SPS COMMERCE

CRAFTING CONSISTENT APIS AT SCALE

EMBRACING LINTING & REUSABLE MODELS

PRAIRIE DEV CON
WEB | DEV | CLOUD | AI

NORTHFIELD
Johnston group
Lotlinx
online business systems
LAMBERT NEMEC GROUP
EXCELLENCE IN RECRUITMENT
CONQUEST
ATLASSIAN

> "Consistency is one of the most powerful usability principles: when things always behave the same, users don't have to worry about what will happen."
>
> **Jakob Nielsen**

TRAVIS GOSSELIN

DISTINGUISHED SOFTWARE ENGINEER

DEVELOPER EXPERIENCE

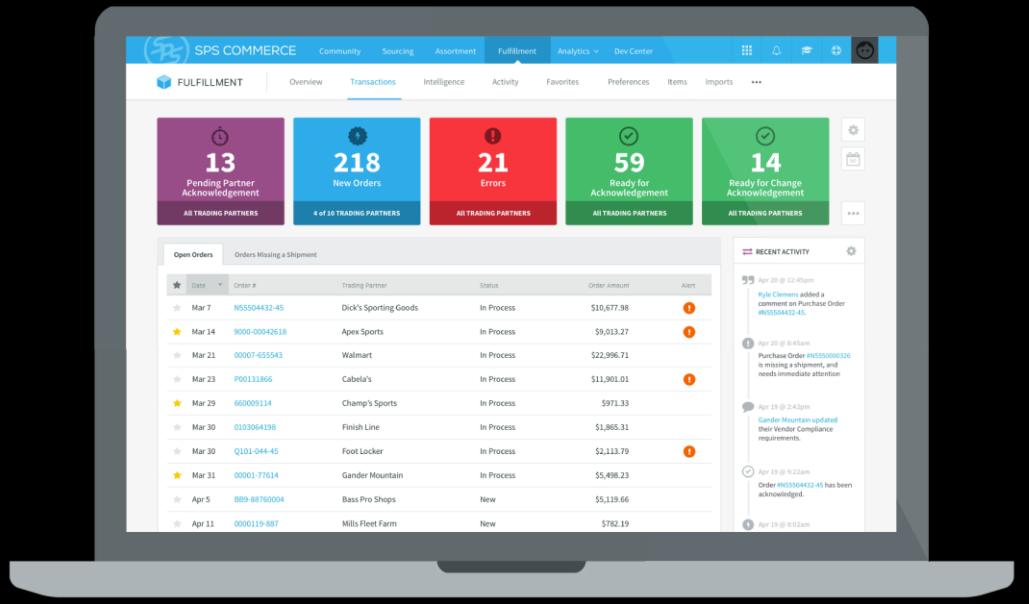
travisgosselin.com

linkedin.com/in/travisgosselin

@travisjgosselin

SPS COMMERCE

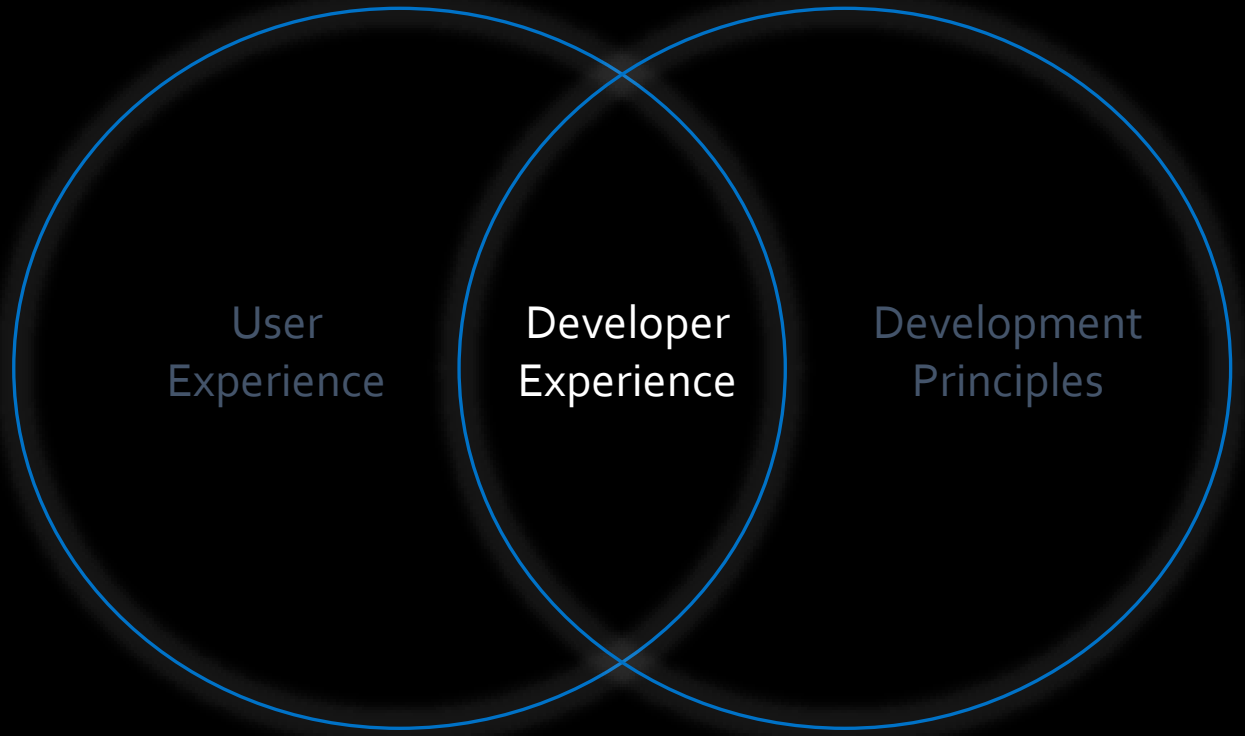
INFINITE RETAIL POWER™



> Developer Experience is the activity of studying, improving and optimizing how developers get their work done.
>
> theappslab.com (2017)

DEVELOPER EXPERIENCE

WHAT IS THAT…EXACTLY?



