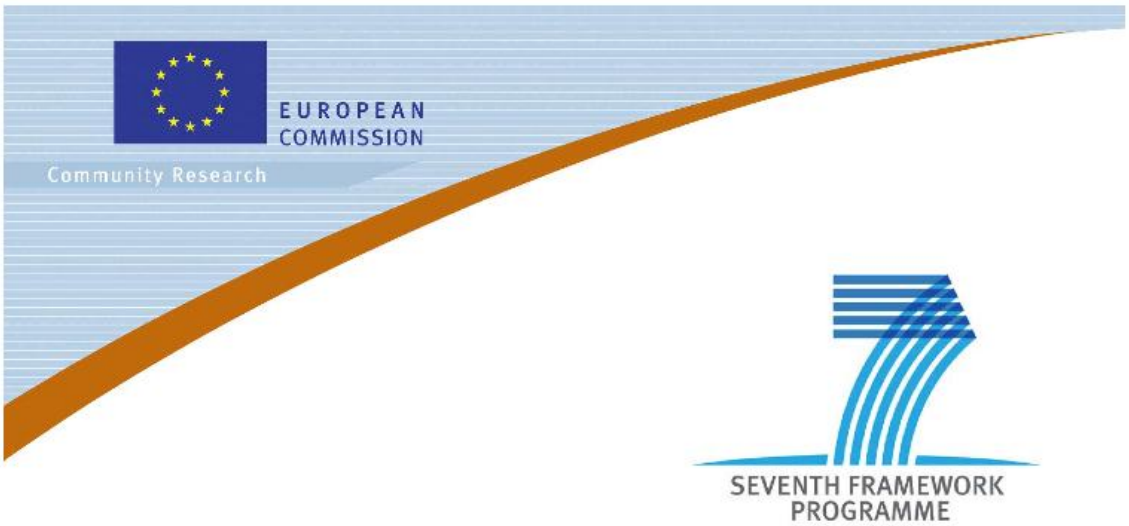


Private Public Partnership Project (PPP)
Large-scale Integrated Project (IP)

D.4.3.1b: FI-WARE Installation and Administration Guide

Project acronym: FI-WARE
Project full title: Future Internet Core Platform
Contract No.: 285248
Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform
Project Document Number: ICT-2011-FI-285248-WP4-D.4.3.1b
Project Document Date: 2012-11-02
Deliverable Type and Security: Public
Author: FI-WARE Consortium
Contributors: FI-WARE Consortium

1.1 Executive Summary

Welcome the Installation and Administration Guide for the Data Center Resource Manager Generic Enabler. This generic enabler is built on Open Source projects, OpenStack and the OCCI API for OpenStack, and so where possible this guide points to the appropriate online content that has been created for the projects.

The system requirements for the installation of a Generic Enabler are outlined with respect to necessary hardware, operating system and software. Each GE has a section dedicated to the software installation and configuration process as well as a section, which describes sanity check procedures for the system administrator to verify that the GE installation was successful.

This document consolidates new contents and also contents in previous issues of Release 1. The reason for re-delivering parts that were already issued is twofold:

- FI-WARE has made an effort to create a unified and improved format. The parts generated in the past are also provided in the new enhanced format for the sake of uniformity and readability.
- A single reference document per chapter is clearer and easier to handle that two incremental issues.

1.2 About This Document

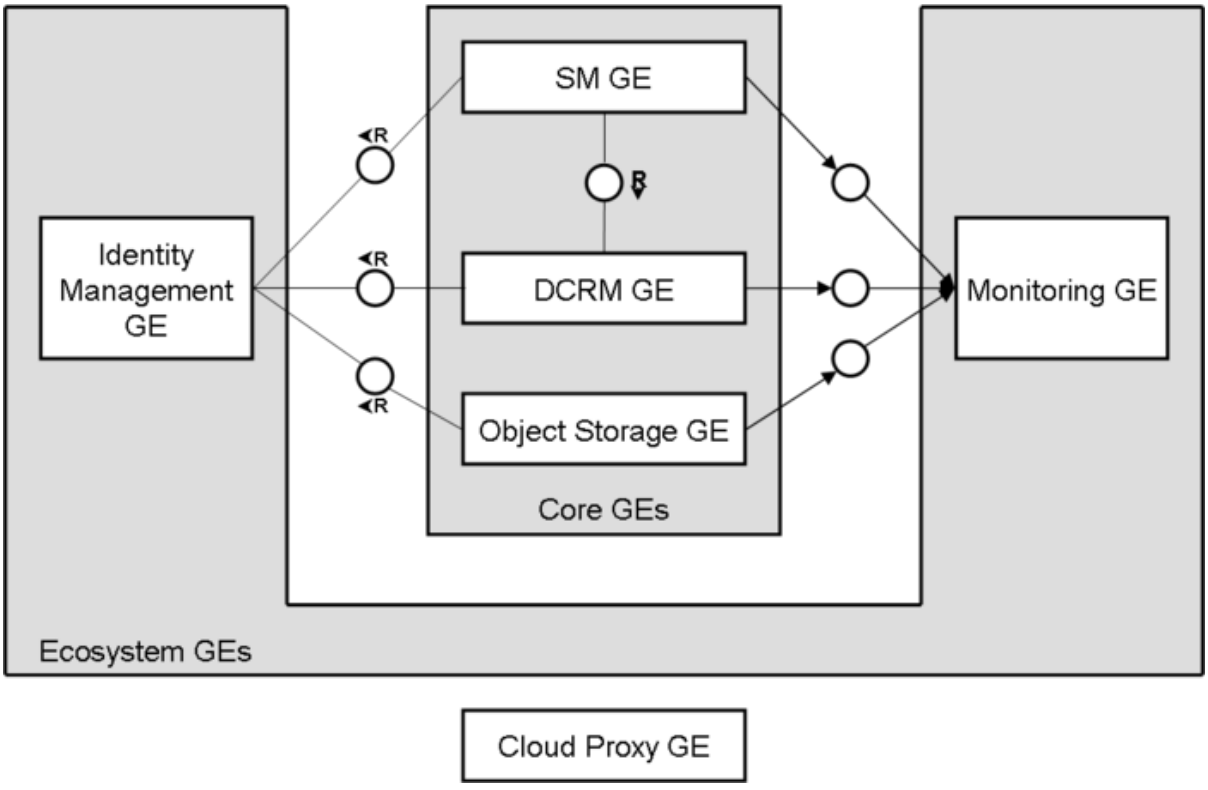
The "FI-WARE Installation and Administration Guide" comes along with the software implementation of components, each release of the document referring to the corresponding software release (as per D.x.2), to facilitate the users/adopters in the installation (if any) and administration of components (including configuration, if any).

1.3 Intended Audience

The document targets system administrators as well as system operation teams of FI-WARE Generic Enablers from the FI-WARE project.

1.4 Chapter Context

The Cloud Chapter offers Generic Enablers that comprise the foundation for designing a modern cloud hosting infrastructure that can be used to develop, deploy and manage Future Internet applications and services. The solution focuses on fundamental cloud capabilities enabling provisioning and life cycle management of virtual machines and associated resources (compute, storage, network, images, etc) hosting FI applications and services, as well as object storage capabilities which can be used directly by FI applications and services via a REST API. In future releases, additional capabilities will be added, notably the support for complex services comprising multiple virtual machines, including monitoring, policy-based elasticity, as well as many others, as outlines in Roadmap of Cloud Hosting.
The following diagram shows the main components (Generic Enablers) that comprise the first release of FI-WARE architecture.



The GEs in the above diagram are grouped into **Core GEs**, providing the core hosting capabilities at different abstraction levels (resources, services, objects, etc) and **Ecosystem**

GEs, addressing various specific needs across the Core GEs, and establishing the ecosystem that enables the end-to-end capabilities provided by a cloud offering.
The **Core GEs** include:

- **Data Center Resource Maangement (DCRM) GE**, offering provisioning and life cycle management of virtualized resources (compute, storage, network) associated with virtual machines.
- **Object Storage GE**, offering provisioning and life cycle management of object-based storage containers and elements
- **Service Management (SM) GE**, offering provisioning and life cycle management of composite services comprising several resources provided by on of the above GEs. In the first release of FI-WARE, Service Management GE will consume resources provided by Data Center Resource Management GE, via the corresponding APIs.

