Design of RESTful API for NFFG verifier

1. Conceptual structure of the resources

Basic structure

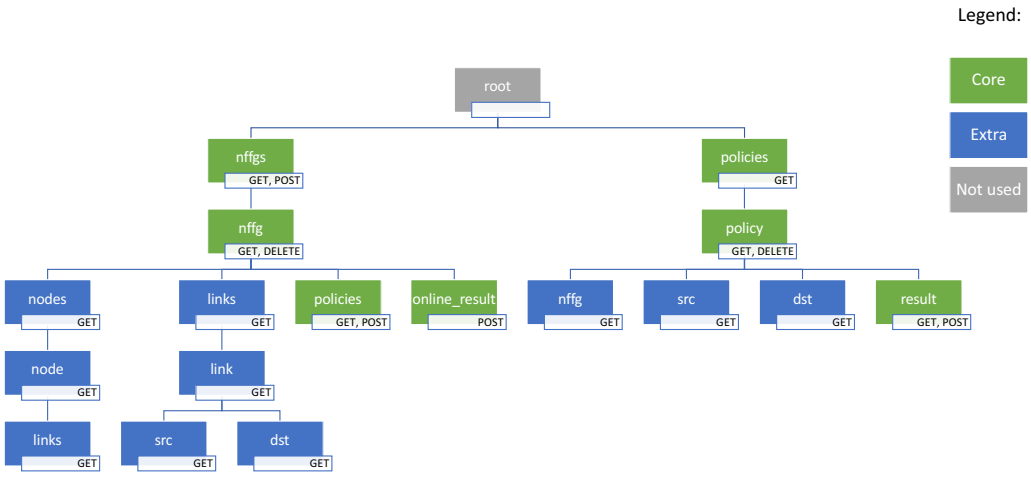
The conceptual structure of the data to be represented in the service has a hierarchical structure that is:

- there is a collection of **NFFGs** (top level resource)
- the collection is a set of **NFFG** (child resource), that contain information about nodes and links
- each NFFG has a child resource that represents **policies**
- policies is a collection of **policy** resources
- each policy can have a **result** resource

```
nffgs
  |
 nffg
  |
policies
  |
 policy
```

This basic schema has resources only for unit of data that need to be manipulated separately. This is not the complete structure that I designed. The complete structure, that is more complete and complex, will be explained in details and is needed in order to make the data available in an easier way for the clients.

Complete structure



The service consists of two top level resources: one for NFFGs and the other one for policies.

Policies placement

The policies are made available as root elements because the clients may want to get the whole set of policies stored inside the server, without being specific on policies belonging to a single NFFG. Since the name of a policy is unique not only iniside a single NFFG, but has a global scope, it is appropriate that also the child resource policy belongs to this subtree.

The policies resource remains as child of nffg only for creation/update and for getting the collection of the policies of a specific nffg.

Policy creation and modification

Single policies can be created and updated using the same procedure: a POST request on the resource policies child of nffg. Because the requirements specify that if a new policy is submitted with the same name as an already stored one, a replacement will occurr, to update a policy the client simply need to send the new version into a POST request using the same name (on the policies resource child of the correct nffg) (TODO check the nffg in the request??).

Ignored fields of POST requests

The POST requests done on resource whose ancestors are identified by a name, may contain a reference to the ancestor. Those fields will be ignored by the service.

For example, a POST request for creating a policy may contain the nffg name. This information, since is already provided by the path on which the POST request is done, is not required and will be ignored.

Other notes on the strucure TODO

TODO

In the following paragraphs the resources are explained in more detail, including the information they store and the HTTP methods allowed on them. (TODO better to avoid two times enumeration of resources. Do it after explaining the mapping to URLs)

nffgs

This is the first out of the two root resources. It represents the collection of NFFG elements.

method	explaination
GET	read the collection of NFFGs
POST	add a new NFFG to the service

Nffg

The NFFG resource stores informations about a nffg: name, last update, nodes, links.

method	explaination

GET	read the info about the NFFG
DELETE	delete the NFFG from the service

Policies

This resource represents the collection of policies.

method	explaination
GET	read the collection of policies
POST	add a new policy to the service

Policy

The policy resource stores informations about a policy: name, source node, destination node, list of functionalities to be traversed (optional), the result of verification (optional), a reference to the NFFG to be considered, the information about the positiveness of the policy.

method	explaination
GET	read the info about the policy
DELETE	delete the policy from the service

Result

method	explaination
GET	read the verification result
POST (called on update nested resource)	requests an update of the verification result

2. Mapping of the resources to URLs

The tree structure of the resources previously shown is reflected on the URLs used. Curly braces are used in the following when the path contains an identifier.

