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

D.4.4.3: FI-WARE User and Programmers 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.4.4.3
Project Document Date: 2014-07-31
Deliverable Type and Security: PU
Author: FI-WARE Consortium
Contributors: FI-WARE Consortium
1.1 Executive Summary

This document includes the information required for Users and Programmers to interact with each of the Generic Enablers developed within the "Cloud Hosting" chapter.
1.2 About This Document

This document comes along with the Software implementation of components, each release of the document being referred to the corresponding Software release (as per D.x.3), to provide documentation of the features offered by the components and interfaces to users/adopters. Moreover, it explains the way they can be exploited in their developments.

1.3 Intended Audience

The document targets users as well as programmers of FI-WARE Generic Enablers.

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, as outlined in Materializing Cloud Hosting in FI-WARE.

The capabilities available in the second release of FI-WARE Cloud Hosting platform are outlined in Roadmap of Cloud Hosting.

The following diagram shows the main components (Generic Enablers) that comprise the second release of FI-WARE architecture.

The architecture comprises a set of Generic Enablers that together provide hosting capabilities of several kinds and at several levels of resource abstraction -- aiming at the needs of different applications hosted on the cloud platform. **IaaS Data Center Resource Management (DCRM) GE** is offering provisioning and life cycle management of virtualized resources (compute, storage, network) associated with **virtual machines**, which can run general purpose Operating Systems as well as arbitrary software stacks. Application developers and providers can use these virtual machines to develop and deploy their own software components that comprise their application stacks. **Object Storage GE** offers provisioning and life cycle management of **object-based** storage containers and elements, which can be efficiently used to store unstructured fixed content (such as images, videos, etc) as well as accompanying metadata. **Job Scheduler GE** offers the application to submit and manage computational jobs in a unified and scalable manner. **Edgelet Management GE** offers the capability to host lightweight application components, called **edgelets**, on devices typically located outside of the Data Center, such as those provided by the **Cloud Proxy GE** (developed jointly by the Cloud chapter and...
the Interfaces to Network and Devices chapter). **Software Deployment and Configuration (SDC) GE** offers a flexible framework for installation and customization of software products within individual virtual machines. **Policy Manager GE** provides a framework for rule-based management of cloud resources, including application auto-scaling based leveraging metrics collected by **Monitoring GE**. Lastly, **PaaS Management GE** uses the above capabilities to offer holistic provisioning and ongoing management of complex workloads comprising sophisticated combination of interdependent VMs and associated resources (such as multi-tier web applications or even complete custom-built PaaS environments), as well as configuration and management of software components within the VMs. Each of the above GEs provides a REST API that can be used programmatically. The human actor represents the programmatic user of the different capabilities of the Cloud GEs via REST APIs. Moreover, the Cloud chapter provides a Web-based **Portal** (part of of the UI layer), which surfaces main capabilities in an interactive manner --such as provisioning and monitoring of VM instances and services.

Cloud Hosting Generic Enablers are using the **Identity Management and Access Control** framework provided by the Security chapter, as outlined in the **Cloud Security Architecture**.

### 1.5 Structure of this Document

The document is generated out of a set of documents provided in the FI-WARE wiki. For the current version of the documents, please visit the wiki at [http://wiki.fi-ware.eu](http://wiki.fi-ware.eu)

The following resources were used to generate this document:

- D.4.4.3_User_and_Programmers_Guide_front_page
- IaaS Data Center Resource Management - User and Programmers Guide
- Object Storage - User and Programmers Guide
- PaaS Management - User and Programmers Guide
- Software Deployment And Configuration - User and Programmers Guide
- Policy Manager - User and Programmers Guide
- Monitoring - User and Programmers Guide
- Self-Service Interfaces - User and Programmers Guide
- Job Scheduler - User and Programmers Guide
- Edgelets - User and Programmers 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 our wikis. 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 part of the links contained in the deliverables generated from wiki pages cannot be rendered to fully working links. This happens for instance when a wiki page references a section within the same wiki page (but there are other cases). In such scenarios we preserve a link for readability purposes but this points to an explanatory page, not the original target page.

In such cases where you find links that do not actually point to the original location, we encourage you to visit the source pages to get all the source information in its original form. Most of the links are however correct and this impacts a small fraction of those in our deliverables.

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, Thales, UPM, INRIA.

1.8 Keyword list

1.9 Changes History

<table>
<thead>
<tr>
<th>Release</th>
<th>Major changes description</th>
<th>Date</th>
<th>Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>First draft</td>
<td>2014-07-24</td>
<td>IBM</td>
</tr>
<tr>
<td>v2</td>
<td>Final review. Ready for delivery</td>
<td>2014-07-28</td>
<td>IBM</td>
</tr>
</tbody>
</table>

1.10 Table of Contents

2 IaaS Data Center Resource Management - User and Programmers Guide........ 7
3 Object Storage - User and Programmers Guide........................................ 16
4 PaaS Management - User and Programmers Guide .................................... 39
5 Software Deployment And Configuration - User and Programmers Guide........ 52
6 Policy Manager - User and Programmers Guide ........................................ 59
7 Monitoring - User and Programmers Guide.............................................. 65
8 Self-Service Interfaces - User and Programmers Guide .......................... 69
9 Job Scheduler - User and Programmers Guide ......................................... 76
10 Edgelets - User and Programmers Guide ............................................... 229
2 IaaS Data Center Resource Management - User and Programmers Guide

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

2.1 Introduction

Welcome to the User and Programmer Guide for the DCRM Generic Enabler. This generic enabler is built on a proprietary solution using standard interface to communicate with and so where possible this guide points to the appropriate online content that has been created for this specific API. 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 using this interface.

2.2 User Guide

The DCRM GE is a backend component, without user interface. Therefore there is no need to provide a user guide. The Cloud Portal can be used for Web-based interaction (but it is not part of this GE).

2.3 Programmer Guide

OpenStack Compute API is based upon HTTP and therefore all devices, which can handle HTTP traffic, are possible clients.

2.3.1 Accessing DCRM from the CLI

To invoke the REST API use the curl program. Curl [1] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, Gopher, Dict, Telnet, LDAP or File) and therefore is also usable for OpenStack Compute API. Use the curl command line tool or use libcurl from within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

**nova-client** implements various Openstack Nova's APIs and provides a simple way to execute them from the CLI. A detailed guide can be found here: [2]

2.3.1.1 Get a valid token

First thing required for any API invocation, either a direct REST request (using curl for example) or by using nova-client is to provide credentials.

2.3.1.1.1 nova-credential authorization

```
source openrc
```

If you don't already have openrc file, download it from Horizon (Openstack's dashboard service): [http://docs.openstack.org/user-guide/content/cli_openrc.html](http://docs.openstack.org/user-guide/content/cli_openrc.html)
2.3.1.1.2  REST authorization

For running API commands, you need a valid Keystone token (tokens are valid for 24 hours). To generate a token:

```bash
curl -d '{"auth": {"tenantName": "demo", "passwordCredentials": {"username": "admin", "password": "password"}}}' -H "Content-type: application/json" http://127.0.0.1:35357/v2.0/tokens
```

Result example:

```
{"access": {"token": {"expires": "2012-06-22T07:50:54Z", "id": "d71c70e2d0834d4baaa7ec9f2b94b7ca", "tenant": {"enabled": true, "id": "a4f4eb48f31c447e84606368a7193c9c", "name": "demo", "description": null}}, "serviceCatalog": [{"endpoints": [{"adminURL": "http://192.168.255.135:8774/v2/a4f4eb48f31c447e84606368a7193c9c"}]}}
```

Extract the first 'id' value, this is your token Extract adminURL, this is the endpoint you will send the requests to

To validate both your endpoint and token, invoke the following command:

```bash
curl -v -X GET -H 'X-Auth-Token: d71c70e2d0834d4baaa7ec9f2b94b7ca' -H "Content-type: application/json" http://192.168.255.135:8774/v2/0/a4f4eb48f31c447e84606368a7193c9c
```

The response should be similar to the one below:

```
* About to connect() to 192.168.255.135 port 8774 (#0)
* Trying 192.168.255.135... connected
* Connected to 192.168.255.135 (192.168.255.135) port 8774 (#0)
  > GET /v2.0/a4f4eb48f31c447e84606368a7193c9c HTTP/1.1
  > User-Agent: curl/7.21.6 (i686-pc-linux-gnu) libcurl/7.21.6 OpenSSL/1.0.0e zlib/1.2.3.4 libidn/1.22 librtmp/2.3
  > Host: 192.168.255.135:8774
  > Accept: */*
  > X-Auth-Token: d71c70e2d0834d4baaa7ec9f2b94b7ca
  > Content-type: application/json
  >
  < HTTP/1.1 300 Multiple Choices
  < Content-Type: application/json
```
2.3.1.2 **ResourceManager Placement APIs**

2.3.1.2.1 **Optimize**

Overtime, as new VMs are scheduled and some VMs have been terminated, it may be desired to check whether there is a better way to schedule the existing VMs on the Compute nodes. This is what the Scheduler's Optimize provide you with. Administrator permissions are required for this API. To manually invoke it:

```
```

2.3.1.2.2 **Put in Maintenance**

Sometimes administrator would like to take down a host for maintenance reasons, among them Hardware upgrade Security patch Update and reboot.

