Overlay and Arazzo

From API Definitions to API Experiences

Gloria Ciavarrini
Principal Software Engineer @ Red Hat

How many of you are already familiar with OpenAPI spec?

OpenAPI

```yaml
openapi: 3.0.2
servers:
  - url: /v3
info:
  description: |-
    This is a sample Pet Store Server based on the OpenAPI 3.0 specification.
  version: 1.0.20-SNAPSHOT
  title: Swagger Petstore - OpenAPI 3.0
paths:
   '/pet/{petId}':
    get:
      tags:
        - pet
      summary: Find pet by ID
      description: Returns a single pet
      operationId: getPetById
      parameters:
        - name: petId
          in: path
          description: ID of pet to return
          required: true
          schema:
            type: integer
            format: int64
      responses:
        '200':
          description: successful operation
          content:
            application/xml:
              schema:
                $ref: '#/components/schemas/Pet'
            application/json:
              schema:
                $ref: '#/components/schemas/Pet'
        '400':
          description: Invalid ID supplied
```

Full example:

OpenAPI

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` *menu option!*

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

Terms of service

Contact the developer

Apache 2.0

Find out more about Swagger

Servers

https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

Swagger UI

OpenAPI

The OpenAPI specs is a Yaml file that describes each endpoints, its input parameters and the output.

She wrote the specs and everything is fine!

The problem is:

What do others want
OpenAPI spec
to look like?

OpenAPI

Examples

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` *menu option!*

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

Terms of service

Contact the developer

Apache 2.0

Find out more about Swagger

Servers

https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

OpenAPI

Examples

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` *menu option!*

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

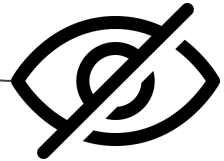
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image



Hide some endpoints

OpenAPI

Examples

Translations

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

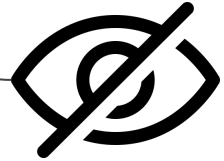
Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image



Hide some endpoints

OpenAPI

Examples

Translations

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

Infrastructure

Hide some endpoints

OpenAPI

Examples

Translations

Security

Swagger Petstore - OpenAPI 3.0 1.0.11 OAS 3.0

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at https://swagger.io. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click here. Alternatively, you can load via the `Edit > Load Petstore OAS 2.0` menu option!

Some useful links:

- The Pet Store repository
- The source API definition for the Pet Store

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://petstore3.swagger.io/api/v3

Authorize

pet Everything about your Pets — Find out more

PUT	/pet	Update an existing pet
POST	/pet	Add a new pet to the store
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

Infrastructure

Hide some endpoints

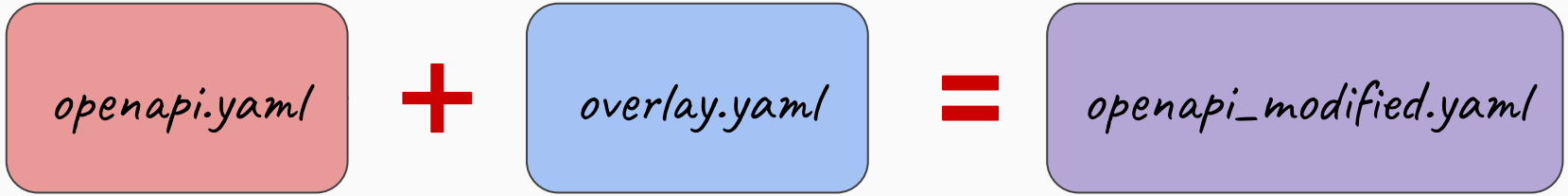
Each user wants
to contribute
to the OpenAPI spec

Each user wants
to contribute
to the OpenAPI spec

How?

Overlay

Overlay: the equation



Overlays - How does it work?

