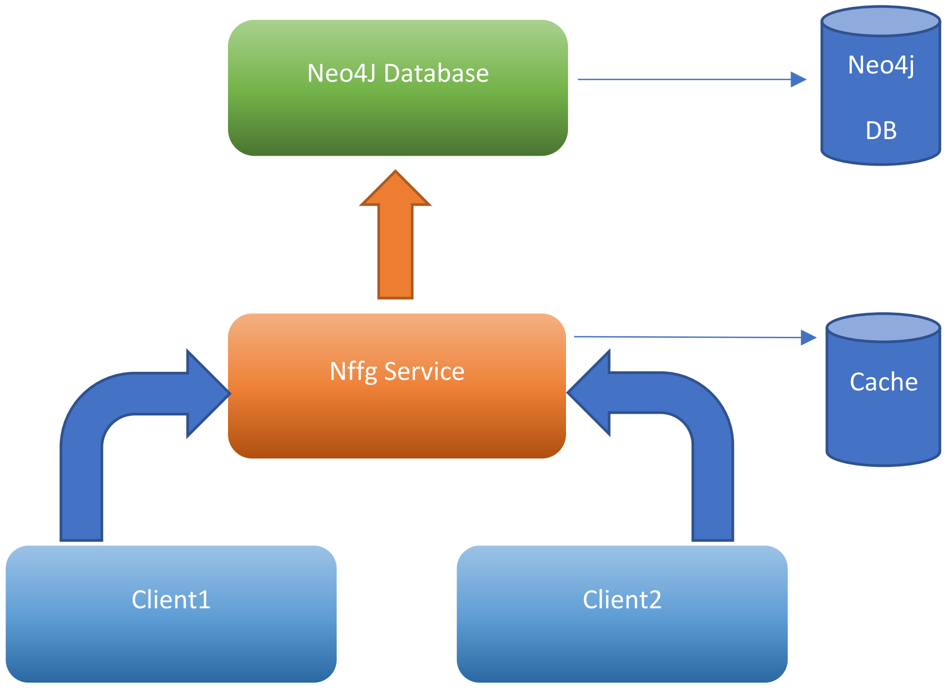
# NffgService REST API Documentation

Riccardo Persiani 225289

Distributed Programming II, A.Y 2016/2017

## 1. The Actual System

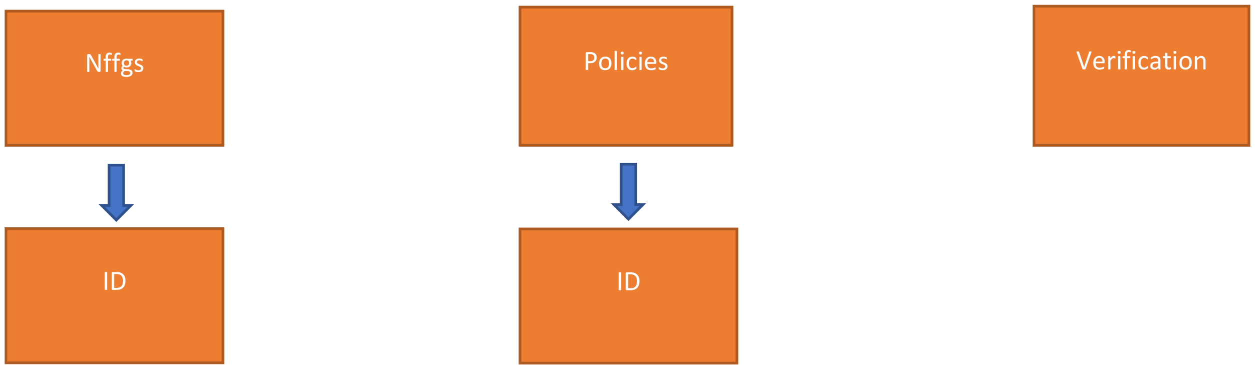


The whole system is divided in three levels:

- Neo4j: a highly performant graph database where Nffgs, Nodes and Relationships element are stored and graphically represented through a friendly user interface at the address http://localhost:7474/browser/
- NffgService: the core element which can be seen as a Client sending HTTP request to the Neo4j database, in order to store informations inside it and as a Server, managing concurrent HTTP requests coming from clients.
- Clients: two clients has been developed in order to test the functions, the correctness and the robustness of the server, sending HTTP requests to the NffgService.

