MOW and config – `148cd44`

  Active when mass-flow estimate exceeds FLOW_THRESHOLD or goes negative

  Can also select to always use median-based estimation

```
config.de1.bump_resist.FLOW_THRESHOLD = 10.0  # g/s
config.de1.bump_resist.FLOW_MULTIPLIER = 1.1
config.de1.bump_resist.SUB_MEDIAN_WEIGHT = True  # when excessive flow
config.de1.bump_resist.USE_MEDIAN_WEIGHT_ALWAYS = False
config.de1.bump_resist.USE_MEDIAN_FLOW_ALWAYS = False
```

- `legacy_to_json.py` converts profiles to JSON format – `40d8a86`
- DE1 supplies `current_frame` property – `cc2ae17`
- ScaleProcessor supplies `current_weight` property – `93ea42f`

12.4.3 Changed

- Requires Python 3.9 or later, supports Python 3.11
- Logic for skip_to_next improved to limit to when it is available in the firmware and sensible (during espresso only) – `62da713`
- Logging
  - `Scale.Period` and `Outbound.Counts` loggers to assist in filtering out "expected, periodic" log messages – `09500d3`, `3b8f87a`
  - Scale: Add logging of exception to `weight_update_handler()` – `472eb90`
  - Frame changes are now logged at the INFO level, including the previous and current frames, as well as an estimate of the weight.
  - Adjusted `MMR0x80LowAddr.for_logging()` to retain uppercase names. Also can return hex if no name associated. MMR Read notifications now logged similar to `INFO [Controller] DE1.CUUID.ReadFromMMR.Notify.FLUSH_TIMEOUT: 3.0` – `b013d7a`, `b455744`
  - Improved message on scale/DE1 disconnect to include previous address – `6e9fe24`
- `None` is now permitted as a return value in the Scale class from `current_weight()` coroutine. `AtomaxSkaleII` returns `None` rather than raising `NotImplementedError` *(It does not appear that the weight can be requested on demand, just notifications.)* – `816e29c`

  See also new `ScaleProcessor.current_weight` property – `93ea42f`
- The `StopAtNotification` has been updated to version 1.1.0 as it adds the current frame, which may be None/null, as well as the action of 'move on by weight'.
- The mode-control objects within the FlowSequencer have been refactored. Any code accessing these directly should examine both `flow_sequencer.py` and `flow_sequencer_impl.py`.
- The signature of the `stop_at_notify` method of the FlowSequencer has been changed to capture information at the time of the call, rather than when the call is serviced. With `skip_to_next` being called, it is possible that the time of service could be in a different frame of the profile.
- The FlowSequencer internal `stop_at_weight` routine has been renamed to act_on_weight and handles both SAW and move-on by weight.
- Python 3.11 compatibility
  - Event loop is now created explicitly based on DeprecationWarning – `b939868`
  - Changes in how enum.IntFlag are string-ified are accommodated for 3.11 and later – `3e953cd`
- Bleak v0.18.1 compatibility
  - Reworked deprecated `BleakScanner.register_detection_callback()` – `d21f1a6`
  - Marked usages of deprecated `BleakClient.set_disconnected_callback()` – `d8c63fb`
  - `WrappedBleakClient()` now passes `*args, **kwargs` to `BleakClient()` – `b5468b4`
  - Added `bleak_version_check.BLEAK_AFTER_0_17` – `7082b8c` (unused as changes in method signatures were reverted in bleak v0.18.1)

12.4.4 Fixed

- Example pyde1.conf comments now properly refer to `purge_deferred` – `5e63756`
- Scan-initiation now properly accepts `null` to accept default timeout – `84308ec`

12.4.5 Removed

- Python 3.8 support removed
- Deprecations in v1.2.0 (2022-03) removed
  - Use of `first_if_found` to initiate scanning removed – `2c957fd`
  - Use of a Boolean when setting Bluetooth scan timeout removed – `2c957fd`

12.5 1.4.0 – 2022-09-12

12.5.1 Overview

Adds ability to patch DE1 from config file on connect.

Support for features in firmware through 1352. Of these, perhaps the steam-purge mode control is the most interesting.