The **Ecosystem GEs** include:

- **Monitoring GE**, collecting metrics associated with each of the Core GEs, and offering them to GEs which are interested to consume such metrics. For example, Service Management GE consumes metrics associated with KPIs of the various service components in order to drive auto-scaling decisions. In the future, more advanced metrics-related capabilities will be provided, such as processing (before it is delivered to the consumer), archival and analysis of metrics.
- **Identity Management GE**, providing a unified management of users, roles and tokens, that can be used by other GEs for authentication and authorization purposes. This GE will be provided by the Security Chapter.

1.5 Structure of this Document

The document is generated out of a set of documents provided in the public FI-WARE wiki.
For the current version of the documents, please visit the public wiki at http://wiki.fi-ware.eu/
The following resources were used to generate this document:

D.4.3.1b FI-WARE Installation and Administration Guide front page
IaaS Data Center Resource Management - Installation and Administration Guide
IaaS Service Management - Installation and Administration Guide
Object Storage - Installation and Administration Guide

1.6 Typographical Conventions

Starting with October 2012 the FI-WARE project improved the quality and streamlined the submission process for deliverables, generated out of the public and private FI-WARE wiki. The project is currently working on the migration of as many deliverables as possible towards the new system.
This document is rendered with semi-automatic scripts out of a MediaWiki system operated by the FI-WARE consortium.

1.6.1 Links within this document

The links within this document point towards the wiki where the content was rendered from. You can browse these links in order to find the "current" status of the particular content.

Due to technical reasons not all pages that are part of this document can be linked document-local within the final document. For example, if an open specification references and "links" an API specification within the page text, you will find this link firstly pointing to the wiki, although the same content is usually integrated within the same submission as well.

1.6.2 Figures

Figures are mainly inserted within the wiki as the following one:

```
[[Image:....|size|alignment|Caption]]
```

Only if the wiki-page uses this format, the related caption is applied on the printed document. As currently this format is not used consistently within the wiki, please understand that the rendered pages have different caption layouts and different caption formats in general. Due to technical reasons the caption can't be numbered automatically.

1.6.3 Sample software code

Sample API-calls may be inserted like the following one.

```
http://[SERVER_URL]?filter=name:Simth*&index=20&limit=10
```

1.7 Acknowledgements

The current document has been elaborated using a number of collaborative tools, with the participation of Working Package Leaders and Architects as well as those partners in their teams they have decided to involve; IBM, Intel, Technicolor, Telefonica.

1.8 Keyword list

FI-WARE, PPP, Architecture Board, Steering Board, Roadmap, Reference Architecture, Generic Enabler, Open Specifications, I2ND, Cloud, IoT, Data/Context Management, Applications/Services Ecosystem, Delivery Framework , Security, Developers Community and Tools , ICT, es.Internet, Latin American Platforms, Cloud Edge, Cloud Proxy.

1.9 Changes History

Release	Major changes description	Date	Editor
v0	First draft of deliverable submission generated	2012-11-02	Automated
v1	First Version	2012-11-02	IBM
v2	Final Version	2012-11-08	IBM

1.10 Table of Contents

2 IaaS Data Center Resource Management - Installation and Administration Guide

You can find the content of this chapter as well in the wiki of fi-ware.

2.1 Introduction

Welcome the Installation and Administration Guide for the Data Centre Resource Manager Generic Enabler. This generic enabler is built on an Open Source project, the OCCI API for OpenStack, and so where possible this guide points to the appropriate online content that has been created for this project. The online documents are being continuously updated and improved, and so will be the most appropriate place to get the most up to date information on installation and administration.

2.2 System Installation

OpenStack system has several key projects that are separate installations but can work together: OpenStack Compute (Nova), OpenStack Object Storage (Swift), OpenStack Identity Service (KeyStone), and OpenStack Image Service (Glance) and a dashboard Service (Horizon), for our installation we would configure Nova, Keystone and Horizon projects. OpenStack installation consists of various softwares (such as Queue software - RabbitMQ, DB software - SQLAlchemy, etc) and the above projects code.
OpenStack can be deployed manually using instructions on http://docs.openstack.org/trunk/openstack-compute/install/content/ or using DevStack. DevStack (http://devstack.org/) is essentially a Shell script which allow the easy deployment of an OpenStack installation from the latest source code. DevStack can be run for development or in other setups too. To setup a multi node OpenStack installation follow the instructions at http://devstack.org/guides/multinode-lab.html (make sure you complete the SSH setup on all relevant hosts as detailed in the provided link, as well as configure the localrc files in each host with the services to be installed/referenced) The latest OpenStack VERSION is Essex (when writing this guide)

Enabelment of OCCI OpenStack API when deploying using DevStack

To get OpenStack and the OCCI OpenStack API up and running all follow the setup routine: A pre-requisite to this is the python dev tools: sudo apt-get install python-pip python-dev build-essential

1. Install pyssf

```
pip install pyssf
```

2. Install devstack

```
git clone git://github.com/openstack-dev/devstack.git
```

3. Configure devstack. Here we have to change the NOVA_REPO location (done with the sed command).

```
cd devstack
```

```
sed -i
's/NOVA_REPO=https\:\/\/github.com\/openstack\/nova.git/NOVA_REPO=ht
tps\:\/\/github.com\/dizz\/nova.git/' stackrc

sed -i 's/NOVA_BRANCH=master/NOVA_BRANCH=bp\/VERSION-open-cloud-
compute-interface/' stackrc
```

4. Set the contents of localrc (you may have to create the file) to:

```
EXTRA_OPTS=( --allow_resize_to_same_host=True --
libvirt_inject_password=True --
enabled_apis=ec2,occiapi,osapi_compute,osapi_volume,metadata )
ENABLED_SERVICES=g-api,g-reg,key,n-api,n-crt,n-obj,n-cpu,n-net,n-
sch,n-novnc,n-xvnc,n-cauth,horizon,mysql,rabbit,n-vol,openstackx
OFFLINE=False
```

5. Run devstack

```
./stack.sh
```

For more information please refer to the OpenStack wiki where a detailed description of OCCI can be found.

Fabric Enhancement Modules
To install and enable Group services:
1. Install Zookeeper (it is recommended to maintain three installations on various nodes)
http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html
(The next two items should be applied to OpenStack code using *git pull* command)
2. Install Zookeeper python drivers by applying a patch to the code
https://github.com/maoy/python-evzookeeper
3. Install SvcGroup services by patching the code from review item
https://review.openstack.org/#/c/6613/
4. Adjust nova.conf with Zookeepers addresses, and configure membership service
5. Start Zookeeper and SvcGroup services, restart Nova

2.3 System Administration

There are many guides and documentations available for administrating OpenStack. Those can be found on the OpenStack website.
Since the OCCI API sits on top of nova the most important documentation can be found here. This link points to the nova compute Administration guide.

2.3.1 Install all required software packages

Openstack installation [1] Zoo keeper, our source control, our nova.conf, NFS [guide],

2.4 System Requirements

2.4.1 Hardware Requirements

The following table contains the minimum resource requirements for running the infrastructure:

Resource	Requirement
# of nodes	5
CPU	8 cores with at least 2.4 GHZ, VT-x enabled
Physical RAM	128GB
Disk Space	1TB The actual disk space depends on the amount of data being stored within the Repositories NoSQL database System.</ref>

2.4.2 Operating System Support

We have been tested against the following Operating Systems:

- Ubuntu 12.04 x86_64

2.4.3 Software Requirements

The following software is needed:

- MySql server - mandatory
- tgt (linux SCSI target user-space tool) - mandatory
- Open-iScsi (iSCSI implementation) - mandatory
- RabbitMQ server - mandatory
- KVM, libvirt (Virtualization software) - mandatory
- Apache server - mandatory
- LVM (Logical volume manager) - mandatory
- ntp (Network time protocol) - mandatory

2.5 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.

2.5.1 End to End testing

Nova comes with a selection of fairly basic smoke tests which can run against the installation. It can be useful to use these to sanity check the configuration: ./run_tests.sh from /opt/stack/nova

Next, log in with the YOUR_USER_NAME and YOUR_PASSWORD to the Dashboard endpoint (HORIZON http web interface), usually located at http://cloud.lab.fi-ware.eu/ and

examine the various sections in the admin tab to make sure all instances, images and services are well defined and accessible.

Testing the OCCI interface

Please note that the following information is required before carrying out this procedure:

- the IP address of the OCCI node
- the IP address of the Openstack Keystone node managing security for the DCRM GE Deployment
- a valid OpenStack (keystone) username and password

1. Verify that **http://occiservice.lab.fi-ware.eu:8787** can be reached. By default, web access will receive a 401 Unauthorized response.
2. Acquire a valid token from the OpenStack Keystone server. Using curl, the following command can be used:

```
   curl -d '{"auth":{"passwordCredentials":{"username":"<Insert
Openstack username here>", "password":"<Insert Openstack password
here>"}}}' -H "Content-type: application/json" http://<Insert IP
address of Keystone node here>:5000/v2.0/tokens
```

The resulting Keystone Token is returned near the beginning of the response.