## 2. Resources

This is the resource overview:



NffgService Rest API includes the following resources:

| Resource URL | Description |
|---|---|
| localhost:8080/NffgService/rest/ | Base URL of the service |
| localhost:8080/NffgService/rest/application.wadl | WADL descriprion |
| localhost:8080/NffgService/rest/nffgs | The set of nffgs |
| localhost:8080/NffgService/rest/nffgs/id | The nffg identified by id |
| localhost:8080/NffgService/rest/policies/ | The set of policies |
| localhost:8080/NffgService/rest/policies/id | The policy identified by id |
| localhost:8080/NffgService/rest/verification/ | The verification resource |

## 3. Operations

This is the operations summary:

| Resource URL | Operation |
|---|---|
| localhost:8080/NffgService/rest/nffgs | **1. POST**<br>**2. GET**<br>**3. DELETE** |
| localhost:8080/NffgService/rest/nffgs/id | **4. GET**<br>**5. DELETE** |
| localhost:8080/NffgService/rest/policies | **6. POST**<br>**7. GET**<br>**8. DELETE**<br>**9. PUT** |
| localhost:8080/NffgService/rest/policies/id | **10. GET**<br>**11.DELETE** |
| localhost:8080/NffgService/rest/verification | **12. PUT**<br>**13. POST** |

As requested, some operations have not been implemented.

## 1. Create Nffg

This method develop the loading of an NFFG element on the server, the NFFG must be valid and must not be already loaded in order to make the request succeed.

*Example Request*

| POST | localhost:8080/NffgService/rest/nffgs |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<NFFG xmlns="http://www.riccardopersiani.com/Schema"
name="Nffg3"last_update_time="2017-01-27T13:50:31.634+01:00">
   <Nodes>
      <Node name="Node1">
            <Service>NAT</Service>
      </Node>
      <Node name="Node2">
            <Service>Mail client</Service>
      </Node>
   </Nodes>
   <Links>
      <Link name="Link1">
         <Source>Node1</Source>
         <Destination>Node2</Destination>
      </Link>
   </Links>
   <Policies>…<Policies/>
</NFFG>
```

*Example Response*

| 201 | CREATED |
|---|---|
| **Content-Type** | application/xml |
| **Location:** | localhost:8080/NffgService/rest/nffgs/NFFG3 |

| HTTP Status Code | Description |
|---|---|
| **201 CREATED** | SUCCESS |
| **400 BAD REQUEST** | FAILURE – NFFG not valid |
| **409 CONFLICT** | FAILURE – NFFG already stored |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 2. Get all Nffgs

This method returns in its body request the list of all the Nffgs that are stored in the server.

*Example Request*

| **GET** | localhost:8080/NffgService/rest/nffgs |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| **200** | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Nffgs xmlns="http://www.riccardopersiani.com/Schema">
   <NFFG name="Nffg4"last_update_time="2017-01-27T13:50:29.972+01:00">…</NFFG>
   <NFFG name="Nffg3"last_update_time="2017-01-27T13:50:31.634+01:00">…</NFFG>
   <NFFG name="Nffg7"last_update_time="2017-01-27T14:24:18.205+01:00">…</NFFG>
   <NFFG name="Nffg2"last_update_time="2017-01-27T13:50:33.055+01:00">…</NFFG>
   <NFFG name="Nffg1"last_update_time="2017-01-27T13:50:34.830+01:00">…</NFFG>
   <NFFG name="Nffg6"last_update_time="2017-01-27T13:50:36.153+01:00">…</NFFG>
   <NFFG name="Nffg5"last_update_time="2017-01-27T13:50:39.695+01:00">…</NFFG>
   <NFFG name="Nffg0"last_update_time="2017-01-27T13:50:42.751+01:00">…</NFFG>
</Nffgs>
```

| **HTTP Status Code** | **Description** |
|---|---|
| **200 OK** | SUCCESS |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 3. Delete all Nffgs

This method allow the client to delete all the nffgs present in the database.