- RESOURCE v3.8.0
- MAPPING v4.2.1

(Includes changes previously pending for 1.3.0)

12.5.2 New

- Patch DE1 from config file when it first connects – `0c22418`
- Support for firmware through version 1352 – `94034a5`

12.5.3 Changed

- Add `last_updated` to DE1 "state" API to resolve ambiguity between API calls and MQTT notifications – `4594ad8`
- JSON profile "version" element can now accept a semantic-version string – `e95e3b4`
- Logging
  - Value that triggered `DE1APIValueError` now in message – `df5d20d`
  - `MMR0x80LowAddr` shown in hex when unknown – `004e287`
  - Added debug logging for "state-less" writes to database – `78cfa38`
- Clarified return value of `DE1().start_notifying()` as an event that triggers when the notification is received. Removed return value of `stop_notifying()` which was always `None`. – `c425703`

12.5.4 Fixed

- Gracefully handle "shots" without weight for Visualizer upload – `a9d9f01`
- Fix `last_mmr0x80()` for pre-1250 firmware version reported – `5249e65`

12.5.5 Removed

- Internal `DE1().write_one_mmr0x80()` was only used in one place and then in a context that replicated other calls. `write_and_read_back_mmr0x80()` is a preferred replacement.

12.6 1.2.0 - 2022-06-20

12.6.1 Changed

- DE1XXL is properly recognized from MMR0x80LowAddr.V13_MODEL – `eca93bb`
- The API_Substates added by FW v1315/1316 were added – `2d38e42`

12.7 1.2.0b1 - 2022-03-19

12.7.1 Overview

RESOURCE and MAPPING changes to enable uploading profiles without requiring DE1 connectivity. Use case suggested by EBengoechea, thanks!

Scale-management reworked in preparation of further changes to support Acaia and other scales that are typically not connected 24x7.

Ending a sequence before flow starts should no longer bloat the database.

12.7.2 Changed

scale: Change logger name to Scale.AtomaxSkaleII - `fd48ec3`

File logging can be disabled and SubscribedEvent can notify without a pipe present (for testing) - `6b5e6cf`, `d78cfa0`

Add config.de1.SEQUENCE_WATCHDOG_TIMEOUT (default, 270 seconds)- `a4a2dda`

12.7.3 Fixed

de1: scale: Quiet all connection attempt/fail logging when "not logging" - `6936f24`

scale: Factory now properly checks keys of name-to-class mapping - `323bbca`

Python 3.10: Change import for Callable from collections to typing - `39f7a57`

Sequences that are terminated before flow starts should no longer continue writing to the database. Watchdog timer also added - `a4a2dda`

12.7.4 Deprecated

“first_if_found”, “id”, “scan” deprecated - `207a492`

To start a scan, the parameter has been changed to prefer a positive number for the timeout, or null (to accept the default). Use of a bool here has been DEPRECATED. The preferred forms include:

```
{'begin': null}
{'begin': 5}
{'begin': 5.0}
```

To start a scan and select the first-found device of the desired type, set the id to ‘scan’. Use of the ‘first_if_found’ key has been DEPRECATED: The preferred forms include:

```
{'id': 'scan'}
{'id': 'aa:bb:cc:dd:ee:ff'}
{'id': null}
```

DEPRECATED forms include:

```
{'begin': true}
{'begin': false}
{'first_if_found': true}
{'first_if_found': false}
```

12.8 1.1.0 – 2022-01-24

Fixed: Long profiles should no longer time out when selecting by ID - `f1f383a`

Other functional changes described at *1.1.0b1 – 2022-01-14*

- Trivial documentation changes from 1.1.0b2
- Updated documentation from 1.1.0b1

12.9 1.1.0b2 – 2022-01-22

12.9.1 Overview

Updated, expanded, and reorganized installation documentation

12.9.2 Changed

Documentation (only)

12.10 1.1.0b1 – 2022-01-14

12.10.1 Overview

Resolves shutdown issue with MQTT unconnected, DE1 config-file values, improves some logging, updates FeatureFlag for FW 1293, improves compatability with Manjaro (OS), fixes documentation-generation issue.

12.10.2 Changed

- Reduce log severity for unimplemented MMRs 0x3820 and 0x3824 – `0125b72`
- `FeatureFlag` includes `sched_idle` flag, active for FW 1293 and later – `64ee7f7`
- Timeouts on CUUID request/notify log changed wording to state that it could also be the write or the lack of a notify received that caused the timeout – `a675f50`
- Removed stray comment from `20-create-dirs.sh` – `6070984`
- Link `README.rst` for documentation generation – `bb640f3`

12.10.3 Fixed

- Shutdown without an MQTT connection does not try (and fail) to close it – `adda65e`
- DE1 is initialized with config-file values, rather than default – `f7d6393`
- HTTP API now returns a more descriptive error if the payload data type is incorrect – `43614df`
- *disconnect-btid.sh* should no longer cause *sh* errors with Manjaro OS – `d3a3c65`
- Service definitions updated to use `StandardError=journal` – `ac0ead7`

12.11 1.0.0 — 2021-12-11

12.11.1 Overview

First release version.