> Developers work in rainforests, not planned gardens.
>
> a16z.com

DEVELOPER EXPERIENCE == APIS

APIs are the Universal Language of Development

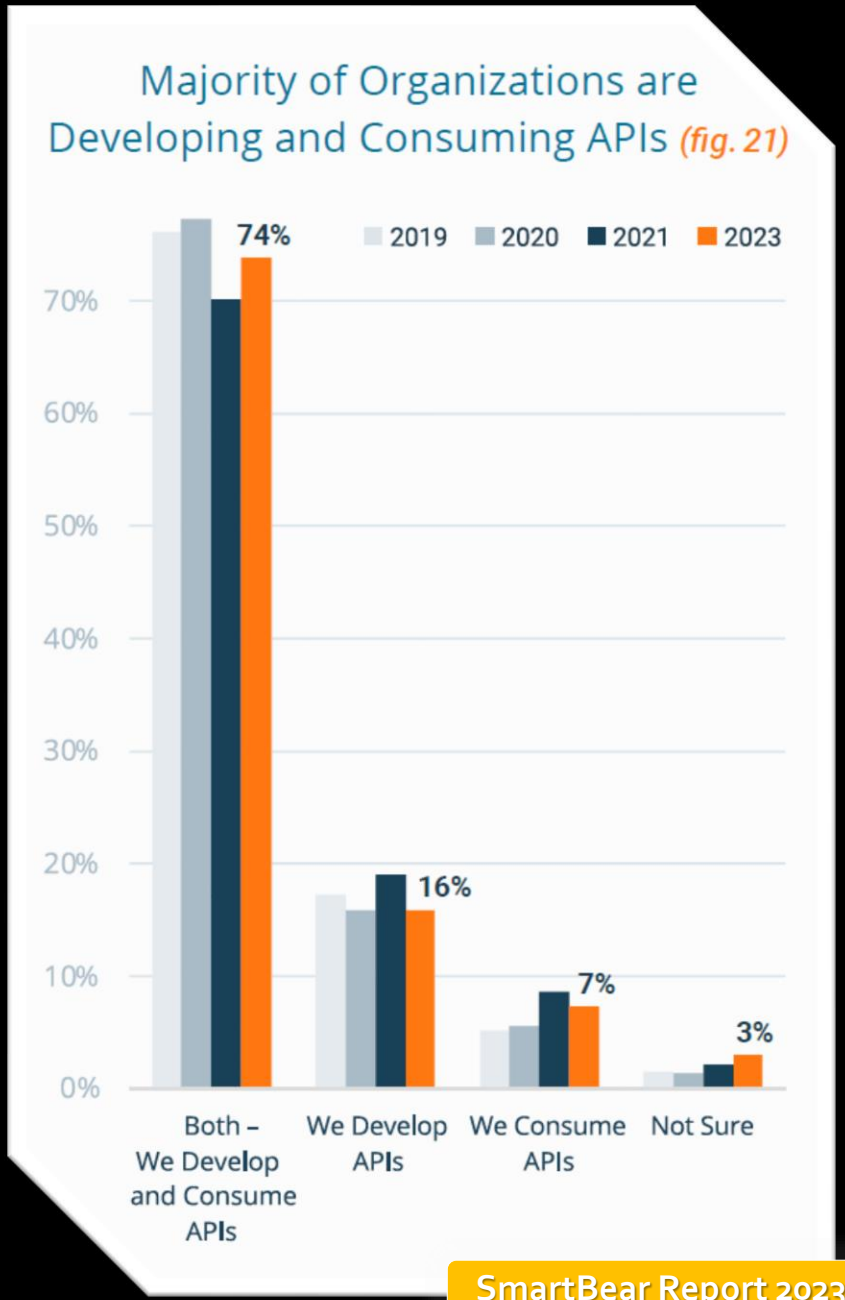
Over 90% of Developers Use APIs

2020 NordicAPI

Developers Spend 30% of Their Time Coding APIs

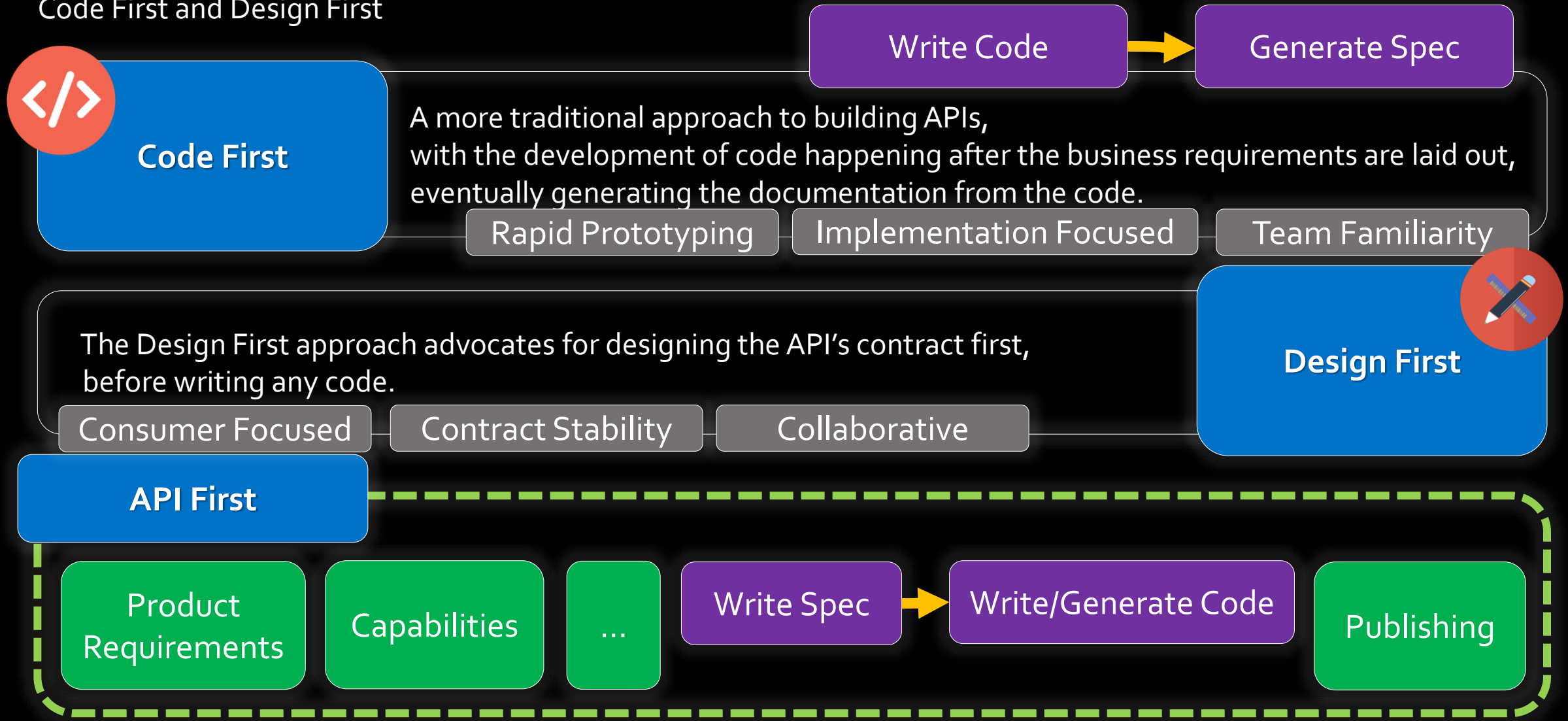
Trends: Most API Consumers are also API Producers

> "When businesses take APIs seriously, it's the single fastest way to extract value out of data. Embracing APIs is a business imperative. It's no longer sufficient to offer a standalone, isolated product."
>
> **Forbes**



APPROACHES TO API DESIGN

Code First and Design First



API ORGANIZATIONAL MODELS

How is API Development Distributed?

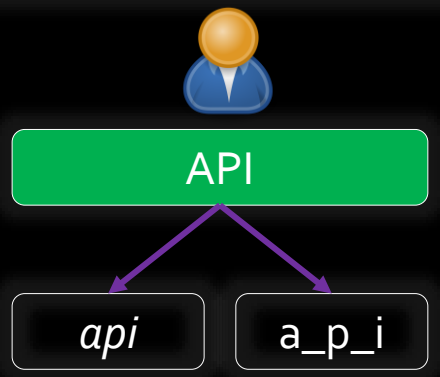
Single Team

Square *Facade*

Single team is responsible for the creation of a façade for all internal APIs behind it.

Limited Governance Needed

Does Not Scale



Autonomous

Granular

Service teams are autonomous and responsible for the entire API including documentation and SDKs, etc.

Some Governance Needed

Highly Inefficient

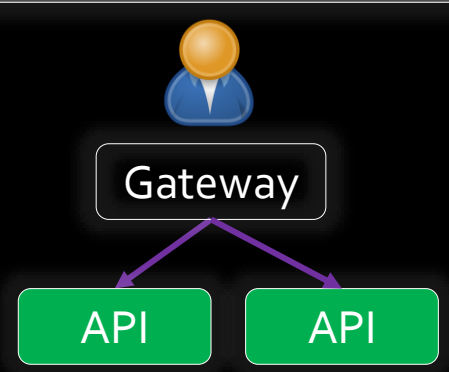
Coordinated

Gateway

Multiple teams delivering APIs for the **domain** they are responsible for centrally versioned and documented.

Substantial Governance

Resource Efficient



API CONSISTENCY

OpenAPI Specification Consistency Drift

Single or Façade API

```yaml
schemas:
  user:
    type: object
    description: A user for HR.
    properties:
      id:
        type: string
        example: 2
      full_name:.
        type: string
        minLength: 1
        maxLength: 100
```

Consistency in Yourself

Less Intentionality

Coordinated API

```yaml
schemas:
  User:
    type: object
    properties:
      id:
        type: string
        format: uid
        example: 12345678-1234-1234-1234-123456789012
      fullName:
        description: The first and last name of the user.
        type: string
        example: "Travis Gosselin"
```