In order to keep the VMs running and prevent new VMs from being scheduled into a host that soon will be taken down, use the PutInMaintenance API to live migrate the relevant VMs into alternate hosts.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>host_id</td>
<td>String</td>
<td>NULL</td>
<td>Compute node to be put in maintenance mode</td>
</tr>
<tr>
<td>mode</td>
<td>String</td>
<td>NULL</td>
<td>enable/disable</td>
</tr>
</tbody>
</table>

Start PutInMaintenance using nova-client:

```
$nova host-update host_id --maintenance enable
```

where host_id is the name of the host you are about to take down for maintenance.

Once maintenance work is completed, use:

```
$nova host-update host_id --maintenance disable
```

2.3.1.3 **ResourceManager Network APIs**

Quantum API is supported and can be found here: [Quantum developer guide](#) and [Quantum API](#).

Additionally, we support the following three resources as extensions to the standard API:

- Policies (policys in Quantum terminology)
- Actions
- Filters

<table>
<thead>
<tr>
<th>Resource</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>/policys</td>
<td>Defines policy for specified networks</td>
</tr>
</tbody>
</table>
Similar to standard Quantum API, we provide CRUD operations through REST API. Operations are specifically, List, Show, Create, Update, and Delete.

Below, we show the Create operations to list the attributes that can be used in the request body. Other operations follow the same pattern as specified in Quantum API v2.0.

### 2.3.1.3.1 Create Policy

Create unidirectional DOVE Policy between pair of networks.

This operation requires a request body.

The request body must contain the following attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_net_name</td>
<td>String</td>
<td>NULL</td>
<td>Source network name</td>
</tr>
<tr>
<td>src_net_id</td>
<td>UUID</td>
<td>NULL</td>
<td>Source network UUID</td>
</tr>
<tr>
<td>dst_net_name</td>
<td>String</td>
<td>NULL</td>
<td>Dst. network name</td>
</tr>
<tr>
<td>dst_net_id</td>
<td>UUID</td>
<td>NULL</td>
<td>Dst. network UUID</td>
</tr>
<tr>
<td>action_id</td>
<td>UUID</td>
<td>NULL</td>
<td>UUID of the action</td>
</tr>
<tr>
<td>ttl</td>
<td>Int</td>
<td>10</td>
<td>Policy time to live in seconds</td>
</tr>
</tbody>
</table>

Sample Request Body: JSON Request

```json
POST v2.0/networks.json Content-Type: application/json
Accept: application/json

{
  "policy": {
    "src_net_name": "Net1",
    "src_net_id": "d6b4d3a5-c700-476f-b609-1493dd9eb101", "dst_net_name": "Net2", "dst_net_id": "d6b4d3a5-c700-476f-b609-1493dd9eb102", "action_id": "d6b4d3a5-c700-476f-b609-1493dd9dadc0", "ttl": 360
  }
}
```

### 2.3.1.3.2 Create Action

Create a policy action to be used by DOVE policy

This operation requires a request body.
The request body must contain the following attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>NULL</td>
<td>Action name</td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>NULL</td>
<td>Action type (Connectivity, QoS, Waypoint, Security, ACL, etc.)</td>
</tr>
<tr>
<td>action</td>
<td>String</td>
<td>NULL</td>
<td>Per type field</td>
</tr>
</tbody>
</table>

Sample Request Body : JSON Request

```
POST v2.0/networks.json Content-Type: application/json
Accept: application/json

{ "Action": {
    "name": "Action1",
    "type": "WayPoint",
    "WayPoint_Action_field": "10.0.0.7,10.0.0.45",
}
}
```

2.3.1.3.3  Create Filter
Create a policy action to be used by DOVE policy
This operation requires a request body.
The request body must contain the following attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>NULL</td>
<td>Filter name</td>
</tr>
<tr>
<td>flow</td>
<td>String</td>
<td>NULL</td>
<td>Flow description</td>
</tr>
<tr>
<td>action</td>
<td>String</td>
<td>NULL</td>
<td>Drop or accept (may be extended)</td>
</tr>
</tbody>
</table>

Sample Request Body : JSON Request

```
POST v2.0/networks.json Content-Type: application/json
Accept: application/json

{ "Filter": {
    "name": "Filter1",
    "flow": "HTTP,FTP",
    "action": "DROP",
}
}
```
2.3.1.4 Mgmt Fabric APIs

2.3.1.4.1 Evacuate

Instance evacuation:
Here we will describe how to perform instance evacuation from the command line interface. Notice that this operation requires administrator token. As cloud administrator, while you are managing your cloud, you may get to the point where one of the cloud compute nodes fails, for example due to hardware malfunction. At that point you may use server evacuation in order to make managed instances available again.

With the information about instance configuration, like if it is running on shared storage, you can choose the required evacuation parameters for your case. Use the nova host-list command to list the hosts and find new host for the evacuated instance. In order to preserve user data on server disk, target host needs to have pre-configured shared storage with the source host. Also, you have to validate that the current vm host is down. Otherwise the evacuation will fail with error.

Instances can be evacuated either to a specified host or, if the target host parameter is not given, the target host will be chosen by nova scheduler automatically. In case the target host is chosen by the scheduler all the compute nodes in the environment should have same shared storage.

Evacuate server and preserve user data
This is relevant only if both the source host and the target host are on the same shared storage (e.g. volume mapping and/or NFS). In such a case user disk data are preserved during instance evacuation. In this scenario the password will remain unchanged.

Evacuate server without shared storage
Nova evacuate performs an instance evacuation from down host to specified (or scheduler selected) host. The instance will be booted from a new disk, but will preserve the configuration, e.g. id, name, uid, ip...etc. The new instance password can be passed to the command using the --password <pwd> option. If not given it will be generated and printed after the command finishes successfully.

```
$ nova evacuate [--host <host>] [--password <password>] [--on-shared-storage] <server>
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>evacuated_server_name</td>
<td>String</td>
<td>NULL</td>
<td>Server to be evacuated</td>
</tr>
<tr>
<td>host_b</td>
<td>String</td>
<td>NULL</td>
<td>Target compute node to rebuild the server on</td>
</tr>
</tbody>
</table>

```
$ nova evacuate evacuated_server_name host_b
```

The command returns a new server password.
The command returns a new server password.

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

The command returns a new server password.

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>

Using REST API:

```
```

Host evacuation:

In some cases, the administrator would prefer to evacuate an entire host. This can be done using nova host-evacuate command. It will attempt to evacuate each instance from the failed host either to a specified host or to one automatically selected by nova scheduler.

```
$ nova host-evacuate [--target_host <target_host>] [--on-shared-storage] <host>
```

the command returns a list of evacuated instances and evacuation status:

```
$ nova host-evacuate evacuated_server_name
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>adminPass</td>
<td>kRAJpErnT4xZ</td>
</tr>
</tbody>
</table>
2.3.1.5  **ResourceManager Capacity APIs : IBM Adaptive Utilization Accelerator for Virtualized Environments (IBM PULSAR)**

2.3.1.5.1  **IBM Pulsar APIs**

Pulsar provides a mechanism to monitor instances and identify idle ones. To allow safe resources over commit, idle instances' resources allocations are reduced to free space for the use of existing/new instances. Once an idle instance is back to work and request more resources - Pulsar identifies it and increase the instance's resources back to its reservation ones. Currently this mechanism deals only with CPU resources.

Pulsar composes of two periodic tasks, utilization_filter, and admission_control.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance_uuid</td>
<td>String</td>
<td>NULL</td>
<td>VM to be throttled/unthrottled</td>
</tr>
</tbody>
</table>

- throttle_instance(instance_uuid)
- unthrottle_instance(instance_uuid)

issues the throttle_instance or unthrottle_instance to change the throttling-state and set the throttle_ratio correspondingly

- detect_idleness() - Uses the metric system to identify idle VMs

For now the APIs are not exposed externally, but are part of internal mechanism.

2.3.2  **Useful APIs details**

2.3.2.1  **Group Affinity Filters**

When creating a VM, specify it's group with group scheduler_hints

```
$ nova boot --image cedef40a-ed67-4d10-800e-17455edce175 --flavor 1 \ 
   --hint group=foo server-1
```

By default, the group foo will be considered anti-affinity type. So in the above example, a VM named server-1 would not be placed with any VMs belonging to group foo. To specify affinity group, use affinity namespace (i.e. --hint group=affinity:foo). Notice that you could specify several affinity groups by stacking them in a list (i.e. --hint group=foo --hint group=foo1)
Or if you use the api:

```
{
    'server': {
        'name': 'server-1',
        'imageRef': 'cedef40a-ed67-4d10-800e-17455edce175',
        'flavorRef': '1'
    },
    'os:scheduler_hints': {
        'group': 'foo'
    }
}
```

```
{
    'server': {
        'name': 'server-1',
        'imageRef': 'cedef40a-ed67-4d10-800e-17455edce175',
        'flavorRef': '1'
    },
    'os:scheduler_hints': {
        'group': ['foo', 'foo1']
    }
}
```

### 2.3.3 Accessing DCRM from a Browser

To send http commands to DCRM using browser, use:

- Firefox RESTClient add-ons [5].
3 Object Storage - User and Programmers Guide

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

3.1 Introduction

Welcome to the User and Programmer Guide for the Object Storage Generic Enabler. This generic enabler is built on an Open Source project, the CDMI 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.