*Example Request*

| DELETE | localhost:8080/NffgService/rest/nffgs/ |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | text/plain |
| **BODY:** | Content-Length: 0 Bytes |

| HTTP Status Code | Description |
|---|---|
| **204 NO CONTENT** | SUCCESS |
| **404 NOT FOUND** | FAILURE – No content |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 4. Get Nffg by ID

This method returns in the body of the request the Nffg specified by ID.

*Example Request*

| GET | localhost:8080/NffgService/rest/nffgs/id |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<NFFG xmlns="http://www.riccardopersiani.com/Schema
name="Nffg3"last_update_time="2017-01-27T13:50:31.634+01:00">
   <Nodes>
        <Node name="Node1">
             <Service>NAT</Service>
        </Node>
        <Node name="Node2">
             <Service>Mail client</Service>
        </Node>
   </Nodes>
   <Links>
        <Link name="Link1">
            <Source>Node1</Source>
           <Destination>Node2</Destination>
        </Link>
   </Links>
   <Policies>…<Policies/>
</NFFG>
```

| HTTP Status Code | Description |
|---|---|
| **200 OK** | SUCCESS |
| **404 NOT FOUND** | FAILURE – No content |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 5. Delete Nffg by ID

This method allow the client to delete the Nffg specified in the request

*Example Request*

| **DELETE** | localhost:8080/NffgService/rest/nffgs/id |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| **200** | OK |
|---|---|
| **Content-Type** | text/plain |
| **BODY:** | |

| **HTTP Status Code** | **Description** |
|---|---|
| **204 NO CONTENT** | SUCCESS |
| **404 NOT FOUND** | FAILURE – Nffg not found |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 6. Create a Policy

This method develop the loading of a Policy element on the server, the Policy must be valid in order to make the request succeed. The client can load a TraversalPolicy or a ReachabilityPolicy. As requested, if a policy is still in the database, it will be updated.

*Example Request*

| **POST** | localhost:8080/NffgService/rest/policies |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<Policy xmlns="http://www.riccardopersiani.com/Schema">
    <TraversalPolicy name="Policy1">
        <Source>WEBCLIENT2</Source>
        <Destination>WEBSERVER2</Destination>
        <Nffg>Nffg0</Nffg>
        <isPositive>true</isPositive>
        <Devices>
            <Device>NAT</Device>
            <Device>Firewall</Device>
            <Device>Mail client</Device>
        </Devices>
    </TraversalPolicy>
</Policy>
```

*Example Response*

| **201** | CREATED |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | Content-Length: 0 Bytes |

| HTTP Status Code | Description |
|---|---|
| **201 CREATED** | SUCCESS – Policy successfully created |
| **204 NO CONTENT** | SUCCESS – Policy already in DB, Updated. |
| **400 BAD REQUEST** | FAILURE – Policy not valid |
| **404 NOT FOUND** | FAILURE – Nffg in Policy not exists |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 7. Get all policies

This method returns in its body request the list of all the Reachability and Traversal policies that are stored in the server.

