Improve your API with OpenAPI

Rob Allen

PHPSW, November 2021

The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service

https://spec.openapis.org/oas/latest.html

It's about

documentation

It's about

design-first

It's about

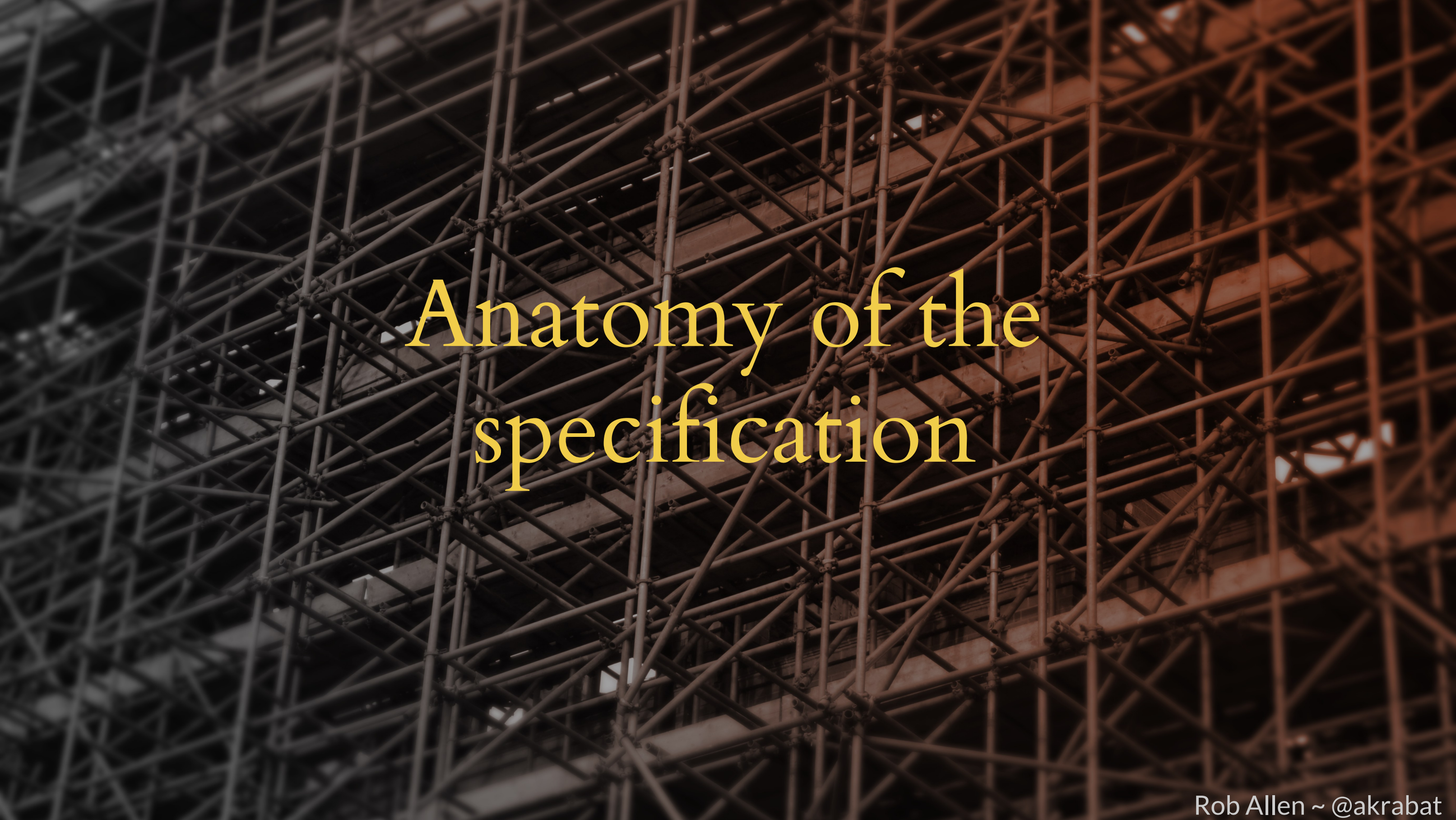
communicating changes

It's about

development workflows

It's about

a contract



Anatomy of the specification

openapi.yaml

```yaml
openapi: "3.1.0"  # or "3.0.3"
info: # ...
servers: # ...
paths: # ...
webhooks: # ...
components: # ...
security: # ...
tags: # ...
externalDocs: # ...
```