URL	resource type	method	usage
/nffgs	nffgs	GET	obtain the collection of NFFGs
		POST	store a new NFFG
/nffgs/{nffg_name}	nffg	GET	obtain a single NFFG given its name
		DELETE	delete a single NFFG given its name
/nffgs/{nffg_name}/policies	policy	GET	obtain the collection of policies belonging to a NFFG whose name is given

		POST	store a new policy belonging to a NFFG whose name is given
/nffgs/{nffg_name}/online_result	result	POST	obtain the result of a policy provided in the request testing it against an existing NFFG given its name
/policies TODO			
/policies/{policy_name}			
policies/{policy_name}			

Additional resources can be used to obtain partial information about the data.

TODO nodes, links, ...

```
/                                                   GET, DELETE
    nffgs/                                          GET, POST
          {nffg_id}/                                GET, DELETE
                    nodes/                          GET
                             {node_id}/             GET
                                        links/      GET
                    links/                          GET
                             {link_id}/             GET
                                        src/        GET
                                        dst/        GET
                    policies/                       GET, POST
                             {policy_id}/           GET, PUT/PATCH
                    online_result/                  POST (because need body
request)
    policies/                                       GET (flat view. readonly??)
          {policy_id}/                              GET, DELETE
                    nffg/                           GET
                    result/                         GET, POST to request an
update of the result
                    src/                            GET
                    dst/                            GET
```

3. Operations by resource

/ verifier root

This is the root resource, also the entry point. Returns all the informations about NFFGs and policies.

method	request type	response type	explaination	errors
GET	-	verifier	get the informations (pagination)	
DELETE	-	OK	delete all the informations	

/nffgs

NFFGs collection

method	request type	response type	explaination	errors
GET	-	nffgs	get the collection of NFFGs (pagination)	
POST	nffg_req	nffg	create a new NFFG	4xx wrong request, 403 already existing

TODO: the id of nffg (named entities) is the name? PROs: same type in POST request and in response, no duplicated unique attribute, no need to store name mappings to ids. CONs: the name is always a valid URL?

/nffgs/{nffg_id}

A single nffg identified by its id

method	request type	response type	explaination	errors
GET	-	nffg	get the NFFG	404: wrong nffg_id
DELETE	-	OK	delete the NFFG	404: wrong nffg_id

/nffgs/{nffg_id}/nodes

Nodes collection for this NFFG

method	request type	response type	explaination	errors
GET	-	nodes	get the nodes	404: wrong nffg_id

/nffgs/{nffg_id}/nodes/{node_id}

A node identified by its id

method	request type	response type	explaination	errors
GET	-	node	get the node	404: wrong nffg_id/node_id

/nffgs/{nffg_id}/links

Links collection for this NFFG

method	request type	response type	explaination	errors
GET	-	links	get the links	404: wrong nffg_id

/nffgs/{nffg_id}/links/{link_id}

A link identified by its id

method	request type	response type	explaination	errors
GET	-	link	get the link	404: wrong nffg_id/link_id

/nffgs/{nffg_id}/policies

Policies collection for this NFFG

method	request type	response type	explaination	errors
GET	-	policies	get the collection of policies (pagination)	404: wrong nffg_id
POST	policy_req	policy	create a new policy	404: wrong nffg_id, 4xx: wrong request; always replace

/nffgs/{nffg_id}/policies/{policy_id}

A policy identified by its id

method	request type	response type	explaination	errors
GET	-	policy	get the policy	404: wrong id
DELETE	-	OK	delete the policy	404: wrong id

/nffgs/{nffg_id}/policies/{policy_id}/result

The result of this policy

method	request type	response type	explaination	errors
GET	-	result	the verification of policy is performed and the result is both stored and returned	404: wrong id

/nffgs/{nffg_id}/online_result

Verification endpoint for client policies, not stored on the service

	request	response

method	type	type	explaination	errors
POST	policy_req	policy/result??	get the result for this policy	404: wrong id, 4xx: wrong request

/policies

This is a flat view on the policies. But should be readonly: POST requests should also contain the NFFG name/id and another type should be created.

There is a queryParam named from that can be used to get policies only from a certain time.

??? TODO: replication of subtree about policies

/policies/{policy_id}

A single policy identified by its id.

/policies/{policy_id}/nffg

The nffg this policy belongs to

/policies/{policy_id}/src

The src node if this is a reachability policy (always)

/policies/{policy_id}/dst

The dst node if this is a reachability policy (always)

/policies/{policy_id}/result

The corresponding result for this policy