3. Execute an OCCI Command that lists the resources that can be provisioned through the OCCI instance:

```
    curl -v -H 'X-Auth-Token: <Insert Keystone Token here>' -X GET
http://occiservice.lab.fi-ware.eu:8787/-/
```

A response should be returned that details the type of resources that can be provisioned by the OCCI instance.

Testing Openstack infrastructure

```
    nova-manage service list
```

Make sure at least three nova-compute are running (:)), nova-volume, nova-scheduler, nova-network, nova-consoleauth, nova-cert

keystone catalog --service ec2 http://dashboard glance index

2.5.2 List of Running Processes

In case that all services run on one machine the following processes should be available.

```
python /opt/stack/keystone/bin/keystone-all
python /opt/stack/nova/bin/nova-api
sg libvirtd /opt/stack/nova/bin/nova-compute
python /opt/stack/nova/bin/nova-compute
python /opt/stack/nova/bin/nova-cert
python /opt/stack/nova/bin/nova-volume
python /opt/stack/nova/bin/nova-network
python /opt/stack/nova/bin/nova-scheduler
python /opt/stack/nova/bin/nova-objectstore
```

Next to these a MySQL server and a RabbitMQ server should be up and running anywhere in the distributed system.
Please note however that depending on your setup maybe only the nova-api process is running, as other services are deployed in a distrbuted system and are accessed through the network.

2.5.3 Network interfaces Up & Open

OpenStack Horizon uses port 80 (WSGI), Keystone listens on 5000 Nova URL : 8774, glance : 9292, ec2 url: 8773, vncproxy:6080, libvirt -d -l
The OCCI interface itself will start on port 8787 by default. It is mentioned however that port 35357 should be open as well. The later is the port for the authentication service Keystone. Which is also used by the OCCI API.

2.5.4 Databases

OpenStack components use a centralized database, installed and configured as part of standard OpenStack installation. In order to verify that the database is operational, the following command can be used:

```
sudo mysql -uroot -p$MYSQL_PASS nova -e 'SELECT * FROM services;'
```

2.6 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.
The following sections have to be filled in with the information or an “N/A” (“Not Applicable”) where needed. Do not delete section titles in any case.

2.6.1 Resource availability

The bare minimum requirements would be the following:

```
Memory:     1GB
CPU:        single-core
Storage:    8GB
Network:    single interface connected with the internet
```

We would however encourage Administrators to use up-to-date Hardware. This includes modern Processors which have extra support for virtualization technologies.

Depending on the environment load and the amount of resources available, it should be consider to use multi-node installation, so each node would run a subset of the services and each service would enjoy more resources.

First thing to look in case of an error, would be to check the nova.conf file located in /etc/nova.
Logging level can be set to debug or info (using verbose, debug and default_log_levels config parameters), and log file location could be set as well (common setup by setting logdir=/var/log/nova). When logging is enabled, check the log at the specified location for the error. For more information see http://docs.openstack.org/trunk/openstack-compute/admin/content/configuring-logging.html

RabbitMQ tools are great tools to monitor queues for undelivered Openstack messages

Restart of the services might fix the problem sometimes

2.6.2 Remote Service Access

Openstack Horizone can be accessed through : **http://<hostname>/**.
The OCCI interface can be accessed through **http://<hostname>:8787/**.

2.6.3 Resource consumption

Resource consumption can get quite high as more VMs are started. There is no general assumption on what the default consumption is.
In nova.conf overcommit parameters can be configured to limit (as for the moment) Compute nodes' amount of Virtual Machines deployed and scheduled and by that limit in a way the resources consumption.

2.6.4 I/O flows

N/A

3 IaaS Service Management - Installation and Administration Guide

You can find the content of this chapter as well in the wiki of fi-ware.

3.1 Claudia Installation

This guide tries to define the procedure to install the IaaS SM GE - Claudia in a machine, including its requirements and possible troubleshooting that we could find during the installation. We have to talk about two nodes, one including the core functionality and deployed in a JBoss server application (JBoss node) and a second one with the OpenStack API functionality and deployed in a Apache Tomcat server (Tomcat node).

3.1.1 Requirements

In order to execute the IaaS SM, it is needed to have previously installed the following software of framework in the two nodes:

- JBoss Node:
  - JavaTM Platform, Standard Edition Development Kit (JDKTM) 6 [1].
  - MySQL Community Server GA 5.1.63 [2].
  - JBoss 5.1.0.GA [3]
- Tomcat Node:
  - JavaTM Platform, Standard Edition Development Kit (JDKTM) 6.
  - MySQL Community Server GA 5.1.63.
  - Apache Tomcat 6 [4]

3.1.2 Database configuration

We start with the JBoss node. After you have installed the MySQL it is necessary execute the following script in order to create the appropriate database to be used by IaaS SM:

```
DROP DATABASE IF EXISTS ClaudiaDB;

CREATE DATABASE ClaudiaDB;

CREATE USER 'claudiauser'@'%' IDENTIFIED BY 'claudiapass';
GRANT ALL PRIVILEGES ON ClaudiaDB.* TO 'claudiauser'@'%';
```

If during the execution of this process we obtain an error due to the user existed previously, we will see the following message:

```
''ERROR 1396 (HY000) at line 5: Operation CREATE USER failed for
'claudiauser'@'%'''
```

We should execute the following commands in that case after the previous SQL sentences:

```
GRANT ALL PRIVILEGES ON ClaudiaDB.* TO 'claudiauser'@'%';
```

It is checked some problems in Linux system in order to create also the user due do the **%**, in that cases, we can change the create user and privileges SQL sentences by the following:

```
DROP DATABASE IF EXISTS ClaudiaDB;
CREATE DATABASE ClaudiaDB;
CREATE USER 'claudiauser'@'localhost' IDENTIFIED BY 'claudiapass';
GRANT ALL PRIVILEGES ON ClaudiaDB.* TO 'claudiauser'@'localhost';
```

Regarding the OpenStack node, in which we have installed a Tomcat and MySQL, we need only to have a Database (openstack_tcloud) and it is created automatically when the Tomcat is started with the OpenStack module on it. At the same time, when the Tomcat is stopped this database is dropped, then the only operation that we have to do is the creation of the database.

```
CREATE DATABASE openstack_tcloud;
```

3.1.3 JBoss Application Server configuration

The recommended path to install the JBoss is */opt/jboss*

```
$ mkdir -p /opt/jboss
$ cp /path/to/jboss-5.1.0.GA.zip /opt/jboss
$ cd /opt/jboss
$ tar xfvz jboss-5.1.0.GA.zip
```

3.1.3.1 ***Step 1: Run JBoss***

JBoss is started with the following command:

```
$ nohup <JBOSS_HOME>/bin/run.sh -b 0.0.0.0 > jboss-default.log &
```

The **-b** parameter (optional) is used in order to catch from all interfaces. If we want to stop the JBoss application server, we execute the following command:

```
```

```
$ <JBOSS_HOME>/bin/shutdown.sh -S [-u admin –p admin]
```

<JBOSS_HOME> in these cases can be changed by *<JBOSS_INSTALLATION_DIR>/jboss-5.1.0.GA* in JBoss 5.1 GA or < *JBOSS_INSTALLATION_DIR >/jboss-eap-5.1/jboss-as* in JBoss EAP 5.1.
We will use the *default server configuration*, which contains everything you need to run a stand-alone J2EE server including web services. Anyway, JBoss is not limited to the existing default server configurations. You can create a custom server configuration that suits your needs best. [5].
After JBoss is started completely (it could need several minutes), we can go to the home page of JBoss in order to check that it is up and running.

```
$ wget http://localhost:8080
```

And it returns the index.html file. We have also the possibility to go to the **Admin Console** (http://localhost:8080/admin-console) in order to see the configuration of the JBoss.

