Appla

Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

Phil Sturgeon
Working as a Platform Engineer @WeWork, helping their systems crash less frequently than I crash my bikes.
Mar 9 · 4 min read

Creating API Specifications from Bulls**t

So you want to get into writing specifications for your API. You're tired of forcing developers to guess what your API contracts might be, you want beautiful documentation, and want to jump on the OpenAPI / JSON Schema bandwagon like all the cool kids.

Getting started can be tough. I don't just mean learning how OpenAPI and JSON Schema work—a talk covering this will be out soon!—but actually writing up your first real actual API seems like a daunting challenge.

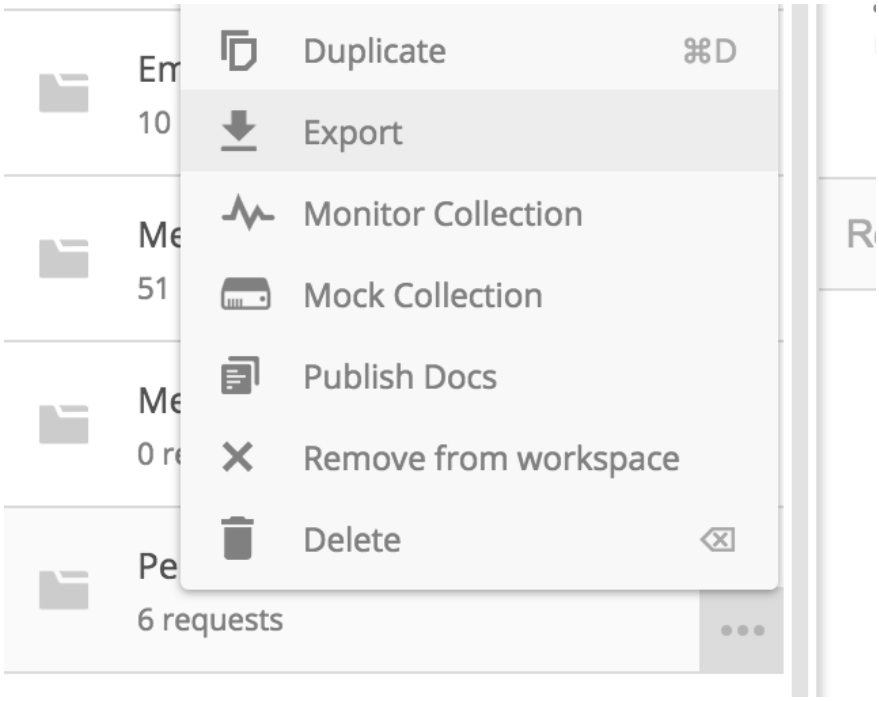
A few tricks are available to help you out.

Convert from Postman

Most teams will at the very least half-assedly update a Postman Collection for their more important endpoints, and sometimes these are vaguely up-to-date.

Postman Collections are essentially just another API specification format, so taking this can be a good start. Accepting that it is not going to be a complete list of endpoints, aware that it is missing all sorts of metadata like field descriptions, formats, enum values, etc., you can use this as a head start.

Step 1 is to Export your Postman Collection.



Open the Postman application, find your API and click Export.

There are a lot of "Postman to Swagger" or "Postman to OpenAPI" tools out there, but if you are luck enough to find one that works, it is definitely outdated. Many of them only work wiht Postman Collection v1, and they all only convert to OpenAPI v2.0…

The best closest CLI tool out there is API Flow, but even after sending 7 pull requests that tool is still OpenAPI v2.0 only, and their version is completely broken. Give my fork a try if you are interested.

Luckily the APImatic Transformer is amazing. You can use their website to upload the Postman Collection, and download the OpenAPI v3.0 in JSON or YAML. Fantastic!

Their output is not perfect, but it is a start. Run the Speccy lint command on the output to get advice on things to add, and use the serve command to get a preview in your browser.

Generate from Real Traffic

From a mere chunk of JSON, a "schema generator" can attempt to make a very basic schema for you. If something looks like a string, and maybe contains a date, it can rather easily list the field as `type`:

`string` and `format: date-time`. That might not be mind blowing, and it's certainly not the most advanced usage of schema, but it is rather useful when you have a whole API with a few thousand fields spread over various resources.

There are a few tools around for OpenAPI and JSON Schema.

Update 2018–05–16: *Smartbear (creators of the Swagger tooling for OpenAPI) have just released Swagger Inspector, which is explained in this blog post. You basically use it like a HTTP client, import it into SwaggerHub, then export as OpenAPI v2 or v3 and use it however you like.*

If you want to create JSON Schema there are a whole bunch of JSON Schema Generators, including one written in Ruby, and one written for NodeJS.

There is also jsonschema.net, which is a really handy online tool that supports loads of different drafts of JSON Schema.

Throw JSON in on the left, get a bunch of JSON Schema out on the right.

Caveat, if you are referencing these JSON Schema files in an OpenAPI v3.0 specification, make sure your JSON Schema Generator is outputting the JSON Sceham as draft4, or it will break OpenAPI.

Generate Specs from Code

Annotation based systems aare terrifying and not something I can understand people wanting to do.

That said, if you have a strictly typed language like Go, you probably have enough information in your code-base to generate some OpenAPI specs. To create a bunch of specs rather quickly, goswagger can help you out.

```
swagger generate spec -o ./swagger.json
```

It will only return OpenAPI v2 right now, but slap the v2 spec through APImatic Transformer and you'll have v3 ready to go.

Onwards

When you have both your data model defined in OpenAPI or JSON Schema, and your service model defined in OpenAPI or JSON HyperSchema, you're off to the races!

What is data model vs what is service model.

Now all you need to do is save those specification files in your API's GitHub repository, create some beautiful documentation, make sure you're keeping them up-to-date, and start looking into other awesome uses of JSON Schema like client-side validation.