Error Format | Serialization | URL Structure | Bulk

Naming | Headers | Paging | Verbs | Status Codes

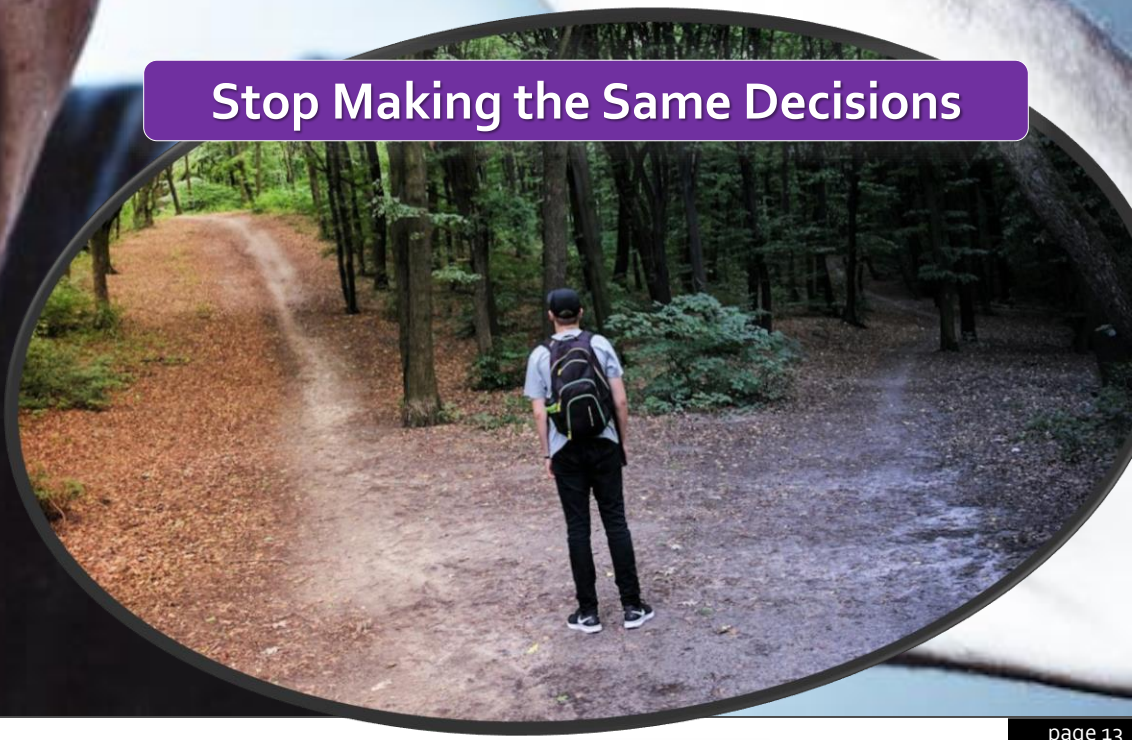
- Organizational Interoperability
- Developer Experience
- Avoiding Toil
- Portability
- Building Abstractions

> "Consistency in API design is essential for today's software engineering landscape. It can guarantee reliability, improve user experience, reduce the need to spend time and resources on maintenance and debugging, and encourage good practices."
>
> Robert Kimani

"Abstraction is one of the greatest existing accomplishments of human logical thought; don't give it up without a fight."

Donald Knuth

Stop Making the Same Decisions



API CONSISTENCY

Does Consistency Matter?

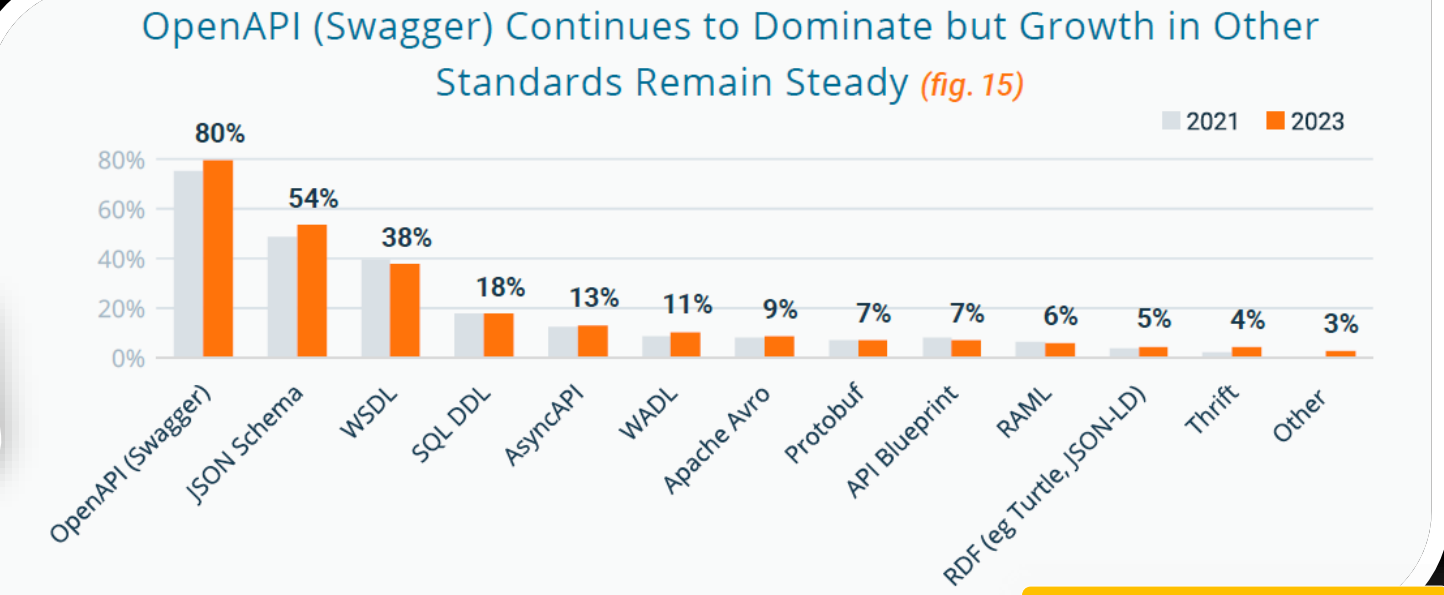
CONSISTENCY AT SCALE

Building Consistency into your Architecture

Precursor

Ubiquitous Design Language

OPENAPI



SmartBear Report 2023

Linting

Reusable Models

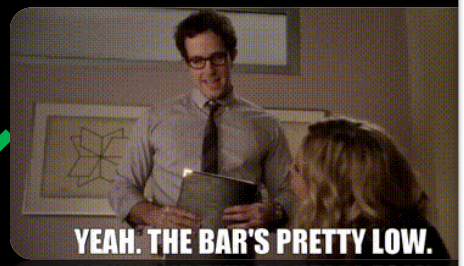
> *Linting:*
>
> *...static code analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs.*
>
> Wikipedia

API LINTING

ENSURE QUALITY AND CONSISTENCY EARLY!

API LINTING

API Linting Maturity Model



Level 0

Valid API Spec & Description

Pretty straightforward… have a description!

Level 1

Basic Standards

- Security Checks
- Useful Metadata
- Design Pattern
- Consistent Data Formats

Level 2

Consistent Interface

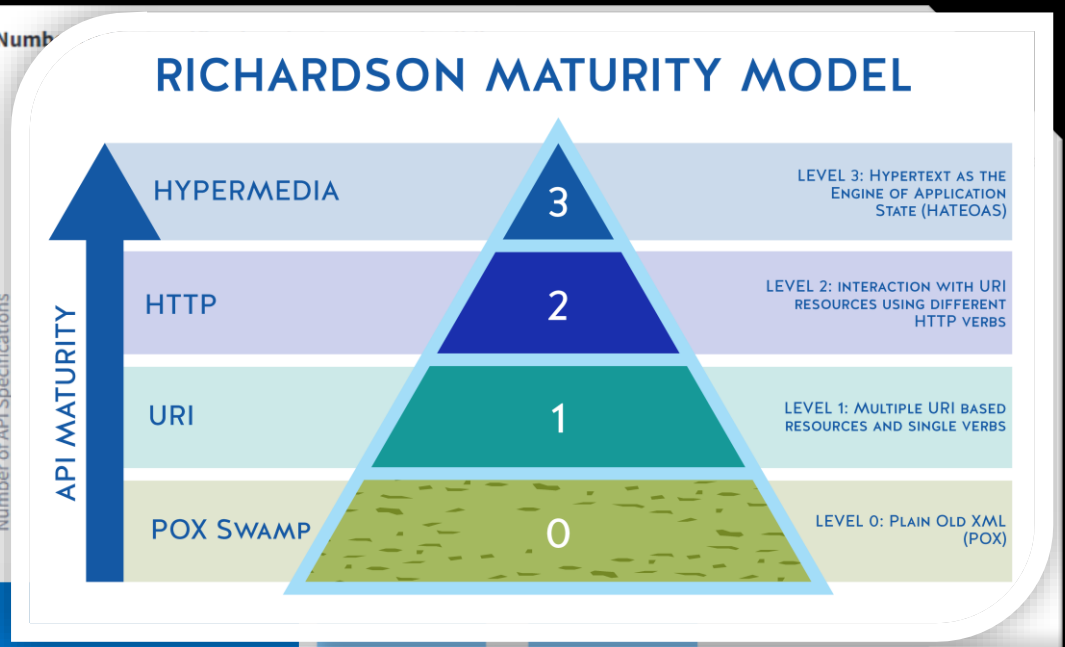
- Mechanics
- Pagination
- Parameters
- Request / Response

Level 3

Joyful Experience

- Simplicity
- Governance
- Intentionality
- Workflow

https://lornajane.net/posts/2024/api-linting-levels



> "In reality, the effort required to design something is inversely proportional to the simplicity of the result."
>
> **Roy T. Fielding**

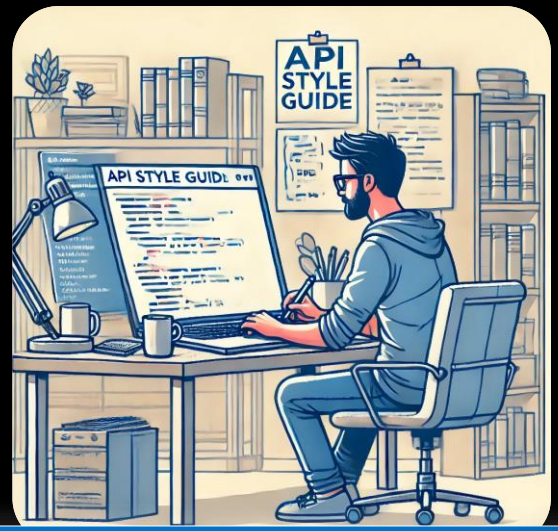
API LINTING

Getting Started

What Are Our Linting Rules?

