Michele Pasqua
michele.pasqua@univr.it

UNIVERSE LAB

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Spec-to-Test: Mastering REST API Automated Black-box Testing with RestTestGen


Davide Corradini
uni.lu


Mariano Ceccato

Rio de Janeiro, BR

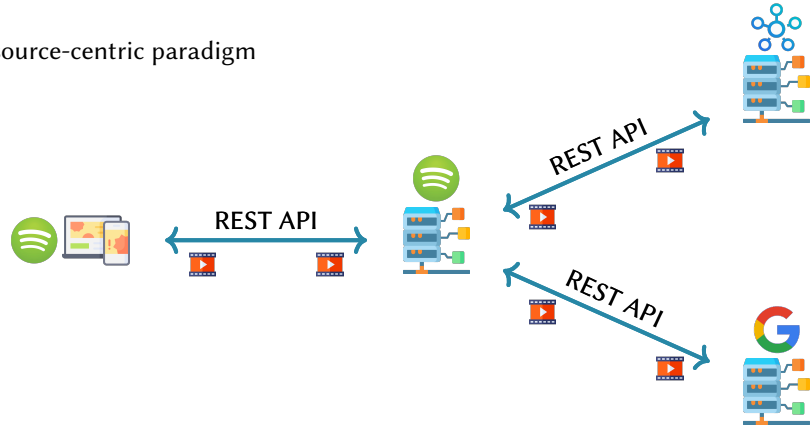
April 12, 2026

19th International Workshop on Search-Based and Fuzz Testing

REST APIs Testing

What is a REST API?

- Resource-centric paradigm



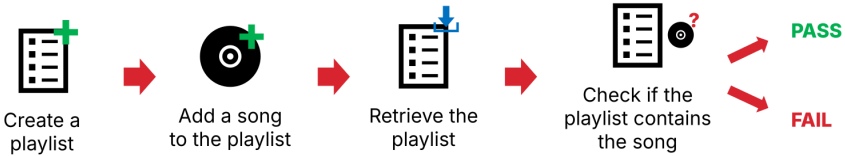
Create
Read
Update
Delete

What is a REST API? (Cont'd)

Testing REST APIs

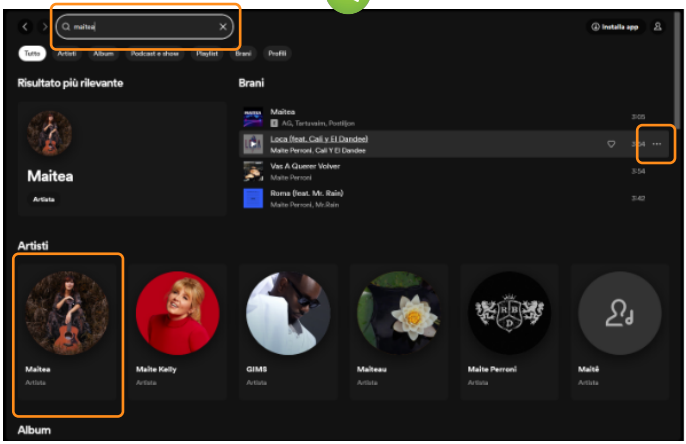
The number of REST APIs grows larger and larger

- REST APIs may contain programming defects and vulnerabilities
- Manual writing of test cases is limiting and costly

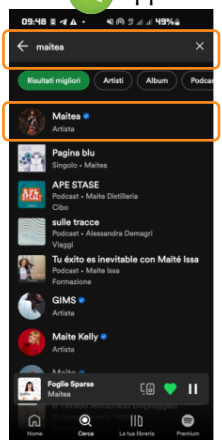


Testing REST APIs (Cont'd)