3.1.1.1 Background and Detail

This User and Programmers Guide relates to the Object Storage GE which is part of the Cloud Hosting chapter. Please find more information about this Generic Enabler in the following Open Specification.

3.2 User Guide

The Object Storage Generic Enabler is a back-end component that provides object storage capabilities which software developers can incorporate into their applications. It is not designed for direct manipulation by end users. However, as this generic enabler is built on top of OpenStack Swift, the OpenStack Horizon user interface can also be used to manipulate the Object Storage system. Detailed Horizon user interface instructions are outside the scope of this generic enabler, but comprehensive OpenStack documentation is available online[1].

3.3 Programmer Guide

3.3.1 Introduction

The Object Storage Generic Enabler exposes Object Storage functionality via a standard API: CDMI.

The complete CDMI specification is freely available online from the Storage Networking Industry Association (SNIA)[2].

CDMI is a RESTful API and so just like OCCI it is built on well-known HTTP operations such as PUT, POST, GET and DELETE. For example, to retrieve a data object from a CDMI container a simple HTTP GET command is used.

For example, the following request:

```
GET /MyContainer/MyDataObject.txt HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.0.1
X-Auth-Token: 0cc2bab32f3246919c2d2cbea314d850
```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.0.1
Content-Type: application/cdmi-object
{
    "objectType": "application/cdmi-object",
    "objectId": "0000706D0010B84FAD185C425D8B537E",
    "objecName": "MyDataObject.txt",
    "parentURI": "/MyContainer/",
    "parentID": "00007E7F00102E230ED82694DAA975D2",
    "domainURI": "/cdmi_domains/MyDomain/",
    "capabilitiesURI": "/cdmi_capabilities/dataobject/",
    "completionStatus": "Complete",
    "mimetype": "text/plain",
    "metadata": {
        "cdmi_size": "37"
    },
    "value": "This is the Value of this Data Object"
}

In this case the CDMI specific media-types are used. Others can be used too if the
HTTP mime-types are used appropriately.

HTTP commands can be constructed and sent from all modern programming
languages. They can also be invoked using a command line utility such as curl[3].

3.3.2 Authentication
A valid token is required to access an object store. This section describes how to get
a valid token assuming an identity management system compatible with OpenStack
Keystone is being used. If the username, password and tenant details are known,
only step 3 is required. If only the username and password are known then all three
steps must be executed in order.

The examples in this section point to the identity management system on the FI-LAB
infrastructure.

3.3.2.1 1. Authentication to get initial token
The response to the following command includes an initial token.

```bash
curl -d '{"auth": {"passwordCredentials":
    {"username":"email@company.com", "password":"mypassword"}}}' \ 
    -H 'Content-type: application/json'
```
The following bash script illustrates how the initial token can be retrieved and stored in a variable $token1.

```bash
username='email@company.com'
password='mypassword'
curl -d '{"auth": {"passwordCredentials":
    {"username": "$username", "password": "$password"}}}'
    -H 'Content-type: application/json'
    http://cloud.lab.fi-ware.org:4730/v2.0/tokens
> auth_token1.dat
token1=$(awk -F"[,:]" '{for(i=1;i<=NF;i++)
    {if($i~/id042/)
        {print $(i+1)}
    }
} auth_token1.dat | awk -F"' "'{print $2; exit}')
```

In python this can be achieved as follows.

```python
import httplib
import json
username='email@company.com'
password='mypassword'
conn = httplib.HTTPConnection('cloud.lab.fi-ware.org:4730')
headers = {'Content-Type': 'application/json'}
body = '{"auth": {"passwordCredentials": "username": '+username+'", "password": '+password+'"}}'
conn.request("POST", "/v2.0/tokens", body, headers)
response = conn.getresponse()
data = response.read()
datajson = json.loads(data)
token1 = datajson['access']['token']['id']
```

3.3.2.2 2. Use initial token to get tenant

The response to the following command includes tenant details.

```bash
curl -H 'x-auth-token: KUCmesWXA0QfEFf1Vb4AoCdx8ZBS9YzKQiqaCRUHNrxWNYKtgtyeLEGdstj1XyxFqGw4WbkjF3tw9_0_KLZA' 
    http://cloud.lab.fi-ware.org:4730/v2.0/tenants
```

The following bash script illustrates how the tenant details can be retrieved and stored in a variable $tenantName. The script assumes a variable $token includes a valid token - as per step one.

```bash
curl -H 'x-auth-token: "$token1 " \
    http://cloud.lab.fi-ware.org:4730/v2.0/tenants \n    > auth_tenant.dat
tenantName=$(awk -F"[,:]" '{for(i=1;i<=NF;i++)
    {if($i~/id042/)
    {print $(i+1)}
    }
}' auth_tenant.dat | awk -F"'" '{print $2; exit}')
```

In python this can be achieved as follows. The code assumes token1 includes a valid token - as per step one.

```python
headers = {'x-auth-token': token1}
conn.request("GET", "/v2.0/tenants", None, headers)
response = conn.getresponse()
data = response.read()
datajson = json.loads(data)
tenant = datajson['tenants'][0]['id']
```

### 3.3.2.3 Authenticate tenant to get token for Object Storage

The response to the following command includes the token to access the object store, and the string necessary to refer to the users object storage space.

```bash
curl -d '{"auth": {"passwordCredentials": {"username":"email@company.com", "password":"mypassword"},"tenantName":"000000000000000000000000000000150"}}' \n    -H 'Content-type: application/json' \n    http://cloud.lab.fi-ware.org:4730/v2.0/tokens \n    
```

The following bash script illustrates how the required token and auth string can be retrieved and stored in a variables $token and $auth. The script assumes variables $username, $password and $tenantName have been initialized as per step one and two.

```bash
curl -v \
    -d '{ "auth" :
    { "passwordCredentials" :
    { "username" : "$username" , "password" :
    "$password" },
    "tenantName" : "$tenantName" }'
In python this can be achieved as follows. The code assumes tenant is a valid tenant as per step two.

```python
headers = {'Content-Type': 'application/json'}
body = '{"auth": {"tenantName": "+tenant1", "passwordCredentials": {"username": "+username1", "password": "+password1"}}}'
conn.request("POST", "/v2.0/tokens", body, headers)
response = conn.getresponse()
data = response.read()
datajson = json.loads(data)
token = datajson['access']['token']['id']
for i in auth_response['access']['serviceCatalog']:
    if i['name'] == 'swift':
        auth_url = i['endpoints'][0]['publicURL']
        break
auth = auth_url[auth_url.find("AUTH"):]"}
```

### 3.3.3 Programming against CDMI

The CDMI interface is built upon the HTTP protocol and so can be manipulated by any programming language that can manipulate HTTP calls.

Once authenticated, the various CDMI commands as defined in the CDMI specification can be invoked. For example, the following Python code illustrates how to create a container:
conn = httplib.HTTPConnection('130.206.82.9:8080')
headers = {'X-Auth-Token': token,  
           'Content-Type': "application/cdmi-container",  
           'Accept': "application/cdmi-container",  
           'X-CDMI-Specification-Version': "1.0.1"}
conn.request('PUT', '/cdmi/' + auth + '/ContainerName/', None, headers)
response = conn.getresponse()
result = "Status: " + str(response.status) + ", Reason: " + response.reason + ", Body: " + response.read()
conn.close()

3.3.3.1 Example Python

The following python illustrates some basic operations against object storage.

```python
#!/usr/bin/env python

""
Example python to authenticate against an object store, create a new container, store text object, retrieve it, delete object, delete container.
"""

import httpplib
import json
import sys
import time
import os
import logging

# Init a simple logger...
logging.basicConfig(level=logging.INFO)
console = logging.StreamHandler()
console.setLevel(logging.DEBUG)
logger = logging.getLogger()
logger.addHandler(console)

