fiware-ofnic Documentation

Release 1

Federico Cimorelli

January 11, 2017

Refer to this page for more general information about OFNIC.

Available manuals:

CHAPTER 1

Welcome to FIWARE OFNIC GE's documentation!

Refer to this page for more general information about OFNIC.

CHAPTER 2

User & Programmers

2.1 Introduction

OFNIC provides a programmable interface to control and retrieve information and control Openflow Networks. The interface is based on the RESTful paradigm and is powered by a Web Server developed in Python programming language. OFNIC relies on the Openflow protocol in order to abstract and virtualize forwarding capabilities of the Openflow Network. This GEi is an extension to the OpenDaylight Openflow controller.

OFNIC is composed of many modules which communicate whith each other with events. It is designed with an event-driven paradigm also to manage information that comes from the network. The OFNIC RESTful interface is an instance of the NetIC API (NetIC Generic Enabler Open Specifications can be found here).

Goal of this document is to provide a useful guide for developers who want to build new applications based on the NetIC API, and a guide for users that might utilize the provided GUI application. Specific sections are dedicated to developer tools intended to help them during development, test and deployment.

CHAPTER 3

User Guide

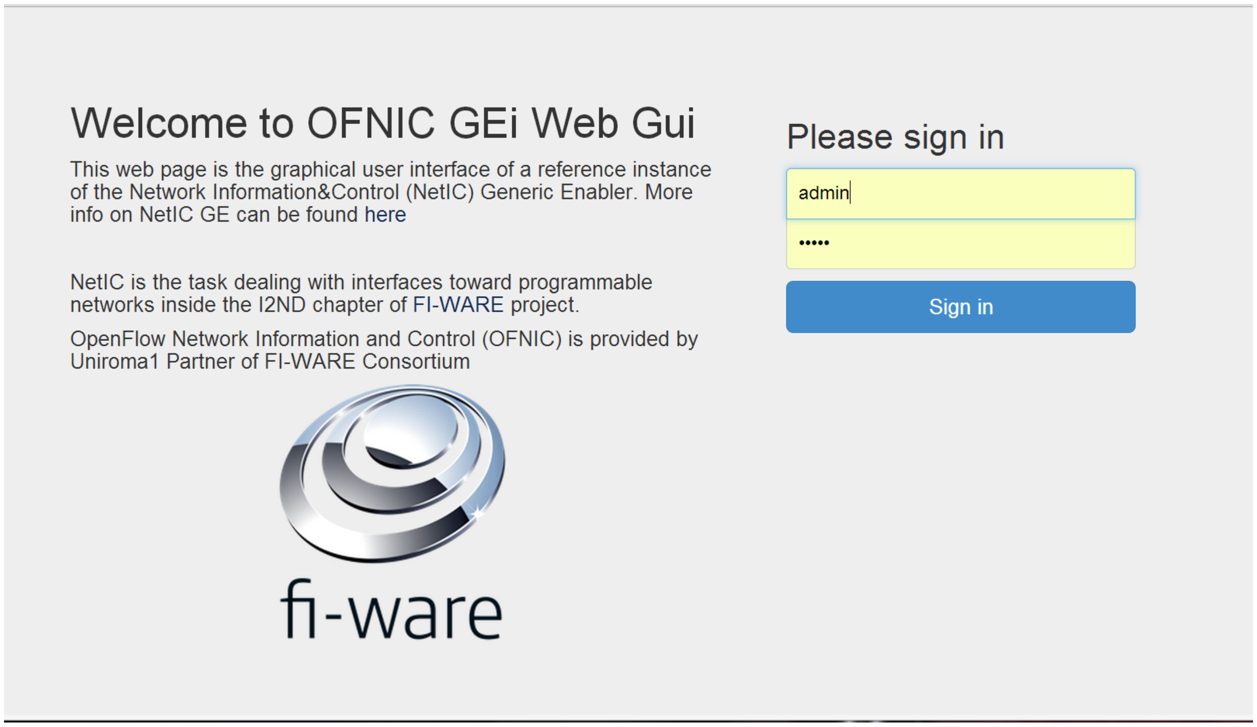
The NetIC_RESTful_API reference provides a page which describes how to use this API in details. It includes a set of commands, which can be used to manipulate and instantiate network resources physical or virtual. Since NetIC RESTful API is based on the HTTP protocol there are a rich variety of tools, which can be used to access the OFNIC GEi interface. The sections of the Programmer Guide will go into more detail on how to use and develop against NetIC RESTful API.