Target some parts of the OpenAPI spec document and mutate them

Overlay - Hello world example

```yaml
overlay: 1.0.0
info:
    title: example overlay
    version: 0.0.0
extends: https://petstore3.swagger.io/api/v3/openapi.json
actions:
  - target: $.info.description
    update: Hello World
```

Overlay - Hello world example

Published 17 October 2024

```yaml
overlay: 1.0.0
info:
    title: example overlay
    version: 0.0.0
extends: https://petstore3.swagger.io/api/v3/openapi.json
actions:
  - target: $.info.description
    update: Hello World
```

Overlay - Hello world example

```yaml
overlay: 1.0.0
info:
    title: example overlay
    version: 0.0.0
extends: https://petstore3.swagger.io/api/v3/openapi.json
actions:
  - target: $.info.description
    update: Hello World
```

Target OpenAPI spec

Overlay - Hello world example

```yaml
overlay: 1.0.0
info:
    title: example overlay
    version: 0.0.0
extends: https://petstore3.swagger.io/api/v3/openapi.json
actions:
  - target: $.info.description
    update: Hello World
```

Array of actions to perform

Overlay - Actions

An action is composed by

- Target
- Mutations
  - Update: merge in a value
  - Remove: remove a value

Overlay - Actions

An action is composed by

- Target ← JSONPath
- Mutations
  - Update
  - Remove

JSONPath
RFC 9535

Overlay - Targeting with JSONPath

JSONPath
RFC 9535

Examples

- `$.paths.*.*` - Wildcards
- `$..description` - Descendants
- `$.paths."/pet/".post` - One specific element
- `$.tags[?(@.name == "pet")]` - Filters/Expressions`

Overlay - Targeting with JSONPath

Select all description fields at any depth, anywhere in the document.

Examples

- `$.paths.*.*` - Wildcards
- `$..description` - Descendants
- `$.paths."/pet/".post` - One specific element
- `$.tags[?(@.name == "pet")]` - Filters/Expressions`

Overlay - Targeting with JSONPath

Examples

- `$.paths.*.*` - Wildcards
- `$..description` - Descendants
- `$.paths."/pet/".post` - One specific element
- `$.tags[?(@.name == "pet")]` - Filters/Expressions`

Start with '$'

Separate the tree structure with '.'

Overlay - Targeting with JSONPath

Examples

- `$.paths.*.*` - Wildcards
- `$..description` - Descendants
- `$.paths."/pet/".post` - One specific element
- `$.tags[?(@.name == "pet")]` - Filters/Expressions

`[?()]`
Filter expression:
evaluates a condition for each element in the *tags* array.

`@`
current element being evaluated in the array

`.name == "pet"`
name of the current element equals *"pet"*?

What’s the Overlay goal?

What’s the Overlay goal?