openapi.yaml

```yaml
info:
  title: Rock-Paper-Scissors
  version: "1.0.0"
  description: >
    An implementation of Rock-Paper-Scissors.
  contact:
    name: "Rob Allen"

servers:
  - url: https://rock-paper-scissors.example.com
    description: "RPS production API"
```

openapi.yaml

```yaml
paths:
  post:
    summary: Create a new game
    description: >
        Create a new game of Rock-Paper-Scissors.
    requestBody:
      # ...
    responses:
      # ...
```

openapi.yaml

```
requestBody:
  description: Game to add
  required: true
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/NewGameRequest'
```

Reuse of objects

`$ref` allows us to define one & use in many places

```
components:
  schemas:
    GameId:
      type: string
      examples:
        - "2BC08389-885A-4322-80D0-EF0DE2D7CD37"
      format: "uuid"
    Player:
      type: string
      example: "Rob"
```

Build on other components

```yaml
schemas:
  NewGameRequest:
    properties:
      player1:
        $ref: '#/components/schemas/Player'
      player2:
        $ref: '#/components/schemas/Player'
    required:
      - player1
      - player2
    examples:
      - '{"player1":"Ian", "player2":"Dave"}'
```

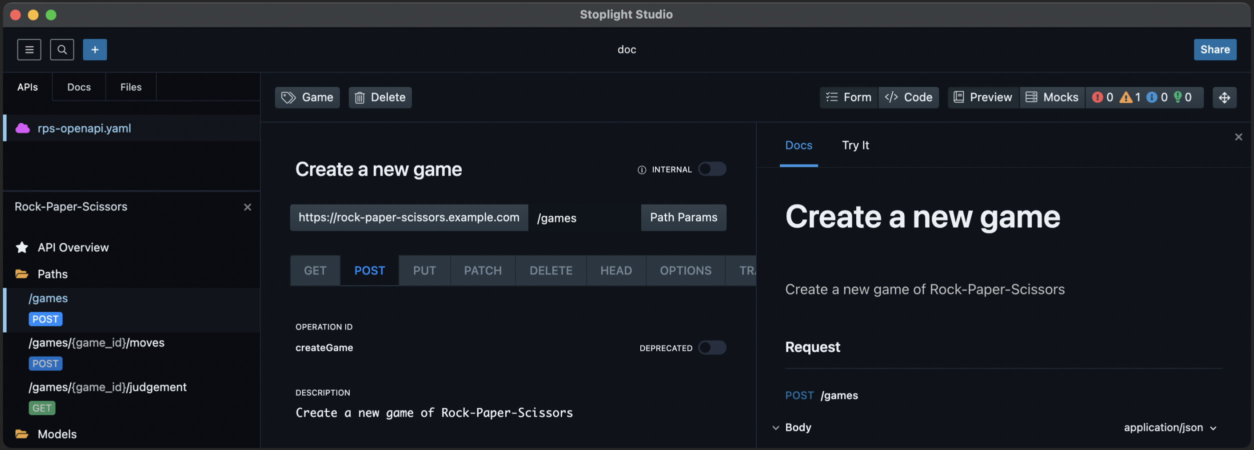
openapi.yaml

```yaml
responses:
  "201":
    $ref: '#/components/responses/NewGameResponse'
  "400":
    $ref: '#/components/responses/NewGameError'
  "500":
    $ref: '#/components/responses/InternalServerError'
```

Writing your spec

Editing

- Editor with plugins: vim, VS Code, etc
- GUI: Stoplight, OpenAPI-GUI, Swagger Editor