3.1.3.2 *Step 2: Logging Service*

By default, JBoss produces output to both the console and a log file (log/server.log). There are 6 basic log levels used: TRACE, DEBUG, INFO, WARN, ERROR and FATAL. The logging threshold on the console is INFO, which means that you will see informational messages, warning messages and error messages on the console but not general debug and trace messages. This effectively means that any TRACE or DEBUG logger from any logger categories will not be logged in any files or the console appender. This setting is controlled through the *jboss.server.log.threshold* property. By default this is INFO. If you were to change this to DEBUG, it would produce much more detailed logging output. In order to change this you can execute JBoss with the following parameter:

```
$ <JBOSS_HOME>/bin/run.sh -b 0.0.0.0  -
Djboss.server.log.threshold=DEBUG
```

3.1.3.3 *Step 3: Users and Roles*

By default the user and password of JBoss is admin/admin. If there is a problem with these data we have to check that they exist and they are not commented in the file *<JBOSS_HOME>/server/default/conf/props/jmx-console-users.properties*

```
# A sample users.properties file for use with the
UsersRolesLoginModule

admin=admin
```

and in *<JBOSS_HOME>/server/default/conf/props/jmx-console-roles.properties*

```
# A sample roles.properties file for use with the
UsersRolesLoginModule

admin=JBossAdmin,HttpInvoker
```

3.1.3.4 *Step 4: JNDI*

We need to create or modify the property *java.naming.provider.url* within the property file *server/default/conf/jndi.properties*

```
java.naming.provider.url=localhost:1199
```

After this change, it is convenience to restart JBoss.

3.1.3.5 *Step 5: Setting up a MySQL datasource*

Download the driver from [6], unrar/unzip it and extract the jar file and then copy it into <*JBOSS_HOME*>*server**default**lib*\

Copy the example datasource file in <*JBOSS_HOME*>*/docs/examples/jca/mysql-ds.xml* to <*JBOSS_HOME*>*/server/default/deploy*. You can select the name that you want but the file must finish with **-ds.xml**. The content of this file will be the following:

```
<?xml version="1.0" encoding="UTF-8"?>


<datasources>
    <xa-datasource>
        <jndi-name>claudiaDataSource</jndi-name>
        <track-connection-by-tx>true</track-connection-by-tx>
        <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-
datasource-class>
        <xa-datasource-property name="ServerName">localhost</xa-
datasource-property>
        <xa-datasource-property name="DatabaseName">ClaudiaDB</xa-
datasource-property>
        <xa-datasource-property name="User">claudiauser</xa-
datasource-property>
        <xa-datasource-property name="Password">claudiapass</xa-
datasource-property>
       <transaction-
isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
    </xa-datasource>
</datasources>
```

IMPORTANT, the name of the database together with its username and password must be the same defined in Database configuration section.

3.1.3.6 *Step 6: JMS and Message-Driven Beans*

We need to configure the JMX Queue to be used by IaaS SM. In order to do that, we need to create the following file into the *<JBOSS_HOME>/server/default/deploy*

- **ExecutionQueue-service.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<server>
    <mbean xmbean-dd="xmdesc/Queue-xmbean.xml"
name="jboss.messaging.destination:service=Queue,name=ExecutionQueue"
code="org.jboss.jms.server.destination.
QueueService">
        <attribute name="JNDIName">ExecutionQueue</attribute>
        <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
        <depends>jboss.messaging:service=PostOffice</depends>
              <attribute name="MaxDeliveryAttempts">1</attribute>
    </mbean>
</server>
```

- **ProvisioningQueue-service.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<server>
    <mbean xmbean-dd="xmdesc/Queue-xmbean.xml"
name="jboss.messaging.destination:service=Queue,name=ProvisioningQue
ue" code="org.jboss.jms.server.destinati
on.QueueService">
        <attribute name="JNDIName">ProvisioningQueue</attribute>
        <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
        <depends>jboss.messaging:service=PostOffice</depends>
              <attribute name="MaxDeliveryAttempts">1</attribute>
    </mbean>
</server>
```

- **UpdatesQueue-service.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<server>
    <mbean xmbean-dd="xmdesc/Queue-xmbean.xml"
name="jboss.messaging.destination:service=Queue,name=UpdatesQueue"
code="org.jboss.jms.server.destination.Qu
eueService">
        <attribute name="JNDIName">UpdatesQueue</attribute>
        <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
        <depends>jboss.messaging:service=PostOffice</depends>
       <attribute name="MaxDeliveryAttempts">2</attribute>
       <attribute name="RedeliveryDelay">10</attribute>
    </mbean>
</server>
```

- **TaskTopic-service.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<server>
    <mbean xmbean-dd="xmdesc/Topic-xmbean.xml"
name="jboss.messaging.destination:service=Topic,name=TasksTopic"
code="org.jboss.jms.server.destination.Topi
cService">
        <attribute name="JNDIName">TasksTopic</attribute>
        <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
        <depends>jboss.messaging:service=PostOffice</depends>
              <attribute name="MaxDeliveryAttempts">1</attribute>
    </mbean>
</server>
```

IMPORTANT, you have to take into account that if we want to limit the number of resending messages, we have to specify it in the configuration of the queues with the following attribute:

```
<attribute name="MaxDeliveryAttempts">1</attribute>
```

Note: The maximum number of resends will be the minimum between dLQMaxResent (by default 5) and MaxDeliveryAttempts in the configuration of the queue.

3.1.3.7 *Step 7: Property file*

Previously to deploy Claudia EAR file, it is necessary to create the configuration file:

```
defaultHostDomain=hi.inet
openstack-tcloud.orgId=demo
openstack-tcloud.vdcId=occivdc
openstack-tcloud.serviceId=occiservice
openstack-tcloud.image.repository.url=http://localhost
openstack-tcloud.image.repository.path=/var/www
openstack-tcloud.image.extension=vmdk
openstack-tcloud.iaas=occi
openstack-tcloud.iaas.virtualSystemType=vmx-07
openstack-tcloud.iaas.network1=gestion
openstack-tcloud.iaas.network1.scope=private
openstack-tcloud.iaas.network2=servicio
openstack-tcloud.iaas.network2.scope=public
openstack-tcloud.clotho.protocol=http://
openstack-tcloud.clotho.host=<IP_SM>
openstack-tcloud.clotho.port=<PORT_SM>
openstack-tcloud.clotho.path=/rest-api-management
openstack-tcloud.ova.extension=ova
openstack-tcloud.ovf.extension=ovf
openstack-
tcloud.keystone.url=http://<IP_KEYSTONE>:<PORT_KEYSTONE>/v2.0/

openstack-tcloud.keystone.user=admin
openstack-tcloud.keystone.token=d000dc7bb6254400a0e89333ee40b9ed
openstack-
tcloud.openstack.url=http://<IP_OCCI_INTERFACE>:<PORT_OCCI_INTERFACE
>/v2/

openstack-tcloud.cloudSystem=FIWARE
```

The fields <IP_OCCI_INTERFACE> and <PORT_OCCI_INTERFACE> are the IP address and port respectively in which the IaaS DCRM with the occi interface is running. The

<IP_SM> is the IP address of the local machine, <PORT_SM> is the port used, usually 8080. The fields <IP_KEYSTONE> and <PORT_KEYSTONE> are the IP address and port respectively in which the OpenStack Keystone component is running.

3.1.3.8 *Step 8: Deploy EAR file in the deploy directory*

Copy the Claudia EAR file in the deploy directory of JBoss (*<JBOSS_HOME> /server/default/deploy/*). And check that the JBoss start without any problem. We can see it executing the following command in the *<JBOSS_HOME>/server/default/log directory*:

```
$ grep –i ‘started in’ server.log
server.log:2012-06-19 16:21:57,339 INFO
[org.jboss.bootstrap.microcontainer.ServerImpl] (main) JBoss
(Microcontainer) [5.1.0 (build: SVNTag=JBPAPP_5_1_0
date=201009150028)] Started in 2m:7s:941ms
```

We can check now that the EJBs was deployed properly if we check the **JMX Agent View** tree in *http://localhost:8080/jmx-console/* and select *service=JNDIView* following by the *invoke* button associated to the list operation in order to see the **JMX MBean Operation View**.