Web



App

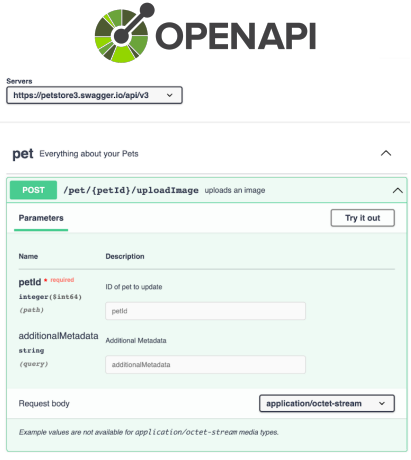


OpenAPI specification

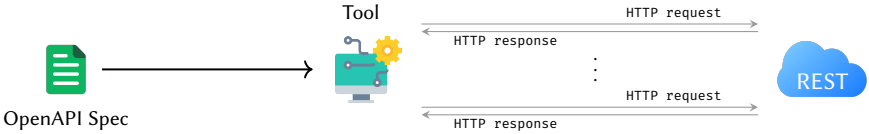
REST API formal documentation

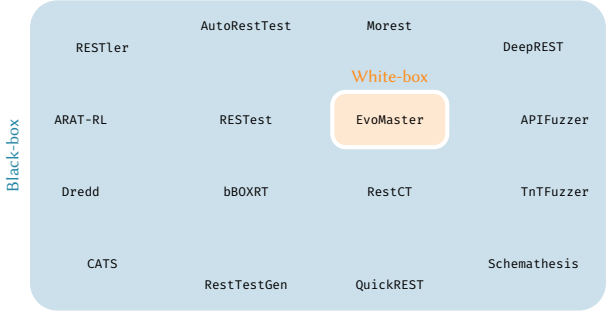
```yaml
openapi: 3.0.3
info:
  title: Swagger Petstore - OpenAPI 3.0
  version: 1.0.11
servers:
  - url: https://petstore3.swagger.io/api/v3
tags:
  - name: pet
    description: Everything about your Pets

paths:
  /pet/{petId}/uploadImage:
    post:
      tags:
        - pet
      summary: uploads an image
      description: ''
      operationId: uploadFile
      parameters:
        - name: petId
          in: path
          description: ID of pet to update
          required: true
          schema:
            type: integer
            format: int64
        - name: additionalMetadata
          in: query
          description: Additional Metadata
          required: false
          schema:
            type: string
      requestBody:
        content:
          application/octet-stream:
            schema:
              type: string
              format: binary
```



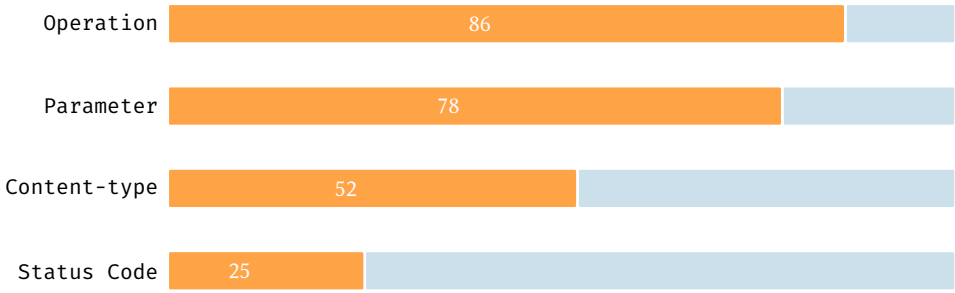
Testing tools





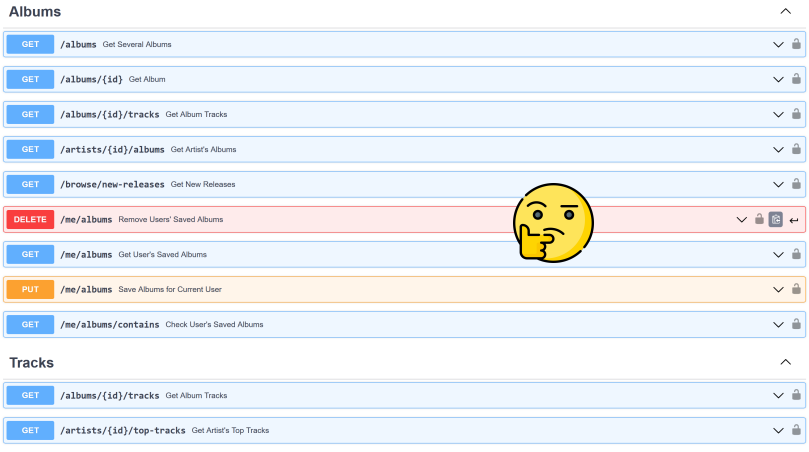
Test adequacy

Black-box coverage metrics



Challenges

Operation testing order



Challenges (Cont'd)

Operation parameter values

What are suitable input values for operation parameters?

- OpenAPI specifications often do not provide example values
- The validity of values might depend on the state of the API (e.g., resource ids)
- The validity of values might depend on multiple parameters
- Parameter types might be missing/wrong/incomplete

Challenges (Cont'd)

The oracle problem

Did the API under test behave as expected during/after test execution?