Create a new game

https://rock-paper-scissors.example.com /games

OPERATION ID
createGame

DESCRIPTION
Create a new game of Rock-Paper-Scissors

Create a new game

Create a new game of Rock-Paper-Scissors

POST /games

Linting

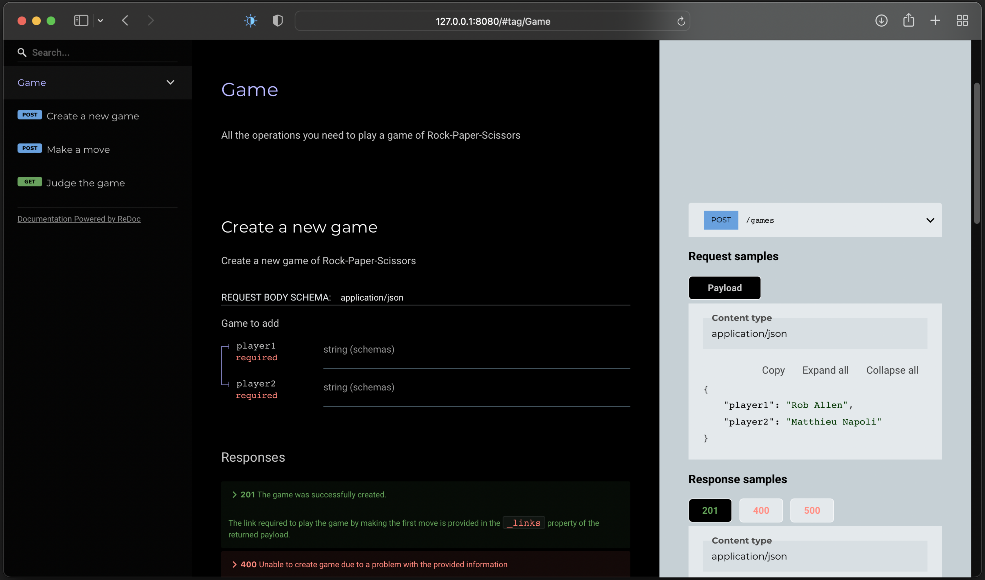
CLI tools: Spectral, openapi-spec-validator, etc.

```
$ spectral lint rps-openapi.yaml

.../slim4-rps-api/doc/rps-openapi.yaml
3:6  warning  info-contact  Info object must have
  "contact" object.  info

□ 1 problem (0 errors, 1 warning, 0 infos, 0 hints)
```

BOLSILLO
POESIA
RETRATOS DEL ARTE
MIGUEL SAYAGO
MATERIALES
MARY BEARD

Docs

Docs


127.0.0.1:8080/#tag/Game

Search...

Game

POST Create a new game

POST Make a move

GET Judge the game

Documentation Powered by ReDoc

Game

All the operations you need to play a game of Rock-Paper-Scissors

Create a new game

Create a new game of Rock-Paper-Scissors

REQUEST BODY SCHEMA: application/json

Game to add

player1 required	string (schemas)
player2 required	string (schemas)

Responses

> 201 The game was successfully created.

The link required to play the game by making the first move is provided in the `_links` property of the returned payload.