3.1 OFNIC GUI

The OFNIC GEi is released with a GUI feature. The NetIC API users might utilize the GUI to navigate the exposed RESTful webservices of the GEi. The GUI has been developed in Javascript language, and uses the FIWARE site template. The web GUI comes embedded in the OFNIC source package release, see the OFNIC Installation-and-Administration guide to download the package.

The GUI relies on the OFNIC GEi, it should up and running correctly. To access the GUI from a web browser navigate to: http://localhost/gui

The GUI page that appears is shown in the figure below.



Welcome to OFNIC GEi Web Gui

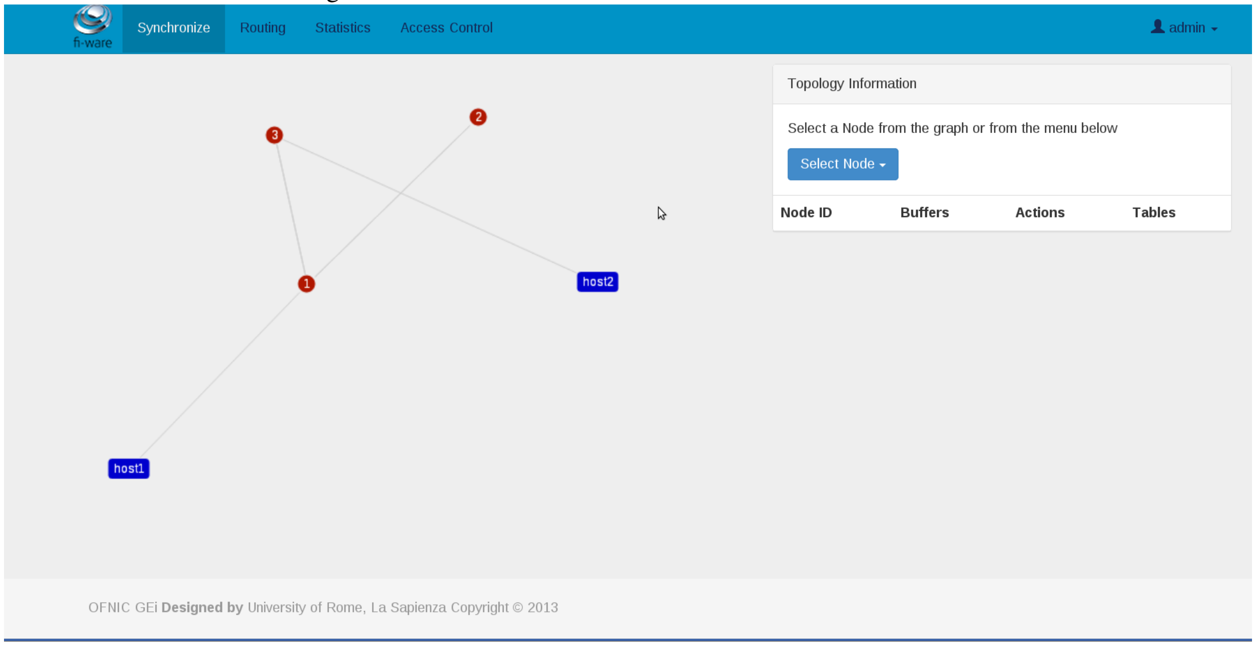
This web page is the graphical user interface of a reference instance of the Network Information&Control (NetIC) Generic Enabler. More info on NetIC GE can be found here

NetIC is the task dealing with interfaces toward programmable networks inside the I2ND chapter of FI-WARE project.

OpenFlow Network Information and Control (OFNIC) is provided by Uniroma1 Partner of FI-WARE Consortium

The default username is ‘admin’ and can be authenticated with the password ‘admin’. After logging in, one can note that the GUI is composed of four navigation tabs: Synchronization,Statistics, Routing, and Access Control. The Synchronization tab shows the network topology with an interactive diagram. Network nodes are clickable and additional information is shown if a node is selected. For example the number of Actions or Ports. Moreover the Displayed ports are clickable and show to which other network entity is the selected node connected to.

The Statistics navigation tab, concerns about the presentation of network statistics collected by the OFNIC GEi. The dynamic network topology is present so the user can select nodes he might want to monitor from the graph. The Statistics tab is shown in the figure below.