Trends: 64% of Organizations Have a Style Guide or Plan To

SmartBear Report 2023



API Style Guide!

API Handyman: Arnaud Lauret
ApiStyleBook.com

URL Structure

Request & Response

Collections

Errors

Serialization

Naming

SPS STYLE GUIDE

SPS COMMERCE REST API STANDARDS

https://github.com/spscommerce/sps-api-standards

cloud foundry

Consistency – understanding one resource informs interacts with the rest.

Discoverability – API responses guide users without the need for external documentation.

Simplicity – complex user workflows constructed from smaller parts.

Opinionatedness – one clear way to do something.

Tolerance – contracts and consumers are forgiving as possible without compromising security.

Automation – lean towards defendable standards, where not compromising

Experience – developer experience is king, and major concerns trump the rest.

API LINTING

Tooling

Spectral

github.com/stoplightio/spectral

- Open Source CLI & JavaScript
- SmartBear (Stoplight)
- Evolved from "speccy".
- Generic YAML/JSON Linter with native Rulesets for OpenAPI
- Support for AsyncAPI

Optic CI

useoptic.com

- Commercial with Lite Version
- Purchased by Atlassian
- Integrated Ecosystem
- OpenAPI Specific

Redocly

redocly.com/docs/cli

- Open Source CLI
- Great Developer Experience
- Beyond Linting
- OpenAPI Specific

Core Criteria

Ease of Use and Integration with Existing Tools!

SmartBear Report 2023

VSCode & GitHub

https://spscommerce.github.io/sps-api-standards

API LINTING

SPS COMMERCE

Spectral

- Every error response (4xx/5xx) SHOULD use the Problem Details RFC 7807 Format

```
spectral lint api.yml --ruleset .spectral.yml
```

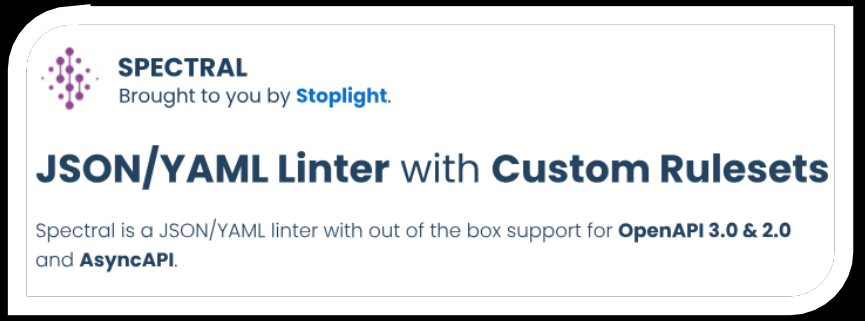
```
spectral lint api.yml --ruleset https://raw.githubusercontent.com/SPSCommerce/sps-api-standards/main/sps-api-standards.spectral.yml
```

```yaml
--- .spectral.yml
extends:
  - "spectral:oas“
  - your-other-rules.yml
rules:
  operation-description: off

problem-details-error-response:
  description: "Every error response SHOULD support RFC 7807
  severity: error
  given: $.paths.[*].responses[?(@property.match(/^(4|5)/))].content.*~
  then:
    function: enumeration
    functionOptions:
      values:
```


```
 1:1   warning  oas3-api-servers       OpenAPI "servers" must be present and non-empty array.
 2:6   warning  info-contact           Info object must have "contact" object.                          info
 2:6   warning  info-description       Info "description" must be present and non-empty string.         info
 7:9   warning  operation-description  Operation "description" must be present and non-empty string.  paths./v1/users.get
 7:9   warning  operation-operationId  Operation must have "operationId".                             paths./v1/users.get
 9:9   warning  operation-tag-defined  Operation tags must be defined in global tags.                 paths./v1/users.get.tags[0]
20:30    error  unknown-error-format   Every error response SHOULD support RFC 7807.                 paths./v1/users.get.responses[400].content.application
```

API LINTING

Spectral Rulesets Examples – API Paths with Environments



https://docs.stoplight.io/docs/spectral/

```
sps-path-no-environment:
  description: Valid API Paths
  message: API paths MUST NOT indicate environment names.
  severity: error
  formats: [oas3]
  given: $.paths.*~
  then:
    function: pattern
    functionOptions:
      notMatch: /prod/|/preprod/|/dev/|/test/|/integration/|/stage/
```

appears in output

error | warn | hint

oas2 | oas3 | oas3_1 | aas2_6 | json-schema | json-schema-2020-12

https://github.com/spscommerce/sps-api-standards

API LINTING

Spectral Rulesets Examples – Query Parameter Casing

```
sps-paths-params-camel-case:
  message: "Path parameter keys MUST use camelCase."
  severity: error
  recommended: false
  given: $.paths.*.*.parameters[?(@.in=='path')].name
  then:
    function: casing
    functionOptions:
      type: camel
      disallowDigits: true
```

extend rulesets

```
extends: [spectral:oas]
extends: [[spectral:oas, all]]
extends: [[spectral:oas, recommended]]
```

Core Function	Description
enumeration	Does the field value exist in this set of possible values?
truthy / falsy	Value should be false, “”, 0, null
length	Count length for min/max
pattern	Regular expression
schema	Matching schema properties
defined/undefined	Value must be present or not present
xor	One and no more than one property is required.

https://github.com/spscommerce/sps-api-standards

API LINTING

Testing

Unit Testing

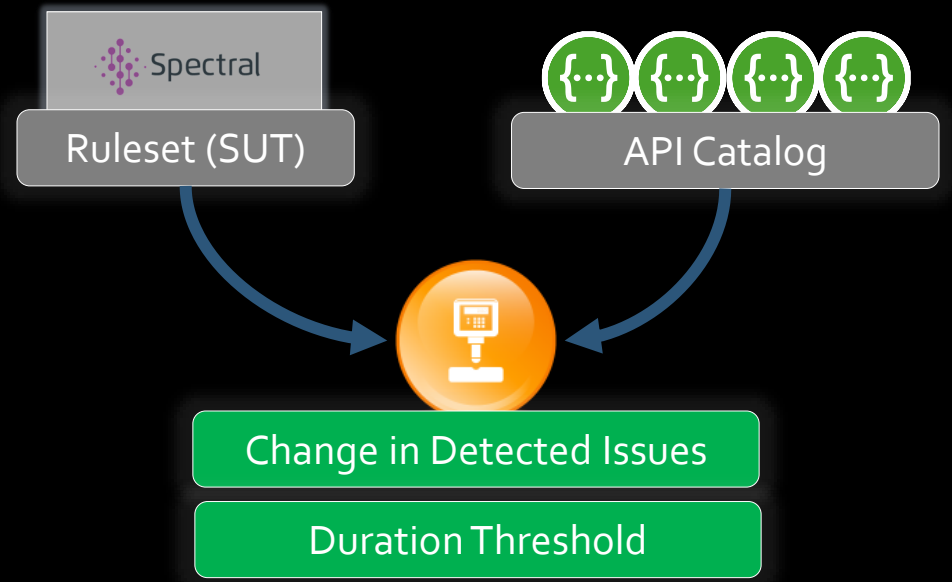
```
describe("sps-path-no-environment", () => {
    let spectral = null;
    const name = "sps-path-no-environment";
    const ruleset = "src/url-structure.ruleset.yml";

    beforeEach(async () => {
        spectral = new SpectralTestHarness(ruleset);
    });

    test("fails with environment names", async () => {
        const s = `
            openapi: 3.1.0
            paths:
                /v1/users/prod/resource:
                    get:
                        summary: thing        `;

        await spectral.validateFailure(s, name, "Error", 1);
    });
});
```

