SE API Standards

Release date : Jul 1, 2020 V3.0

Change Log

Version	Release Date	Changes
V1.0	Jan 31, 2019	Initial Version
V2.0	Jun 1, 2019	• Versioning rules • camelCase for field names • Added 2 slides, at the end, on REST
V3.0	Jul 1,2020	To add • Basepath rule • Additional Design Rules • Golden Rules

Summary

There exists many different types of APIs. This document only focuses on the standards for REST API (it is the most popular API type).
We use OAS – **O**pen **A**PI **S**pecification to describe API definition (Contract).

This document is primarily for standards, please read the companion documents/resources –

- Getting Started page for details about the access, process, training,etc.
- API eXperience Playbook.

Golden Rules

Golden Rules

API is an interface to the backend service(s)
API name **MUST NOT** reveal implementation aspects, e.g., API name should not include mySE, BFO,etc..
Every API must be compliant to '**SE API Standard**'.
Default is **REST** and NOT **SOAP**. Follow standard description (OAS aka Swagger)
Design API First, before developing service. Once service is built then publish the same as API onto Google APIGEE. We have 2 instances of Google APIGEE, use as per guidelines given below -

- 👉Onto SEAPIM (SEACat id 50600), if its IT API
- 👉Onto Exchange.se.com if its OT API

Must use AViD (Stoplight.io) to -

- visually design new/modify API contracts
- persist existing API contracts in one place

Third party APIs are not mandated to be published AS-IS onto the SEAPIM Platform unless SE data is exposed. Don't publish Google Map API, but do publish bFO APIs as later exposes SE data.

Use SEAPIM (Google APIGEE),
WHEN backend is 1 or more webservices and/or microservices (either orchestrate or as passthrough).

API Versioning

API Versioning

Rule : Do NOT maintain more than 3 versions of the same API. Best practice is to maintain 2.

Rule : API Version format major.minor* is permissible for APIs e.g., v1.0 v1.1, v2.0
However, **only major version number should appear in an URI**. If version is omitted, it maps to the latest version by default. This is to reduce number of changes to the URI & hence less impact on the Consumer side.

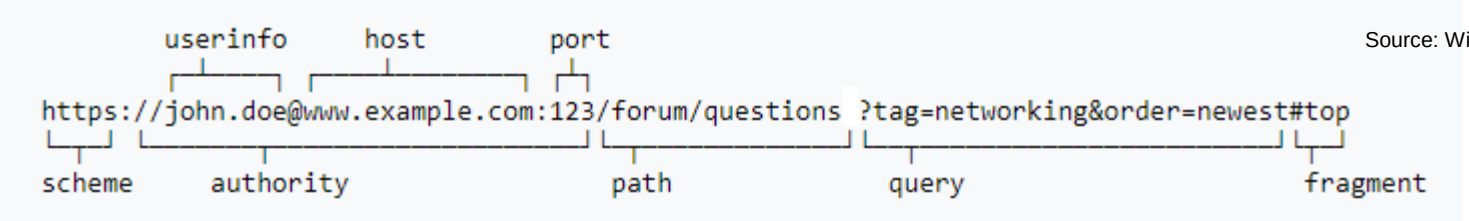
*Source = https://semver.org/
Where Major = MAJOR version, when one makes incompatible/breaking** API changes, MINOR version, when one adds functionality in a backwards-compatible manner

**Breaking changes include:

- Renaming fields and/or resource paths
- Adding or removing (required) input parameters
- Renaming or removing fields from the response structure
- Fixing poor choices of HTTP verbs, response codes, or inconsistent design across existing API endpoints

API Design

API Design – URI format



Source: Wikipedia

URI example:-
https://api.example.com/device-management/managed-devices?model=iphone#year

Rule: Use lowercase only
Rule: Use hyphens; avoid underscore
Rule: No forward slash (/) at the end
Rule: Do not use file extension such as xml or json*

*See examples below:
- https://api.example.com/device-management/managed-devices.xml /*Do not use it*/
- https://api.example.com/device-management/managed-devices /*This is correct URI*/

API Design – domain (authority) name

- API endpoint – https://api.<..se>.com
- Developer portal – https://developer.<..se>.com

← Production

Sandbox → •API – https://api.sandbox.<..se>.com

Any deviation must be reported for exception reporting, tracking!

Note: Based on API DX Mandate # 3

API Design – API Resource Naming

RESTful URI (Universal Resource Identifier) must refer to a resource which is a thing (noun) instead of referring to an action (verb). Use HTTP Verbs (GET,PUT, POST,DELETE,etc..) to act on the URI.

- Use **Noun** and not Verb*

e.g., do not use ‘getEmployees’ instead use ‘employees’

- Use **Plural** not singular**

e.g. ‘products’ and not ‘product’

- Use **Hyphens** (-) not Underscores

e.g. ‘managed-devices’ and not ‘managed_devices’

Use ‘camelCase’ for attribute’s name in the request/response payload

Verb	Objective
GET	Retrieve items from resource
POST	Create new item in resource
PUT	Replace existing item in resource
PATCH	Update existing item in resource
DELETE	Delete existing item in resource

*Exception: URI returns a calculated value or executes an action instead of returning a resource e.g.,
- https://api.example.com/cart-management/users/{id}/cart/checkout
- https://api.example.com/song-management/users/{id}/playlist/play

**To identify singular resource use identifier after the API name, e.g.,
- https://api.example.com/device-management/managed-devices/{device-id}
- https://api.example.com/user-management/users/{id}

API Basepath Rules

New APIs on APIGEE must follow the following rules. Deviations to be approved by Remi Poujeaux or Hirok Choudhury.