3.1.4 Apache Tomcat configuration

Ensure the non-free section is enabled for the APT repository configuration in “/etc/apt/sources.list”, e.g. “deb http://ftp.de.debian.org/debian testing main contrib non-free”. Remember that it is necessary to have Sun Java 6 installed.

```
apt-get update
apt-get install sun-java6-jdk
echo ‘JAVA_HOME="/usr/lib/jvm/java-6-sun"' >> /etc/environment
echo ‘JRE_HOME="/usr/lib/jvm/java-6-sun/jre"’ >> /etc/environment
```

Now, we can install Apache Tomcat 6 with the following command:

```
apt-get install tomcat6
```

To configure Tomcat as a Linux Service you should link that files as follows:

```
ln -s /etc/init.d/tomcat6 /etc/rc1.d/K01tomcat6
ln -s /etc/init.d/tomcat6 /etc/rc2.d/S17tomcat6
```

At the /var/lib/tomcat6/conf/server.xml file, we need to add the next line before “</Host>”:

```
```

```
"<Context docBase=="<WAR_FILE>" path="/v2.0" reloadable="true"/>"
```

Where <WAR_FILE> is the name of the OpenStack API Web Application Archive to be deployed. Configure to listen port 8774 adding this after "<Service name="Catalina">":

```
"<Connector port="8774" protocol="HTTP/1.1"
connectionTimeout="20000"
               URIEncoding="UTF-8"
               redirectPort="8443" />"
```

And finally, at the /var/lib/tomcat6/conf/context.xml file, we need to add the following lines:

```
<Resource name="jdbc/openstack_tcloud" auth="Container"
type="javax.sql.DataSource"
        driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/openstack_tcloud"
        username="<SQL_ADMIN_USER_NAME>"
password="<SQL_ADMIN_USER_PASWORD>" maxActive="20" maxIdle="10"
        maxWait="-1" />
```

Where <SQL_ADMIN_USER_NAME> and <SQL_ADMIN_USER_PASSWORD> must be the user name and password defined in the installation of MySQL.

3.1.4.1 *Step 1: Run Apache Tomcat*

Apache is now configured to be started as a service through initd. It could be started manually using this command:

```
"/etc/init.d/tomcat start"
```

It could be stopped manually using this command:

```
"/etc/init.d/tomcat stop"
```

3.1.4.2 *Step 2: MySQL connectivity*

Download the driver from [7], unrar/unzip it and extract the jar file and then copy it into /var/lib/tomcat6/lib

3.1.4.3 *Step 3: Property file*

Previously to deploy the OpenStack API WAR file, it is necessary to update the SystemConfiguration.properties file with the following data:

```
openstack-tcloud.vdcId=DEFAULT
openstack-tcloud.clotho.protocol=http://
openstack-tcloud.clotho.host=<IP_SM>
openstack-tcloud.clotho.port=<SERVER_PORT_SM>
openstack-tcloud.clotho.path=/rest-api-management
openstack-
tcloud.keystone.url=http://<KEYSTONE_HOST>:<KEYSTONE_HOST>/v2.0/
openstack-tcloud.keystone.token=<KEYSTONE_ADMIN_TOKEN>
openstack-tcloud.openstack.url=http://<OPENSTACK_HOST>:8774/v2/
openstack-tcloud.cloudSystem=FIWARE
```

Where <IP_SM> is the IP address of the IaaS SM, <SERVER_PORT_SM> is the port of the IaaS SM, <KEYSTONE_HOST> and <KEYSTONE_HOST> are respectively the IP address and port of the Keystone component, <KEYSTONE_ADMIN_TOKEN> is the admin_token parameter defined in the keystone.conf file and finally, <OPENSTACK_HOST> is the IP address of the OpenStack.

3.1.4.4 *Step 4: Deploy WAR file in the deploy directory*

Copy the OpenStack API WAR file in the deploy directory of Tomcat /var/lib/tomcat6/webapps

3.2 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.

3.2.1 End to End testing

Although one End to End testing must be associated to the Integration Test, we can show here a quick testing to check that everything is up and running. For this purpose we send a request to our API in order to test the credentials that we have from then and obtain a valid token to work with.

We use the curl for that, which is a command line tool that allows us to transfer data with URL syntax. The sentence will be:

```
curl -d '{"auth": {"tenantName": $TENNANTNAME,
"passwordCredentials":{"username": $USERNAME, "password":
$PASSWORD}}}'
-H "Content-type: application/json" -H "Accept: application/xml"
http://$KEYSTONE_IP:35357/v2.0/tokens
```

And this must return the following results:

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns="http://docs.openstack.org/identity/api/v2.0">
  <token expires="2012-10-26T07:51:21Z"
id="c2a48354570543ab97f2c87233f5f5e0">
    <tenant enabled="true" name="demo"
id="c8da25c7a373473f8e8945f5b0da8217"/>
  </token>
  <serviceCatalog>
    <service type="compute" name="nova">
      <endpoint
adminURL="http://130.206.80.11:8774/v2/c8da25c7a373473f8e8945f5b0da8
217" region="RegionOne"
internalURL="http://130.206.80.11:8774/v2/c8da25c7a373473f8e8945f5b0
da8217"
publicURL="http://130.206.80.11:8774/v2/c8da25c7a373473f8e8945f5b0da
8217"/>
    </service>
    <service type="image" name="glance">
      <endpoint adminURL="http://130.206.80.11:9292/v1"
region="RegionOne" internalURL="http://130.206.80.11:9292/v1"
publicURL="http://130.206.80.11:9292/v1"/>
    </service>
    <service type="volume" name="volume">
      <endpoint
adminURL="http://130.206.80.11:8776/v1/c8da25c7a373473f8e8945f5b0da8
217" region="RegionOne"
internalURL="http://130.206.80.11:8776/v1/c8da25c7a373473f8e8945f5b0
da8217"
publicURL="http://130.206.80.11:8776/v1/c8da25c7a373473f8e8945f5b0da
8217"/>
    </service>
    <service type="ec2" name="ec2">
      <endpoint adminURL="http://130.206.80.11:8773/services/Admin"
region="RegionOne"
internalURL="http://130.206.80.11:8773/services/Cloud"
publicURL="http://130.206.80.11:8773/services/Cloud"/>
    </service>
    <service type="sm" name="service_manager">
      <endpoint
adminURL="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0d
a8217" region="RegionOne"
internalURL="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5
b0da8217"
```

```
publicURL="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0
da8217"/>
    </service>
    <service type="object-store" name="swift">
      <endpoint adminURL="http://130.206.80.102:8080/v1"
region="RegionOne"
internalURL="http://130.206.80.102:8080/v1/AUTH_c8da25c7a373473f8e89
45f5b0da8217"
publicURL="http://130.206.80.102:8080/v1/AUTH_c8da25c7a373473f8e8945
f5b0da8217"/>
    </service>
    <service type="identity" name="keystone">
      <endpoint adminURL="http://130.206.80.100:35357/v2.0"
region="RegionOne" internalURL="http://130.206.80.100:5000/v2.0"
publicURL="http://130.206.80.100:5000/v2.0"/>
    </service>
  </serviceCatalog>
  <user username="admin" id="4f9788a991e14f3c8c51b889f681a29b"
name="admin">
    <role id="2d7519f6af294c6c9df0517c7b83cc02" name="admin"/>
  </user>
</access>
```

After we have obtained the token id and tenant id from the previous execution (c2a48354570543ab97f2c87233f5f5e0 and c8da25c7a373473f8e8945f5b0da8217), we can now check the Claudia making the following simple operation

```
curl -v -H 'X-Auth-Token: c2a48354570543ab97f2c87233f5f5e0' -H
"Content-Type: application/xml" -H "Accept: application/xml" -H
"Access-Control-Request-Method: GET" -H "Access-Control-Request-
Headers: Content-Type, X-Auth-Token, Origin, Accept" -H "Origin:
http://127.0.0.1" -X GET
"http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da8217/fla
vors"
```

Note that this operation not only check the normal operation from Claudia but also the integration with the IaaS DCRM, the answer that this operation should return is the following one:

```
* About to connect() to 130.206.80.91 port 8774 (#0)
*   Trying 130.206.80.91... connected
* Connected to 130.206.80.91 (130.206.80.91) port 8774 (#0)
> GET /v2.0/c8da25c7a373473f8e8945f5b0da8217/flavors HTTP/1.1
```

```
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0)
libcurl/7.19.7 OpenSSL/0.9.8r zlib/1.2.3
> Host: 130.206.80.91:8774
> X-Auth-Token: c2a48354570543ab97f2c87233f5f5e0
> Content-Type: application/xml
> Accept: application/xml
> Access-Control-Request-Method: GET
> Access-Control-Request-Headers: Content-Type, X-Auth-Token,
Origin, Accept
> Origin: http://127.0.0.1
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: http://127.0.0.1
< Access-Control-Allow-Credentials: true
< Www-Authenticate: Keystone uri='http://130.206.80.100:35357/v2.0/'
< Content-Type: application/xml
< Content-Length: 1128
< Date: Thu, 25 Oct 2012 07:58:15 GMT
<
* Connection #0 to host 130.206.80.91 left intact
* Closing connection #0
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flavors xmlns:ns2="http://www.w3.org/2005/Atom"
xmlns:ns3="http://docs.openstack.org/compute/api/v1.1">
  <ns3:flavor id="001" name="M1_TINY 0Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/001"/>
  </ns3:flavor>
  <ns3:flavor id="002" name="M1_SMALL 20Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/002"/>
  </ns3:flavor>
  <ns3:flavor id="003" name="M1_MEDIUM 40Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/003"/>
  </ns3:flavor>
```