- Response status code might be unreliable
- OpenAPI specification might not adhere to the implementation
- Faults may affect interaction chains (e.g., vulnerabilities)

RestTestGen Framework

Develop your REST API testing tool

RestTestGen: REST API automated test case generation

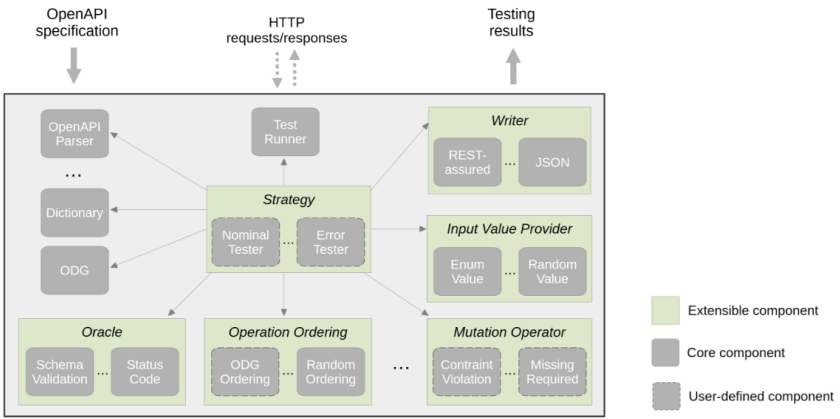
- Nominal and error testing
- Security testing (mass-assignment)

Developed by using the RTG Framework

- From design to implementation in minutes
- Wiki at `https://seunivr.github.io/RestTestGen-Wiki`
- DeepREST is a DRL-extension of RestTestGen

Develop your REST API testing tool (Cont'd)

Java-based extensible framework



OpenAPI specification parser

Robust parser, that deals with the common syntax mismatches and inconsistencies

- Fill internal API-specific structures to be used at testing time

Operation
method : enum // GET, POST, PUT, DELETE, … endpoint : string pathParameters: Set<Parameter> operationSemantics : enum // CREATE, READ, UPDATE, DELETE, … contentType : string requestBody : StructuredParameter responseBody : StructuredParameter …
getAllRequestParameters() : Collection<Parameter> getHeaderParameters() : Collection<Parameter> getPathParameters() : Collection<Parameter> getCookieParameters() : Collection<Parameter> …

```
Parameter
├─ StructuredParameter
│  ├─ ArrayParameter
│  └─ ObjectParameter
└─ LeafParameter
   ├─ StringParameter
   ├─ BooleanParameter
   ├─ NumberParameter
   └─ NullParameter
```

Test case

A `TestSequence` is an ordered list of HTTP interactions // test case

- A `TestInteraction` is a single HTTP data exchange with the API

TestSequence
testInteractions : List<TestInteraction>
isExecuted( ) : boolean inferVariablesFromContext( ) : void

TestRunner
instance : TestRunner
run(TestSequence) : void trytestInteractionExecution(TestInteraction) : void

TestInteraction	
referenceOperation : Operation	
requestMethod : enum	// GET, POST, PUT, DELETE, …
requestURL : string	
requestHeader : string	
requestSentAt : TimeStamp	
responseProtocol : string	// HTTP/1.1, …
responseStatusCode : HttpStatusCode	// 200, 404, 500, …
responseBody : string	
responseReceivedAt : TimeStamp	

Operation Dependency Graph

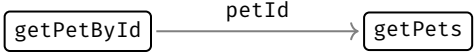
Producer-consumer relation between operations

```
/pets:
  get:
    summary: List all pets
    operationId: getPets
    tags:
      - pets
    responses:
      '200':
        description: PetIds
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  petId:
                    type: integer
```

output

```
/pets/{petId}:
  get:
    summary: Info for a specific pet
    operationId: getPetById
    tags:
      - pets
    parameters:
      - name: petId
        in: path
        required: true
        schema:
          type: string
```

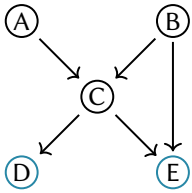
input



Operation Dependency Graph (Cont'd)

Leaf nodes are selected (no outgoing edges)

- No input needed
- Ordered based on CRUD semantics

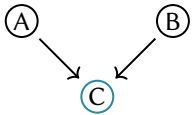


1. HEAD
2. POST
3. GET
4. PUT/PATCH
5. DELETE

Operation Dependency Graph (Cont'd)