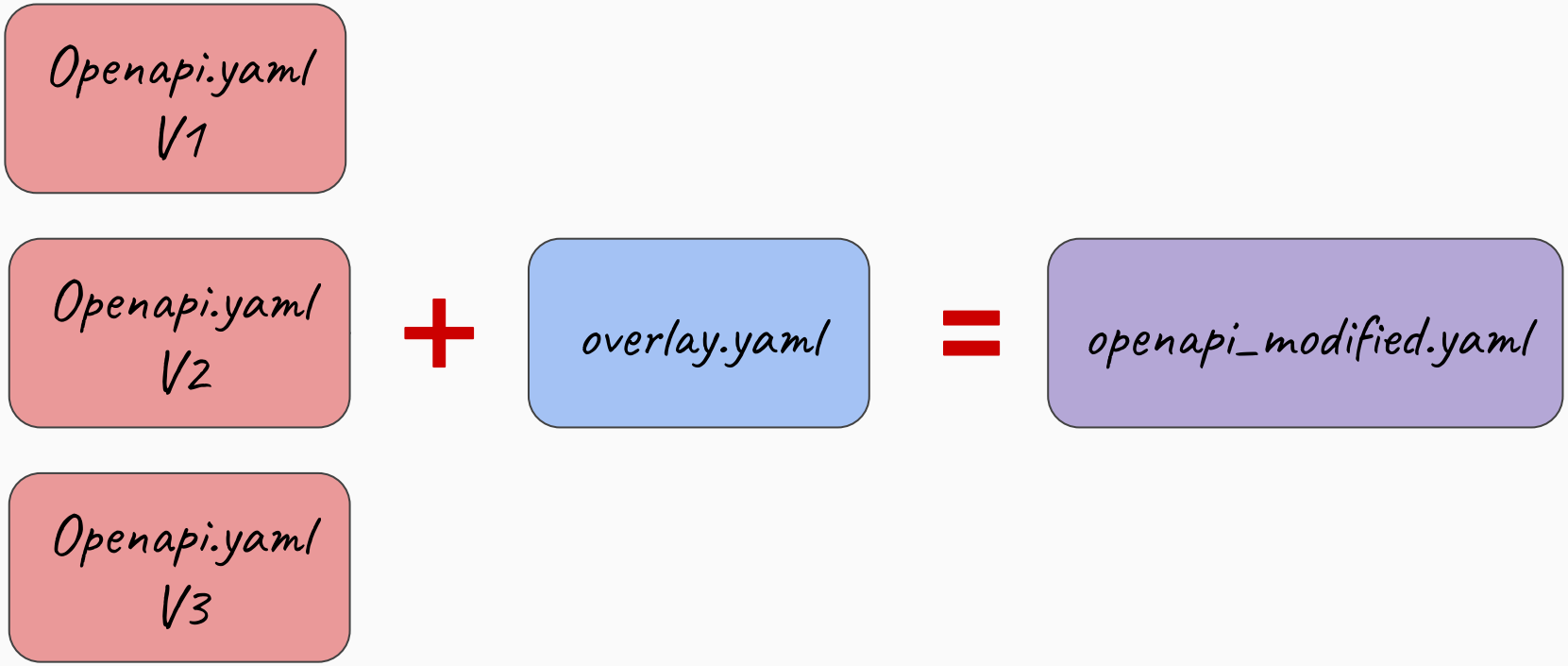
Non disruptive updates

Examples of Overlays

- Maintaining a translate version of an API
- Annotate APIs to include infrastructure detail, security.
- Don’t expose publicly an endpoint (`x-internal: true`)

Different concerns!

Overlay: different input same overlay



Are there any drawbacks?

Are there any drawbacks?

Of Course.

Overlay - Drawbacks

- You can create invalid API ➡️ Validation needed!
- Incomplete APIs ➡️ Rely on Overlay to complete the API
- JSONPath ➡️ Easy to write erroneous target

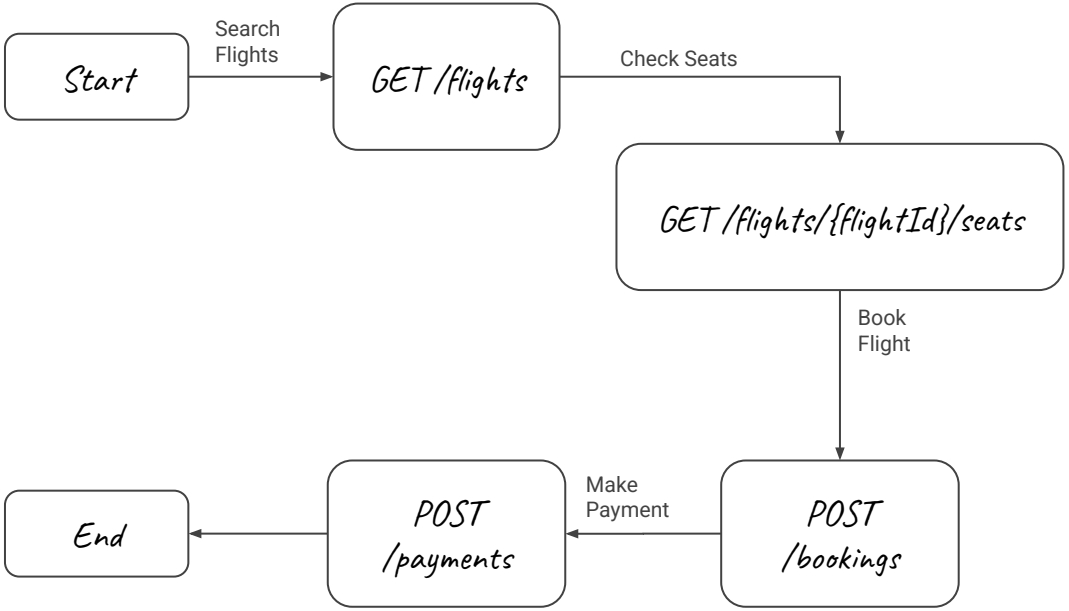
Overlay - Some Tools

- Overlay CLI (prototype)
  https://github.com/ponelat/overlays-cli
- Speakeasy - OpenAPI Overlay (alpha)
  https://github.com/speakeasy-api/openapi-overlay
- Speakeasy - Overlay Playground
  https://overlay.speakeasy.com/

Alright,
we've got the spec and overlays.
But how do I actually use the endpoints to get something done?