Topology Information

Select a Node from the graph or from the menu below

Node ID	Buffers	Actions	Tables

OFNIC GEi **Designed by** University of Rome, La Sapienza Copyright © 2013

The presence of the Openflow network is not mandatory, if no network is attached to the OFNIC GEi, the following warning will appear on the GUI: “no network nodes detected”

CHAPTER 4

Programmer Guide

This section describes the way a programmer can interact ith the OFNIC GEi. It is assumed that the OFNIC GEi is up and running. See the OFNIC Installation-and-Administration guide for this.

4.1 Navigation with browser

The root path of the Web Server is:

http://localhost:8000/api-docs

This page, by using the Swagger plugin, shows all commands that can be sent to the REST API of OFNIC.

4.2 Examples

In this example the OFNIC GEi is connected to an Openflow network composed of three Openflow nodes. The nodes run Openvswitch for implementing the Openflow protocol. The OFNIC GEi is located at the same machine from which the browser is used. Using the browser, type in the location bar the following URL string and then press Enter.

https://127.0.0.1/netic.v1/OFNIC/synchronize/network

The result displayed, should be the list of nodes present in the network:

{ “paths”:[], “nodes”:[“00:00:00:00:00:00:00:02”, “00:00:00:00:00:00:00:01”, “00:00:00:00:00:00:00:03”] “hosts”:[]}

As one can note from the examples, all response bodies are in JSON format. For the full RESTful API specification refers to https://fiware-uniroma1.github.io/FIWARE-OFNIC/

CHAPTER 5

Welcome to FIWARE OFNIC GE's documentation!

Refer to this page for more general information about OFNIC.

CHAPTER 6

Installation & Administration

CHAPTER 7

Goal of the document

The OFNIC is an implementation of the NetIC Generic Enabler Open Specifications. This GEi is in charge of providing a common programmable interface to an Openflow Network, by collecting information and statistics regarding the managed Openflow network node's. This interface is based on the NetIC Open API RESTful specifications. The OFNIC GEi is an extension of the open-source OpenDaylight Controller. It relies on the Openflow protocol to retrieve network information about the managed network. The OFNIC GEi provides also a Graphical User Interface (GUI) based on web technologies. Basically this is a web page with javascript code that communicates with the RESTful interface of the GEi.

Goal of this document is to provide a useful guide for the installation of OFNIC GEi, together with its GUI. The document starts describing basic software and hardware required to support the OFNIC GEi on top of a device. It then follows with specific technical information that might help users and administrators: running processes, diagnosis tests, network flows etc.

CHAPTER 8

OFNIC GEi

8.1 Software and Hardware environment

The OFNIC OpenFlow controller runs in a Java Virtual Machine. Being a Java application, it can (potentially) runs on any machine that supports Java. However, all the software have been tested on recent Linux distributions, so we recommend the following:

- A recent Linux distribution (for example Ubuntu 14.04 LTS or Debian 7.x)
- Java Virtual Machine 1.7

8.2 Prerequisites

As the previous paragraph, being a Java application, the only requistite is Java Virtual Machine.

On an Ubuntu machine, you can satisfy these requirements with:

> sudo apt-get install openjdk-7-jdk git

8.3 Getting OFNIC

There are different options for obtaining the OFNIC Controller. The first option is to download the pre-built current build. The second option is to getting the source code of the component and build the code on your machine. The last is using the Docker Image

The pre-built package can be downloaded here:

> https://github.com/FIWARE-UNIROMA1/FIWARE-OFNIC/releases

Another way of getting the source code is to pull the code by cloning the controller repository on GitHub with the following commands:

> git clone https://github.com/FIWARE-UNIROMA1/FIWARE-OFNIC.git

8.4 Build the code

8.5 Prerequisites

The following are required for building the codebase:

- Maven 3.x.y

If you use a Debian or Ubuntu machine, you can install Maven with the following command:

sudo apt-get install maven

and check the installed version with:

mvn -v

Using a system shell locate in the main OFNIC source code directory, where is located the pom.xml maven's configuration file. Run the following commands:

mvn clean install

8.6 Running

The command reported below starts the OFNIC controller: *./start.sh*

Note: If you are building OFNIC from the source, you have to download the pre-build package, then replace the .jar file with the one produced by maven during the build

Docker

Docker allows you to deploy an OFNIC container in a few minutes. This method requires that you have installed docker, see the [documentation](https://docs.docker.com/installation/) on how to do this.

