REST Principles

DD Owned by David Cherney DISH •••
Last updated: Jun 21, 2023 • Initializing...

Contents

REST is short for representational state transfer.

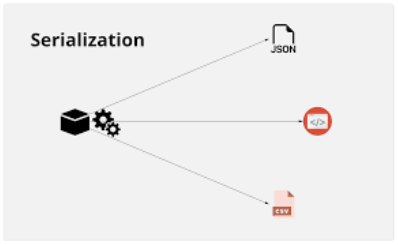
RESTfull is a title that is claimed, not earned. REST is not a protocol or architecture, but rather a collection of design principles that one should adhere to when trying to make an API that will be called RESTfull. Every system that claims to adhere to REST principles will do so imperfectly. That includes the 5G SBA.

Before presenting those principles, it will pay to have some terminology.

REST Terminology

Resource Serialization

Resources are (often) information that needs to be sent. To allow the sending of that information, the information needs to be written as a document: JSON format string, YAML document, HTML, et cetera. This writing of information into a document is called serialization. The serialization should be the choice of the client; if the client requests JSON then the client should get JSON.



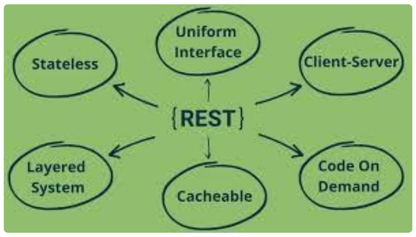
Resource Representation

Each time a resource is serialized into a document a representation of resource is formed. A representation of a resource is not the resource itself. Representation is what puts the “RE” in REST.

The 5G core SBA consumers (clients) only request JSON, and all producers (servers) serialize information into JSON documents. Thus SBA adheres to this principle closely in the sense that it can serialize in the requested way. On the other hand, in general, an API is said to be representational if it can support serialization into multiple formats, and in this sense the 5G core’s architecture, SBA, is not representational.

(It would be nice to find some documentation on serialization options for the NEF, who is meant to be a server for applications which presumably would like to run `GET` methods with specification for serialization other than JSON.)

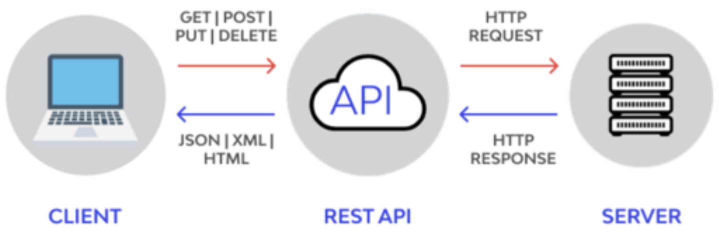
The 6 Principles



1) Client/Server Principle

In all REST APIs, the role of client shall be clearly distinguished from the role of server.