12.11.2 Changed

- Allow request of Idle from a refill state (apparently not acted on by the DE1) - `55d81bb`
- Allow "force" of DE1 Idle from any state, enabled through config - `05adc93`
- Prereqs updated to current versions - `5d320cb`

RESOURCE 3.6.0

- Add `NO_REQUEST` mode to trigger a report from the DE1 - `a52cd6f`
- Add `END_STEAM` mode to support steam-to-temperature - `24d7b52`

12.11.3 Fixed

- Double-counting of scale delay was removed, improving scale-to-DE1 time alignment - `886016a`

12.12 0.10.0 – 2021-11-21

12.12.1 Overview

Documentation, including installation, added. Installation scripts, tested with Raspberry Pi OS Lite (Release date: October 30th 2021, Kernel version: 5.10) available in the source repo.

12.12.2 New

- Documentation viewable at https://pyde1.readthedocs.io/en/latest/
- Install scripts in the source repo in the `install` directory
- Provide config for TLS for MQTT clients - `427b3e0`

12.12.3 Changed

- Documentation reorganized and consolidated into the `docs` directory
- `disconnect-btid.sh` is now expected at `/usr/local/bin/pyde1-disconnect-btid.sh` by `pyde1.service`

MAPPING 4.0.1

- MODULES_FOR_VERSIONS consistent with requirements - `40c4ce0`

12.12.4 Fixed

- utils: data_as_readable() now handles “undecodable” byte sequences - `08aef05`
- packaging: Include schema and service files - `4caf736`

12.13 0.9.0 – 2021-10-31

12.13.1 Overview

Functionality for the beta release completed and tested.

12.13.2 New

- The flush-control features of *experimental* Firmware 1283 were implemented and include control of target duration, temperature, and flow. - `46c0481`
- Clean, Descale, and Transport functionality is now available through the API. - `65f2ac9`
- Provide asynchronous firmware upload through API. - `d6a2dbc`, `32436a9`
- GET of DE1_STATE enabled. - `2b4435e`
- Rewrite of logging and logging configuration. "Early" logging is captured and routed to the log file, once it is opened. Log levels and formatters can be easily configured through the YAML config files. - `b759168`, `39c714d`, `7df0397`, `d3e128c`, `cabab97`
- Provide logging over MQTT for client use (in addition to console and log file). - `019bed0`
- Profile frame logging provides "not" names for unset FrameFlags to clarify log messages. For example, the absence of `CtrlF` is now rendered as `CtrlP`. - `c842565`
- MQTT "Will" implemented, reporting unexpected MQTT disconnects. - `22d06b4`
- Feature flags have been added to formalize access to DE1 and firmware abilities. - `d7405b0`

12.13.3 Changed

- `c_api` was updated with new information. - `46c0481`
- The firmware version is read early in the DE1 initialization to determine the range of valid MMRs and how to efficiently read them. - `46c0481`
- The `ModeControl` class was refactored into `flow_sequencer`. - `46c0481`
- MMRs that are not able to be decoded (such as not implemented), are logged along with the value received. - `2d0fa24`
- Return 400 Bad Request for PATCH/PUT with no content. - `d00bd24`
- Change MQTT to not request retaining messages from pyDE1. - `8a8ba5e`
- Logging level and wording changes. - `99ec22f`, `b31c850`
- Rework imports to remove order dependencies and simplify. - `c895f7d`, - `b31c850`
- Improve reconnection algorithm for DE1 and Scale. - `6be3e5a`
- Improve camelcase_from_underscore(). - `0b40fe9`
- Do not try to reconnect DE1 or Scale while shutting down. - `bd21a93`
- Inbound (HTTP) API: Check DE1 and scale is_ready instead of is_connected. - `5de28e7`

MAPPING 4.0.0

- Rewrites `IsAt` to use an enum, rather than the class to define the target, simplifying package inclusion. - `78cea85`

12.13.4 Fixed

- Loop-level, exception-initiated shutdowns now terminate more cleanly. - `0b593d0`
- An error condition when no scale was present during a "shot" has been resolved. ffae2f
- An error condition when a DE1 connected and the profile was not yet known has been resolved - `58bbfad`
- AutoTareNotification and StopAtNotification now populate sender. - `9f39d08`
- A very early termination of the program (before processes are defined) now terminates more cleanly. - `4f95c34`
- ScaleProcessor: Reset the history if a gap in reports is too long, such as from a disconnect-reconnect sequence. - `48a35ca`

12.13.5 Removed

- Remove unused Config.set_logging(). - `2b104e6`
- Remove feature.py as previously incorporated into FeatureFlag. - `469ee96`

12.14 0.8.0 – 2021-09-28

12.14.1 Overview

This release focused on converting command-line executables to robust, self-starting, and supervised services. Both the core pyDE1 controller and the Visualizer uploader now can be started with `systemd` automatically at boot. Configuration of many parameters can be done through YAML files (simple, human-friendly syntax), by default in `/usr/local/pyde1/`. Command-line parameters, usable by the service unit files, can be used to override the config-file location.