# hosts
HOST_AUTH = 'cloud.lab.fi-ware.org:4730'
HOST_CDMI = '130.206.82.9:8080'
```
def authentication_request(username, password):
    """
    Request authentication of user
    """
    conn = httplib.HTTPConnection(HOST_AUTH)

    # retrieve initial token
    headers = {'Content-Type': 'application/json'}
    body = '{"auth": {"passwordCredentials":{"username": ""+username+"", "password": ""+password+""}}}'}
    conn.request("POST", "/v2.0/tokens", body, headers)
    response = conn.getresponse()
    data = response.read()
    datajson = json.loads(data)
    initialtoken = datajson['access']['token']['id']

    logger.info('Initial Token is: ' + initialtoken)

    # retrieve tenant
    headers = {'x-auth-token': initialtoken}
    conn.request("GET", "/v2.0/tenants", None, headers)
    response = conn.getresponse()
    data = response.read()
    datajson = json.loads(data)
    tenant = datajson['tenants'][0]['id']

    logger.info('Tenant is: ' + tenant)

    # retrieve authentication json
    headers = {'Content-Type': 'application/json'}
    body = '{"auth": {"tenantName": "+tenant+"", "passwordCredentials":{"username": "+username+"", "password": "+password+""}}}'}
    conn.request("POST", "/v2.0/tokens", body, headers)
    response = conn.getresponse()
data = response.read()

return json.loads(data)

def cdmi_request(verb, resource, headers, body):
    '''
    Do a HTTP request defined by HTTP verb, a Url, a dict of
    headers and a body.
    '''
    conn = httplib.HTTPConnection(HOST_CDMI)
    conn.request(verb, '/cdmi/' + resource, body, headers)
    response = conn.getresponse()

    if response.status not in [200, 201, 202, 204]:
        logger.error(response.reason)
        logger.warn(response.read())
        sys.exit(1)

    result = 'Status: ' + str(response.status) + ' , Reason: ' + response.reason + ' , Body: ' + response.read()

    conn.close()

    return result

def check_capabilities(token, auth):
    headers = {
        "X-Auth-Token": token,
        "Accept": "application/cdmi-capability",
        "X-CDMI-Specification-Version": "1.0.1"}
    body = None
    url = auth + '/cdmi_capabilities/

    return cdmi_request('GET', url, headers, body)

def create_container(token, auth, name):
headers = {
    "X-Auth-Token": token,
    "Content-Type": "application/cdmi-container",
    "Accept": "application/cdmi-container",
    "X-CDMI-Specification-Version": "1.0.1"
}
body = None
url = auth + "/" + name + "/

return cdmi_request('PUT', url, headers, body)

def list_container(token, auth, name):
    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/cdmi-container",
        "Accept": "*/*",
        "X-CDMI-Specification-Version": "1.0.1"
    }
    body = None
    url = auth + "/" + name + "/

    return cdmi_request('GET', url, headers, body)

def store_text(token, auth, container_name, object_name, object_text):
    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/cdmi-object",
        "Accept": "application/cdmi-object",
        "X-CDMI-Specification-Version": "1.0.1"
    }
    body = '{"mimetype":"text/plain", "metadata":{}, "value": "" + object_text + '"'}
    url = auth + "/" + container_name + "/" + object_name

    return cdmi_request('PUT', url, headers, body)

def retrieve_text(token, auth, container_name, object_name):
    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/cdmi-object",
        "Accept": "*/*",
        "X-CDMI-Specification-Version": "1.0.1"
    }

"X-CDMI-Specification-Version": "1.0.1"

body = None
url = auth + "/" + container_name + "/" + object_name

return cdm_request('GET', url, headers, body)

def delete_object(token, auth, container_name, object_name):
    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/cdmi-object",
        "X-CDMI-Specification-Version": "1.0.1"
    }
    body = None
    url = auth + "/" + container_name + "/" + object_name

    return cdm_request('DELETE', url, headers, body)

def delete_container(token, auth, container_name):
    headers = {
        "X-Auth-Token": token,
        "Content-Type": "application/cdmi-container",
        "X-CDMI-Specification-Version": "1.0.1"
    }
    body = None
    url = auth + "/" + container_name

    return cdm_request('DELETE', url, headers, body)

if __name__ == '__main__':
    if len(sys.argv) < 3:
        print 'Usage: cdmi_demo.py <username> <password>'
        sys.exit(128)

    username = sys.argv[1]
    password = sys.argv[2]

    # display basic info
    logger.info('CDMI host is: ' + HOST_CDMI)
    logger.info('Authorisation host is: ' + HOST_AUTH)
```python
# get authentication response
auth_response = authentication_request(username, password)

# extract token
token = auth_response['access']['token']['id']
logger.info('Security token is: ' + token)

# extract authentication string required for addressing users resources
for i in auth_response['access']['serviceCatalog']:
    if i['name'] == 'swift':
        auth_url = i['endpoints'][0]['publicURL']
        break

auth = auth_url[auth_url.find("AUTH_"):]  
logger.info('Authentication string is: ' + auth)

# perform some basic Object Store operations
response = check_capabilities(token, auth)
logger.info('Object Store capabilities: ' + response)

response = create_container(token, auth, TEST_CONTAINER_NAME)
logger.info('Create Container Response: ' + response)

response = list_container(token, auth, TEST_CONTAINER_NAME)
logger.info('List Container Response: ' + response)

response = store_text(token, auth, TEST_CONTAINER_NAME, TEST_OBJECT_NAME, TEST_TEXT)
logger.info('Store Text Response: ' + response)

response = list_container(token, auth, TEST_CONTAINER_NAME)
logger.info('List Container Response: ' + response)
```
3.3.4 Command Line Interface to CDMI

For ad-hoc manipulation of the Object Storage generic enabler the command line interface can be used. The curl tool available for many operating systems allows HTTP commands to be sent and received from the command line.

3.3.4.1 Individual Commands

The following examples in curl illustrate some basic object storage manipulation commands. The examples assume that three variables are assigned containing appropriate values for the IP address of the object storage CDMI interface, a valid object storage token, and the auth string to help construct the users object storage url. The token and auth variables are populated during the authentication process.

node_cdmi='130.206.82.9'
node_cdmi=130.206.82.9

node_cdmi='130.206.82.9'
token='730ff2774823b49e3916e99cd4640'
auth='AUTH_00000000000000000000000000000150'

3.3.4.1.1 List Capabilities of the Object Store

```bash
curl -v \
   -X GET \
   -H 'X-Auth-Token: '$token \n   -H 'Accept: application/cdmi-capability' \
   -H 'X-CDMI-Specification-Version: 1.0.1' \n   http://$node_cdmi:8080/cdmi/$auth/cdmi_capabilities/
```

3.3.4.1.2 Create a container

```bash
curl -v -X PUT \
   -H 'X-Auth-Token: '$token \
```
3.3.4.1.3 **List capabilities of the container**

```bash
curl -v
-X GET
-H 'X-Auth-Token: '$token
-H 'Accept: application/cdmi-capability'
-H 'X-CDMI-Specification-Version: 1.0.1'

http://$node_cdmi:8080/cdmi/$auth/cdmi_capabilities/container/mynewcontainer/
```

3.3.4.1.4 **Store simple text in a container**

```bash
curl -v -X PUT
-H 'X-Auth-Token: '$token
-H 'Accept: application/cdmi-object'
-H 'Content-Type: application/cdmi-object'
-d '{"mimetype":"text/plain", "metadata":{}, "value": "Hello CDMI World"}'

http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/mydata
```

3.3.4.1.5 **List contents of a container**

```bash
curl -v
-X GET
-H 'X-Auth-Token: '$token
-H 'Content-Type: application/cdmi-container'
-H 'Accept: */*
-H 'X-CDMI-Specification-Version: 1.0.1'

http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/
```

3.3.4.1.6 **Retrieve object from a container**

```bash
curl -v
-X GET
-H 'X-Auth-Token: '$token
-H 'Accept: application/cdmi-object'
-H 'X-CDMI-Specification-Version: 1.0.1'

http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/mydata
```

3.3.4.1.7 **Retrieve only the value of an object from a container**

```bash
curl -v
```
3.3.4.1.8  *Delete an object from the object store*

```bash
curl -v \
-X DELETE \ 
-H 'X-Auth-Token: '$token \
-H 'X-CDMI-Specification-Version: 1.0.1' \ 
http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/mydata
```

3.3.4.1.9  *Store a file on the object store*

```bash
myobject='CDMI_object_test_data.dat'
curl -v \
-X PUT \ 
-H 'X-Auth-Token: '$token \ 
-H 'Content-Type: application/stream-octet' \ 
-H 'Accept: */*' \ 
--data-binary "@$myobject" \ 
http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/mybinaryobject
```

3.3.4.1.10  *Retrieve file from object store*

```bash
myobjectreceived='CDMI_object_test_data_received.dat'
curl -X GET \ 
-H 'X-Auth-Token: '$token \ 
http://$node_cdmi:8080/cdmi/$auth/mynewcontainer/mybinaryobject \ 
--output $myobjectreceived
```

3.3.4.2  **Example Script**

The following bash script illustrates curl commands that authenticate and perform basic manipulation of the object store. It assumes there is a file called CDMI_object_test_data.dat in the local directory that contains some data to store.

