Displaying My Washing Machine's Remaining Time With curl, jq, and pizauth

April 11 2023

A couple of months ago our washing machine produced a considerable quantity of smoke when I opened its door, which I interpreted as a suggestion that a replacement was needed. After rummaging around the internet for advice, a couple of days later a new Miele washing machine took its place in our home.

As well as boggling my mind at how much better the new machine was than the cheap machine it replaced, I was pleased to discover that Miele make available a third party API which one can use to interact with the machine. Putting aside any worries about connecting a high-speed rotating device to the internet, I decided to have a poke around. Although the API is more limited in practise than in theory – the API has support for setting values, but those mostly aren't used – I eventually realised it can help me solve one recurring problem.

Like most washing machines, ours beeps to notify me that it's finished. However, I can't always empty it straight away, the beep is obnoxious and repetitive, and it ends up disturbing everyone in the house. I can turn off the beep, but then I don't know when the machine has finished. Miele's app can notify me, but it regularly logs me out, and finding out the time remaining is a chore. What I really wanted is a countdown of the remaining time and notification on my desktop computer. Fortunately, I can do exactly what I want on my desktop computer using basic Unix tools. Here's what a sped-up version of the result looks like (the middle part is hugely sped up; the end part is sped up slightly less so):

In this post I'm going to explain how I got this working with the Unix shell, curl, jq, and pizauth. Interested readers should not have much difficulty adapting these techniques for other similar situations.

Blog archi
Last 10 blog posts
Displaying My Was Machine's Remai Time With curl, jq pizauth
pizauth: dump and
How Big Should a Programming La Be?
Rust's Two Kinds o Make for Better C
Scheduling my Ele Usage
Why Aren't Progra Language Specif Comprehensive?
Distinguishing an I from a Compiler
try_repeat release
Why We Need to K and Recursive De Parsing Techniqu
Compiled and Inter Languages: Two Saying Tomato

To get started, I first had to register my washing machine to my email address with Miele's app. It's not the smoothest experience: I had to give it a couple of tries before it worked, but at least it's a one-off task.

With my washing machine registered to my email address, I then needed to set up OAuth2 so that I can use the API from my computer. First, I needed an OAuth2 client ID and client secret [1]. Miele allows anyone to generate their own client ID and client secret by registering ourselves as a developer with Miele which means giving them a random "app name" and the same email address used to register the device in the app. I then had to register that app as one I'm allowing to use on my Miele account.

I then needed an OAuth2 tool: I used pizauth because, well, I wrote it. My `~/.config/pizauth.conf` file contains Miele's authorisation and token URIs and the client ID and client secret [2] I got by registering myself with Miele:

```
account "miele" {
  auth_uri = "https://api.mcs3.miele.com/thirdparty/login/";
  token_uri = "https://api.mcs3.miele.com/thirdparty/token/"
  client_id = "8nka83ka-38ak-38a9-38ah-ah38uaha82hi";
  client_secret = "HOaniszumazr978toh789th789aAGa83";
}
```

I then ran `pizauth server` which causes pizauth to listen for requests for access tokens. The first time that pizauth is asked to display an access token it will report an error which contains the URL needed to authenticate yourself with Miele:

```
$ pizauth show miele
ERROR - Access token unavailable until authorised with URL h
```

Plugging that URL into a web browser, using the same email address and password as I used in Miele's App, and selecting a country (in my case "Great Britain") completes authentication, and pizauth now works as expected.

Getting the device ID

The Miele API is a RESTful API which we can query using curl, though we have to send an OAuth2 access token each time we want it to do something for us. Let's start by asking the API to list all the devices I've registered with Miele:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $(pizauth show miele)" \
  https://api.mcs3.miele.com/v1/devices
```

The only surprising part of this is the `--header` part: `pizauth show miele` displays an access token on `stdout` which is incorporated into the HTTP request. A couple of seconds later, curl prints a vast wodge of JSON to stdout:

```
{"000173036828":{"ident":{"type":{"key_localized":"Device ty
```

is irritating because in amongst that JSON wodge is the device ID of my washing machine. That ID is required to use later parts of the API so I need to fish it out somehow. Fortunately, jq allows us to easily extract the field in question:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $(pizauth show miele)" \
  https://api.mcs3.miele.com/v1/devices \
  | jq -r '.[] | .ident.deviceIdentLabel.fabNumber'
000173036828
```

That jq command actually prints out every device ID I've registered with Miele. Since I only have a single Miele device it's fairly obvious that the single ID displayed is for my washing machine, but if you have multiple IDs, how can you tell them apart? The curl command stays the same as before, but now I use jq to fish out the model number and even a human readable name:

```
$ curl ... \
  | jq -r \
    '.[]
     | .ident
     | (.deviceIdentLabel
     | (.fabNumber + ": " + .techType))
       + " " + .type.value_localized'
000173036828: WEG365 Washing machine
```

It's now clear that "000173036828" is the device ID I want going forward.