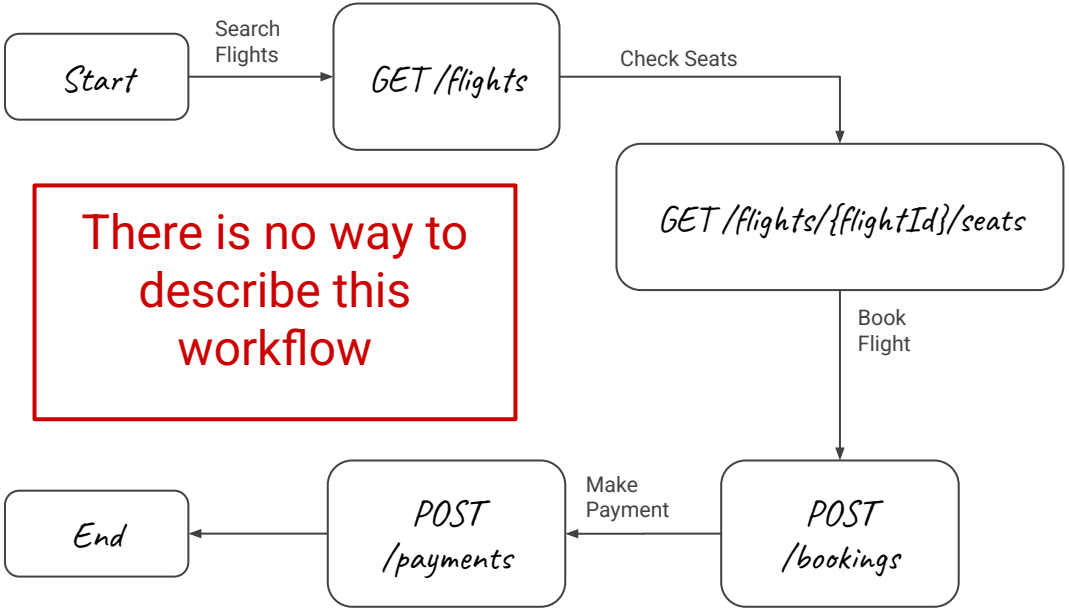
Problem

Usually we need more than just a single call to achieve our goal



Problem

Usually we need more than just a single call to achieve our goal



There is no way to describe this workflow

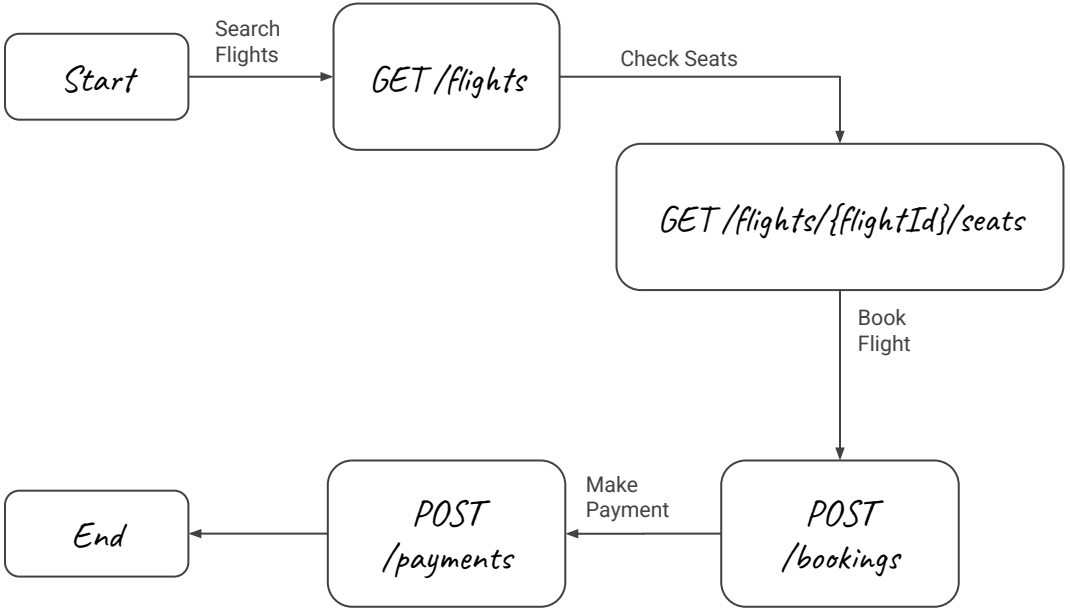
Arazzo

A tapestry for API workflows

What is Arazzo spec goal?

Describes use-case oriented workflows
in a programmatically readable format.

Book a Flight Example



Arazzo - How does it look like?

```yaml
arazzo: 1.0.1

info:
  title: Flight Booking API - Book & Pay
  version: 1.0.0
  description: >
  This API workflow allows you to search for flights, check seat availability,
  book a ticket, and make a payment.

sourceDescriptions:
  - name: flight-booking
  url: ./openapi.yaml
  type: openapi

workflows:
- workflowId: book-a-flight
  summary: Book a flight from origin to destination.
  description: >
    This workflow demonstrates how to search for flights, check seat availability,
    book a flight, and confirm payment.
  inputs:
    $ref: "#/components/inputs/book_a_flight_input"
  steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
    - stepId: check-seats
      description: Check seat availability for the selected flight.
      operationId: get-seats
      parameters:
        - name: flight_id
          in: path
          value: $steps.search-flights.outputs.flight_id
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        available_seats: $outputs.data.available_seats
    - stepId: book-flight
      description: Book the selected flight.
      operationId: create-booking
      requestBody:
        contentType: application/json
        payload:
          flight_id: $steps.search-flights.outputs.flight_id
          passenger_name: "John Doe"
      successCriteria:
        - condition: $statusCode == 201
      outputs:
        booking_id: $outputs.data.booking_id
    - stepId: make-payment