https://github.com/spscommerce/sps-api-standards

https://www.npmjs.com/package/@stoplight/spectral

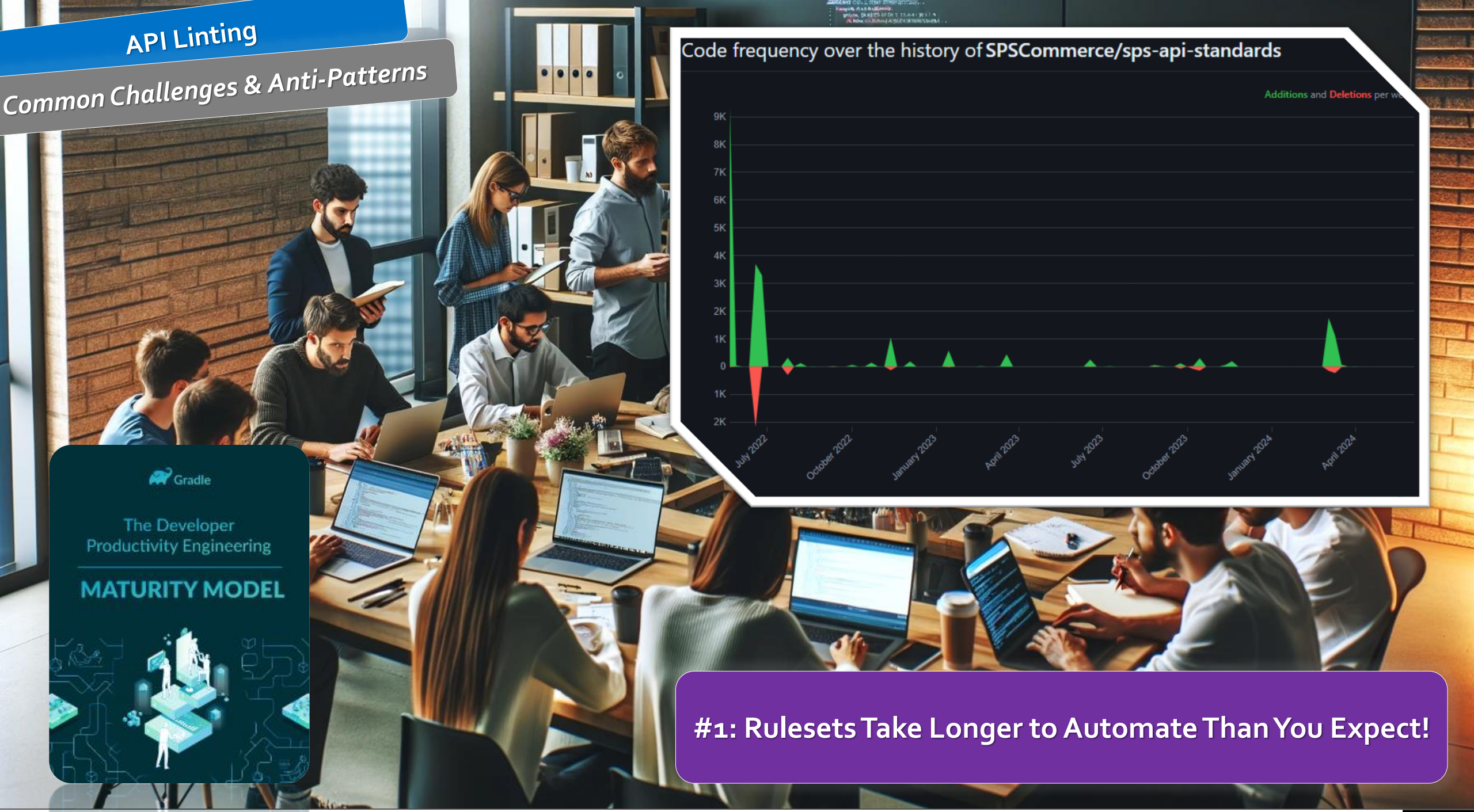
Integration Testing

- Does It work On Complete Spec?
- Impact Analysis
- Performance Validation



API Linting

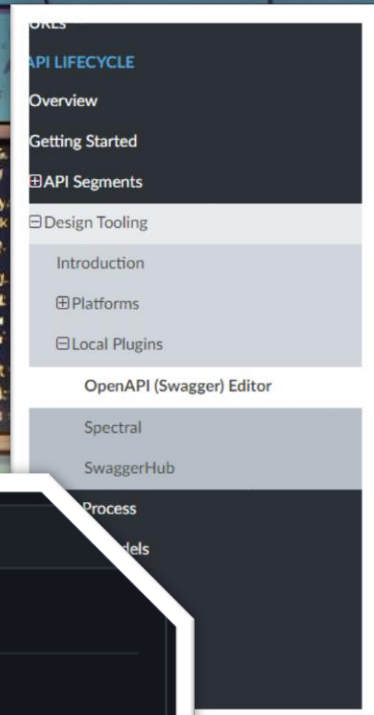
Common Challenges & Anti-Patterns



#1: Rulesets Take Longer to Automate Than You Expect!

API Linting

Common Challenges & Anti-Patterns



Spectral

Using the 42Crunch local Editor plugin gets you a lot. Using the Spectral plugin gets you the rest of the way there with the abilit, execute SPS API Standard validation and verification, along with many additional best practices around OpenAPI linting. This is the same automation used in the documentation publishing and verification pipelines at SPS. It integrates perfectly with the Open API editor from 42Crunch and can be used together nicely.

VSCode: Spectral (By: Stoplight)

IntelliJ: Spectral (By: Schwarz IT)

Note

After you have installed the Spectral plugin, you will need to configure it to use the SPS API Standards ruleset, which can be used remotely from: `https://raw.githubusercontent.com/SPSCommerce/sps-api-standards/main/sps-api-standards.spectral.yml`

SwaggerHub

SwaggerHub also provides a local plugin hat allows you to view, navigate and update files directly on SwaggerHub through your IDE. However, since we use GitHub as our source of truth the plugin is most useful for viewing and navigating versions as part of your workflow quickly. Additionally, it can be really useful in finding, using and viewing available domains in your API spec. This plugin can be used in conjunction with all other plugins mentioned above.

VSCode: SwaggerHub for VS Code (By: SmartBear (SwaggerHub))

IntelliJ: SwaggerHub Plugin for IntelliJ IDEA (By: SmartBear (SwaggerHub))

SPSCommerce-VSTS-BOT commented last month

API Validation Results

RESULT: ❌ **Breaking Changes Detected** 🔥

API Service ID: d47ea086-64a1-4d88-af26-d222c3320c4e
API Name: sps-ref-dotnetcore-api
API Preview URL: SwaggerHub App Designer

✅ API Standards

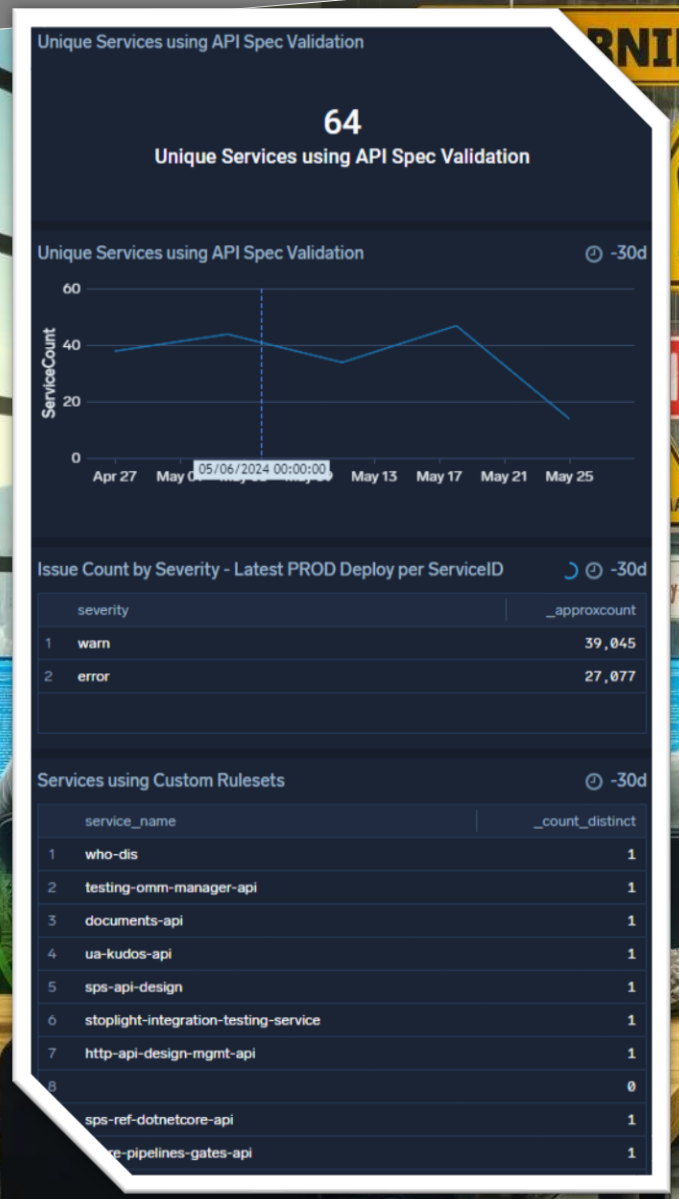
```
==========================================================================
==                        API LINTING RESULTS                           ==
==========================================================================

RESULT: 0 Problems (0 errors, 0 warnings, 0 info/hints)
```