```bash
#!/bin/bash
#
# Simple script to illustrate accessing CDMI using curl
#
# Basic flow is:
```
# Retrieve token
# Check container 'capabilities'
# Create a container, check it
# Put an object in the container, read it, delete it - simple data
# Put an object in the container, read it, delete it - user supplied data file
# Delete the container
#
# There are some pauses in case the user wants to check container using the web
# portal
#
# -------------------
#Configuration settings
#node_auth='cloud.lab.fi-ware.org'
node_cdmi='130.206.82.9'
mycontainer=CDMI_TEST
mydata='HelloWorld.txt'
myobject='CDMI_object_test_data.dat'
myobjectreceived='CDMI_object_test_data_received.dat'
#
# Prompt user for account details
#
read -p "Please enter your username (typically an email address): " username
read -s -p "Please enter your password: " password
#
# Initialisation check
#
echo " "
echo " "
if [ -f $myobject ]
then
echo "Initialisation check complete"
else
  echo "Please create a file " $myobject " containing some test content."
  exit
fi
#
#
# Note .. assume write access to local directory
#
# Get initial token for the username/password combination
#
echo " 

echo " 
curl -d '{"auth": {"passwordCredentials":{"username":"$username", "password":"$password"}}}'} "
  -H 'Content-type: application/json' 
  http://$node_auth:4730/v2.0/tokens
>

#
token1=$(awk -F"[:]," '{for(i=1;i<=NF;i++)
  {if($i~/id\042/)
    {print $(i+1)}
  }
}' auth_token1.dat | awk -F"" '{print $2; exit}')
#
echo " 
  echo " 

echo "Initial token: $token1"
#
# Now get a valid tenantName for this token
#
echo " 
  echo " 

curl -H 'x-auth-token: '$token1 "
  http://$node_auth:4730/v2.0/tenants
>

tenantName=$(awk -F"[\,:]" '{for(i=1;i<=NF;i++)
    if($i~/id\042/)
        print $(i+1)}
} auth_tenant.dat | awk -F"'" '{print $2; exit}')

#
echo " "
echo " "
echo "Tenant name: $tenantName"
#
# Now get valid token for user/password/tenant combination
#
echo " "
echo " "
curl -v \
    -d '{ "auth" :
        { "passwordCredentials" :
            { "username" : "$username" , "password" : "$password" },
                "tenantName" : "$tenantName" } }
} ' http://$node_auth:4730/v2.0/tokens
>
auth_token2.dat

#
token=$(awk -F"[\,:]" '{for(i=1;i<=NF;i++)
    if($i~/id\042/)
        print $(i+1)}
} auth_token2.dat | awk -F"'" '{print $2; exit}')

auth=$(awk -F"[\,:]" '{for(i=1;i<=NF;i++)
    if($i~/publicURL\042/)
        print $(i+3)}
} auth_token2.dat | \
}
```bash
  grep "v1/AUTH" | awk -F'"'}' '{print $1;}' | awk -F"/
' '{print $3;}'
#
# We need both token and auth for access to the CDMI proxy
#
  echo " "
  echo " "
  echo "TOKEN is : $token"
  echo "AUTH is : $auth"

  export token auth
  echo " "
  echo " "
  echo "Authentication phase complete. Press Enter to continue (or Ctrl-C to abort)."

  read
  echo " "
  echo " "

# # Now use acquired info to enquire the cdmi capabilities
#
  echo "*** Enquire CDMI Capabilities"

  curl -v \
    -X GET \
    -H 'X-Auth-Token: '$token \
    -H 'Accept: application/cdmi-capability' \
    -H 'X-CDMI-Specification-Version: 1.0.1' \
    http://$node_cdmi:8080/cdmi/$auth/cdmicapabilities/

#
#
  echo " "
  echo " "

  echo "*** Create a new Container"

  curl -v -X PUT \
    -H 'X-Auth-Token: '$token \
    -H 'Content-Type: application/cdmi-container' \
    -H 'Accept: application/cdmi-container' \
    -d '{"metadata": {}}' \
    http://$node_cdmi:8080/cdmi/$auth/$mycontainer/
```
```bash
# echo " "
echo " "
echo "*** Check the 'capabilities' on the container"
curl -v \
   -X GET \
   -H 'X-Auth-Token: '$token \n   -H 'Accept: application/cdmi-capability' \
   -H 'X-CDMI-Specification-Version: 1.0.1' \
   http://$node_cdmi:8080/cdmi/$auth/cdmi_capabilities/container/
$mycontainer/
#
#
echo " "
echo " "
echo "*** Place a Data Object in the container"
curl -v -X PUT \
   -H 'X-Auth-Token: '$token \n   -H 'Accept: application/cdmi-object' \
   -H 'Content-Type: application/cdmi-object' \
   -d '{"mimetype":"text/plain", "metadata":{}, "value": "Hello CDMI World"}' \
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/$mydata
#
#
echo " "
echo " "
echo "*** List Contents of a container"
curl -v \
   -X GET \
   -H 'X-Auth-Token: '$token \n   -H 'Content-Type: application/cdmi-container' \
   -H 'Accept: */*' \
   -H 'X-CDMI-Specification-Version: 1.0.1' \
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/
#
#```
Future Internet Core Platform

```
echo " "
echo " "
echo -n "There should now be an object in the container. Press Enter to continue."
read # #
echo " "
echo " "
echo " *** Read Contents of Data Object"

curl -v \
-X GET \
# #
echo " "
echo " "
echo " *** Read only the Value of a Data Object"

curl -v \
-X GET \
-H 'X-Auth-Token: '$token \nhttp://$node_cdmi:8080/cdmi/$auth/$mycontainer/$mydata
# #
echo " "
echo " "
echo " *** Delete the Data Object"

curl -v \
-X DELETE \
# #
```
```bash
echo " "
echo " "
echo "*** List Contents of a container"
curl -v \
   -X GET \
   -H 'X-Auth-Token: "$token" \
   -H 'Content-Type: application/cdmi-container' \
   -H 'Accept: */*' \
   -H 'X-CDMI-Specification-Version: 1.0.1' \
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/
#
#
echo " "
echo " "
echo " "
read
#
#
echo " "
echo " "

** Copy a file up to the Data Object**
curl -v \
   -X PUT \
   -H 'X-Auth-Token: "$token" \
   -H 'Content-Type: application/stream-octet' \
   -H 'Accept: */*' \
   --data-binary "@$myobject" \
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/$myobject
#
#
echo " "
echo " "
echo " "

*** List Contents of a container"
curl -v \
   -X GET \
   -H 'X-Auth-Token: "$token" \
   -H 'Content-Type: application/cdmi-container' \
```
-H 'Accept: */*
-H 'X-CDMI-Specification-Version: 1.0.1'
http://$node_cdmi:8080/cdmi/$auth/$mycontainer/

#
#
echo " "
echo " "
echo -n "There should now be an object in the container.
Press Enter to continue."
read
#
#
echo " "
echo " "
echo "*** Read Data back from Data Object"
curl -X GET \
   -H 'X-Auth-Token: '$token \
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/$myobject \
   --output $myobjectreceived
#
#
echo " "
echo " "
echo "*** Delete the Data Object, tidy up"
curl -v \
   -X DELETE \
   -H 'X-Auth-Token: '$token \
   -H 'Content-Type: application/cdmi-object' \
   -H 'X-CDMI-Specification-Version: 1.0.1'
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/$myobject
#
#
echo " "
echo " "
echo "*** Delete the Container, tidy up"
curl -v \
   -X DELETE \
   -H 'X-CDMI-Specification-Version: 1.0.1'
   http://$node_cdmi:8080/cdmi/$auth/$mycontainer/$mycontainer
Future Internet Core Platform

```
#
#
echo " "
echo "*** Done"
exit
```

3.4 References

1. ↑ [http://docs.openstack.org/developer/horizon/](http://docs.openstack.org/developer/horizon/)
2. ↑ [http://www.snia.org/cdmi](http://www.snia.org/cdmi)
3. ↑ [http://curl.haxx.se/](http://curl.haxx.se/)
4 PaaS Management - User and Programmers Guide

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

4.1 Introduction

Welcome the User and Programmer Guide for the PaaS Manager GE. This generic enabler is built on a proprietary solution using standard interface to communicate with and so where possible this guide points to the appropriate online content that has been created for this specific API. 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 using this interface.

4.1.1.1 Background and Detail

This User and Programmers Guide relates to the Scalability Manager GE which is part of the Cloud Hosting Chapter. Please find more information about this Generic Enabler in the following Open Specification.

4.2 User Guide

This GE does not provide a GUI, it is for the use of developers who will invoke it programmatically.

4.3 Programmer Guide

The PaaS API reference offers a [1] page, which describes how to use this API in detail. It includes a set of commands, which can be used to manipulate instance in the Cloud.

PaaS Manager API is based upon HTTP and therefore all devices, which can handle HTTP traffic, are possible clients. This also means that most programming languages can be used to access PaaS Manager through PaaS Manager API.

To give a feeling of how PaaS Manager work lets take a look at an HTTP request and the corresponding response:

* About to connect() to 130.206.80.112 port 8080 (#0)
* Trying 130.206.80.112... connected
* Connected to 130.206.80.112 (130.206.80.112) port 8080 (#0)
> GET /paasmanager/rest/catalog/environment HTTP/1.1
> User-Agent: curl/7.21.1 (i686-pc-mingw32) libcurl/7.21.1 OpenSSL/0.9.8r zlib/1.2.3
> Host: 130.206.80.112:8080
> Accept: */*
> Access-Control-Request-Method: GET
> X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb
>
In this case the HTTP GET operation is used. PaaS Manager API uses the Create Retrieve Update Delete (CRUD) operations, which map almost to the HTTP verbs POST, GET, PUT and DELETE.

PaaS Manager API deals with two media types, application/xml and application/json, both Content-Type and Accept, which means that we can send a xml content and receive and json response and vice versa. For simplicity, only XML examples are going to be provided in this page.