Logging configuration may change prior to "beta". At this time it is only configurable in the output format and level for the *stderr* and *file* loggers.

By default, the *stderr* logger is at the WARNING level abd without timestamps, as it is managed through `systemd` when being run as a service. A command-line parameter allows for timestamped output at the DEBUG level for interactive use.

12.14.2 New

- Services run under `systemd`
  - Service ("unit") files for `pyde1.service` and `pyde1-visualizer.service`
  - Config files in YAML form
- Auto-off, configurable
- Track the IDs of connected Bluetooth devices for cleanup under Linux and disconnect them at the Bluez level in the case of a non-graceful exit
- MQTT supports authorization and access-control lists

- Visualizer: Don't upload short "shots", such as for flushing (configurable)
- Stop-at-weight offset configurable through `pyde1.conf`
- Database:
  - Self-initialize, if needed
  - Check for the proper schema at start
- Replay: config file and command-line switches allow easier configuration, including sequence ID and MQTT topic root

12.14.3 Changed

Warning: SIGHUP is no longer used for log rotation. It is a termination signal.

- Paths changed to `/var/log/pyde1` and `/var/lib/pyde1/pyde1.sqlite` by default (configurable)
- Refactored and unified shutdown processes
- Refactored supervised processes to handle uncaught exceptions and properly terminate for automated restart
- Visualizer: log to `pyde1-visualizer.log` by default
- Stop-at-weight internally includes 170 ms to account for the "fall-time" from the basket to the cup.
- Logging:
  - Switched to a file-watcher handler so that log rotation should be transparent, without the need of a signal
  - Provide better control of formatting and level for use with `systemd` (service) infrastructure
  - Change default file name to `pyde1.log`
  - Add `--console` command-line flag to provide timestamped, DEBUG-level output to assist in development and debugging
  - Adjust some log levels so that INFO-level logs are more meaningful
  - Removed last usages of `aiologger`
- The outbound API reports "disconnected" for the DE1 and scale when initialized

12.14.4 Fixed

- MQTT (outbound) API will now detect connection or authentication failures with the broker and terminate pyDE1
- FlowSequencer no longer raises exception when trying to report that the steam time is not managed directly by the software. (It is managed by the DE1 firmware.)
- Mass-flow estimates had an off-by-one error that was corrected
- Replay now properly reports sequence_id on gate notifications

12.14.5 Deprecated

- `find_first_and_load.py` (Use the APIs. It would have already been removed if previously deprecated)

12.14.6 Removed

- `ugly_bits.py` (previously deprecated)
- `try_de1.py` (previously deprecated)
- `DE1._recorder_active` and dependencies, including `shot_file.py` (previously deprecated)
- Profile `from_json_file()` (previously deprecated)
- `replay_vis_test.py` – Use `replay.py` with config or command-line options

12.15 0.7.0 – 2021-08-12

12.15.1 Schema Upgrade Required

> **Warning:** Backup your database before updating the schema.

See SQLite `.backup` for details if you are not familiar.

This adds columns for the `id` and `name` fields that are now being sent with `ConnectivityUpdate`

12.15.2 New

- Stand-alone app automatically uploads to Visualizer on shot completion
- PUT and GET of DE1_PROFILE_ID allows setting of profile by ID
- A stand-alone "replay" utility can be used to exercise clients, such as web apps
- Both the DE1 and scale will try to reconnect on unexpected disconnect
- Add `DE1IncompleteSequenceRecordError` for when write is not yet complete
- Variants of the EB6 profile at different temperatures

12.15.3 Changed

- Better logging when waiting for a sequence to complete times out
- Capture pre-sequence history at all times so "sync" is possible on replay
- Removed read-back of CUUID.RequestedState as StateInfo provides current state
- Removed "extra" last-drops check
- Allow more API requests when DE1 or scale is not ready
- Use "ready" and not just "connected" to determine if the DE1 or scale can be queried
- Allow [dis]connect while [dis]connected

- `ConnectivityChange` notification includes `id` and `name` to remove the need to call the API for them
- Improve error message on JSON decode by including a snippet around the error
- Set the default first-drops threshold to 0.0 for fast-flowing shots

RESOURCE 3.0.0

- Changes previously unimplemented UPLOAD_TO_ID

```
DE1_PROFILE_ID
DE1_FIRMWARE_ID
```