Use VSCode/IntelliJ Plugins for Design
REST API Standards Reference Documentation
Run Spectral Linting CLI Locally

#2: Prefer Linting Locally Over Relying on CI/CD

API Linting

Common Challenges & Anti-Patterns




```
overrides:
  - files:
      - "**#/paths/~1v1~1_webhooks"
        #/paths//v1/_webhooks
    rules:
      sps-request-support-json: off
      sps-schema-names-pascal-case: off
```

#4: Provide Escape Hatches

> API Linting is table stakes, but reuseable API models are differentiators in API design productivity and governance.
>
> Travis Gosselin

REUSEABLE API MODELS

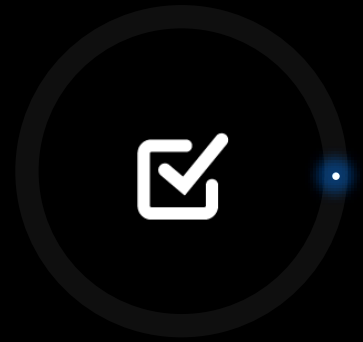
THE MOST EFFICIENT CODE IS THE CODE THAT YOU NEVER HAVE TO WRITE

(ROB PIKE)

```yaml
components:
  schemas:
    User:
      type: object
      properties:
        ????
```

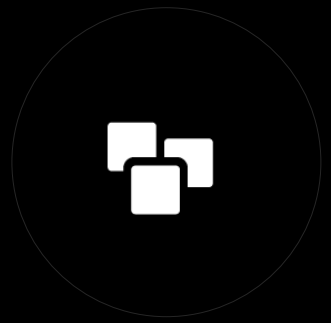
Efficiency & Productivity

API Design with Building Blocks



Governance & Consistency

Referenceable and Centralized



Scalability

Concrete Definitions made Highly Accessible

REUSEABLE API MODELS

Local Components

```yaml
paths:
  /v1/users/{id}:
    get:
      summary: Get User by ID
      parameters:
        - $ref: '#/components/parameters/UserId'
      responses:
        '200':
          description: User with the ID.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
        '500':
          $ref: '#/components/responses/Error'
```

```yaml
components:
  schemas:
    UserId:
      description: A unique id for a user.
      type: string
      format: uuid
      example: 12345678-1234-1234-1234-123456789012
    User:
      description: System User
      type: object
      properties:
        id:
          $ref: '#/components/schemas/UserId'
        name:
          description: Users first/last name.
          type: string
          example: John
  parameters:
    UserId:
      name: id
      in: path
      required: true
      description: A unique id for the User.
      schema:
        $ref: '#/components/schemas/UserId'
```

schemas	parameters
securitySchemas	requestBodies
headers	examples
links	callbacks

REUSEABLE API MODELS

External Components

api.oas.yml

```yaml
paths:
  /v1/users/{id}:
    get:
      …
        '500':
          $ref: '#/components/responses/Error'

components:
  responses:
    Error:
      $ref: 'errors.yml#/components/responses/ProblemDetailsError'
```

errors.yml

```yaml
…
components:
  responses:
    ProblemDetailsError:
      description: 'Internal Server Error'
      content:
        application/problem+json:
          schema:
            type: object
            properties:
              title:
                type: string
              status:
                type: integer
                format: int32
                example: 403
              detail:
                type: string
              instance:
                type: string
              type:
                type: string
                format: url
```

REUSEABLE API MODELS

Remote Components

api.oas.yml

```yaml
paths:
  /v1/users/{id}:
    get:
      …
        '500':
          $ref: '#/components/responses/Error'

components:
  responses:
    Error:
      $ref: 'https://myurl.com/errors#/components/responses/ProblemDetailsError'
```

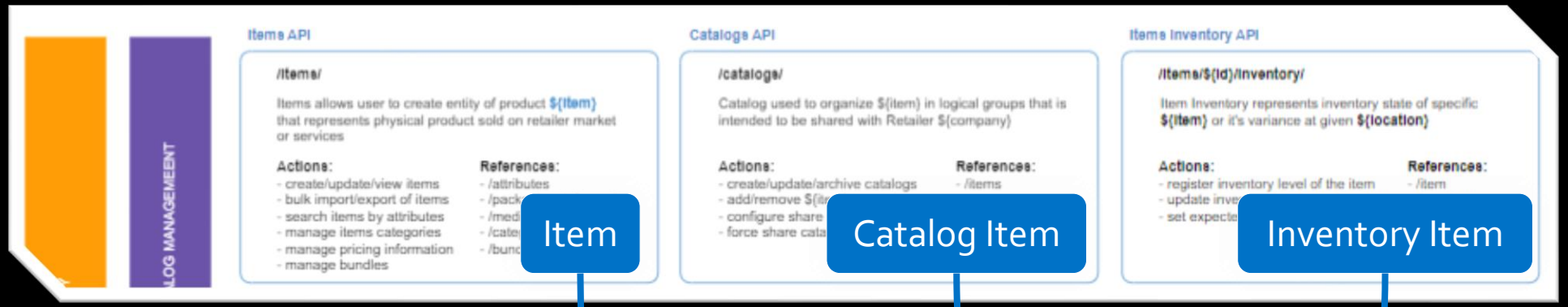
https://myurl.com/errors

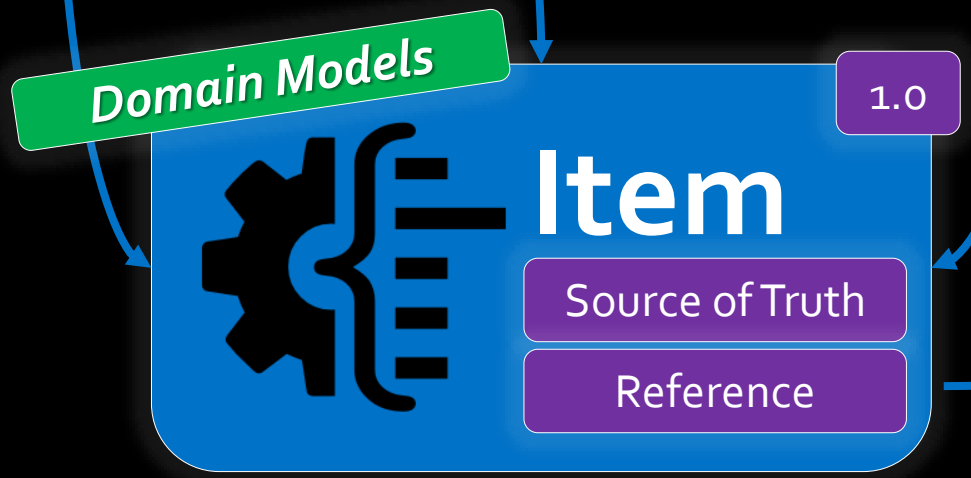
```yaml
…
components:
  responses:
    ProblemDetailsError:
      description: 'Internal Server Error'
      content:
        application/problem+json:
          schema:
            type: object
            properties:
              title:
                type: string
              status:
                type: integer
                format: int32
                example: 403
              detail:
                type: string
              instance:
                type: string
              type:
                type: string
                format: url
```

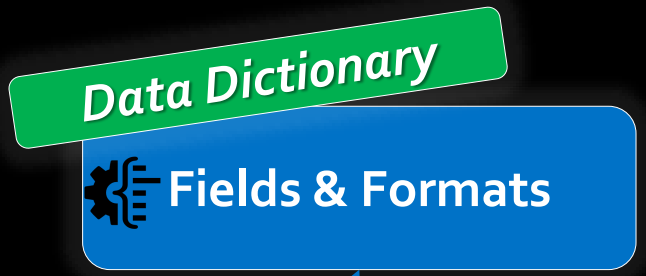
- Common Definitions
- Loosely Coupled Data Models
- Compositional API Design
- Governance & Lifecycle

REUSEABLE API MODELS

API Strategy and Building Blocks

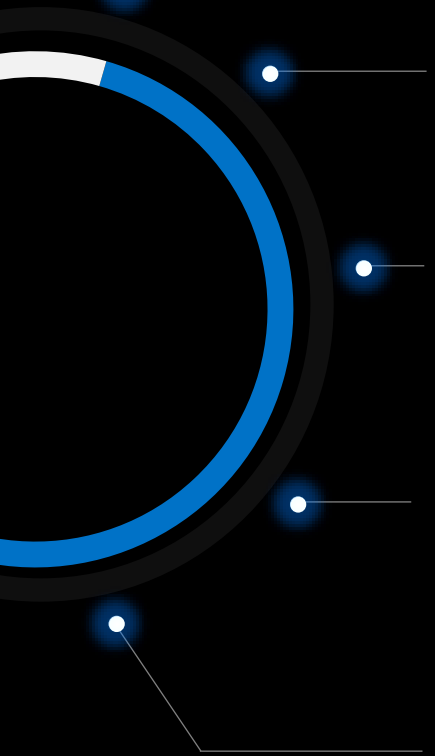






REUSEABLE API MODELS

Producer Considerations



Standardized Schema

Metadata, ownership, and legal.

