OpenAPI Extension Proposal

What is an OpenAPI Extension ?

The OpenAPI Specification version 2.0 allows for custom properties to be added at several places within an OpenAPI document., allowing API providers to extend the meta-data provided for their REST APIs as needed.

For example, it would be possible to :

- add a price list and a description of the different offers.
- Add description of the limits of the API in term of maximum throughput, number of requests per second, hour, etc…
- …

Our extension proposal

Our contribution consists of adding description of the non-functional aspects of the API. Among all existing non-functional properties defined in software engineering, a portion of them can be applied to a web service :

- Response time / Performance : Describes the time between request sending and response arrival in milliseconds, seconds, … Or the number of requests per unit.
- Availability : Average status of response to a periodic ping (every x minute, hours, …)
- Adaptability : Ease with which the service can be changed to fit changes in requirements.
- Security : Measures for confidentiality, authenticity and integrity, i.e., the capability of system or service to protect information so that unauthorized services / applications cannot read or modify them, and authorized services / applications are not denied access to them.

Motivations

In its current form, the OpenAPI Specification describes the API purely from a functional point of view. It provides information about the input and output of each endpoints, what to expect from the server in response of what we sent.

In future, when a developer will consider using purposefully an external service, he will choose from a list of different services performing the same function. Even though these services are different, they fulfil the same function. Nothing differentiates them from a functional point of view. Everything would be different if the developer was aware that one of the services doesn't allow more than x requests per hour, or if another insure a low latency service 24/7.

Today, this developer would have to check the different plans to find more information about the limits of the API in term of throughput. Moreover, about latency and response time, he would have to test himself all the API's to figure out which one is better (this can be subject to pure code performance, network quality or even geographical deployment).

With this Non-Functional OpenAPI Extension, the developer will immediately know which service will fit best to its necessities.

Dimensions

For its first version, our Non-Functional OpenAPI Extension will focus on two properties : latency and uptime.

Latency

Latency is all about API performance, it defines the time spent between the moment an HTTP request is sent to the moment the response of this request comes back to the client. This is measured in milliseconds.

Before describing latency testing configuration, we define :

- An operation record as a measure of one operation of the API at given time from a given region.
- An operation test as all the operation records during the whole period from a given region.
- An API test as the sequentially measure of all operations contained in the API at a given time from a given region.
- A parameter definition strategy as the strategy used by the tool to define parameters in order to complete the request requirements (in body, path, query, header or formdata), there are different :
    - The "initial" strategy which consists of assigning the initial value of the type to the parameter.
        - If the type is primitive, it will use '0' for a number or the empty string for a string, for example.
        - If the type is not primitive, but a schema, it will assign the reference and apply the same strategy for these reference parameters.
    - The "default" strategy which consists of assigning the default value provided by the OpenAPI Specification.
    - The "random" strategy which consists of assigning a random value to the parameter. Same as the "initial" strategy, according to the type, primitive or not, the assignment will change.
    - The "example" strategy which consists of assigning the examples provided by the OpenAPI Specifications to the parameter.

    - The "provided" strategy which consists of assigning values provided by the tester to the parameter.

To configure latency testing, multiple information must be provided :

- The number of times the API test will occur.
- The time interval between API tests.
- The list of regions of the world from which the API tests will occur.
- The threshold value after which a request is considered as timed out.
- The parameter definition strategy which the tool will use.

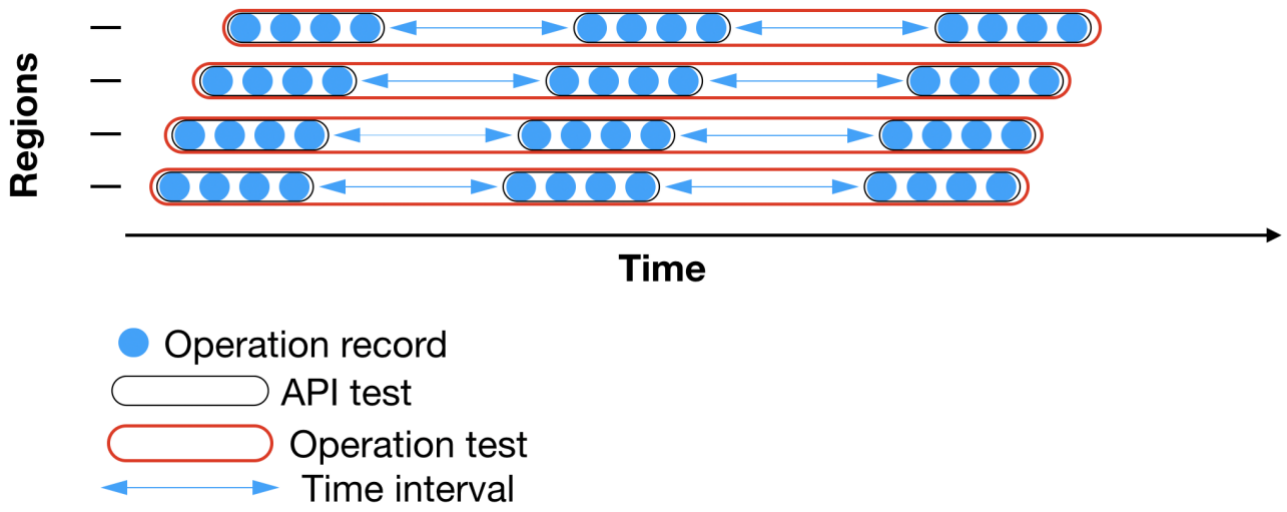
This configuration is enough for a tool to test the API and gather the latency measures.