Database Schema 2

See `upgrade.001.002.sql`

```
PRAGMA user_version = 2;

BEGIN TRANSACTION;

ALTER TABLE connectivity_change ADD COLUMN id TEXT;
ALTER TABLE connectivity_change ADD COLUMN name TEXT;

END TRANSACTION;
```

12.15.4 Fixed

- Legacy "shot" files handle zero flow in "resistance" calculation
- Properly end recording of a sequence if it is interrupted
- FlowSequencer last-drops gate set during sequence
- Correct logic error in stopping recorder at end of sequence
- Correct reporting of not-connected conditions to HTTP API
- Correct scale-presence checking for PUT and PATCH requests
- Handle missing Content-Length header
- Incorrect error message around API request for Sleep removed
- `pyDE1.scanner` should now import properly into other code
- Steam-temperature setter now can set 140-160 deg. C
- Type errors in validation of API inputs properly report the expected type

12.16 0.6.0 – 2021-07-25

The Mimoja Release

> I am not sure how / where to store shots and profiles. I hate it to only have it browser local.

So do I. Wonder no longer.

12.16.1 New

A SQLite3 database now saves all profiles uploaded to the DE1, as well as capturing virtually all real-time data during all flow sequences, including a brief set of data from *before* the state transition.

Profiles are unique by the content of their "raw source" and also have a "fingerprint" that is common across all profiles that produce the same "program" for the DE1. Changing a profile's name alone does not change this fingerprint. Changing the frames in a profile without changing the name changes both the ID of the profile, as well as its fingerprint. These are both calculated using SHA1 from the underlying data, so should be consistent across installs for the same source data or frame set.

Profiles can also be searched by the customary metadata:

- Title
- Author
- Notes
- Beverage type
- Date added

`aiosqlite` and its dependencies are now required.

Legacy-style shot data can be extracted from the database by an application other than that which is running the DE1. Creating a Visualizer-compatible "file" for upload can be done in around 80-100 ms on a RPi 3B. If written to a physical file, it is also compatible with John Weiss' shot-plotting programs. See `pyDE1/shot_file/legacy.py`

The database retains the last-known profile uploaded to the DE1. If a flow sequence beings prior to uploading a profile, it is used as the "most likely" profile and identified in the database with the `profile_assumed` flag.

Note: The database needs to be manually initialized prior to use.

One approach is

```
sudo -u <user> sqlite3 /var/lib/pyDE1/pyDE1.sqlite3 \
< path/to/pyDE1/src/pyDE1/database/schema/schema.001.sql
```

12.16.2 Changed

Upload limit changed to 16 kB to accommodate larger profiles.

FlowSequencer events are now notified over `SequencerGateNotification` and include a `sequence_id` and the `active_state` for use with history logging.

`Profile.from_json()` now expects a string or bytes-like object, rather than a dict. This change is to ease capture of the profile "source" for use with history logging.

`ProfileByFrames.from_json()` no longer rounds the floats to maintain the integrity of the original source. They will still be rounded at the time that they are encoded into binary payloads.

Standard initialization of the DE1 now includes reading `CUUID.Versions` and `ShotSettings` to speed first-time store of profiles.

Robustness of shutdown improved.

Internal `Profile` class extended to capture "raw source", metadata, and UUIDs for both the raw source and the resulting "program" sent to the DE1.

12.16.3 Fixed

In `find_first_and_load.py`, `set_saw()` now uses the passed mass

12.16.4 Deprecated

`Profile.from_json_file()` as it is no longer needed with the API able to upload profiles. If needed within the code base, read the file, and pass to `Profile.from_json()` to ensure that the profile source and signatures are properly updated.

`DE1._recorder_active` and the contents of `shot_file.py` have been superseded by database logging.

12.16.5 Known Issues

The database name is hard-coded at this time.

`Profile.regenerate_source()` is not implemented at this time.

Occasionally, during shutdown, the database capture reports that it was passed `None` and an exception is raised. This may be due to shut down, or may be due to failure to retrieve an earlier exception from the task.

12.17 0.5.0 – 2021-07-14

12.17.1 New

Bluetooth scanning with API. See `README.bluetooth.md` for details

API can set scale and DE1 by ID, by first_if_found, or None

A list of logs and individual logs can be obtained with GET `Resource.LOGS` and `Routine.LOG`

`ConnectivityEnum.READY` added, allowing clients to clearly know if the DE1 or scale is available for use.

> **Warning:** Previous code that assumed that `.CONNECTED` was the terminal state should be modified to recognize `.READY`.

`examples/find_first_and_load.py` demonstrates stand-alone connection to a DE1 and scale, loading of a profile, setting of shot parameters, and disconnecting from these devices.