Open for Extension

Enable teams to dynamically include more properties.

Model Granularity

Referenceable building blocks at just the right size!

Model Dependencies

Should models be in their own file? Should models reference other models?

API Linting with Models

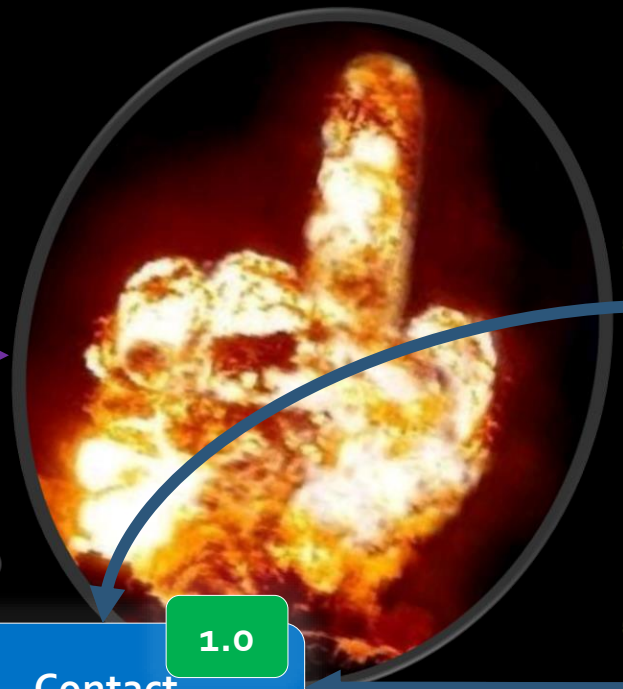
Referencing models only in other reusable models (Private Models)?

```yaml
openapi: 3.0.0
info:
  title: Contacts Domain
  version: "1.0"
  termsOfService: https://developer.sps.com/#/terms-of-use
  contact:
    name: SPS Commerce Support
    url: https://www.spscommerce.com/customer-support/support/
  x-sps-domain: Contacts
components:
  schemas:
    Contact:
      type: object
      additionalProperties: false
      properties:
        name:
          type: string
          nullable: true
          example: "John Doe"
          minLength: 1
          maxLength: 60
        emails:
          type: array
          nullable: true
          items:
            $ref: "#/components/schemas/_Email"
    _Email:
      type: string
      format: email
      example: "test@test.com"
      minLength: 1
      maxLength: 80
```

REUSEABLE API MODELS

Change Strategy

```yaml
Email:
  type: string
  format: email
  pattern: "$abc^"
  example: "t@test.com"
  minLength: 1
  maxLength: 80
```



Service 1

API Spec

Email 1.0

Data Dictionary

Contact 1.0

Domain

Service 2

API Spec

New Property

Contact 1.1

Domain

Service 3

API Spec

Schema Changes

1 . 0 . 0

Domain Changes

Metadata Changes

REUSEABLE API MODELS

Consumer Considerations

api.oas.yml

```yaml
paths:
  /v1/users:
    post:
      summary: Create User
      responses:
        '201':
          ...
        '500':
          $ref: '#/components/responses/Error'
        '409':
          allOf:
            - $ref: "#/components/responses/Conflict"
            - description: "User Email Already Exists"
              content:
                application/problem+json:
                  example:
                    title: Unique Email Conflict
                    detail: Each user…
components:
  responses:
    Error:
      $ref: "https://...components/responses/Error"
    Conflict:
      $ref: "https://...components/responses/Conflict"
```

https://myurl.com/errors/1.o

```yaml
components:
  responses:
    Error:
      description: 'Internal Server Error'
      content:
        application/problem+json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            title: Internal Server Error
            status: 500
            requestId: b6d9a290-9f20-465b-bcd3-4a5166eeb3d7
            detail: Request for resource failed.
            instance: https://example.com/resource/23
    Conflict:
      description: 'Conflict'
      content:
        application/problem+json:
          schema:
            $ref: '#/components/schemas/Error'
          example:
            title: Conflict
            status: 409
            requestId: b6d9a290-9f20-465b-bcd3-4a5166eeb3d7
            detail: Resource 'resource/23' already exists.
            instance: https://example.com/resource/23
```

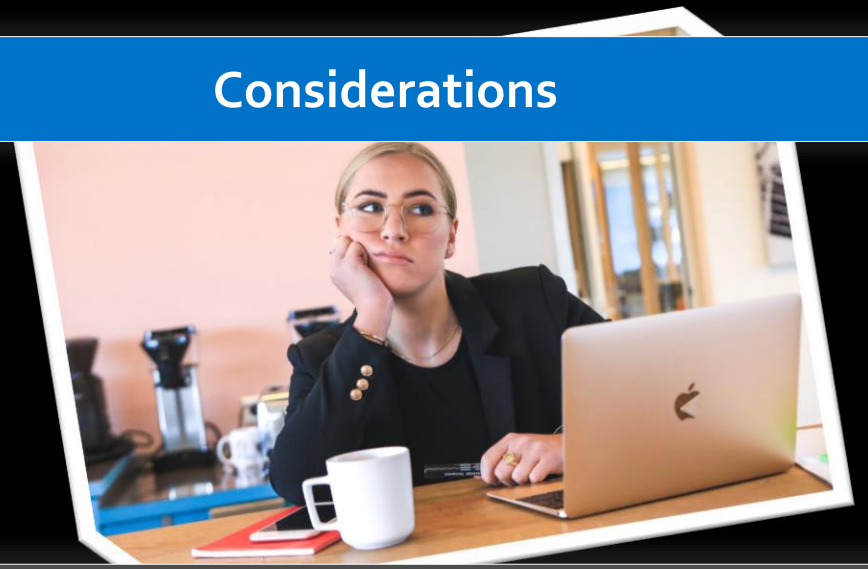
REUSEABLE API MODELS

Model Composition Limitations

Excluding Properties

```yaml
"409":
  allOf:
    - $ref: "#/components/responses/InvalidData"
    - type: object
      not:
        content:
          application/problem+json:
            schema:
              properties:
                requestId: {}
            example:
              requestId: {}
```

Considerations



- Open API 2.0 & 3.0 NOT JSON Schema Compliant
- Open API 3.1 IS JSON Schema Compliant… BUT…
- Reusable Model References Between OAS Versions
- Alternative: Build Models Compositionally Additive