*Example Request*

| GET | localhost:8080/NffgService/rest/policies |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Policies xmlns="http://www.riccardopersiani.com/Schema">
   <ReachabilityPolicy name="RP1">…</ReachabilityPolicy>
   <ReachabilityPolicy name="RP2">…</ReachabilityPolicy>
   <TraversalPolicy name="RP3">…</TraversalPolicy>
   <TraversalPolicy name="TP1">…</TraversalPolicy>
   <TraversalPolicy name="TP2">…</TraversalPolicy>
</Policies>
```

| HTTP Status Code | Description |
|---|---|
| **200 OK** | SUCCESS |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 8. Delete all Policies

This method allow the client to delete all the policies present in the database.

*Example Request*

| **DELETE** | localhost:8080/NffgService/rest/policies/ |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| **200** | OK |
|---|---|
| **Content-Type** | text/plain |
| **BODY:** | Content-Length: 0 Bytes |

| **HTTP Status Code** | **Description** |
|---|---|
| **204 NO CONTENT** | SUCCESS |
| **404 NOT FOUND** | FAILURE – No content |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 9. Get Policy by ID

This method returns in the body of the request the Nffg specified by ID.

*Example Request*

| GET | localhost:8080/NffgService/rest/policies/id |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Policy xmlns="http://www.riccardopersiani.com/Schema">
    <TraversalPolicy name="TP1">
        <Source>Node1</Source>
        <Destination>Node2</Destination>
        <Nffg>Nffg0</Nffg>
        <isPositive>true</isPositive>
        <Devices>
            <Device>NAT</Device>
            <Device>Firewall</Device>
            <Device>Mail client</Device>
        </Devices>
    </TraversalPolicy>
</Policy>
```

| HTTP Status Code | Description |
|---|---|
| **200 OK** | SUCCESS |
| **404 NOT FOUND** | FAILURE – No content |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 10. Delete Policy by ID

This method allow the client to delete the Policy specified in the request

*Example Request*

| **DELETE** | localhost:8080/NffgService/rest/policies/id |
|---|---|
| **Content-Type** | |
| **BODY:** | Content-Length: 0 Bytes |

*Example Response*

| **200** | OK |
|---|---|
| **Content-Type** | text/plain |
| **BODY:** | |

| **HTTP Status Code** | **Description** |
|---|---|
| **204 NO CONTENT** | SUCCESS |
| **404 NOT FOUND** | FAILURE – Policy not found |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 11. Update Policy

This method allow the client to update a policy already stored in the database. The policy to put in the body request must be valid in order to succeed the substitution of the old policy.

*Example Request*

| PUT | localhost:8080/NffgService/rest/policies/ |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```
<Policy xmlns="http://www.riccardopersiani.com/Schema">
    <ReachabilityPolicy name="RP1">
        <Source>Node1</Source>
        <Destination>Node2</Destination>
        <Nffg>Nffg0</Nffg>
        <isPositive>true</isPositive>
        <Verification>
            <Time>2016-09-26T10:58:20.000+02:00</Time>
            <Message>Policy verificarion result true</Message>
            <Result>true</Result>
        </Verification>
    </ReachabilityPolicy >
</Policy>
```

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | Content-Length: 0 Bytes |

| HTTP Status Code | Description |
|---|---|
| **204 NO CONTENT** | SUCCESS |
| **400 BAD REQUEST** | FAILURE – Policy not valid |
| **404 NOT FOUND** | FAILURE – Policy not found |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 12. Verify one or more Policies

This operation permits to verify the policy that are send to the server.

*Example Request*

| **PUT** | localhost:8080/NffgService/rest/verification/ |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<PoliciesToBeVerified xmlns="http://www.riccardopersiani.com/Schema">
    <name>ReachabilityPolicy2</name>
</PoliciesToBeVerified>
```

*Example Response*

| **200** | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<PoliciesVerified xmlns="http://www.riccardopersiani.com/Schema">
    <ReachabilityPolicy name="ReachabilityPolicy2">
        <Source>Node2</Source>
        <Destination>Node4</Destination>
        <Nffg>NFFG1</Nffg>
        <isPositive>true</isPositive>
        <Verification>
            <Time>2017-01-28T17:11:15.902+01:00</Time>
            <Message>Policy Positive and Reachable</Message>
            <Result>true</Result>
        </Verification>
    </ReachabilityPolicy>
```

| HTTP Status Code | Description |
|---|---|
| **200 OK** | SUCCESS |
| **400 BAD REQUEST** | FAILURE – PoliciesToBeVerifed not valid |
| **404 NOT FOUND** | FAILURE – Policy not found |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 13. Verify one or more Policies without storing them

This operation permits to verify the policy that are send to the server and that are not stored in the database.

*Example Request*

| PUT | localhost:8080/NffgService/rest/verification/ |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<PoliciesNotStored xmlns="http://www.riccardopersiani.com/Schema">
    <ReachabilityPolicy name="ReachabilityPolicy2">
        <Source>Node2</Source>
        <Destination>Node4</Destination>
        <Nffg>NFFG1</Nffg>
        <isPositive>true</isPositive>
    </ReachabilityPolicy>
<PoliciesNotStored >
```