`scale_factory(BLEDevice)` returns an appropriate `Scale` subtype

`Scale` subtypes need to register their advertisement-name prefix, such as

```
Scale.register_constructor(AtomaxSkaleII, 'Skale')
```

Timeout on `await` calls initiated by the API

Use of connecting to the first-found DE1 and scale, monitoring MQTT, uploading a profile, setting SAW, all through the API is shown in `examples/find_first_and_load.py`

Example profiles: EB6 has 30-s ramp vs EB5 at 25-s

Add `timestamp_to_str_with_ms()` to `pyDE1.utils`

On an error return to the inbound API, an exception trace is provided, when available. This is intended to assist in error reporting.

12.17.2 Changed

HTTP API PUT/PATCH requests now return a list, which may be empty. Results, if any, from individual setters are returned as dict/obj members of the list.

Some config parameters moved into `pyDE1.config.bluetooth`

"find_first" functionality now implemented in `pyDE1.scanner`

`de1.address()` is replaced with `await de1.set_address()` as it needs to disconnect the existing client on address change. It also supports address change.

`Resource.SCALE_ID` now returns null values when there is no scale.

There's not much left of `ugly_bits.py` as its functions now should be able to be handled through the API.

On connect, if any of the standard register reads fails, it is logged with its name, and retried (without waiting).

An additional example profile was added. EB6 has 30-s ramp vs EB5 at 25-s. Annoying rounding errors from Insight removed.

MAPPING 3.1.0

Add Resource.SCAN and Resource.SCAN_RESULTS

See note above on return results, resulting in major version bump

Add `first_if_found` key to mapping for `Resource.DE1_ID` and `Resource.SCALE_ID`. If True, then connects to the first found, without initiating a scan. When using this feature, no other keys may be provided.

RESOURCE 2.0.0

Add

```
SCAN = 'scan'
SCAN_DEVICES = 'scan/devices'
```

```
LOG = 'log/{id}'
LOGS = 'logs'
```

12.17.3 Deprecated

`stop_scanner_if_running()` in favor of just calling `scanner.stop()`

`ugly_bits.py` for manual configuration now should be able to be handled through the API. See `examples/find_first_and_load.py`

12.17.4 Removed

`READ_BACK_ON_PATCH` removed as PATCH operations now can return results themselves.

`device_adv_is_recognized_by` class method on DE1 and Scale replaced by registered prefixes

Removed `examples/test_first_find_and_load.py`, use `find_first_and_load.py`

12.17.5 Known Issues

At least with BlueZ, it appears that a connection request while scanning will be deferred.

Implicit scan-for-address in the creation of a `BleakClient` does not cache or report any devices it discovers. This does not have any negative impacts, but could be improved for the future.

12.18 0.4.1 – 2021-07-04

12.18.1 Fixed

Import problems with `manual_setup` resolved with an explicit reference to the `pyDE1.ugly_bits` version. Local overrides that may have been in use prior will likely no longer used. TODO: Provide a more robust config system to replace this.

Non-espresso flow (hot water flush, steam, hot water) now have their accumulated volume associated with Frame 0, rather than the last frame number of the previous espresso shot.

12.19 0.4.0 – 2021-07-03

12.19.1 New

Support for non-GHC machines to be able to start flow through the API

More graceful shutdown on SIGINT, SIGQUIT, SIGABRT, and SIGTERM

Logging to a single file, `/tmp/log/pyDE1/combined.log` by default. If changed to, for example, `/var/log/pyDE1/`, the process needs write permission for the directory.

Note: Keeping the logs in a dedicated directory is suggested, as the plan is to provide an API where a directory list will be used to generate the `logs` collection. `/tmp/` is used for ease of development and is not guaranteed to survive a reboot.

Log file is closed and reopened on SIGHUP.

Long-running processes, tasks, and futures are supervised, with automatic restart should they unexpectedly terminate. A limit of two restarts is in place to prevent "thrashing" on non-transient errors.

12.19.2 Changed

Exceptions moved into `pyDE1.exceptions` for cleaner imports into child processes.

String-generation utilities moved from `pyDE1.default_logger` into `pyDE1.utils`

- `data_as_hex()`
- `data_as_readable()`
- `data_as_readable_or_hex()`

Remove inclusion of `pyDE1.default_logger` and replace with explicit calls to `initialize_default_logger()` and `set_some_logging_levels()`

Change from `asyncio-mqtt` to "bare" `paho-mqtt`. The `asyncio-mqtt` module is still a requirement as it is used in `examples/monitor_delay.py`

Controller now runs in its own process. Much of what was in `try_de1.py` is now in `controller.py`

Log entries now include the process name.