4.3.1 Accessing Service Manager from the CLI

The access through the CLI is made using the curl program. Curl [2] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, Gopher, Dict, Telnet, LDAP or File) and therefore is also usable for OpenStack Compute API. Use the curl command line tool or use libcurl from within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

The normal operations sequence to deploying an environment and an application on top of it could be summarized in the following list:

Some operations related to the management are the software can involve:

1.- Get the environment list from the catalogue

```
```

Obtaining:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environmentDtoes>
  <environment>
    <environmentType>
      <id>24</id>
      <name>java_web_server</name>
  </environmentType>
</environment>
</environmentDtoes>
```
2. - Deploying an environment

The following request URL for deploying an environment including hardware resources (servers) as well as the products required. The environment is composed by a set of tiers which correspond for server. The number of servers to be deployed is specified in the payload. In addition, the products to be installed in those servers. The server and product features to be deployed are specified in the payload. In this case, information about the IP or its identification are specified in the server element. In addition, the product (in this case tomcat) and the version is indicated are specified in the payload request.

```
```

Obtaining:
The result for provisioning the environment is a Task indicating the operation status:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task
href="http://130.206.80.112:8080/paasmanager/rest/org/{org-id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-08T09:13:18.311+01:00" status="RUNNING"/>
```
With the URL obtained in the href in the Task, it is possible to monitor the operation status. Once the environment has been deployed, the task status should be SUCCESS.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><task href="http://130.206.80.112:8080/paasmanager/rest/org/{org-id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-08T09:13:19.567+01:00" status="SUCCESS">
  <description>Deploy environment {environment-name}</description>
  <vdc>{vdc-id}</vdc>
</task>
```

3.- Get information about the installed environment

```
```

With the following result:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><environmentInstanceDtoes>
  <environmentInstance>
    <environmentInstanceName>java_tomat_instance</environmentInstanceName>
    <vdc>{vdc-id}</vdc>
    <environment>
      <name>java_tomat</name>
      <tiers>
        <initial_number_instances>1</initial_number_instances>
        <maximum_number_instances>1</maximum_number_instances>
        <minimum_number_instances>1</minimum_number_instances>
      </tiers>
    </environment>
  </environmentInstance>
</environmentInstanceDtoes>
```
<name>tomcat_tier</name>
<productReleases>
  <product>tomcat</product>
  <version>7.0</version>
  <description>Tomcat 7.0</description>
  <productType>
    <id>6</id>
    <name>ApplicationWebServer</name>
    <description>Application Web Server</description>
  </productType>
</productReleases>
</tiers>
</environment>
<tierInstances>
  <id>138</id>
  <date>2012-11-29T13:03:36.056+01:00</date>
  <status>INSTALLED</status>
  <tier>
    <name>tomcat</name>
    <productReleases>
      <product>test</product>
      <version>0.1</version>
    </productReleases>
  </tier>
  <currentNumberInstances>1</currentNumberInstances>
</tierInstances>

<fgn>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.replicas.1</fgn>

  <name>java_tomat_instance_tomcat</name>
  <productInstances>
    <date>2012-11-29T13:03:27.834+01:00</date>
  </productInstances>
</tierInstances>

<name>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.replicas.1.tomcat_7.0</name>
  <status>INSTALLED</status>
  <productRelease>
    <product>tomcat</product>
    <version>7.0</version>
  </productRelease>
4.- Deploying an application in an environment already installed

From an environment instance already installed (called {environment-id}), it is possible to deploy an application (including all its artefacts).


Once installed, it is possible to query it, and check that the version has been updated.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<applicationReleaseDto>
    <applicationName>warapplication</applicationName>
    <version>1.0</version>
    <artifacts>
        <artifact>
            <name>thewarfile</name>
            <attributes>
                <key>webapps_url</key><value>http://Artefacts/WAR/tomcatFixedLocalPostgresDB/flipper.war</value>
                <key>webapps_name</key><value>flipper.war</value>
                <key>type</key><value>war</value>
                <productRelease>
                    <version>7.0</version>
                    <product>tomcat</product>
                </productRelease>
            </attributes>
        </artifact>
    </artifacts>
</applicationReleaseDto>"
<artifact>
  <name>the properties file</name>
  <attributes>
    <key>properties_url</key><value>http://configuration.properties</value>
    <key>properties_name</key><value>flipper.properties</value>
  </attributes>
</artifact>

The result for provisioning the application is a Task indicating the operation status:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="http://130.206.80.112:8080/paasmanager/rest/org/{org-id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-08T09:13:18.311+01:00" status="RUNNING">
  <description>Deploy application {application-name}</description>
  <vdc>{vdc-id}</vdc>
</task>
```

With the URL obtained in the href in the Task, it is possible to monitor the operation status. Once the application has been deployed, the task status should be SUCCESS.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<task href="http://130.206.80.112:8080/paasmanager/rest/org/{org-id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-08T09:13:19.567+01:00" status="SUCCESS">
  <description>Deploy application {application-name}</description>
  <vdc>{vdc-id}</vdc>
</task>
```

5.- Get information about an application already deployed

It obtains the information of the application already deployed.

Obtaining:

```xml
<applicationInstances>
  <id>206</id>
  <date>2012-12-03T08:42:21.294+01:00</date>
  <name>warapplication_instance</name>
  <status>INSTALLED</status>  
  <vdc>{vdc-id}</vdc>
  <applicationName>warapplication</applicationName>
  <version>1.0</version>
  <artifacts>
    <artifact>
      <name>thewarfile</name>
      <attributes>
        <key>webapps_url</key><value>http://Artefacts/WAR/tomcatFixedLocalPostgresDB/flipper.war</value>
        <key>webapps_name</key><value>flipper.war</value>
        <key>type</key><value>war</value>
        <productRelease>
          <version>7.0</version>
          <product>tomcat</product>
        </productRelease>
      </attributes>
    </artifact>
    <artifact>
      <name>the properties file</name>
      <attributes>
        <key>properties_url</key><value>http://configuration.properties</value>
        <key>properties_name</key><value>flipper..prperties</value>
      </attributes>
    </artifact>
  </artifacts>
</applicationInstances>
```
<environmentInstance>
  <environmentInstanceName>java_tomat_instance</environmentInstanceName>
  <vdc>{vdc-id}</vdc>
  <environment>
    <name>java_tomat</name>
    <tiers>
      <initial_number_instances>1</initial_number_instances>
      <maximum_number_instances>1</maximum_number_instances>
      <minimum_number_instances>1</minimum_number_instances>
      <name>tomcat_tier</name>
      <productReleases>
        <product>tomcat</product>
        <version>7.0</version>
        <description>Tomcat 7.0</description>
        <productType>
          <id>6</id>
          <name>ApplicationWebServer</name>
          <description>Application Web Server</description>
        </productType>
      </productReleases>
    </tiers>
  </environment>
  <tierInstances>
    <id>138</id>
    <date>2012-11-29T13:03:36.056+01:00</date>
    <status>INSTALLED</status>
    <tier>
      <name>tomcat</name>
      <productReleases>
        <product>test</product>
      </productReleases>
    </tier>
  </tierInstances>
</environmentInstance>
4.3.2 Accessing Service Manager from a Browser

We are using the Chrome browser [3] with the Simple REST Client plugin[4] in order to send http commands to the PaaS Manager. You can use it also in Firefox through RESTClient add-ons [5].

We follow the same sequence that we take previously in order to deploy environments and applications:

1.- Deploying an environment
With the URL obtained in the href in the Task, it is possible to monitor the operation status. Once the environment has been deployed, the task status should be SUCCESS.

2.- Get information about the installed environment

3.- Deploying an application in an environment already installed
4.- Get information about an application already deployed
5 Software Deployment And Configuration - User and Programmers Guide

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

5.1 Introduction

Welcome the User and Programmer Guide for the SDC Manager Generic Enabler. This generic enabler is built on a proprietary solution using standard interface to communicate with and so where possible this guide points to the appropriate online content that has been created for this specific API. 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 using this interface.

5.1.1.1 Background and Detail

This User and Programmers Guide relates to the Scalability Manager GE which is part of the Cloud Hosting Chapter. Please find more information about this Generic Enabler in the following Open Specification.

5.2 User Guide

This GE does not provide a GUI, it is for the use of developers who will invoke it programmatically.

5.3 Programmer Guide

The SDC API reference offers a [1] page, which describes how to use this API in details. It includes a bunch of commands, which can be used to manage software in the Cloud. Since SDC API is based on the HTTP protocol there are a rich variety of tools, which can be used to access the PaaS Manager interface.

SDC API is based upon HTTP and therefore all devices, which can handle HTTP traffic, are possible clients. This also means that most programming languages can be used to access SDC through SDC API.

To give a feeling of how it work lets take a look at an HTTP request and the corresponding response:

```
* About to connect() to 130.206.80.112 port 8080 (#0)
* Trying 130.206.80.112... connected
* Connected to 130.206.80.112 (130.206.80.112) port 8080 (#0)
> GET /sdc/rest/catalog/product HTTP/1.1
> User-Agent: curl/7.21.1 (i686-pc-mingw32) libcurl/7.21.1 OpenSSL/0.9.8r zlib/1.2.3
> Host: 130.206.80.112:8080
> Access-Control-Request-Method: GET
> Origin: http://130.206.80.93
> Content-Type: application/xml
```

D.4.2.3 FI-WARE User and Programmer Guide Page 52
In this case the HTTP GET operation is used. SDC API uses the Create Retrieve Update Delete (CRUD) operations, which map almost to the HTTP verbs POST, GET, PUT and DELETE.