```
  <ns3:flavor id="004" name="M1_LARGE 80Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/004"/>
  </ns3:flavor>
  <ns3:flavor id="005" name="M1_XLARGE 160Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/005"/>
  </ns3:flavor>
  <ns3:flavor id="006" name="M2_TINY 0Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/006"/>
  </ns3:flavor>
</flavors>
```

3.2.2 List of Running Processes

Due to the Claudia basically is running over the JBoss Application Server, the list of processes must be only the JBoss and MySQL. If we execute the following command:

```
ps -ewF | grep 'mysql\|jboss' | grep -v grep
```

It should show something similar to the following:

```
UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
mysql      757     1  0 34611  4632   0 Jun12 ?        00:03:05
/usr/sbin/mysqld
root     14419 13184  0   510   488   0 13:45 pts/0    00:00:00
/bin/sh ./run.sh -b 0.0.0.0 -Djboss.server.log.threshold=WARN
root     14443 14419  1 490505 883040 0 13:45 pts/0    00:04:08 java
-Dprogram.name=run.sh -server -Xms512M -Xmx1024M -
XX:MaxPermSize=312M ...
```

Where you can see the MySQL daemon , and the run process to launch the JBoss and the java process that the previous command launch.
In case of the Tomcat node, if we execute the following command:

```
ps -ewF | grep 'mysql\|apache' | grep -v grep
```

It should show something similar to the following:

```
UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
```

```
mysql     1903     1  0 77905 14132   0 Jun22 ?        00:24:38
/usr/sbin/mysqld

root      2155     1  0 21954  2828   0 Jun22 ?        00:00:21
/usr/sbin/apache2 -k start

www-data 18981  2155  0 21956  2252   0 Jul01 ?        00:00:00
/usr/sbin/apache2 -k start

tid      18988  2155  0 78697 36856   0 Jul01 ?        00:00:04
/usr/sbin/apache2 -k start

tid      18989  2155  0 62411 37700   0 Jul01 ?        00:00:03
/usr/sbin/apache2 -k start

tid      18990  2155  0 62401 37208   0 Jul01 ?        00:00:03
/usr/sbin/apache2 -k start

www-data 18991  2155  0 78437  5912   0 Jul01 ?        00:00:00
/usr/sbin/apache2 -k start

www-data 18992  2155  0 94841  6028   0 Jul01 ?        00:00:00
/usr/sbin/apache2 -k start

www-data 18994  2155  0 78469  6060   0 Jul01 ?        00:00:00
/usr/sbin/apache2 -k start
```

3.2.3 Network interfaces Up & Open

Taking into account the results of the ps commands in the previous section, we take the PID in order to know the information about the network interfaces up & open. To check the ports in use and listening, execute the command:

```
netstat –p –a | grep $PID/java
```

Where $PID is the PID of Java process obtained at the ps command described before, in the previous case 757 (mysqld), 14419 (run.sh), 14443 (java). The expected results must be something similar to the following:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
tcp        0      0 localhost:41958         *:*
LISTEN      14443/java
tcp        0      0 *:4712                  *:*
LISTEN      14443/java
tcp        0      0 *:8009                  *:*
LISTEN      14443/java
tcp        0      0 *:4713                  *:*
LISTEN      14443/java
tcp        0      0 *:4457                  *:*
LISTEN      14443/java
```

```
tcp        0      0 *:1098                  *:*
LISTEN      14443/java

tcp        0      0 localhost:mysql         *:*
LISTEN      757/mysqld

tcp        0      0 *:rmiregistry           *:*
LISTEN      14443/java

tcp        0      0 *:23533                 *:*
LISTEN      14443/java

tcp        0      0 *:http-alt              *:*
LISTEN      14443/java

tcp        0      0 *:8083                  *:*
LISTEN      14443/java

tcp        0      0 *:8787                  *:*
LISTEN      14443/java

tcp        0      0 *:4444                  *:*
LISTEN      14443/java

tcp        0      0 *:4445                  *:*
LISTEN      14443/java

tcp        0      0 *:4446                  *:*
LISTEN      14443/java

tcp        0      0 *:3873                  *:*
LISTEN      14443/java

tcp        0      0 localhost:38611         localhost:mysql
ESTABLISHED 14443/java

tcp        0      0 localhost:mysql         localhost:38611
ESTABLISHED 757/mysqld



Active UNIX domain sockets (servers and established)

Proto RefCnt Flags       Type       State         I-Node
PID/Program name    Path

unix  2      [ ACC ]     STREAM     LISTENING     7595
757/mysqld          /var/run/mysqld/mysqld.sock

unix  2      [ ]         STREAM     CONNECTED     1123535
14443/java
```

Apart from these ports, the JBoss Web Interface (Coyote) is listening in port 8080 and the MySQL is listening in port 3307

In case of Apache Tomcat, the result will we the following:

```


Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
```

```
tcp        0      0 *:www                       *:*
LISTEN      2155/apache2
tcp        0      0 <DCRM_HOST>:45326     <DCRM_HOST>:8774
ESTABLISHED 18989/apache2
tcp        0      0 <DCRM_HOST>:45324     <DCRM_HOST>:8774
ESTABLISHED 18990/apache2



Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node
PID/Program name    Path
unix  2      [ ACC ]     STREAM     LISTENING     5560971
2155/apache2        /var/run/apache2/wsgi.2155.2.1.sock
unix  2      [ ACC ]     STREAM     LISTENING     5560974
18981/apache2       /var/run/apache2/cgisock.2155
```

3.2.4 Databases

The last step in the sanity check, once that we have identified the processes and ports is to check the different databases that have to be up and accept queries. Fort he first one, if we execute the following commands:

```
mysql –u$MYSQL_USER –p$MYSQL_PASS -e "show databases;"
```

$MYSQL_USER and $MYSQL_PASS must be the same defined in the installation process. It should show something similar to the following results:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ClaudiaDB          |
| mysql              |
+--------------------+
```

Where we can see the Database ClaudiaDB, if we execute now the following commands:

```
mysql –u$MYSQL_USER –p$MYSQL_PASS -e "use ClaudiaDB;"
mysql –u$MYSQL_USER –p$MYSQL_PASS ClaudiaDB -e "show tables;"
```

in order to show the different tables contained in the ClaudiaDB, we should see the following:

```
+----------------------------------------+
| Tables_in_ClaudiaDB                    |
+----------------------------------------+
| Capacity                               |
| ComplexOperation                       |
| ComplexOperation_Message               |
| ComplexOperation_globalData            |
| InfrastructureConfiguration            |
| LogEntry                               |
| MeasuredValue                          |
| MeasuredValue_ipAddresses              |
| Media                                  |
| MediaFile                              |
| Message                                |
| Message_metadata                       |
| MonitoringSample                       |
| Network                                |
| NodeDirectory                          |
| OperatingSystem                        |
| Organization                           |
| Organization_attributes                |
| ResourceConsumption                    |
| ServiceApplication                     |
| ServiceApplication_Network             |
| ServiceApplication_domainAffinities    |
| ServiceApplication_domainAntiAffinities |
| ServiceApplication_hostAffinities      |
| ServiceApplication_hostAntiAffinity    |
| ServiceApplication_siteAffinities      |
| ServiceApplication_siteAntiAffinities  |
| Snapshot                               |
| Task                                   |
| TaskResult                             |
| TaskResult_attributes                  |
| VDC                                    |
| VMTemplate                             |
| VirtualMachine                         |
```

```
| VirtualMachine_availableOperations      |
| VirtualMachine_availablePowerOperations |
| Zone                                    |
+-----------------------------------------+
```

Now, we can execute a simple test query in order to check the content of the table:

```
mysql –u$MYSQL_USER –p$MYSQL_USER ClaudiaDB -e "select * from
OperatingSystem;"
```

And it should return with the following information