Follow these steps:

1. Download [OFNIC' source code](https://github.com/FIWARE-UNIROMA1/FIWARE-OFNIC) from GitHub (*git clone https://github.com/FIWARE-UNIROMA1/FIWARE-OFNIC.git*)
2. *cd FIWARE-OFNIC/docker*
3. Using the command-line and within the directory you created type: *docker build -t fiware-ofnic .*.

After a few seconds you should have your OFNIC image created. Just run the command *docker images* and you see the list.

To execute the OFNIC image, execute the command *docker run fiware-ofnic*.

You can obtain the IP address of the machine just executing *docker-machine ip*.

If you want to stop the scenario you have to execute *docker ps*, take the Container ID and execute *docker stop cID* or *docker kill cID*.

CHAPTER 10

Sanity check procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

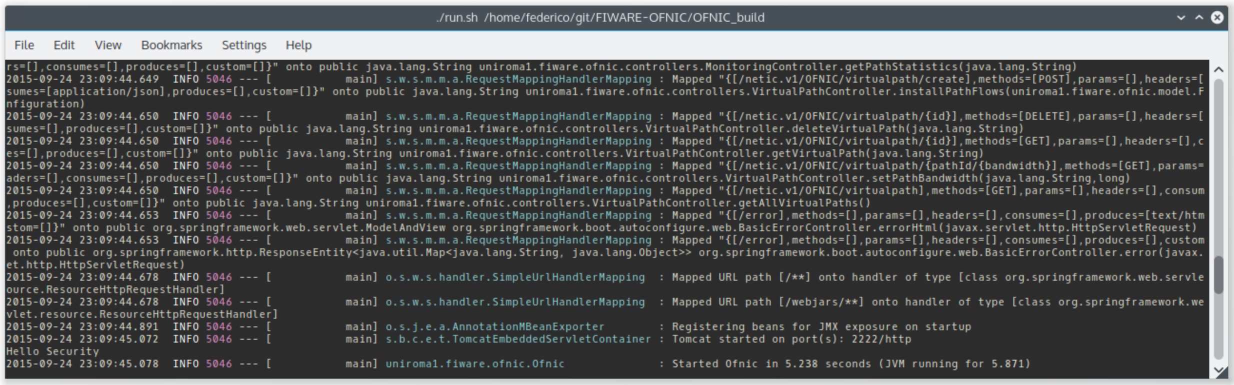
10.1 End to End testing

This is basically quick testing to check that everything is up and running.

1. Launch ofnic with the command:

   ./start.sh

2. To verify that the OFNIC GEi is loaded correctly it should display bootstrap complete in the terminal on which it was launched.



```
rs=[],consumes=[],produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.MonitoringController.getPathStatistics(java.lang.String)
2015-09-24 23:09:44.649  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/netic.v1/OFNIC/virtualpath/create],methods=[POST],params=[],headers=[
sumes=[application/json],produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.VirtualPathController.installPathFlows(uniroma1.fiware.ofnic.model.F
nfiguration)
2015-09-24 23:09:44.650  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/netic.v1/OFNIC/virtualpath/{id}],methods=[DELETE],params=[],headers=[
sumes=[],produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.VirtualPathController.deleteVirtualPath(java.lang.String)
2015-09-24 23:09:44.650  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/netic.v1/OFNIC/virtualpath/{id}],methods=[GET],params=[],headers=[],c
es=[],produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.VirtualPathController.getVirtualPath(java.lang.String)
2015-09-24 23:09:44.650  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/netic.v1/OFNIC/virtualpath/{pathId/{bandwidth}],methods=[GET],params=
aders=[],consumes=[],produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.VirtualPathController.setPathBandwidth(java.lang.String,long)
2015-09-24 23:09:44.650  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/netic.v1/OFNIC/virtualpath],methods=[GET],params=[],headers=[],consum
,produces=[],custom=[]}" onto public java.lang.String uniroma1.fiware.ofnic.controllers.VirtualPathController.getAllVirtualPaths()
2015-09-24 23:09:44.653  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],methods=[],params=[],headers=[],consumes=[],produces=[text/htm
stom=[]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest)
2015-09-24 23:09:44.653  INFO 5046 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],methods=[],params=[],headers=[],consumes=[],produces=[],custom
 onto public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.
et.http.HttpServletRequest)
2015-09-24 23:09:44.678  INFO 5046 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servle
ource.ResourceHttpRequestHandler]
2015-09-24 23:09:44.678  INFO 5046 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.we
vlet.resource.ResourceHttpRequestHandler]
2015-09-24 23:09:44.891  INFO 5046 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2015-09-24 23:09:45.072  INFO 5046 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 2222/http
Hello Security
2015-09-24 23:09:45.078  INFO 5046 --- [           main] uniroma1.fiware.ofnic.Ofnic              : Started Ofnic in 5.238 seconds (JVM running for 5.871)
```