SDC API deal with two media types, application/xml and application/json, both Content-Type and Accept, which means that we can send a xml content and receive and json response and vice versa.

5.3.1 Accessing Service Deployment and Configuration from the CLI

The access through the CLI is made using the curl program. Curl [2] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, Gopher, Dict, Telnet, LDAP or File) and therefore is also usable for SDC API. Use the curl command line tool or use libcurl from within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

Some operations related to the management are the software can involve:

1.- Get the product list from the catalogue


Obtaining:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<products>
  <product>
    <name>tomcat</name>
    <description>tomcat J2EE container</description>
  </product>
</products>
```

2.- Installing a product in a VM

The following request URL for deploying software. The VM and product features to be deployed are specified in the payload. In this case, information about the IP or its identification are specified in the VM element. In addition, the product (in this case tomcat) and the version is indicated are specified in the payload request.

```xml
<productInstanceDto>
  <vm>
    <ip>{NODE_IP}</ip>
    <fqn>fqn</fqn>
  </vm>
  <product>
    <productDescription/>
    <productName>tomcat</productName>
    <version>6</version>
  </product>
</productInstanceDto>
```

After that, it is possible to check that the product has been deployed, concretely in this case by [http://{NODE_IP}:8080](http://{NODE_IP}:8080).

### 3.- Get information about the product installed


With the following result:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<productInstance>
  <id>{PRODUCT_ID}</id>
  <date>2012-11-07T15:56:30.590+01:00</date>
  <status>INSTALLED</status>
  <vm>
    <ip>130.206.80.114</ip>
    <hostname>rhel-5200ee66c6</hostname>
    <fqn>fqn</fqn>
    <osType>95</osType>
  </vm>
  <vdc>{vdc-id}</vdc>
</productInstance>
```
4.- Updating the product version in the VM

From a product instance already installed (called {PRODUCT_ID}), it is possible to update the product version (in this case to update Tomcat from version 6 to 7).

```bash
```

Once installed, it is possible to query it, and check that the version has been updated.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<productInstance>
  <id>{PRODUCT_ID}</id>
  <date>2012-11-07T15:56:30.590+01:00</date>
  <status>INSTALLED</status>
  <vm>
    <ip>130.206.80.114</ip>
    <hostname>rhel-5200ee66c6</hostname>
    <fqdn>fqn</fqdn>
    <osType>95</osType>
  </vm>
  <vdc>{vdc-id}</vdc>
  <product>
    <releaseNotes>Tomcat server 7</releaseNotes>
    <version>7</version>
  </product>
</productInstance>
```
5.- Reconfigure a product already installed
From a product instance already installed (called \{PRODUCT_ID\}), it is possible to update its attributes (like the port where the tomcat is deployed).

```
> <attributes>
  > <key>port</key>
  > <value>8082</value>
  > <description>The listen port</description>
</attributes>
```
After that, it is possible to check that the product has been deployed, concretely in this case by \http://\{NODE_IP\}::8082\n
5.3.2 Accessing Service Manager from a Browser
We are using the Chrome browser [3] with the Simple REST Client plugin[4] in order to send http commands to the SDC. You can use it also in Firefox through RESTClient add-ons [5].

We follow the same sequence that we take previously in order to install software:
1.- Deploying an environment
2.- Get information about the product installed

3.- Updating the product version in the VM
4. Reconfigure a product already installed

5. Uninstall product
6 Policy Manager - User and Programmers Guide

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

6.1 Introduction

Welcome the User and Programmer Guide for the Policy Manager Generic Enabler. 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 using this interface.

6.1.1 Background and Detail

This User and Programmers Guide relates to the Policy Manager GE which is part of the Cloud Hosting Chapter. Please find more information about this Generic Enabler in the following Open Specification.

6.2 User Guide

The Policy Manager GE is a backend component, without user interface. Therefore there is no need to provide a user guide. The Cloud Portal can be used for Web-based interaction (but it is not part of this GE).

6.3 Programmer Guide

Policy Manager API is based upon HTTP and therefore all devices, which can handle HTTP traffic, are possible clients.

6.3.1 Accessing Policy Manager from the CLI

To invoke the REST API use the curl program. Curl [1] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, GOPHER, DICT, TELNET, LDAP or FILE) and therefore is also usable for Policy Manager API. Use the curl command line tool or use libcurl from within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

In order to make a probe of the different functionalities related to the Policy Manager, we make a list of several operations to make a probe of the execution of these GEis.

1. Get a valid token for the tenant that we have (It is not a Policy Manager operation but a IdM operation).

Due to all operations of the Policy Manager are using the security mechanism which is used in the rest of the cloud component, it is needed to provide a security token in order to continue with the rest of operations.
Both $TENNANT (Project), $USERNAME and $PASSWORD must be values previously created in the OpenStack Keystone. The IP address 10.95.171.115 and the Port 35357 are the data of our internal installation of IdM, if you planned to execute it you must changed it by the corresponding IP and Port of the FIWARE Keystone or IdM IP and Port addresses.

We obtained two data from the previous sentence:

- X-Auth-Token

  `<token expires="2012-10-25T16:35:42Z" id="a9a861db6276414094bc1567f664084d">`

- Tenant-Id

  `<tenant enabled="true" id="c907498615b7456a9513500fe24101e0" name=$TENNANT>`

### 2. Get tenant information

This is the first real operation about our GEi, by which we can obtain the information about the Policy Manager, together with the information about the window size fixed for the execution of the GEi. For more information about the window size and its meaning.

```
curl -v -H 'X-Auth-Token: a9a861db6276414094bc1567f664084d' -X GET http://130.206.81.71:8000/v1.0/c907498615b7456a9513500fe24101e0
```

This operation will return the information regarding the tenant details of the execution of the Policy Manager

```
< HTTP/1.0 200 OK
< Date: Wed, 09 Apr 2014 08:25:17 GMT
< Server: WSGIServer/0.1 Python/2.6.6
< Content-Type: text/html; charset=utf-8
{
  "owner": "Telefonica I+D",
  "runningfrom": "14/04/09 07:45:22",
```
3. Create a rule for a server

This operation allows to create a specific rule associate to a server:

```
curl -v -H 'X-Auth-Token: 86e096cd4de5490296fd647e21b7f0b4' -X POST http://130.206.81.71:8000/v1.0/6571e342ad84f7d828ce2f30373b3d4/servers/32c23ac4-230d-42b6-81f2-db9bd7e5b790/rules/ -d '{"action": {"actionName": "notify-scale", "operation": "scaleUp"}, "name": "ScaleUpRule", "condition": { "cpu": { "value": 98, "operand": "greater" }, "mem": { "value": 95, "operand": "greater equal"}}}'
```

The result of this operation is the following content:

```
< HTTP/1.0 200 OK
< Date: Wed, 09 Apr 2014 10:14:11 GMT
< Server: WSGIServer/0.1 Python/2.6.6
< Content-Type: text/html; charset=utf-8
{
    "serverId": "32c23ac4-230d-42b6-81f2-db9bd7e5b790",
    "ruleId": "68ed416-bfc6-11e3-a8b9-fa163e202949"
}
```

4. Subscribe the server to the rule

Through this operation we can subscribe a rule to be monitored in order to evaluate the rule to be processed.

```
curl -v -H 'X-Auth-Token: a9a861db6276414094bc1567f664084d' -X POST http://130.206.81.71:8000/v1.0/6571e342ad84f7d828ce2f30373b3d4/servers/32c23ac4-230d-42b6-81f2-db9bd7e5b790/subscription -d '{ "ruleId": "ruleid", "url": "URL to notify any action" }'
```

An the expected result is the following.