```
+------------+-----------------------------------------+------+-----
----+
| internalId | description                             | id   |
version |
+------------+-----------------------------------------+------+-----
----+
|          5 | Other 2.6x Linux (64-bit)               | 1    | 5
|
|          8 | Other 2.6x Linux (64-bit)               | 1    | 5
|
|          9 | Other 2.6x Linux (64-bit)               | 1    | 5
|
|         10 | Other 2.6x Linux (64-bit)               | 1    | 5
|
|         11 | Other 2.6x Linux (64-bit)               | 1    | 5
|
|         33 | Red Hat Enterprise Linux 5 (32-bit)     | 79   | 5
|
|         81 | Cirros 0.3.0 - x86_64 - OPENSTACK Image | 79   | 5
|
|        106 | Cirros 0.3.0 - x86_64 - OPENSTACK Image | 79   | 5
|
|        109 | Cirros 0.3.0 - x86_64 - OPENSTACK Image | 79   | 5
|
|        111 | Cirros 0.3.0 - x86_64 - OPENSTACK Image | 79   | 5
|
|        114 | Cirros 0.3.0 - x86_64 - OPENSTACK Image | 79   | 5
|
+------------+-----------------------------------------+------+-----
----+
```

In case of Tomcat node, if we execute the following command:

```
mysql –u$MYSQL_USER –p$MYSQL_PASS -e "show databases;"
```

It should show something similar to the following results:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| openstack_tcloud   |
+--------------------+

mysql –u$MYSQL_USER –p$MYSQL_PASS -e "use openstack_tcloud;"
mysql –u$MYSQL_USER –p$MYSQL_PASS openstack_tcloud -e "show tables;"
```

in order to show the different tables contained in the ClaudiaDB, we should see the following:

```
+----------------------------+
| Tables_in_openstack_tcloud |
+----------------------------+
| Flavor                     |
| HardwareConfiguration      |
| IP                         |
| Image                      |
| ImageDecodeID              |
| Metadata                   |
| Ova                        |
| OvaInstantiation           |
| Ova_OvaInstantiation       |
| OvfInstantiation           |
| Personality                |
| Server                     |
| ServerDecodeID             |
| Server_IP                  |
| Server_Metadata            |
| Server_Personality         |
| Server_Snapshot            |
```

```
| Snapshot                    |
| configuration_properties    |
+-----------------------------+
```

And finally, we can execute a simple test query in order to check the content of the table:

```
mysql –u$MYSQL_USER –p$MYSQL_USER ClaudiaDB -e "select * from Flavor
limit 10;
```

And it should return with the following information:

```
+-----------------+-----+
| name            | id  |
+-----------------+-----+
| M1_TINY 100Mb   | 001 |
| M1_TINY 10.1Gb  | 002 |
| M1_TINY 20.1Gb  | 003 |
| M1_TINY 30.1Gb  | 004 |
| M1_TINY 40.1Gb  | 005 |
| M1_TINY 50.1Gb  | 006 |
| M1_SMALL 100Mb  | 007 |
| M1_SMALL 10.1Gb | 008 |
| M1_SMALL 20.1Gb | 009 |
| M1_SMALL 30.1Gb | 010 |
+-----------------+-----+
```

3.3 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

3.3.1 Resource availability

The resource availability in the JBoss node should be at least 2Gb of RAM and 8GB of Hard disk in order to prevent enabler’s bad performance. In the Tomcat node should be at least 2Gb of RAM and 8GB of Hard disk in order to prevent enabler's bad performance. This means that bellow these thresholds the enabler is likely to experience problems or bad performance.

3.3.2 Remote Service Access

We have internally two components to connect, the OpenStack API component and the Claudia itself. After that two internals component, we should connect with the IaaS DCRM and the IdM GE (aka Keystone in this first release). An administrator to verify that such links are available will use this information.
The first step is to check that the Claudia is up and running, for this purpose we can execute the following curl command, which is a simple GET operation:

```
root@fiware:~# curl http://<IP/host>:8080/rest-api-
management/api/org/fla
```

The <IP/Host> variable will be the IP direction in which we have installed the Claudia. This request should return one <org> element (Organization or Tennant) in the following xml response structure:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Org xmlns="http://schemas.tcloud.telefonica.com/tcloud/1"
xmlns:tcloud="http://schemas.tcloud.telefonica.com/tcloud/1"
href="http://<IP/Host>:8080/rest-api-management/api/org/fla"
name="fla">
  <Link href="http://<IP/Host>:8080/rest-api-
management/api/org/fla/action/instantiateVdc" rel="add"
type="application/vnd.telefonica.tcloud.vdc+xml"/>
  <Link href="http://<IP/Host>:8080/rest-api-
management/api/org/fla/task" rel="tasks"
type="application/vnd.telefonica.tcloud.tasklist+xml"/>
</Org>
```

In order to check the connectivity between the Claudia and the IdM GE, due to it must obtain a valid token and tenant for a user and organization with the following curl commands:

```
root@fiware:~# curl -d '{"auth": {"tenantName": "<MY_ORG_NAME>",
"passwordCredentials":{"username": "<MY_USERNAME>", "password":
"<MY_PASS>"}}}' -H "Content-type: application/json" -H "Accept:
application/xml"  http://<KEYSTONE_HOST>:<KEYSTONE_PORT>/v2.0/tokens
```

The <MY_ORG_NAME> will be the name of my Organization/Tennat/Project predefined in the IdM GE (aka Keystone). The <MY_USERNAME> and <MY_PASS> variables will be the user name and password predefined in the IdM GE and finally the <KEYSTONE_HOST> and <KEYSTONE_PORT> variables will be the IP direction and port in which we can find the IdM GE (aka Keystone). This request should return one valid token for the user credentials together with more information in a xml format:

```
<?xml version="1.0" encoding="UTF-8"?>
<access xmlns="http://docs.openstack.org/identity/api/v2.0">
```

```
  <token expires="2012-06-30T15:12:16Z"
id="9624f3e042a64b4f980a83afbbb95cd2">
    <tenant enabled="true" id="30c60771b6d144d2861b21e442f0bef9"
name="FIWARE">
      <description>FIWARE Cloud Chapter demo project</description>
    </tenant>
  </token>
  <serviceCatalog>
  …
  </serviceCatalog>
  <user username="fla" id="b988ec50efec4aa4a8ac5089adddbaf9"
name="fla">
    <role id="32b6e1e715f14f1dafde24b26cfca310" name="Member"/>
  </user>
</access>
```

With this information (extracting the token id), we can perform a GET operation to the IaaS DCRM in order to get the list of available flavors. For this purpose we can execute the following curl commands:

```
curl -v -H "Access-Control-Request-Method: GET" -H "Access-Control-
Request-Headers: Content-Type, X-Auth-Token" -H "Origin: http://<IP
LOCAL>" -H 'X-Auth-Token: a9a861db6276414094bc1567f664084d' -H
"Content-Type: application/xml" -H "Accept: application/xml" -X GET
"http://<OPENSTACK API
HOST>:8774/v2.0/c8da25c7a373473f8e8945f5b0da8217/flavors"
```

The <OPENSTACK API HOST> variable will be the IP direction in which we have installed the OpenStack API functionality. The <IP LOCAL> is the IP address of the local machine from which the petition is launched. This request should return one valid token for the user credentials that we send through the curl command in the following xml response structure:

```
* About to connect() to <OPENSTACK API HOST> port 8774 (#0)
*   Trying <OPENSTACK API HOST>... connected
* Connected to <OPENSTACK API HOST> (<OPENSTACK API HOST>) port 8774
(#0)
> GET /v2.0/c8da25c7a373473f8e8945f5b0da8217/flavors HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0)
libcurl/7.19.7 OpenSSL/0.9.8r zlib/1.2.3
> Host: <OPENSTACK API HOST>:8774
> Access-Control-Request-Method: GET
> Access-Control-Request-Headers: Content-Type, X-Auth-Token
> Origin: http://<IP LOCAL>
```

```
> X-Auth-Token: a9a861db6276414094bc1567f664084d
> Content-Type: application/xml
> Accept: application/xml
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Origin: http://<IP LOCAL>
< Access-Control-Allow-Credentials: true
< Www-Authenticate: Keystone
uri='http://<KEYSTONE_HOST>:<KEYSTONE_PORT>/v2.0/'
< Content-Type: application/xml
< Content-Length: 1128
< Date: Wed, 24 Oct 2012 12:46:49 GMT
<
* Connection #0 to host <OPENSTACK API HOST> left intact
* Closing connection #0
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<flavors xmlns:ns2="http://www.w3.org/2005/Atom"
xmlns:ns3="http://docs.openstack.org/compute/api/v1.1">
  <ns3:flavor id="001" name="M1_TINY 0Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/001"/>
  </ns3:flavor>
  <ns3:flavor id="002" name="M1_SMALL 20Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/002"/>
  </ns3:flavor>
  <ns3:flavor id="003" name="M1_MEDIUM 40Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/003"/>
  </ns3:flavor>
  <ns3:flavor id="004" name="M1_LARGE 80Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/004"/>
  </ns3:flavor>
  <ns3:flavor id="005" name="M1_XLARGE 160Gb">
```