Getting the remaining time

Now that I have its device ID, I can use a different part of the API to see my washing machine's current state:

```
$ curl \
  --silent \
  --header "Authorization: Bearer $(pizauth show miele)" \
  https://api.mcs3.miele.com/v1/devices/000173036828/state
```

This gives me another huge wodge of JSON of which only the `remainingTime` field is interesting:

```
$ curl ... \
  | jq -r '.remainingTime'
[
  0,
  56
]
```

That means I've got 0 hours and 56 minutes left — but that formatting is rather horrible. My first attempt might be:

```
$ curl ... \
  | jq -r '.remainingTime[0] + .remainingTime[1]'
54
```

Instead, I want to convert each integer to a string and then concatenate them:

```
$ curl ... \
  | jq -r \
    '(.remainingTime[0] | tostring)
     + ":"
     + (.remainingTime[1] | tostring)'
0:54
```

What happens when the remaining time goes below 10?

```
$ curl ... \
  | jq -r
    '(.remainingTime[0] | tostring)
     + ":"
     + (.remainingTime[1] | tostring)'
0:9
```

That's rather confusing! We need to make sure the minutes are always two digits. There is no builtin way of padding numbers in jq, but we can multiply strings by integers so with a bit of thought we can write:

```
$ curl ... \
  | jq -r \
    '(.remainingTime[0] | tostring)
     + ":"
     + (.remainingTime[1] | tostring
        | (length | if . >= 2 then "" else "0" * (2 - .) enc
     + (.remainingTime[1] | tostring)'
0:09
```

The API also tells us what phase the washing machine is currently in, which I find interesting, so let's print that out:

```
$ curl ... \
  | jq -r '.programPhase.value_localized'
Rinsing
```

A Countdown

At this point, I can print out the remaining time for the current load once: what I really want to do is update this so that I have a meaningful countdown. Before we try and go further, let's bundle what we've got into a simple script so that we don't have to continually enter commands into a terminal:

```
#! /bin/sh

set -e

DEVICE_ID=000173036828

get() {
  curl \
    --silent \
    --header "Authorization: Bearer $(pizauth show miele)" \
    https://api.mcs3.miele.com/v1/devices/${DEVICE_ID}/state
  | jq -r "$1"
case $1 in
  phase) get .programPhase.value_localized ;;
  remaining_time)
    get \
      '(.remainingTime[0] | tostring)
       + ":"
       + (.remainingTime[1] | tostring
          | (length
             | if . >= 2 then "" else "0" * (2 - .) end))
       + (.remainingTime[1] | tostring)'
    ;;
esac
```

Let's call that script `washing_machine`:

```
$ washing_machine remaining_time
0:42
$ washing_machine phase
Rinsing
```

What we now want to do is create a loop which prints out the remaining time. A first cut, which updates the remaining time every 30 seconds is as follows:

```
while [ true ]; do
  printf "\r%s (%s)" \
    $(washing_machine remaining_time) \
    $(washing_machine phase)
  sleep 30
done
```

That works surprisingly well, but has two flaws. First, when the load is finished, the loop doesn't terminate. Second, `\r` is "carriage return" which means that the countdown keeps displaying text on the same line. This works well provided any new text is the same, or longer, length than what came before. However, if the new text is shorter we end up with odd output such as:

```
0:32 (Rinsing)h)
```

We can fix the first problem by noticing that the load is complete when `remaining_time` returns "0:00":

```
while [ true ]; do
  t=$(washing_machine remaining_time)
  if [[ $t == "0:00" ]]; then
    break
  fi
  printf "\r%s (%s)" \
    "$t" \
    "$(washing_machine phase)"
  sleep 30
done
```