On the right is the list of capabilities and value to be used in the URI.
Use the Workbook to get a basepath for your API.

API URI format and examples:

https://<host-name>/**<base-path>**/<resource>[/<id-template>].....

<base-path> format: v<version-no>/<capability>[/<activity>]/<api-name>

https://api.se.com/v2/reference-data/installed-base/installed-products
https://api.se.com/v1/customer-support/alarm/cases/{case-id}
https://api.se.com/v1/it/system/myse-guided-search/{country}/{language}/search

General Rules:

1. Basepath components must be singular words
2. No special characters but compound words may have hyphens
3. Do not introduce extra components in the basepath format
4. Must select one of 15 capability values
5. Do not introduce a new capability value

Capability Name	Value in URI
Customer/Persona journey	customer-journey
OfferDesign/Offer Creation Management	offer-design
Marketing Engagement	marketing-engagement
Marketing Planning & Execution	marketing-plan-exec
Sales Operations	sales-operation
Pricing & Quotations	pricing-quote
Customer Order Intake	customer-order
Fulfillment	fulfillment
CustomerSupport	customer-support
IT	it
Finance, Controlling & Legal	finance-legal
HR Operations	hr
Security	security
Reference Data	reference-data
Architecture, Technology & Platform	technology

API Design – additional rules

Common Resources can be accessed from below links:
OAS2 version
OAS3 version

Streaming: Response code 206 is required for a request using streaming.

Paging – Standard Common Resources template in AViD should be use.

Filtering – TBD

Sorting - TBD

Additional API Management Rules

Additional rules & guidelines

- Default is OAuth 2.0
- Use TLS 1.2 or above to connect to api.se.com endpoints
- Long-lived tokens are discouraged, requires exception
- Message payload size limit is 10MB . Use streaming for bigger size
- Check explorer before creating a new API design.
- To subscribe (use) an existing API, search in the catalog, then ‘try it’ to check suitability and once found ‘fit for purpose’ then subscribe to it.
- APIs with no traffic in last 90 days will be candidate for decommissioning
- Minimize number of hops between API Management layer and the actual source of record to improve performance (API response time).
- Use full capabilities of the SEAPIM platform such as orchestration, caching, extensions, etc..
- Ensure backend webservice development is complete before asking for publishing onto APIGEE.

Acknowledgement

Acknowledgements

We thank Remi Poujeaux and Melissa Johnston for defining the API Basepath Rules. <included in V3.0>

Appendix

Key Input Sources

- Book - REST API Design Rulebook by Mark Masse
- https://blog.octo.com/en/design-a-rest-api/
- eBook "Web API Design: The Missing Link” from Apigee
- https://apigee.com/about/blog/taglist/api-design
- https://www.baeldung.com/rest-versioning
- https://apigee.com/about/blog/technology/restful-api-design-nouns-are-good-verbs-are-bad
- https://www.thoughtworks.com/insights/blog/rest-api-design-resource-modeling

REST API – simplified view

‘Request’ API to get ‘Response’.

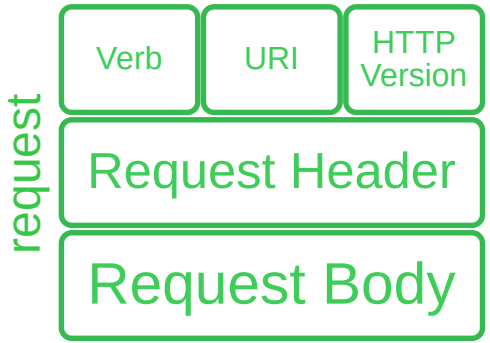
In this context, API is analogous to someone (waiter) who serves us at a restaurant ☺.

A contract is the agreement/specification for the API. An analogy of API contract would be USB or wall sockets. Probably some of us don’t even recall what USB stands for, which is OK as its part of our DNA (daily life).
However we shall continue caring about API till it gets into our DNA!

We use the ‘Open API Specification’ formerly known as Swagger for describing an API.

REST API

You 'Request' API to get 'Response'



An HTTP Request has five major parts −

- ✓ Verb − Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.
- ✓ URI − Uniform Resource Identifier (URI) to identify the resource on the server.
- ✓ HTTP Version − Indicates the HTTP version. For example, HTTP v1.1.
- ✓ Request Header − Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc.
- ✓ Request Body − Message content or Resource representation.

An HTTP Response has four major parts −

- ✓ Status/Response Code − Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is ok.
- ✓ HTTP Version − Indicates the HTTP version. For example HTTP v1.1.
- ✓ Response Header − Contains metadata for the HTTP Response message as key value pairs. For example, content length, content type, response date, server type, etc.
- ✓ Response Body − Response message content or Resource representation.