3. Network nodes side, on ovs switch give the following command to verify that the device is correctly connected with the controller :

   ovs-vsctl show

it should display the following log message:

- Bridge “br0”
- Controller “tcp:127.0.0.1” is_connected: true
- Port “br0”
- Interface “br0” type: internal

4. After this two checks have been done, the GEi should be up and ready. To test that is actually running a simple check can be done from the browser. With an internet browser application, go to the following address:

   http://localhost:2222/api-docs

should display the list of all API of OFNIC GEi. The credentials to access the page are the one in the ofnic.conf file.

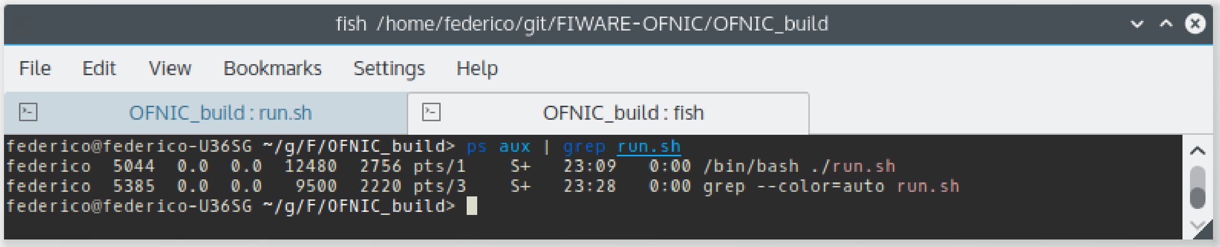
10.2 List of Running Processes

In order to list running processes on a Linux distribution one can use ps aux command. In order to get more filtered results one can use this more articulated command:

ps aux | grep "Name_of_process"

In the machine that hosts the OFNIC GEi the run.sh process is required to be active. So by typing in the terminal:

ps aux | grep run.sh



```
federico@federico-U36SG ~/g/F/OFNIC_build> ps aux | grep run.sh
federico  5044  0.0  0.0  12480  2756 pts/1    S+   23:09   0:00 /bin/bash ./run.sh
federico  5385  0.0  0.0   9500  2220 pts/3    S+   23:28   0:00 grep --color=auto run.sh
federico@federico-U36SG ~/g/F/OFNIC_build>
```

10.3 Network interfaces Up & Open

The OFNIC GEi listens to the ports 6633, 2222 and 8080, with the following command you can verify it:

netstat -lnptu | grep tcp

in the terminal you will see a list of the process listen on port 6633, 8080 and 2222.

CHAPTER 11

Diagnosis Procedures

OFNIC logs to the stdout on the terminal on which it was launched.

11.1 Resource availability

The required RAM depends on many factors such as network topology, number of flows in the network, frequency of the statistics updates, frequency of web service requests, etc.

- Generally RAM size varies from 100 MB to 250 MB.
- Usually the disk size required during run time is negligible.

11.2 Remote Service Access

User can verify the correct execution of the OFNIC, by directing the browser (all types are supported) to the following page:

> http://localhost:8000/api-docs

which should display a list of commands that can be sent to the interface. Note that if the browser app is not on the same machine of the GE, the remote IP address of the GEi can be used.

11.3 Resource consumption

The resource consumption is highly dependent on the number of network events processed. The minimum amount of RAM is nearly 200 MB so eventually in any lower amount of RAM means that the application did not load properly. Under normal working conditions RAM size reaches the order of 250 MB so values of greater orders mean that there is some malfunctioning. CPU percentage ranges and is highly dependent on the processor speed. However it should be noted that at idle state the OFNIC processor consumption can be even lower that 0.1%.

11.4 I/O flows

Port 2222 for RESTful API and port 6633 for the communication with the network nodes via OpenFlow plugin..