Design of RESTful API for NFFG verifier

1. Conceptual structure of the resources

Basic structure

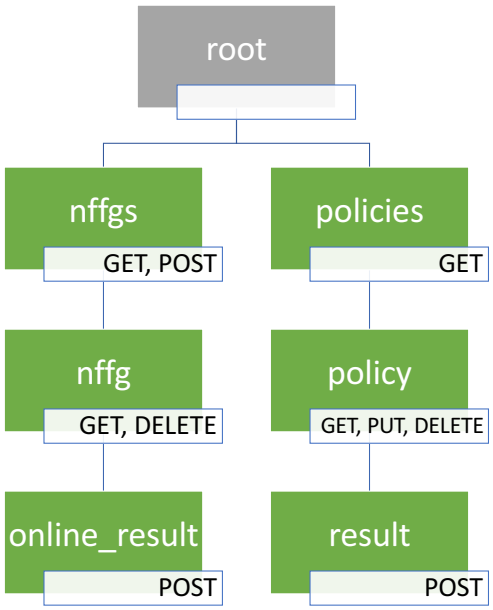
The conceptual structure of the data to be represented in the service has a hierarchical structure that is:

- 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
   |
 result
```

This basic schema has resources only for unit of data that need to be manipulated separately. This is not the actual structure that I designed. The complete structure, that has some more resources and moves the policies resource as a top level one, 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. 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.

Therefore the policies resource has no more purpose of existing.

Policy creation and modification

Single policies can be created and updated using the same procedure: a PUT request on the policy resource child of policies. 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 or to create a policy the client can use the same PUT request. The HTTP method chosen is PUT, because it is idempotent.

Other notes on the strucure TODO

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
/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	policies	GET	obtain the collection of all the policies
/policies/{policy_name}	policy	GET	obtain a single policy given its name
		PUT	store a policy on this resource (both creation or update)
		DELETE	delete a single policy given its name
			obtain (if any) the stored result for the stored

/policies/{policy_name}/result	result	GET	policy given its name
		POST	recompute and obtain the result of a stored policy

IDs

Since both the NFFGs and the policies are identified by their name (as specified in the requirements of assignment 1) and since their pattern is quite strict (only alphabetical characters), the names can be used directy as path elements and therefore the ID of the resources is chosen by the client.

For the NFFGs creation, the POST method is used because the creation is not idempotent as explained in the requirements (creation of an NFFG with the same name as an existing one is forbidden). Instead for the policies, since the creation of a policy with the same name as one already stored in the service is allowed and is used for replacement, the creation of a policy uses the same method as the replacement/update. In this way the client directly submits a PUT request to the path that contains the name of the policy, and since PUT is idempotent the creation follows the same approach as an update.

3. Operations by resource

All the operations are available with Content-Type (both requests and responses) application/xml or application/json. The types used are the ones contained in the XSD.

Errors on all the resources: 406 500 400

/nffgs

NFFGs collection

method	request type	response type	explaination	result	errors
GET	-	nffgs	get the collection of NFFGs	200 OK	-
POST	nffg	nffg	create a new NFFG	201 CREATED	422(400) validation error, 409 already existing

The POST request must contain the field name, that will be the identifier of the created resource if the request succeeds. In case the name is already used by another stored NFFG, the service returns a HTTP 403 error. Instead if the request itself contains an error when doing validation of the data contained, the service returns a HTTP 422 error.

/nffgs/{nffg_name}

A single nffg identified by its name.

method	request type	response type	explaination	result	errors

GET	-	nffg	get the NFFG	200 OK	404: no NFFG exists with this name
DELETE	-	nffg	delete the NFFG	200 OK	404: no NFFG exists with this name

/nffgs/{nffg_name}/online_result

Verification endpoint for client policies, not stored on the service

method	request type	response type	explaination	result	errors
POST	policy	policy	verify this policy	201 CREATED	404: wrong id, 422(400): validation error

/policies

Policies collection

method	request type	response type	explaination	result	errors
GET	-	policies	get the collection of policies	200 OK	

queryParam:

- nffg: only get policies for a specific NFFG
- from: get policies only from a certain time

/policies/{policy_name}

A single policy identified by its id.

method	request type	response type	explaination	result	errors
GET	-	policy	get the policy	200 OK	404: no policy exists with this name
DELETE	-	-	delete the policy	200 OK	404: no policy exists with this name
PUT	policy	policy	update/create the policy	200 OK, 201 CREATED	404: no nffg with this name, 422: validation error

/policies/{policy_name}/result

The corresponding result for this policy.

method	request	response	explaination	result	errors

	type	type			
POST	-	policy	update the policy result	200 OK	404: no policy exists with this name