The process of measuring is the following :

According to the configuration, a server from the region will test sequentially all the operations, then wait for the given time interval before testing it again, repeating it the number of times given.

These measurements are described as follows :

- An operation record is defined by the time it occurred and :
    - If the response resulted to an error, the error code, a message explaining the error and latency measure set to null.
    - If the response didn't result to an error, the code, the latency measure in milliseconds.
- An operation test is defined by the total number of records, the success and latency average of these records. The latency average doesn't include records resulting in an error.



Uptime

Uptime is all about API availability, it defines the capability of an API to respond requests. The uptime can take only two values, true or false, which can be translated to available or unavailable.

In this dimension, it is important to differentiate between API and Web Server. An API is supposed to run on one or multiple servers (to performance, geographical distributivity and other concerns…), if one server is down and the routing is not properly, it will result to an API unavailability, even if the API is available from another terminal, country, etc.

Before describing uptime testing configuration, we define :

- An uptime record as a measure of the API at given time from a given region.
- An uptime test as all the uptime records during the whole period from a given region.

To configure uptime test, the following information must be provided :

- The number of times the uptime test will occur.
- The interval between each uptime tests.

This configuration is enough for a tool to test the API and gather the uptime measures.

The process of uptime testing consists of sending an HTTP request at the root endpoint of the API (url + basePath) with the OPTIONS method.

The measurement of an uptime record contains this information :

- The result of the test, as defined above, true or false.
- The date at which the test occurred.

OpenAPI Specifications Injection proposal

How OpenAPI can be extended

The OpenAPI Specification version 2.0 allows for custom properties to be added at several places within an OpenAPI document., allowing API providers to extend the meta-data provided for their REST APIs as needed. Extension properties are always prefixed by "x-" and can have any valid JSON format value. (Source : GitHub)

Currently extension properties are supported in the info object, the paths object, the path-items object, the operation object, the parameter object, the responses object, the tag object, the security-scheme object.

What to inject, and where

We have 3 different information to implement in the OpenAPI Specifications :

- Latency and Uptime configurations
- Latency measurements
- Uptime measurements

Latency and Uptime configurations

We propose to insert in the info object all information relative to the configuration of these tests.

```
"x-nonFunctionalPropertiesConfiguration": {
  "latency": {
    "repetitions": 150,
    "interval": "P1D",
    "parameterDefinitionStrategy" : "initial",
    "timeoutThreshold": 5000,
    "zones": [
      "asia-east2",
      "australia-southeast1"
    ]
  },
  "uptime": {
    "repetitions": 500,
    "interval": "PT1H30M",
    "zones": [
      "asia-northeast2",
      "northamerica-northeast1"
    ]
  }
}
```

For the latency part :

- Repetitions (number) represents the number of times the API will be tested.
- Interval (string), represent the time of interval between tests, in ISO8601 format.
- ParameterDefinitionStrategy (string), represent the strategy used by the tool to define parameters in operations.
- TimeoutThreshold (number) represent the time the tool should wait until considering the request to be timed out, in milliseconds.

- Zones (array of strings) contains all identifier for each region from which the tests will occur. (*Note to SOM : Maybe we should add information about the cloud service provider, as these information are supposed to be computer readable, which means that the tool parsing this file should know which provider use in order to test, as AWS, GCP and Azure doesn't use the same nomenclature of region names. So maybe it should look like : [{ regionId : "asia-northeast2", provider : "GCP"}, { regionId : "australia1", provider : "AWS"}]).*

For the uptime part, it is the same as latency part, except we don't have timeout threshold value *(Or should we ?)*.

Latency measurements

We propose to insert in the operation object all information relative to the results of latency tests.

In this object, we will find objects named with the identifier of region set in the configuration with :

- Records (array) which contains the list of records defined by :
  - The date (string) the test occurred, in ISO8601 format.
  - The error (boolean) :
    - If there was an error, then latency will be set at 0 and the code should be >= 400.
    - If there was no error, then latency will be >0 and the code should be < 400.
  - The code (string) returned by the API.
  - The latency (number) which represent, as defined above, the time between the request sending and the response returning, in milliseconds.
- The mean latency (number) which represent the average latency of all successful operation records (which means it doesn't count error resulting records), in milliseconds.
- The average success (number) of operation records, in percentage.

Uptime measurements

We propose to insert in the info object all information related to the results of uptime tests.

In this object, we will find objects names with the identifier of region set in the configuration which contains :

- Records (array) which contains the list of records defined by :
  - The date (string) the test occurred, in ISO8601 format.
  - The state (boolean) of the API.
- The overall availability (number) based on records, and which represent the availability of the API during the testing period.