*Example Response*

| 200 | OK |
|---|---|
| **Content-Type** | application/xml |
| **BODY:** | |

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<PoliciesNotStored xmlns="http://www.riccardopersiani.com/Schema">
    <ReachabilityPolicy name="ReachabilityPolicy2">
        <Source>Node2</Source>
        <Destination>Node4</Destination>
        <Nffg>NFFG1</Nffg>
        <isPositive>true</isPositive>
        <Verification>
            <Time>2017-01-28T17:11:15.902+01:00</Time>
            <Message>Policy Positive and Reachable</Message>
            <Result>true</Result>
        </Verification>
    </ReachabilityPolicy>
<PoliciesNotStored >
```

| HTTP Status Code | Description |
|---|---|
| **200 OK** | SUCCESS |
| **400 BAD REQUEST** | FAILURE – PoliciesNotStored not valid |
| **404 NOT FOUND** | FAILURE – Policy not found |
| **500 INTERNAL SERVER ERROR** | FAILURE – Error inside the server |

## 5. Design

The class used to develop the server are divided in several packages:

- it.polito.dp2.NFFG.sol3.service.resources: contains the implementation of the HTTP operations;
- it.polito.dp2.NFFG.sol3.service.database: contains the implementation of the nffgs database and the policies database, in addiction there are two classes, NffgInfo and PolicyInfo that represents nffg and policy elements inside the databases;
- it.polito.dp2.NFFG.sol3.service.neo4j: contains the types of Neo4J
- it.polito.dp2.NFFG.sol3.service.jaxb: contains the types of the NffgService
- it.polito.dp2.NFFG.sol3.service: contains one class (NffgService.java) which is the core of the web service, infact it is responsable for the intercommunication between the server and the Neo4J database and between multiple clients and the server. As requested, inside there are methods which allow to store Nffgs and policies inside the local database (actually a cache) and send HTTP request to Neo4J, for storing only the Nffgs.
- it.polito.dp2.NFFG.sol3.service.validation: contains classes that verify the correctness of the client requests (More details in Cap 8).

## 6. Persistency

The Nffgs are stored inside the Neo4J database and the same informations are available in the NffgService server cache which saves both nffgs and policies in two different DBs.
The first one is NffgsDB which refers to the nffg informations; it is implemented with an hash map <NffgName, NffgInfo> where NffgInfo is a class that memorize everything about the Nffg, including the NFFG Object.
The second is PoliciesDB which refers to the policy informations; it is implemented with an hash map <PolicyName, PolicyInfo> where PolicyInfo is a class that memorize everything about the policy, but not the ReachabilityPolicy and/or TraversalPolicy Object.
It must be clear that the information about a single policy is stored in the PoliciesDB under several variables of PolicyInfo and also in the NffgsDB inside the NFFG Object; This makes the policies data a bit redundant, but improve performance.

When the server have to return the whole nffg, the policies are already inside it and when must be returned just a single policy the server takes it fastly from the PolicesDB.

## 7. Concurrency

Because of the initial choice of realizing two databases (local caches), develop concurrency is really hard.
In the NffgService.java class where all the methods, regarding the client request, are implemented there are several synchronized blocks that must synchronize the double write on the two different databases.
In both the databases the has map has been declared as concurrent hash map.
The Unmarshaller in the validators, in addiction is not thread-safe, so in order to admit the concurrency, it has been created in the method readFrom() and not in the constructor.

## 8. Validation

3 classes has been developed in order to check if the client requests are valid or not.
One for the NFFG, one for the Policy and one for the PoliciesToBeVerified element.
As said previously the unmarshaller is created in the readFrom() method, because it is not Thread Safe.