> 400 Unable to create game due to a problem with the provided information

POST /games

Request samples

Payload

Content type
application/json

Copy Expand all Collapse all

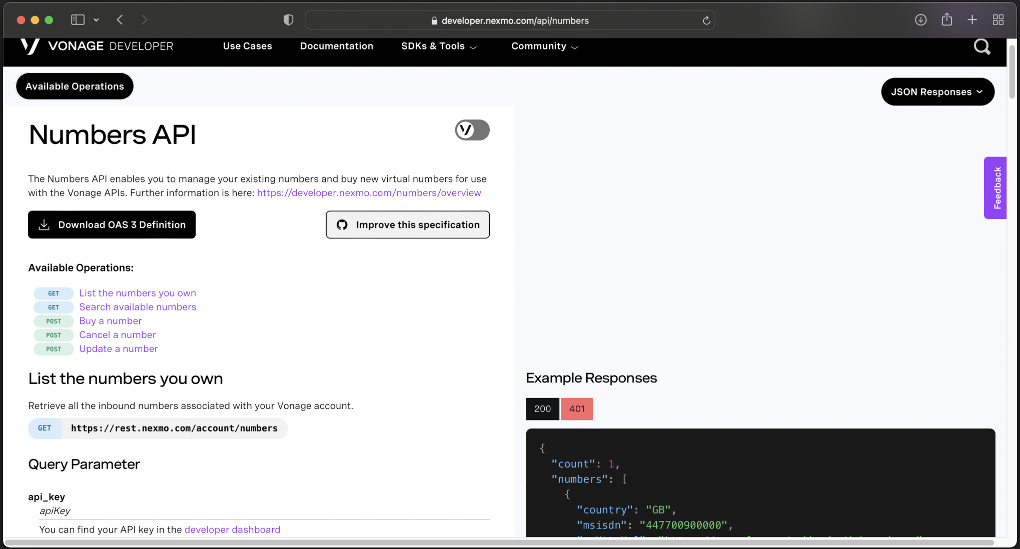
```
{
    "player1": "Rob Allen",
    "player2": "Matthieu Napoli"
}
```

Response samples

201 400 500

Content type
application/json

Docs



Numbers API

The Numbers API enables you to manage your existing numbers and buy new virtual numbers for use with the Vonage APIs. Further information is here: https://developer.nexmo.com/numbers/overview

Download OAS 3 Definition | Improve this specification

Available Operations:

- GET List the numbers you own
- GET Search available numbers
- POST Buy a number
- POST Cancel a number
- POST Update a number

List the numbers you own

Retrieve all the inbound numbers associated with your Vonage account.

`GET https://rest.nexmo.com/account/numbers`

Query Parameter

api_key
apiKey
You can find your API key in the developer dashboard

Example Responses

200 401

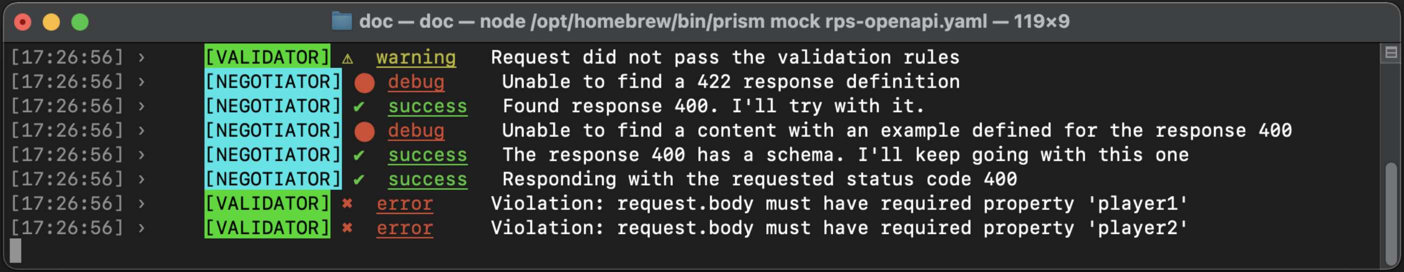
```
{
  "count": 1,
  "numbers": [
    {
      "country": "GB",
      "msisdn": "447700900000",
```

Rob Allen ~ @akrabat

Developers

Mock server

```
$ prism mock rps-openapi.yaml

$ curl http://127.0.0.1:4010/games -d '{}'
{"message":"Must provide both player1 and player2"}
```



doc — doc — node /opt/homebrew/bin/prism mock rps-openapi.yaml — 119×9

```
[17:26:56] ›     [VALIDATOR] ⚠  warning   Request did not pass the validation rules
[17:26:56] ›     [NEGOTIATOR] ●  debug     Unable to find a 422 response definition
[17:26:56] ›     [NEGOTIATOR] ✔  success   Found response 400. I'll try with it.
[17:26:56] ›     [NEGOTIATOR] ●  debug     Unable to find a content with an example defined for the response 400
[17:26:56] ›     [NEGOTIATOR] ✔  success   The response 400 has a schema. I'll keep going with this one
[17:26:56] ›     [NEGOTIATOR] ✔  success   Responding with the requested status code 400
[17:26:56] ›     [VALIDATOR] ✖  error     Violation: request.body must have required property 'player1'
[17:26:56] ›     [VALIDATOR] ✖  error     Violation: request.body must have required property 'player2'
```

Rob Allen ~ @akrabat

Validation

The `schema` section can be used to validate the request *and* response

- Validate early and return a 422
- Validate that we return what we say we will
- Put it in CI to prevent regressions

But I already have validation!