We can fix the second problem in various ways, but the most portable I know of uses `tput el` after the carriage return. This causes all text between the cursor (which `\r` has moved to the start of the line) and the end of the line to be cleared:

```
  t=$(washing_machine remaining_time)
  if [[ $t == "0:00" ]]; then
    break
  fi
  printf "\r%s%s (%s)" \
    $(tput el) \
    "$t" \
    "$(washing_machine phase)"
  sleep 30
done
```

At this point, I'm into nitpicking mode: it irritates me that my countdown has a blinking cursor next to it. I can turn that off and on with `tput civis` and `tput cnorm` respectively. However, I need to make sure that if my program is terminated early (e.g. because I press Ctrl-C) that cursor blinking is restored. Fortunately I can use the `trap` command to restore blinking if this happens, so I can adjust my program as follows:

```
tput civis
trap "tput cnorm" 1 2 3 13 15
while [ true ]; do
  ...
done
tput cnorm
```

Last, but not least, when the load has finished I use `notify-send` to pop a message up in my desktop telling me the load is complete:

```
notify-send -t 60000 "Washing finished"
```

Now that I've got that sorted out, I can put it into my script (keeping the now-recursive calls as-is), which now looks as follows:

```
#! /bin/sh

set -e

DEVICE_ID=000173036828

get() {
  curl \
    --silent \
    --header "Authorization: Bearer $(pizauth show miele)" \
    https://api.mcs3.miele.com/v1/devices/${DEVICE_ID}/state
  | jq -r "$1"
}

case $1 in
  countdown)
    tput civis
    trap "tput cnorm" 1 2 3 13 15
    while [ true ]; do
      t=$(washing_machine remaining_time)
      if [[ $t == "0:00" ]]; then
        break
      fi
      printf "\r%s%s (%s)" \
        $(tput el) \
        "$t" \
    done
    tput cnorm
    echo
    notify-send -t 60000 "Washing finished"
    ;;
  phase) get .programPhase.value_localized ;;
  remaining_time)
    get \
      '(.remainingTime[0] | tostring)
       + ":"
       + (.remainingTime[1] | tostring
          | (length
             | if . >= 2 then "" else "0" * (2 - .) end))
       + (.remainingTime[1] | tostring)'
    ;;
esac
```

And now each time I use my washing machine I need only execute the following at the command line:

```
$ washing_machine countdown
```

Summary

Open standards are great, especially when they can be used with simple tools. Nearly everyone has curl installed on their machine already and most people can easily install jq via their favourite package manager. At the time of writing, I think OpenBSD is the only OS which makes pizauth easily installable, but perhaps that will change as time goes on. Still, hopefully the underlying message of this post is clear: we can often do a lot with simple tools!

If you'd like updates on new blog posts: follow me on Mastodon or Twitter; or subscribe to the RSS feed; or subscribe to email updates:

email@example.com Subscribe

Footnotes

[1] The latter doesn't make much sense in this context, and it's not required by the OAuth2 standard, but Miele seem to require it.
[2] No, these are not my real client CD and client secret, but they do follow the same format as the real thing. I'll be doing something similar for other IDs in this post too.

Comments

Name

Homepage (e.g. https://example.com/) (optional)

email (e.g.email@example.com) (used only to verify your comment: it is not displayed)

[✓] Notify me of further comments

Comment Preview

bert (2023-04-12 10:29:11) Permalink

Thanks, Laurence. This was a very enjoyable read and a cute little exercise. Subscribed!

Alan Howlett (2023-04-12 11:35:49) Permalink

Nice writeup. Your style is very helpful in clarifying the processes involved.

Lucien (2023-04-12 12:08:30) Permalink

I never manage to understand the point of over-engineered projects like these. It doesn't solve an actual problem, it's an absurd solution looking for a problem. The washing machine already tells you on its own when it's done, and before starting the machine it will also tell you the required programme time on its display, so you can use your wrist watch, or an egg timer, or just "sleep XYZ; notify-send blah" in a terminal. All this effort brings no actual practical benefit. I don't get it.

@Lucien (2023-04-12 13:39:28) Permalink

It's funny, there's always this one guy in the comments who's apparently never heard of the concept called "fun".

urig (2023-04-12 15:46:02) Permalink

Thanks for the clear write-up. I've saved it to my pocket in case I need to do something similar in the future.

It's a bit sad that one has to go through Miele's servers to get the data from one room in the house to another. Imagine if the REST server (securely) ran on the washing machine and was available through LAN. :)