Tested operations are removed

- New operations become leaf nodes
- Enabled to be tested



The class `OperationSorter` is responsible for deciding the order of operations in a test sequence

Static Ordering is performed before starting test execution

Dynamic Ordering defined during the test execution (e.g., based on ODG)

Parameter value generation

The interface `ParameterValueProvider` provides a value for an operation parameter

`ExampleValueProvider`

- Random value from OAS examples

`DefaultValueProvider`

- Default value from OAS

`RandomValueProvider`

- Random value from OAS constraints

`DictionaryValueProvider`

- Value from a dictionary of previously observe values

Parameter value generation (Cont'd)

Wrapper classes combining multiple value provider for an operation parameters

`RandomSelectorInputValueProvider`

- Random value providers selected from those compatible with the parameters

`EnumExamplePriorityInputValueProvider`

- Higher priority is given to enum and example value provider

`GlobalDictionaryInputValueProvider`

- Higher priority is given to the dictionary value provider

Predefined fuzzers

The `Fuzzer` class generates test sequences (operation order and input values)

`NominalFuzzer`

- Simulate different valid input scenarios to test an operation

`ErrorFuzzer`

- Simulate different erroneous inputs to test API error handling

`MassAssignmentFuzzer`

- Generate sequences to spot mass assignment vulnerabilities

Mutation operators

A mutator changes the value of a given parameter

- To intensify testing of an operation
- To try make a successfully tested operation to fail

`OperationMutator`

- Randomly change a parameter value, toggle a required parameter, …

`ParameterMutator`

- Force OAS constraints violation, inject values with wrong type, …

Test oracle

An oracle makes a decision on a test sequence, by emitting a test result

PENDING	The test case has not yet been executed
PASS	The test case has passed (no fault)
FAIL	The test case did not pass (fault revealed)
ERROR	The test case has encountered an error during execution
UNKNOWN	The oracle is not able to make a decision

Test oracle (Cont'd)

`StatusCodeOracle`

- PASS when all interactions get a 2XX
- FAIL when at least one interaction gets a 5XX
- UNKNOWN when at least one interaction gets a 4XX

`SchemaValidationOracle`

- PASS when data in the HTTP response matches the OAS schema
- FAIL otherwise

`MassAssignmentOracle`

- FAIL if the vulnerability has been successfully exploited
- PASS otherwise

Oracle example

SchemaValidationOracle

```yaml
responses:
  '200':
    description: Expected response to a valid request
    content:
      application/json:
        schema:
          $ref: "#/components/schemas/Pet"
```

```yaml
components:
  schemas:
    Pet:
      type: object
      required:
        - id
        - name
        - tag
      properties:
        id:
          type: integer
          format: int64
        name:
          type: string
        tag:
          type: string
```

```
{
  "id": 1,
  "name": "doggy",
  "tag": "dog"
}
```

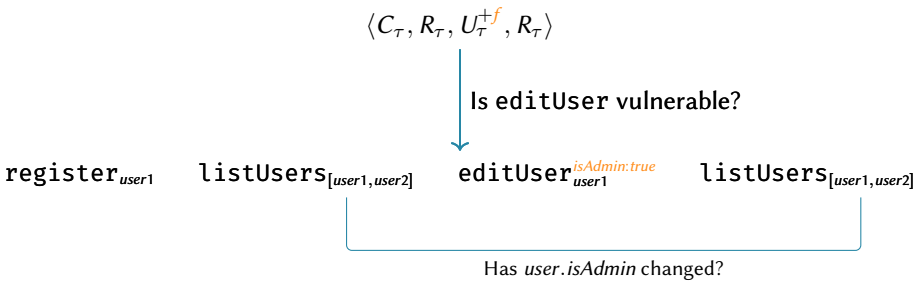
```
{
  "id": 1,
  "name": "doggy",
}
```

```
{
  "id": 1,
  "name": "doggy",
  "tag": 5
}
```

Oracle example (Cont'd)

`MassAssignmentOracle`

- Unauthorized create/update of read-only parameters



Result writer

Export the details of executed test sequences to files for later analysis/usage

`ReportWriter`

- To JSON file, including HTTP requests/responses and test oracle results

`HtmlReportWriter`

- To HTML file, including HTTP requests/responses and test oracle results

`CoverageReportWriter`

- To JSON file, including detailed coverage results

`RestAssuredWriter` and `JUnitWriter`

- To Java test cases using RESTAssured or JUnit libraries

Coverage metrics

Input coverage metrics

- Path coverage
- Operation coverage
- Parameter coverage
- Parameter value coverage
- Request content-type coverage