      description: Make payment for the booked flight.
      operationId: process-payment
      requestBody:
        contentType: application/json
        payload:
          booking_id: $steps.book-flight.outputs.booking_id
          payment_method: "credit_card"
      successCriteria:
        - condition: $statusCode == 200
components:
  inputs:
  book_a_flight_input:
    type: object
    properties:
      origin:
        type: string
        description: The origin airport code.
      destination:
        type: string
        description: The destination airport code.
      date:
        type: string
        format: date-time
```

Solution

Arazzo spec version
(released 16 Jan 2025)

```yaml
arazzo: 1.0.1

info:
  title: Flight Booking API - Book & Pay
  version: 1.0.0
  description: >
    This API workflow allows you to
    search for flights, check seat
    availability, book a ticket, and
    make a payment.

sourceDescriptions:
  - name: flight-booking
    url: ./openapi.yaml
    type: openapi
```

Solution

Metadata about API workflows defined in this Arazzo document

```
arazzo: 1.0.1

info:
  title: Flight Booking API - Book & Pay
  version: 1.0.0
  description: >
    This API workflow allows you to
    search for flights, check seat
    availability, book a ticket, and
    make a payment.

sourceDescriptions:
  - name: flight-booking
    url: ./openapi.yaml
    type: openapi
```

Solution

```yaml
arazzo: 1.0.1

info:
  title: Flight Booking API - Book & Pay
  version: 1.0.0
  description: >
    This API workflow allows you to
    search for flights, check seat
    availability, book a ticket, and
    make a payment.

sourceDescriptions:
  - name: flight-booking
    url: ./openapi.yaml
    type: openapi
```

Describes a source description (such as an OpenAPI description) that will be referenced by one or more workflows described within an Arazzo Description.