REUSEABLE API MODELS

Tooling

> "Discoverability Drives Reuse"
>
> James Higginbotham

Custom CDN

42crunch

Data Dictionary

Stoplight

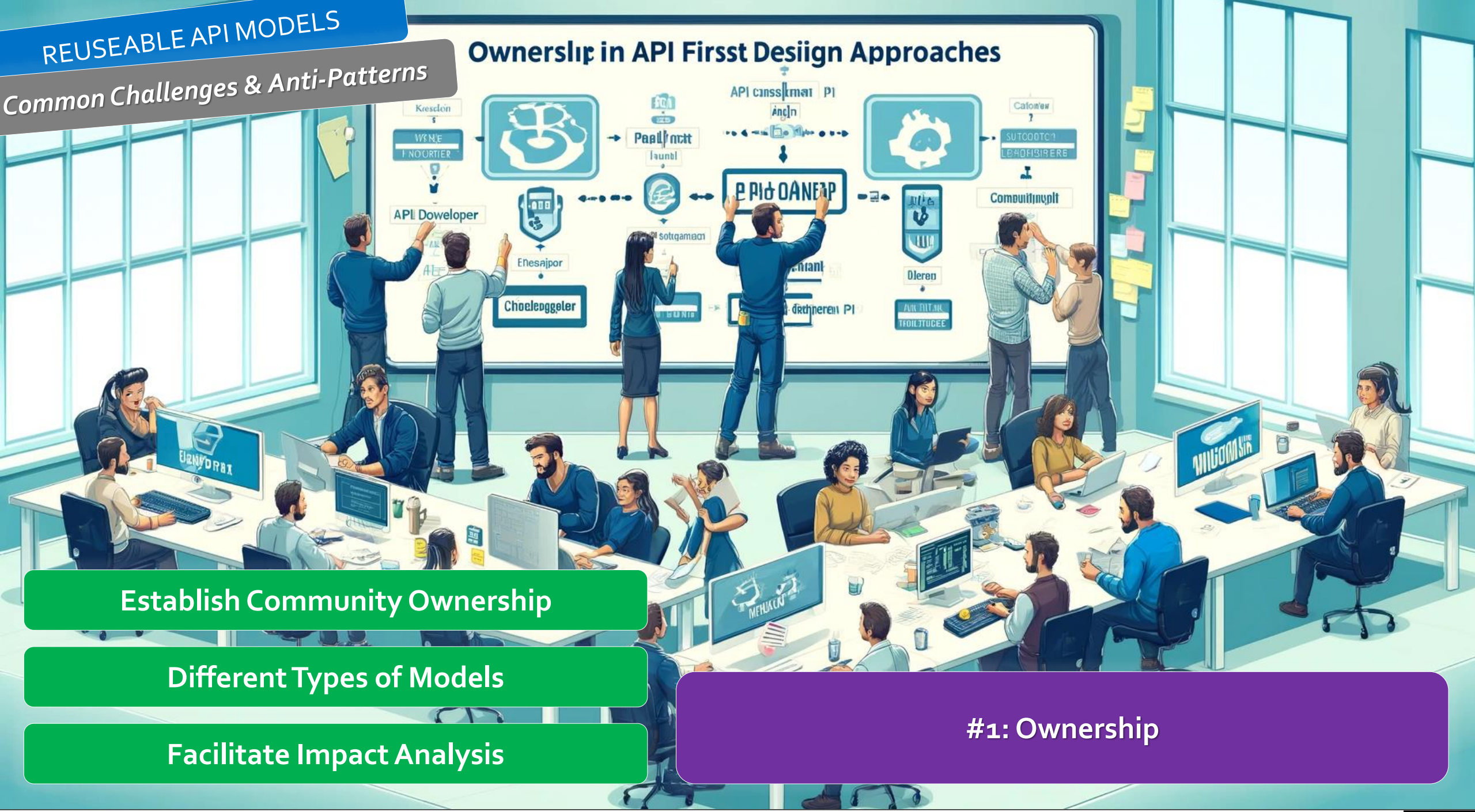
Component Library

SMARTBEAR SwaggerHub

Domains

REUSEABLE API MODELS

Common Challenges & Anti-Patterns



Establish Community Ownership

Different Types of Models

Facilitate Impact Analysis

#1: Ownership

REUSEABLE API MODELS

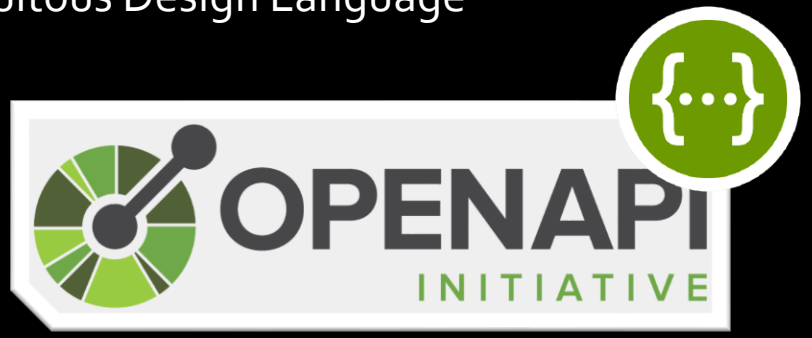
Common Challenges & Anti-Patterns

- Can Models Reference Other Models?
- Can Domains Reference Other Domains?
- Classification and Tiers of Models?
- Private or Internal Models?

#2: Complexity

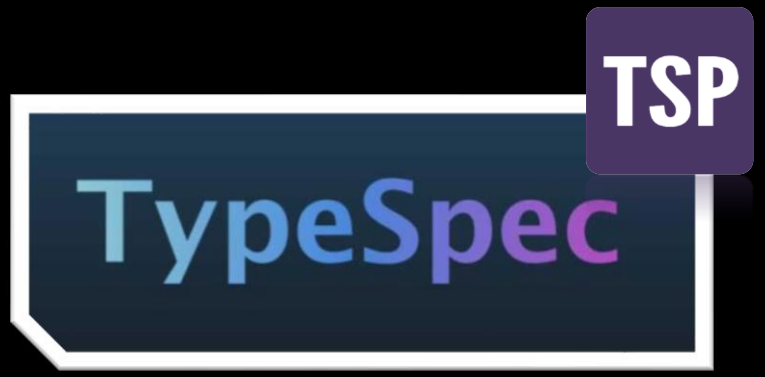
CRAFTING APIs AT SCALE

Ubiquitous Design Language



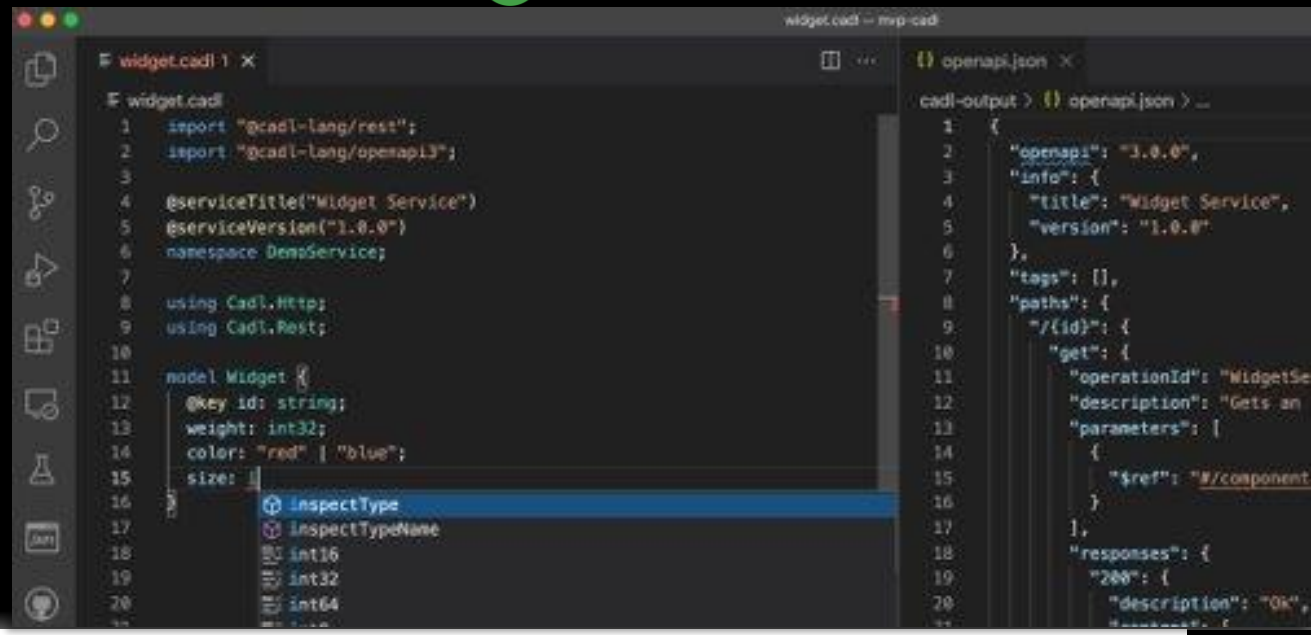
https://www.openapis.org/

- Widely Accepted Standard for API Design ✅
- Extensive Tooling Support ✅
- Robust Ecosystem of Support and Community ✅
- Straightforward Learning Curve ✅
- Painful Version Upgrades ❌
- Tooling Interoperability Not 100% Compliant ❌
- Tedious Handcrafted Files at Times ❌



https://typespec.io/

- A new language for HTTP, JSON Schema, Protobuf ✅
- Abstraction to generate OpenAPI and other assets ✅
- Highly extensible DSL that promotes code reuse ✅❌
- Version Aware ✅



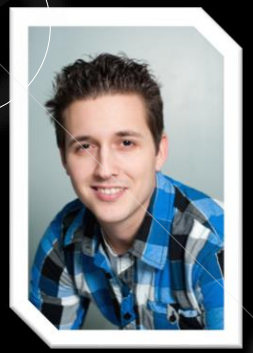
> Software is eating the world, and APIs are the teeth. Scale and adaptability in APIs are not just technical requirements; they are business imperatives.
>
> Marc Andreessen

- API Linting
- Reuseable API Models

Is Your Organization Ready to Scale?

CRAFTING CONSISTENT APIS AT SCALE

EMBRACING LINTING & REUSABLE MODELS



TRAVIS GOSSELIN

travisgosselin.com

linkedin.com/in/travisgosselin

@travisjgosselin