IPC between the controller and outbound (MQTT) API now uses a pipe and `loop.add_reader()` to improve robustness and ease graceful shutdown.

Several internal method signatures changed to accommodate changes in IPC. These are considered "internal" and do not impact the two, public APIs.

Significant refactoring to move setup and run code out of `try_de1.py` and into more appropriate locations. The remaining "manual" setup steps are now in `ugly_bits.py`. See also `run.py`

MAPPING 2.1.1

- Handle missing modules in "version" request by returning `None` (`null`)

RESOURCE 1.2.0

- Adds to `DE1ModeEnum` Espresso, HotWaterRinse, Steam, HotWater for use by non-GHC machines
- `.can_post` now returns False, reflecting that POST is and was not supported

Response Codes

- 409 — When the current state of the device does not permit the action
- `DE1APIUnsupportedStateTransitionError`
- 418 — When the device is incapable of or blocked from taking the action
- `DE1APIUnsupportedFeatureError`

12.19.3 Fixed

Resolved pickling errors related to a custom exception. It now is properly reported to and by the HTTP server.

Changed BleakClient initialization to avoid `AttributeError: 'BleakClientBlueZDBus' object has no attribute 'lower'` and similar for `'BleakClientCoreBluetooth'`

Exiting prior to device connection no longer results in `AttributeError: 'NoneType' object has no attribute 'disconnect'`

12.19.4 Deprecated

`try_de1.py` is deprecated in favor of `run.py` or similar three-liners.

12.19.5 Removed

"null" outbound API implementation — Removed as not refactored for new IPC. If there is a need, the MQTT implementation can be modified to only consume from the pipe and not create or use an MQTT client.

12.19.6 Known Issues

Exceptions on a non-supervised task or callback are "swallowed" by the default handler. They are reported in the log, but do not terminate the caller.

The API for enabling and disabling auto-tare and stop-at can only do so within the limits of the FlowSequencer's list of applicable states. See further `autotare_states`, `stop_at_*_states`, and `last_drops_states`

The main process can return a non-zero code even when the shutdown appeared to be due to a shutdown signal, rather than an exception.

The hard limit of two restarts should be changed to a time-based limit.

12.20 0.3.0 — 2021-06-26

12.20.1 New

Upload of profile (JSON "v2" format) available with PUT at de1/profile

> curl -D - -X PUT –data @examples/jmk_eb5.json http://localhost:1234/de1/profile

Line frequency GET/PATCH at de1/calibration/line_frequency implemented. Valid values are 50 or 60. This does not impact the DE1, only if 1/100 or 1/120 is used to calculate volume dispensed.

The HTTP API now checks to see if the request can be serviced with the current DE1 and Scale connectivity. This should help enable people that don't have a Skale II connected.

`BleakClientWrapped.willful_disconnect` property can be used to determine if the on-disconnect callback was called as a result of an intentional (locally initiated) or unintentional disconnect.

`BleakClientWrapped.name` provides the advertised device name under BlueZ and should not fail under macOS (or Windows).

12.20.2 Changed

MAPPING 2.1.0

- Adds `IsAt.internal_type` to help validate the string values for `DE1ModeEnum` and `ConnectivityEnum`. JSON producers and consumers should still expect and provide `IsAt.v_type`
- Enables `de1/profile` for PUT

RESOURCE 1.1.0

- Adds `DE1_CALIBRATION_LINE_FREQUENCY = 'de1/calibration/line_frequency'`

`DE1`, `FlowSequencer`, and `ScaleProcessor` are now `Singleton`.

`DE1()` and `Scale()` no longer accept an address as an argument. Use the `.address` property.

`BleakClientWrapped` unifies `atexit` to close connected devices.

12.20.3 Fixed

Better error reporting if the JSON value can not be converted to the internal enum.

Python 3.8 compatibility: Changed "subscripted" type hints for `dict`, `list`, and `set` to their capitalized versions from `typing`, added replacement for `str.removeprefix()`

Running on macOS with `bleak` 0.12.0 no longer raises device-name lookup errors. This was not a `bleak` issue, but due to hopeful access to its private internals.

12.20.4 Removed

`DE1()` and `Scale()` no longer accept an address as an argument. Use the `.address` property.

12.21 0.2.0 — 2021-06-22

12.21.1 Inbound Control and Query API

An inbound API has been provided using a REST-like interface over HTTP. The API should be reasonably complete in its payload and method definitions and comments are welcomed on its sufficiency and completeness.

Both the inbound and outbound APIs run in separate *processes* to reduce the load on the controller itself.

GET should be available for the registered resources. See, in `src/pyDE1/dispatcher`

- `resource.py` for the registered resources, and
- `mapping.py` for the elements they contain, the expected value types, and how they nest.