Output coverage metrics

- Status code class coverage
- Status code coverage
- Response content-type coverage

Automatically computed as defined by Martin-Lopez et al.[1]

- By using the provided classes (e.g., `OperationCoverage`)

[1] A. Martin-Lopez, S. Segura, and A. Ruiz-Cortés. *Test coverage criteria for RESTful web APIs* (A-TEST 2019)

Testing strategy

The `Strategy` class is the entry point of the framework

- Represent the tool logic for generating test cases
- Coordinate the framework components, possibly after customization

Some are already available

- `NominalAndErrorStrategy`, `MassAssignmentSecurityTestingStrategy`, ...

`DemoStrategy`

RestTestGen Setup

Installation

Requirements

- Java >= 11 // required
- Gradle (via Gradle Wrapper) // no separate install needed
- Docker // optional, to run APIs
- IntelliJ IDEA // optional, recommended for development
- Ubuntu 22.04 LTS // recommended OS

Installation (Cont'd)


```
# Clone the repository
git clone https://github.com/SeUniVr/RestTestGen.git
cd RestTestGen

# Make the Gradle wrapper executable
chmod +x gradlew

# Build the project (dependencies are auto-downloaded)
./gradlew assemble
```

Demo REST API

List all books

Gather data of a book

Create a new book

Update data of a book

Delete a book


```
[
  {
    "id": 1,
    "title": "Software Engineering, Vol. 1",
    "author": "Mariano Ceccato",
    "price": 16.5,
    "copiesSold": 10
  }
]
```

Configuration

Start the REST API

```
# Download from DockerHub and run
docker pull davidecorradini94/bookstore:latest
docker run -p 80:8080 davidecorradini94/bookstore:latest

# Verify it is running
curl -v http://localhost:80/books
```

`api-config.yml` ~RestTestGen/apis/bookstore/

```
name: Book Store
specificationFileName: bookstore-openapi.json    # relative to 'bookstore/specifications' dir
host: http://localhost:80   # base URL of the running API instance

# authenticationCommands:
#   default: python3 apis/bookstore/scripts/bookstore-auth.py
#   other: 'echo {"name": "apikey", "value": "davide", "in": "query", "duration": 6000}'
```

Configuration (Cont'd)

rtg-config.yml ~RestTestGen/

```yaml
apiUnderTest: bookstore    # matches the 'apis' subdirectory name
strategyClassName: DemoStrategy
logVerbosity: INFO
testingSessionName: my-first-session
resultsLocation: local    # writes to 'apis/bookstore/results' directory
```

Start testing session

```
# Run with Gradle
./gradlew run
```

REST API Testing Ecosystem

Testing the testing tool

In the context of REST APIs, it is hard to:

- Defining effective testing strategies
- Find working, testable case studies
- Compute testing metrics
- Retrieve the ground truth

We provide a toolchain to support REST API testing development

https://github.com/SeUniVr

Benchmarking infrastructure

RESTgym

- Extensible container-based testing infrastructure
- Automated orchestration engine
- Built-in test case generation tools and APIs
- Built-in testing metrics