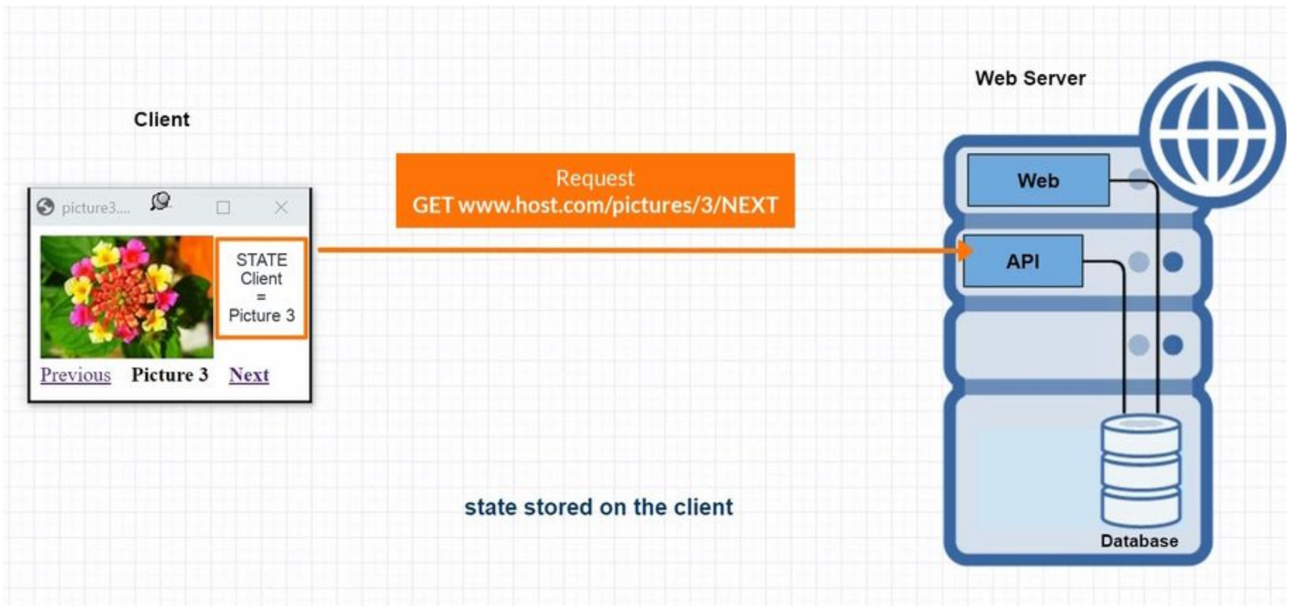
In 5G SBA these terms are replaced with consumer and producer, and this principle is tightly adhered to.



2) Stateless Principle

In a client-server interaction, the server shall not keep any state information about clients. Instead, the client shall send information about its state.

For example, you are going through a slide deck and click the next button. An interaction that follows the stateless principle would send to a server “currently on slide 3, send next slide”. An interaction that would *not* follow the principle is for you to send to the server “look up where I am in the slide deck and give me the next slide” since that would require “where I am in the slide deck” to be stored in the server.



Note that the stateless principle is a bit of a misnomer; one might think the principle is about *state* (that client and server should not have a state). This is not the case. Rather, the stateless principle is about *state transfer (*who stores and transmits information about states in a client server interaction). That is “X is stateless” means something different than “X adheres to the stateless principle”.

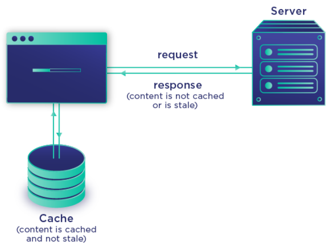
In order to adhere to the stateless principle 5G NFs utilize storage. For example, the AMF and NRF can use the UDSF (unstructured data support function) to store state information about UEs and NFs so that the information is not stored in the AMF or NRF; when the AMF is the client for a API call to change a UE’s configuration, it does not remember the UE’s current configuration, but rather uses an API `GET` call to the UDSF to get the current configuration, then makes an API `POST` call to the UDSF to make a change.

By their nature `PUT` and `POST` API calls to the UDSF are not stateless, but the separation of client role and storage role better fit the design principle than combining roles would.

3) Cacheable Principle

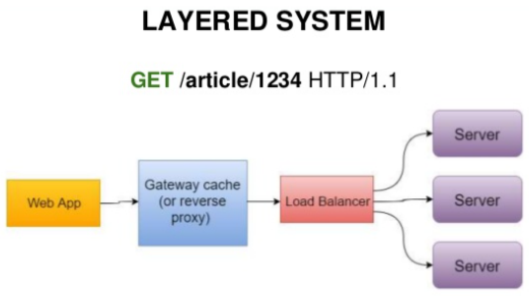
Clients should be informed if the representations they receive are representations of information that will not change. This allows the client to know if the data can be safely cached in the client for later use by the client, or if such caching has the potential for not-timely data use.