```
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/005"/>
  </ns3:flavor>
  <ns3:flavor id="006" name="M2_TINY 0Gb">
    <ns2:link rel="self"
href="http://130.206.80.91:8774/v2.0/c8da25c7a373473f8e8945f5b0da821
7/flavors/006"/>
  </ns3:flavor>
</flavors>
```

3.3.3 Resource consumption

State the amount of resources that are abnormally high or low. This applies to RAM, CPU and I/O. For this purpose we have differentiated between:

- Low usage, in which we check the resources that the JBoss or Tomcat requires in order to load the IaaS SM.
- High usage, in which we send 100 concurrent accesses to the Claudia and OpenStack API.

The results were obtained with a top command execution over the following machine configuration:

Machine Info		
	JBoss Node	**Tomcat Node**
Type Machine	Virtual Machine	Physical Machine
CPU	1 core @ 2,4Ghz	Intel(R) Xeon(R) CPU X5650 Dual Core @ 2.67GHz
RAM	1,4GB	4GB
HDD	9,25GB	100GB
Operating System	Ubuntu 11.10	Debian 6.0

The results of requirements both RAM, CPU and I/O to HDD in case of JBoss node is shown in the following table:

Resource Consumption (in JBoss node)		
	Low Usage	**High Usage**

RAM	1,2GB ~ 70%	1,4GB ~ 83,5%
CPU	1,3% of a 2400MHz	95% of a 2400MHZ
I/O HDD	6GB	6GB

And the results of requirements both RAM, CPU and I/O to HDD in case of Tomcat node is shown in the following table:

Resource Consumption (in Tomcat node)		
	Low Usage	**High Usage**
RAM	1GB ~ 63%	3GB ~ 78%
CPU	0,8% of a 2400MHz	90% of a 2400MHZ
I/O HDD	6GB	6GB

3.3.4 I/O flows

The application WAR is hearing from port 8774 and the EAR application (by default) is hearing in the port 8080. Please refer to the installation process in order to know exactly which was the Claudia port selected.

4 Object Storage - Installation and Administration Guide

You can find the content of this chapter as well in the wiki of fi-ware.

4.1 Introduction

Welcome to the Installation and Administration Guide for the Object Storage Generic Enabler. This generic enabler is built on an Open Source project, and so where possible this guide points to the appropriate online content that has been created for this project. The online documents are being continuously updated and improved, and so will be the most appropriate place to get the most up to date information on installation and administration. The required parts of this generic enabler are OpenStack's Swift as well as Cloud Data Management Interface (CDMI) for accessing the object storage.

4.2 System Installation

The following steps need to be performed to get the CDMI interface for OpenStack up & running:

```
1. Install Openstack with Swift according to the Instructions from
[OpenStack].
2. Install the CDMI interface from [github] using the command
'python setup.py install'
3. Configure the Swift proxy server.
```

To configure the proxy server edit the file /etc/swift/proxy-server.conf and add the "cdmi" filter before "proxy-server"

```
    [pipeline:main]
    pipeline = healthcheck cache tempauth *cdmi* proxy-server
```

Also add the following section to the same file:

```
    [filter:cdmi]
    use = egg:cdmiapi#cdmiapp
```

4.3 System Administration

Please see the OpenStack Swift Documentation at http://swift.openstack.org

4.4 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.

4.4.1 End to End testing

Please note that the following information is required before carrying out this procedure:

- the IP address of the CDMI node (e.g. on the FI-WARE testbed this is cdmiservice.lab.fi-ware.eu)
- the IP address of the Openstack Keystone node managing security for the Object Storage GE Deployment (e.g. on the FI-WARE testbed this is 130.206.80.100)
- a valid OpenStack (keystone) username and password

1. Verify that **http://cdmiservice.lab.fi-ware.eu:8080/cdmi** can be reached. By default, web access will receive a 401 Unauthorized response.
2. Acquire a valid token from the OpenStack Keystone server. Using curl, the following command can be used:

```
   curl -d '{"auth":{"passwordCredentials":{"username":"<Insert
Openstack username here>", "password":"<Insert Openstack password
here>"}}}' -H "Content-type: application/json" http://<Insert IP
address of Keystone node here>:5000/v2.0/tokens
```

The resulting Keystone Token is returned near the beginning of the response.

3. Verify that you can retrieve the capabilities of the root Object container:

```
   curl -v -X GET -H 'X-Auth-Token: <Insert Keystone Token here>' -H
'X-CDMI-Specification-Version: 1.0.1' http://cdmiservice.lab.fi-
ware.eu:8080/cdmi/cdmi_capabilities/AUTH_<tenant_id>/
```

Sample output of this command:

```
 > GET
/cdmi/cdmi_capabilities/AUTH_d418851c6d294381bbe6e082849686d6/
HTTP/1.1
 > User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0
OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
 > Host: 130.206.80.102:8080
 > Accept: */*
 > X-Auth-Token: e40f876706d1470facf783a5ce656d63
 > X-CDMI-Specification-Version: 1.0.1
 >
 < HTTP/1.1 200 OK
 < Content-Type: application/cdmi-capability
 < X-Cdmi-Specification-Version: 1.0.1
 < Content-Length: 200
 < X-Trans-Id: txe226c68ba16a481b9f5c6838006e4b50
 < Date: Mon, 29 Oct 2012 15:04:35 GMT
 <
 {
   "capabilities": {
    "cdmi_list_children": true,
```

```
     "cdmi_create_container": true
   },
   "objectName": "AUTH_d418851c6d294381bbe6e082849686d6",
   "objectType": "application/cdmi-capability"
 }
```

4. Create a test container:

```
   curl -v -X PUT -H 'X-Auth-Token: <Insert Keystone Token here>' -H
'Content-tType: application/directory'-H 'Content-Length: 0'
http://cdmiservice.lab.fi-
ware.eu:8080/cdmi/AUTH_<tenant_id>/<container_name>/
```

Expected output:

```
 > PUT /v1/AUTH_288e7938013a48529a62a30d62edf0cc/test_container
HTTP/1.1
 > User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0
OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
 > Host: 192.168.0.10:8080
 > Accept: */*
 > X-Auth-Token: 87efddec36c34e8da15fd2909ca7e8cd
 > Content-tType: application/directory-H
 >
 < HTTP/1.1 201 Created
 < Content-Length: 18
 < Content-Type: text/html; charset=UTF-8
 < X-Trans-Id: tx128ef4eb253b4535b23c46a68603b173
 < Date: Thu, 26 Jul 2012 08:12:16 GMT
 <
 201 Created
```

4.4.2 List of Running Processes

In case OpenStack has been installed in the directory '/opt/stack/' the following command will allow the admin to see all process running out of that directory:

```
 ps -auxw | grep /opt/stack
```

The output should include the following two services:

```
 /opt/stack/keystone/bin/keystone-all --config-file
/etc/keystone/keystone.conf --log-config /etc/keystone/logging.conf
-d --debug
 /opt/stack/swift/bin/swift-proxy-server /etc/swift/proxy-
server.conf -v
```

Many other services based on your installation might also be available. Also noted that the 'keystone' is optional here.

4.4.3 Network interfaces Up & Open

- TCP port 8080 should be accessible to the client
- Proxy server and Storage Servers in the Swift Ring should be in the same network and have access to each other.

4.4.4 Databases

Swift does not use traditional databases.

4.5 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

4.5.1 Resource availability

- Verify that enough diskspace is left using the UNIX command 'df'
- Ensure that no error messages can be found in /var/log

4.5.2 Remote Service Access

Please make sure port 8080 is accessible. When using Keystone or similar port 5000 needs to be open too.

4.5.3 Resource consumption

Swift and the CDMI interface have very minimal resource constraints. Since this is a storage solutions disk space is required. A swift installation is recommended to have at least 5 storage nodes for redundancy purposes.

4.5.4 I/O flows

Clients access the Object Storage through the CDMI interface. This is simple HTTP traffic. Based on the type of the object and the server it is stored on I/O data is read/written and sent over the network within the Swift installation and towards the Client.