Run and initialize REST APIs

Setup and run (black-box) testing tools

Manage the parallel execution of testing sessions

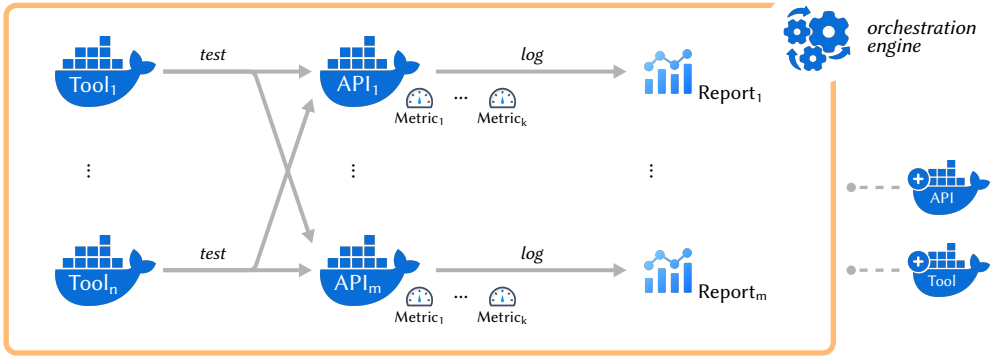
Perform health checks during testing

Collect experimental data from testing sessions

Aggregate results and provide detailed reports

Overview

Built-in test case generation tools and APIs retrieved from Docker Hub



User-provided test case generation tools and APIs locally built

More details

Eleven state-of-the-art REST API testing tools

Seventeen real-world REST APIs *from 2 to 59 operations / from 500 to 83k LoC*

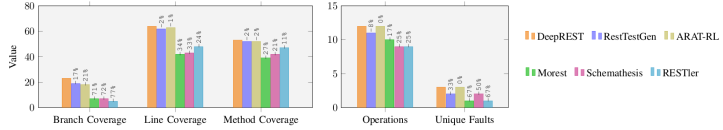
Three consolidated REST API testing metrics:

Code Coverage – Operation Coverage – Unique Faults

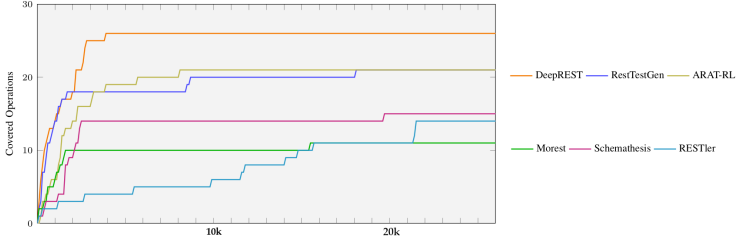
Tutorial on how to plug-in new tools and APIs

Testing reports

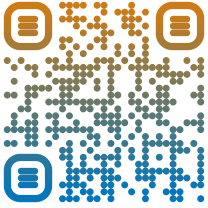
Aggregate effectiveness results on all APIs



Efficiency trends on a single API



REST API testing toolchain

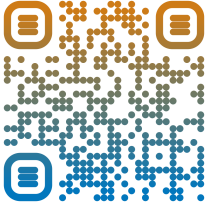


RTG

https://github.com/SeUniVr/RestTestGen

RESTgym

https://github.com/SeUniVr/RESTgym



Clapping (hopefully)

Thanks for the attention!

LaTeX is the way
... and `beamer` is not an enemy

Additional Slides

Testing strategy


```
OperationsSorter sorter = new GraphBasedOperationsSorter();
while (!sorter.isEmpty()) {
  Operation operationToTest = sorter.getFirst();
  NominalFuzzer nominalFuzzer = new NominalFuzzer(operationToTest);
  List<TestSequence> nominalSequences =
     nominalFuzzer.generateTestSequences(config.getNumberOfSequences());
  for (TestSequence testSequence : nominalSequences) {
    // run test sequence
    TestRunner testRunner = TestRunner.getInstance();
    testRunner.run(testSequence);
    // evaluate sequence with oracles
    StatusCodeOracle statusCodeOracle = new StatusCodeOracle();
    statusCodeOracle.assertTestSequence(testSequence);
    // write report to file
    ReportWriter reportWriter = new ReportWriter(testSequence);
    reportWriter.write();
    RestAssuredWriter restAssuredWriter = new RestAssuredWriter(testSequence);
    restAssuredWriter.write();
  }
  sorter.removeFirst();
}
```

Testing strategy (Cont'd)


```
private TestSequence generateTestSequence() {
  editableOperation = operation.deepClone();
  resolveCombinedSchemas();
  populateArrays();
  setValueToLeaves();
  // create a test interaction from the operation
  TestInteraction testInteraction = new TestInteraction(editableOperation);
  // encapsulate the test interaction into a test sequence
  TestSequence testSequence = new TestSequence(this, testInteraction);
  String sequenceName =
     !editableOperation.getOperationId().isEmpty() ?
     editableOperation.getOperationId() :
     editableOperation.getMethod().toString() + "-" + editableOperation.getEndpoint();
  testSequence.setName(sequenceName);
  testSequence.appendGeneratedAtTimestampToSequenceName();
}
```

Authentication script

drive-auth.py

Google Drive API

```python
import json

# Get token from https://developers.google.com/oauthplayground/
# WARNING: do not use your personal Google Drive account
#  - RestTestGen may erase your data on Google Drive!

token = "your_token_here"

rtg_info = {
  "name": "Authorization",
  "value": "Bearer " + token,
  "in": "header",
  "duration": 600
}

print(json.dumps(rtg_info))
```