`None` or `null` are often used to me "no value", such as for stop-at limits. As a result, though similar, this is not an RFC7368 JSON Merge Patch.

In Python notation, `Optional[int]` means an `int` or `None`. Where `float` is specified, a JSON value such as `20` is permitted.

GET presently returns "unreadable" values to be able to better show the structure of the JSON. When a value is unreadable, `math.nan` is used internally, which is output as the JSON `NaN` token.

GET also returns empty nodes to illustrate the structure of the document. This can be controlled with the `PRUNE_EMPTY_NODES` variable in `implementation.py`

Although PATCH has been implemented for most payloads, PUT is not yet enabled. PUT will be the appropriate verb for`DE1_PROFILE` and `DE1_FIRMWARE` as, at this time, in-place modification of these is not supported. The API mechanism for starting a firmware upload as not been determined, as it should be able to abort as it runs in the background, as well as notify when complete. Profile upload is likely to be similar, though it occurs on a much faster timescale.

If you'd like the convenience of a GET of the same resource after a PATCH, you can set `READ_BACK_ON_PATCH` to `True` in `dispacher.py`

> The Python `http.server` module is used. It is not appropriate for exposed use. There is no security to the control and query API at this time. See further https://docs.python.org/3/library/http.server.html

It is likely that the server, itself, will be moved to a uWSGI (or similar) process.

With either the present HTTP implementation or a future uWSGI one, use of a webserver, such as `nginx`, will be able to provide TLS, authentication, and authorization, as well as a more "production-ready" exposure.

12.21.2 Other Significant Changes

- `ShotSampleWithVolumeUpdates` (v1.1.0) adds `de1_time`. `de1_time` and `scale_time` are preferred over `arrival_time` as, in a future version, these will be estimates that remove some of the jitter relative to packet-arrival time.
- To be able to keep cached values of DE1 variables current, a read-back is requested on each write.
- `NoneSet` and `NONE_SET` added to some `enum.IntFlag` to provide clearer representations
- Although `is_read_once` and `is_stable` have been roughed in, optimizations using them have not been done

- Disabled reads of `CUUID.ReadFromMMR` as it returns the request itself, which is not easily distinguishable from the data read. These two interpret their `Length` field differently, making it difficult to determine if `5` is an unexpected value or if it was just that 6 words were requested to be read.
- Scaling on `MMR0x80LowAddr.TANK_WATER_THRESHOLD` was corrected.

12.22 0.1.0 — 2021-06-11

12.22.1 Outbound API

An outbound API (notifications) is provided in a separate process. The present implementation uses MQTT and provides timestamped, source-identified, semantically versioned JSON payloads for:

- DE1
  - Connectivity
  - State updates
  - Shot samples with accumulated volume
  - Water levels
- Scale
  - Connectivity
  - Weight and flow updates
- Flow sequencer
  - "Gate" clear and set
    - Sequence start
    - Flow begin
    - Expect drops
    - Exit preinfuse
    - Flow end
    - Flow-state exit
    - Last drops
    - Sequence complete
  - Stop-at-time/volume/weight
    - Enable, disable (with target)
    - Trigger (with target and value at trigger)

An example subscriber is provided in `examples/monitor_delay.py`. On a Raspberry Pi 3B, running Debian *Buster* and `mosquitto` 2.0 running on `::`, median delays are under 10 ms from *arrival_time* of the triggering event to delivery of the MQTT packet to the subscriber.

Packets are being sent with *retain* True, so that, for example, the subscriber has the last-known DE1 state without having to wait for a state change. Checking the payload's `arrival_time` is suggested to determine if the data is fresh enough. The *will* feature of MQTT has not yet been implemented.

A good introduction to MQTT and MQTT 5 can be found at HiveMQ:

- https://www.hivemq.com/mqtt-essentials/
- https://www.hivemq.com/blog/mqtt5-essentials-part1-introduction-to-mqtt-5/

One good thing about MQTT is that you can have as many subscribers as you want without slowing down the controller. For example, you can have a live view on your phone, live view on your desktop, log to file, log to database, all at once.

12.22.2 Scan For And Use First DE1 And Skale Found

Though "WET" and needing to be "DRY", the first-found DE1 and Skale will be used. The Scale class has already been designed to be able to have each subclass indicate if it recognizes the advertisement. Once DRY, the scanner should be able to return the proper scale from any of the alternatives.

Refactoring of this is pending the formal release of `BleakScanner.find_device_by_filter(filterfunc)` from bleak PR #565

CHAPTER

THIRTEEN

INDICES AND TABLES

- genindex
- modindex
- search