# API Governance without tears

Lorna Mitchell, Redocly

#### API governance

Good API governance is invisible

#### Without it:

- designing changes is more difficult
- changes get rejected and need repeat work
- APIs become inconsistent
- more difficult to adopt an API

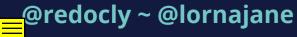
#### Start with standards

## Hypertext links

HAL: Hypertext Application Language https://en.wikipedia.org/wiki/Hypertext\_Application\_Language

```
{
  "_links": {
    "self": {
        "href": "http://example.com/api/book/hal-cookbook"
    }
},
  "id": "hal-cookbook",
  "name": "HAL Cookbook"
}
```

Common for pagination, and nested data resources



#### Problem details RFC7807

https://datatracker.ietf.org/doc/html/rfc7807

Common body format with error responses

#### Authentication

Adopt an existing standard

- OAuth2
- JWT
- access token / API key
- basic authentication

Use 401: Unauthorized and 403: Forbidden appropriately



#### OpenAPI

Machine-readable API description https://openapis.org

#### OpenAPI structure

```
openapi: 3.1.0
                                          // (this example is not to scale)
info:
servers: [ https://example.com ]
paths:
  "/events":
    post:
      operationId: createEvent
      responses:
        200:
    get:
      operationId: listEvent
      responses:
        200:
components:
  schemas:
    event: {}
```

# Design good governance

Start with an existing ruleset

Rules for compliance

- Every endpoint must have security defined
- Contact information for API owner
- Headers required for all endpoints
- License information provided

Pull request workflow provides an audit

Rules for happy engineers

- Operation IDs must be unique
- Every operation has an error response
- Paths cannot be ambiguous

```
paths:
    "/user/{name}":
    ...

"/{team}/admin_user":
    ...
```

A matter of taste

- Use plurals in path segments
- Standardise on case (kebab-case or camelCase)
- No trailing slashes in paths

Rules for developer delight

- Tags should have descriptions
- Parameters must have descriptions and examples

```
parameters:
- in: query
  name: userId
  description: |
    User ID, returned by the `listUser` endpoints.
    (tip: find your own user ID by calling `/user/me`)
  schema:
    type: string
    example: "U0123"
```

#### Non-lintable standards

The machines cannot do everything \*

Humans must review changes too

- would you want to use this API?
- is the naming sensible and intuitive?
- is the change consistent with the existing API?

\*Try asking an AI "does this naming make sense?"



# API change management

API governance: it's a practice, not a job title

#### Design-first APIs

Change the OpenAPI files only, open a pull request

#### Then, use CI to:

- publish docs
- spin up a mock server
- check it meets the standards
- get humans to do their review

# Tools for API Governance

https://openapi.tools

## API lifecycle tools

- Linting, standard rulesets or make your own
- Documentation, build a preview of proposed changes
- Mock servers, try before you build
- Extensions, improve an existing description

## Improve API experience

- Publish the OpenAPI
- Add more examples to your API descriptions
- Use writers to improve the words and descriptions
- Add metadata to help code generators
- Use a design-first, code-style workflow

# Good API governance (without tears)

#### Resources

- https://lornajane.net
- https://redocly.com
- https://apistylebook.com
- https://openapi.tools
- https://github.com/APIs-guru/openapi-directory
- https://apisyouwonthate.com