Solution

Each workflow describes the steps to be taken to achieve an objective.

```
workflows:
- workflowId: book-a-flight
  summary: Book a flight from
     origin to destination.
  description: >
    This workflow demonstrates how to search for flights,
    check seat availability, book a flight,
    and confirm payment.
  inputs:
    $ref: "#/components/inputs/book_a_flight_input"
  steps:
    …
```

Solution

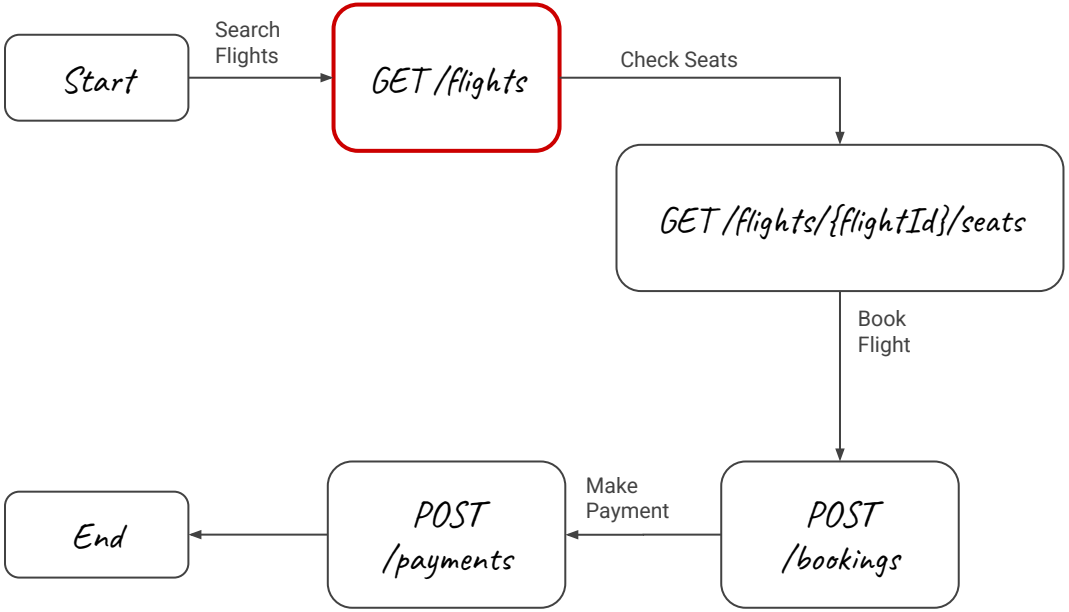
```
workflows:
- workflowId: book-a-flight
  summary: Book a flight from
     origin to destination.
  description: >
    This workflow demonstrates how to search for flights,
    check seat availability, book a flight,
    and confirm payment.
  inputs:
    $ref: "#/components/inputs/book_a_flight_input"
  steps:
    …
```

The workflow object may define inputs needed in order to execute workflow steps

Solution

```yaml
components:
  inputs:
    book_a_flight_input:
      type: object
      properties:
        origin:
          type: string
          description: The origin airport code.
        destination:
          type: string
          description: The destination airport code.
        date:
          type: string
          format: date-time
```

Book a Flight Example



Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

Steps represent a call to an API operation or another workflow, and a set of outputs.

Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

The location of the parameter.
Possible values are:
path, query, header, or cookie

Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

Value can either be a constant or a runtime expression

Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

This is a runtime expression that pulls the value of the origin input provided when the workflow is executed.

Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

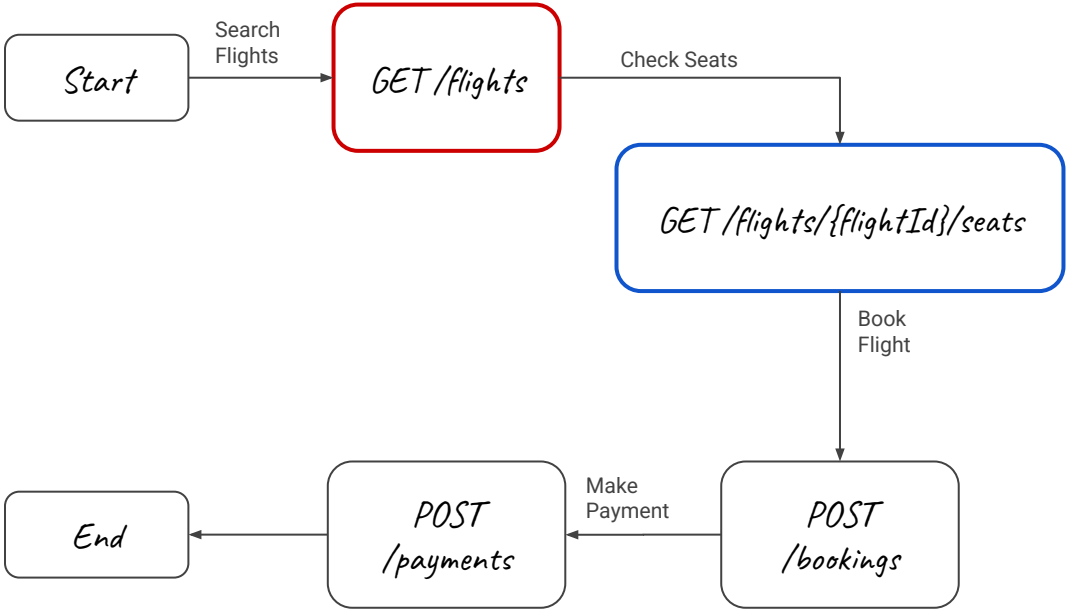
A list of assertions to determine if this action SHALL be executed.
All criteria assertions MUST be satisfied for the action to be executed.

Solution

```yaml
steps:
    - stepId: search-flights
      description: Search for available flights.
      operationId: get-flights
      parameters:
        - name: origin
          in: query
          value: $inputs.origin
        - name: destination
          in: query
          value: $inputs.destination
        - name: date
          in: query
          value: $inputs.date
      successCriteria:
        - condition: $statusCode == 200
      outputs:
        flight_id: $outputs.data[0].id
```

A map between a friendly name and a dynamic output value.

Book a Flight Example



Solution

```
steps:
    - stepId: search-flights
      [...]
      outputs:
        flight_id: $outputs.data[0].id
```

```
- stepId: check-seats
      description: Check seat availability for the selected
flight.
      operationId: get-seats
      parameters:
        - name: flight_id
          in: path
          value: $steps.search-flights.outputs.flight_id
      successCriteria:
        - condition: $statusCode == 200
```

Solution

```yaml
- stepId: book-flight
  description: Book the selected flight.
  operationId: create-booking
  requestBody:
    contentType: application/json
    payload:
      flight_id: $steps.search-flights.outputs.flight_id
      passenger_name: "John Doe"
  successCriteria:
    - condition: $statusCode == 201
  outputs:
    booking_id: $outputs.data.booking_id

- stepId: make-payment
  description: Make payment for the booked flight.
  operationId: process-payment
  requestBody:
    contentType: application/json
    payload:
      booking_id: $steps.book-flight.outputs.booking_id
      payment_method: "credit_card"
  successCriteria:
    - condition: $statusCode == 200
```

Solution

```yaml
- stepId: book-flight
    description: Book the selected flight.
    operationId: create-booking
    requestBody:
      contentType: application/json
      payload:
        flight_id: $steps.search-flights.outputs.flight_id
        passenger_name: "John Doe"
    successCriteria:
      - condition: $statusCode == 201
    outputs:
      booking_id: $outputs.data.booking_id
```

```yaml
- stepId: make-payment
      description: Make payment for the booked flight.
      operationId: process-payment
      requestBody:
        contentType: application/json
        payload:
          booking_id: $steps.book-flight.outputs.booking_id
          payment_method: "credit_card"
      successCriteria:
        - condition: $statusCode == 200
```

Cool but…

Use cases should be described in a human readable format.

Arazzo GPT

Arazzo GPT is an expert assistant designed to help you understand and work with the Arazzo Specification.

What Does Arazzo GPT Do?

- Parses and explains Arazzo Documents
- Generates Arazzo Documents from OpenAPI descriptions
- Validates workflows against the Arazzo Specification
- Provides code examples and diagrams for better understanding
- Improves API usability and integration by defining clear workflows

Visual Tool

Questions?

Thank you!

Slides available here!

Gloria Ciavarrini

📨 gloria.ciavarrini@gmail.com
www.linkedin.com/in/gloria-ciavarrini