5G SBA follows this principle carefully.


Server
request
response
(content is not cached or is stale)
Cache
(content is cached and not stale)

4) Layered System Principle

Between a client and a server with the desired resource, there may be a series of servers serving as intermediaries. These servers might provide security, load balancing, etc. Each server in the chain should be from a distinct functional layer. Further, Clients shall be unaware of the specific chain of servers they are accessing, and aware only of the service the server at the end of the chain provides.



For example, if the AMF acts as a client to an NRF, no information about which NRF instance is acting as server is passed to the AMF.

Clients shall be unaware of any actions of their server as clients. For example, if the AMF acts as a client to the NRF in requesting a list of SMFs, the AMF shall be unaware that the NRF queries the UDSF.

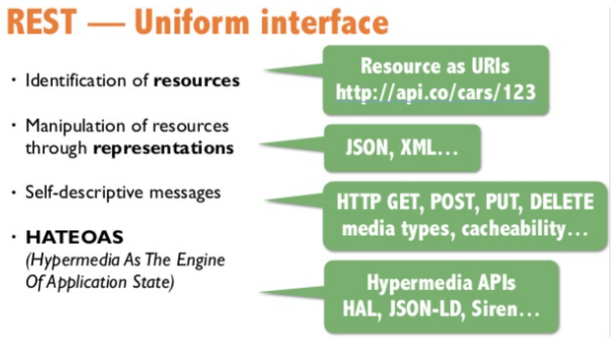
5) Code on Demand Principle

Pieces of executable code shall be downloadable from servers to clients. This allows the extension of functionality between client and server in future interactions.

This principle is not currently implemented in 5G because SBA does not currently require this flexibility.

6) Uniform Interface Principle

This principal has 4 parts.


REST — Uniform interface
• Identification of **resources**
Resource as URIs http://api.co/cars/123
• Manipulation of resources through **representations**
JSON, XML...
• Self-descriptive messages
HTTP GET, POST, PUT, DELETE media types, cacheability...
• **HATEOAS** (Hypermedia As The Engine Of Application State)
Hypermedia APIs HAL, JSON-LD, Siren...

1: Resource Identification:

Each resource shall have a unique address. Further, it is not the servers that have unique addresses, but the resources that they provide. This means that if two servers provide the same resource, the address of that resource does not *need to* specify which server to access.

SBA follows this part of the principle by allowing either specific instances of NFs to be addressed, or equivalent instances. In the example above the API endpoint `https://nrf7.slice-v2x.opx.3gpp/nrf-nfm/v1/nf-instances/smf5-slicev2x` specified a particular NRF instance by the name of `nrf7.slice-v2x.opx.3gpp`. By 3GPP specs, the same resource would be obtained from the endpoint `https://nrf.opx.3gpp/nrf-nfm/v1/nf-instances/smf5-slicev2x` which addresses any/all NRF insistences that are equivalent to `nrf7.slice-v2x.opx.3gpp` as potential clients.

(The addressing system is not crystal clear to me, and neither is the method by which one of the equivalent NF instances is chosen as server.)

2: Resource Manipulation Through Representation:

Resources can be modified or replaced by a client via the client sending an API call that includes a representation of all or part of the resource.

For example a `PATCH` method API call from and SMF to a NRF can modify the service profile entry in the NRF by including either all of the service profile information, or just the part that needs to be updated.

SBI follows this part of the principle closely.

3: Self Descriptive Messages:

Requests and responses shall include enough information to indicate how the message should be processed.

4: Hypermedia and the Engine of Application State (HATEOAS)

Response messages shall inform the client of further actions that can be taken on the resource while the resource is in its current state. This information shall be in the form of hyperlinks.

This part of the principle has yet to be applied to any service in SBA.

Recap

RESTful is a descriptive term for API calls. It describers adherence to a collection of principles. No API adheres to these principles perfectly.