NffgService Documentation

Silvia Vitali 231748

Table of Contents

Design Description

The NffgService provides the possibility to create and access to a set of Nffgs. A Nffg (*Network Function Forwarding Graph*) allows to model a a specific network as a graph where nodes are network-functions and end-hosts while arcs are packets forwarding paths. It is possible to verify on the server different policies, already present or not in the service.

The design of the service is based on a constrained design approach, in order to follow as much as possible REST principles.

The service is based on two interfaces:

- *Nffg interface*: allow access to a set of Nffgs, create a new Nffg or delete a specific Nffg.
- *Policy interface*: allow access to a set of policies. It is possible to create, delete or update a specific policy. Using this interface the client can also verify if one or more policies are satisfied.

In order to increase performances, minimize coupling and design a service that could be flexible and opened to future changes, different strategies have been applied: the service is organized in order to show to the client only what is needed, in according with the *information hiding* principle and the granularity level is a trade-off between too fine grain and too coarse grain partitioning. The number of interactions necessary to access to the services is limited and well defined.

The service is designed also to foresee all particular cases in order to be robust and secure.

The service is design following the *RESTful principles*. Resources are organized in a hierarchical way, they are uniquely identified and correspond to collections of entities or to a single entity. Interfaces are used to define resources while actions are implemented using http methods. URLs use plural names for collections and singular names for entities and reproduce the hierarchical relationship of resources. Resources relationships are represented, when needed, in a nested way.

Version information

Version : 1.0.0

URI scheme

Host : localhost:8080
BasePath : /NffgReader/rest
Schemes : HTTP

Tags

- nffg : A specific Nffg
- nffgs : A collection of Nffgs
- policies : A collection of Policies
- policy : A single Policy

Paths

Create a new Nffg

```
POST /nffg
```

Description

Metod used to add a new Nffg to the service if it is not present yet.

Parameters

Type	Name	Schema
Body	**body** *optional*	nffg

Responses

HTTP Code	Description	Schema
201	Nffg Created	No Content
400	Bad request	No Content
500	System Error	No Content

Consumes

- `application/xml`

Produces

- `application/xml`

Tags

- nffg

Get a Nffg

```
GET /nffg/{id}
```

Description

Metod used to get a specific Nffg from the server passing the corrispondent id.

Parameters

Type	Name	Schema
Path	**id** *required*	string

Responses

HTTP Code	Description	Schema
200	OK	nffg
404	Not Found	No Content
500	Internal Server Error	No Content

Produces

- `application/xml`

Tags

- nffg

Delete a Nffg

```
DELETE /nffg/{id}
```

Parameters

Type	Name	Schema
Path	**id** *required*	string

Responses

HTTP Code	Description	Schema
200	OK	No Content
404	Not Found	No Content

HTTP Code	Description	Schema
500	Internal Server Error	No Content

Tags

- nffg

Get all Nffgs

```
GET /nffgs
```

Description

Metod used to return all the Nffgs of the service.

Responses

HTTP Code	Description	Schema
200	OK	nffgs
204	The service is empty	No Content
500	Internal Server Error	No Content

Produces

- `application/xml`

Tags

- nffgs

Get and test Policies

```
GET /policies
```

Description

Metod used to return all the Policies of the service, with the verify parameter setted to true is possible to test policies passing and array of ids.

Parameters

Type	Name	Schema
Query	**id** *optional*	< string > array
Query	**verify** *optional*	string

Responses

HTTP Code	Description	Schema
200	OK	No Content
204	The service is empty	No Content
500	Internal Server Error	No Content

Produces

- `application/xml`

Tags

- policies

Test Policies

```
PUT /policies
```

Description

Metod used to test Policies without save them on the server.

Parameters

Type	Name	Schema
Query	**verify** *optional*	string
Body	**body** *optional*	policies

Responses

HTTP Code	Description	Schema
200	OK	array result
400	Bad request	No Content
404	Nffgs or Nodes not found	No Content
500	Internal Server Error	No Content

Consumes

- `application/xml`

Produces

- `application/xml`

Tags

- policies

Create a new Policy

```
POST /policy
```

Description

Metod used to create a new Policy.

Parameters

Type	Name	Schema
Body	**body** *optional*	policy

Responses

HTTP Code	Description	Schema
201	Created	No Content
400	Bad request	No Content
404	Nffg or nodes not found	No Content
500	Internal Server Error	No Content

Consumes

- `application/xml`

Produces

- `application/xml`

Tags

- policy

Get or test a single Policy

```
GET /policy/{id}
```

Description

Return a Policy passing the corrispondent id. With the verify parameter setted to true the Policy is tested.

Parameters

Type	Name	Schema
Path	**id** *required*	string
Query	**verify** *optional*	string

Responses

HTTP Code	Description	Schema
200	OK	policy
404	Policy not found	No Content
500	Internal Server Error	No Content

Produces

- `application/xml`

Tags

- policy

Update a Policy

```
PUT /policy/{id}
```

Description

Update a Policy already in the service.

Parameters

Type	Name	Schema
Path	**id** *required*	string
Body	**body** *optional*	policy

Responses

HTTP Code	Description	Schema
201	OK	No Content
400	Bad request	No Content
404	Nffgs or Nodes not found	No Content
500	Internal Server Error	No Content

Consumes

- `application/xml`

Produces

- `application/xml`

Tags

- policy

Delete a Policy

```
DELETE /policy/{id}
```

Parameters

Type	Name	Schema
Path	**id** *required*	string

Responses

HTTP Code	Description	Schema
200	OK	No Content
404	Policy not found	No Content
500	Internal Server Error	No Content

Tags

- policy

Definitions

link

Name	Schema
dest *optional*	string
name *optional*	string
source *optional*	string

nffg

Name	Schema
link *optional*	< link > array
name *required*	string
nodeType *required*	< nodeType > array

Name	Schema
updateTime *optional*	string(date-time)

nodeType

Name	Schema
functionalType *required*	string
name *optional*	string

policies

Name	Schema
policy *optional*	< policy > array

policy

Name	Schema
dest *optional*	string
functionalType *optional*	< string > array
name *optional*	string
nffgName *optional*	string
policyDesc *required*	string
result *optional*	result
source *optional*	string
value *optional*	boolean

result

Name	Schema
message *required*	string
resultValue *optional*	boolean
vTime *optional*	string(date-time)