Your code:

- isn't good enough!
- isn't reusable!
- doesn't match the docs!

But I already have validation!

Your code:

- isn't good enough!
- isn't reusable!
- doesn't match the docs!

However…

Business logic validation still needed!

Validation in PHP

OpenAPI 3.0:

`league/openapi-psr7-validator`

OpenAPI 3.1:

`opis/json-schema`

Validation middleware

```
public function process($request, $handler)
{
  // Determine OAS schema for this route into $schema

  // Validate request
  $builder = (new ValidatorBuilder())->fromSchema($schema);
  $validator = $builder->getServerRequestValidator();
  try {
    $match = $validator->validate($request);
  } catch (ValidationFailed $e) {
    throw new RequestValidationFailed($e);
  }
```

Validation middleware

```
public function process($request, handler)
{
  // Determine OAS schema for this route into $schema

  // Validate request
  $builder = (new ValidatorBuilder())->fromSchema($schema);
  $validator = $builder->getServerRequestValidator();
  try {
    $match = $validator->validate($request);
  } catch (ValidationFailed $e) {
    throw new RequestValidationFailed($e);
  }
```

Validation middleware

```
public function process($request, handler)
{
    // Determine OAS schema for this route into $schema

    // Validate request
    $builder = (new ValidatorBuilder())->fromSchema($schema);
    $validator = $builder->getServerRequestValidator();
    try {
        $match = $validator->validate($request);
    } catch (ValidationFailed $e) {
        throw new RequestValidationFailed($e);
    }
```

Validation middleware (cont)

```
  // Process request
  $response = $handler->handle($request);

  // Validate response
  try {
    $builder->getResponseValidator()
      ->validate($match, $response);
  } catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
  }
  return $response;
}
```

Validation middleware (cont)

```
    // Process request
    $response = $handler->handle($request);

    // Validate response
    try {
      $builder->getResponseValidator()
        ->validate($match, $response);
    } catch (ValidationFailed $e) {
      throw new ResponseValidationFailed($e);
    }
    return $response;
}
```

Validation middleware (cont)

```
  // Process request
  $response = $handler->handle($request);

  // Validate response
  try {
    $builder->getResponseValidator()
      ->validate($match, $response);
  } catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
  }
  return $response;
}
```

Validation middleware (cont)

```
  // Process request
  $response = $handler->handle($request);

  // Validate response
  try {
    $builder->getResponseValidator()
      ->validate($match, $response);
  } catch (ValidationFailed $e) {
    throw new ResponseValidationFailed($e);
  }
  return $response;
}
```

Validation Middleware in Laravel

```
// composer install softonic/laravel-psr15-bridge

// app/Http/Kernel.php
use \League\OpenAPIValidation\PSR15\ValidationMiddleware;
protected $routeMiddleware = [
  // ...
  'openapi-validation' => ValidationMiddleware::class,
];

// Routes file
Route::post('/game', 'GameController@create')
    ->middleware('openapi-validation');
```

To sum up

Resources

- https://www.openapis.org
- https://openapi.tools

- https://github.com/akrabat/slim4-rps-api
- https://github.com/thephpleague/openapi-psr7-validator
- https://github.com/primitivesocial/openapi-initializer

Thank you!

https://joind.in/talk/08577

Photo credits

- Scaffolding: https://www.flickr.com/photos/pagedooley/49683539647
- Writing: https://www.flickr.com/photos/throughkikslens/14516757158
- Books: https://www.flickr.com/photos/eternaletulf/41166888495
- Computer code: https://www.flickr.com/photos/n3wjack/3856456237
- Rocket launch: https://www.flickr.com/photos/gsfc/16495356966
- Stars: https://www.flickr.com/photos/gsfc/19125041621