```
< HTTP/1.0 200 OK
< Date: Wed, 09 Apr 2014 10:16:11 GMT
< Server: WSGIServer/0.1 Python/2.6.6
< Content-Type: text/html; charset=utf-8
```
5. Manual simulation of data transmission to the server

This operation simulate the operation that the context broker used to send data to the Policy Manager, the normal execution of this process will be automatically once that the Policy Manager subscribes a rule to a specific server. The operation is the following:

curl -v -H "Content-Type: application/json" -X POST 
http://127.0.0.1:5000/v1.0/6571e3422ad84f7d828ce2f30373b3d4/servers/serverI1 -d '{
>     "contextResponses": [
>         {
>             "contextElement": {
>                 "attributes": [
>                     {
>                         "contextValue": "6",
>                         "name": "users",
>                         "type": "string"
>                     },
>                     {
>                         "contextValue": "1",
>                         "name": "usedMemPct",
>                         "type": "string"
>                     },
>                     {
>                         "contextValue": "0.14",
>                         "name": "cpuLoadPct",
>                         "type": "string"
>                     },
>                     {
>                         "contextValue": "0.856240",
>                         "name": "freeSpacePct",
>                         "type": "string"
>                     }
>             
>         }
>     }
> }
}
> "id": "Trento:193.205.211.69",
> "isPattern": "false",
> "type": "host"
> },
> "statusCode": {
> "code": "200",
> "reasonPhrase": "OK"
> }
> }
> }
> '
>
Which produces the following result after the execution:

* About to connect() to 127.0.0.1 port 5000 (#0)
* Trying 127.0.0.1...
* Adding handle: conn: 0x7fa2e2804000
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x7fa2e2804000) send_pipe: 1, recv_pipe: 0
* Connected to 127.0.0.1 (127.0.0.1) port 5000 (#0)
> POST /v1.0/33/servers/44 HTTP/1.1
> User-Agent: curl/7.30.0
> Host: 127.0.0.1:5000
> Accept: */*
> Content-Type: application/json
> Content-Length: 1110
> Expect: 100-continue

> < HTTP/1.1 100 Continue
> < HTTP/1.1 200 OK
> < Content-Type: text/html; charset=utf-8
> < Content-Length: 0
> < Date: Wed, 09 Apr 2014 00:11:49 GMT
> <
> * Connection #0 to host 127.0.0.1 left intact
6. Unsubscribe the previous rule

In order to stop the process to evaluate rules, it is needed to unsubscribe the activated rule. We can do it with the following operation:

```
curl -v -H 'X-Auth-Token: a9a861db6276414094bc1567f664084d' -X DELETE http://130.206.81.71:8000/v1.0/6571e3422ad84f7d828ce2f30373b3d4/servers/serverI1/subscriptions/Subscriptio
```

```
< HTTP/1.0 200 OK
< Date: Wed, 09 Apr 2014 10:16:59 GMT
< Server: WSGIServer/0.1 Python/2.6.6
< Content-Type: text/html; charset=utf-8
```

6.3.2 Accessing Policy Manager from a Browser

To send http commands to Policy Manager using browser, use:

- Chrome browser [2] with the Simple REST Client plugin [3]
- Firefox RESTClient add-ons [4].
7 Monitoring - User and Programmers Guide

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

7.1 Introduction

Welcome the User and Programmers Guide for the Monitoring Generic Enabler. This GE is built up from different distributed components, as depicted in the following figure:
7.1.1.1 **Background and Detail**
This User and Programmers Guide relates to the Scalability Manager GE which is part of the **Cloud Hosting Chapter**. Please find more information about this Generic Enabler in the following **Open Specification**.

7.2 **User Guide**
This GE does not provide an interactive user interface, hence there is no User Guide. The following section elaborates on programmatic usage.

7.3 **Programmer Guide**
According to the architecture aforementioned, there are several APIs involved in the monitoring process:
- NGSI Adapter API
- Context Broker API
- Monitoring (Query Manager) API

7.3.1 **NGSI Adapter API**
Probe raw data should be sent as body of a POST request to the adapter, identifying the source entity being monitored in the query parameters. For example, given the following scenario:

<table>
<thead>
<tr>
<th>Monitored host</th>
<th>178.23.5.23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring tool</td>
<td>Nagios</td>
</tr>
<tr>
<td>Monitoring probe name</td>
<td>check_load</td>
</tr>
<tr>
<td>NGSI Adapter endpoint</td>
<td><a href="http://adapterhost:1337">http://adapterhost:1337</a></td>
</tr>
</tbody>
</table>

then requests would look like:

```
HTTP POST
http://adapterhost:1337/check_load?id=178.23.5.23&type=host
Content-Type: text/plain
OK - load average: 0.36, 0.25, 0.24
load1=0.360;1.000;1.000;0; load5=0.250;5.000;5.000;0; load15=0.240;15.000;15.000;0;
```

Please take into account that NGSI standard identify entities (in this case, the resources being monitored) using a pair `<entityId, entityType>`. This identification of the monitored resource has to be provided as the query parameters `id` and `type`, respectively. The probe name included in the URL lets NGSI Adapter know the originating monitoring probe, therefore selecting the proper parser for it.

Monitoring framework is expected to schedule the execution of probes and send the raw data been gathered to the NGSI Adapter. Depending on the tool that has been chosen, this would require the development of a custom component (a kind of **monitoring collector**) to automatically forward such data to the adaptation layer.
NGSI Adapter processes requests asynchronously, trying to locate a valid parser named after the originating probe, located at lib/parsers/. If probe is unknown, response status will be 404; otherwise, response status will be 200, parser will be dynamically loaded, and then its `parseRequest()` and `getContextAttrs()` methods will be invoked. With the attribute list returned by the latter, Context Broker will be invoked.

Custom parsers for new probes may be easily added to NGSI Adapter, just extending a base abstract object and implementing the aforementioned methods. For example, suppose we want to support a new "myProbe" whose data is a comma-separated list of values of two attributes `myAttr0` and `myAttr1`:

```javascript
// module "lib.parsers.myProbe"

var baseParser = require('./common/base').parser,
    myParser   = Object.create(baseParser);

myParser.parseRequest = function() {
    return { data: this.request.body };;
};

myParser.getContextAttrs = function(data, optionalData) {
    var items = data.split(',');
    return { myAttr0: items[0], myAttr1: items[1] };;
};
```
7.3.2 Context Broker API

Please refer to Context Broker Programmers Guide. This will give us access to the last updates of monitoring data available, but not to historic data.

7.3.3 Monitoring API

Retrieval of data stored at the BigData GE is handled by the Query Manager component, whose API is described in Monitoring Open RESTful API Specification (PRELIMINARY) section.
8 Self-Service Interfaces - User and Programmers Guide

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

8.1 Introduction

Welcome to the User Guide of the Self Service Interfaces. The Self Service Interfaces provide a support for the users of the cloud infrastructure and platform to manage their services and resources deployed in cloud. For the moment it consist of open source implementation of a User Portal and Scripts. The User Portal is implemented in a form of a Web GUI following the same functionality as the OpenStack Dashboard. All about the implementation and the functionality of the OpenStack Dashboard can be found under the Horizon project. The Scripts facilitate direct approach to the underlying cloud resources through a command line and is addressed for administrators. This user guide describes the User Portal part of the Self Service Interfaces.

Note: The SelfService Interfaces is a front end component, therefore for this GE there is no need to provide a programmers guide.

8.1.1 Background and Detail

This User and Programmers Guide relates to the Self-Service Interfaces GE which is part of the Cloud Chapter. Please find more information about this Generic Enabler in the following Open Specification.

8.2 User Guide

The User Portal offers a stand-alone open-source implementation of the OpenStack Dashboard. Initially it provides the same functionality as the OpenStack Dashboard. Some interactions include (create, delete, update) over the virtual resources (images, instances, flavors, services, etc.).

The User Portal is a Backbone-based Web client-side HTML5 application, implemented in JavaScript. The backbone model-view-controller methodology contributes for better dynamism and makes it a stand-alone client application with no need for Web server to interact with the backend. Instead, it aims to improving the user experience by using AJAX for dynamic rendering of the views. Moreover it contributes to improving the user-experience of the OpenStack portal by offering responsive design which makes it adaptable to multiple device screens (desktop, smart phone, tablet, etc), customizable object oriented CSS and internationalization i18n to support different languages. The User Portal functionality is tightly related to a JStack Library - a JavaScript equivalent of the OpenStack API that represent the Scripts Interface.

The design perspective of the User Portal is the following. There is one implementation that follows exactly the same functionality and design pattern of the OpenStack Dashboard. It is aimed for all use-case and third-party projects that want to use it as an independent component.

The other implementation is build upon the same methodology. However, apart from the basic functionality, it is designed to further extend the functionality of the OpenStack Dashboard and offer advanced interaction with the Service Manager GE,
the Resource Manager GE, Object Storage GE and Cloud Proxy GE. Some of the new operations will include: creation and management of vApps, VDC, possibility to create elasticity rules and display monitoring statistics etc. The design would be changed according the new functionalities and style guide requirements of the FI-Ware project. The audience of this second implementation is the Cloud Hosting architecture adopters who would choose the User Portal as integral part of the architecture, to manage resources and platform components in coordination with the rest of the GEs.

To be able to try the user portal individually over a different setup of the OpenStack infrastructure, one should download the code from the GitHub page. No additional fine-tuning is required in order for this to work properly now.

Now that integration with all DCRM GE and PaaS Manager GE is finished, the user of the portal is able to make the same operations to the infrastructure components, through these components. Furthermore, additional functionalities are leveraged by the PaaSManager for the user to inter-operate with the cloud resources.

### 8.2.1 Examples of Use

Next we present some example operations in the portal for a normal user and admin.

#### 8.2.1.1 Login to User Portal

Figure 1 shows the initial login page on the user portal. Once the user obtains his credentials he can enter the User Portal page. At the moment, one can test the User Portal functionality through the Testbed page of the User Portal [http://cloud.testbed.fi-ware.org](http://cloud.testbed.fi-ware.org)

![Figure 1: User Portal Login Page](image)
8.2.1.2 **Images List**

Once the user has logged in, he is redirected to the panel of his account. We can see in **Figure 2** the page with the available Images. The user can choose an image type and launch an instance out of that image.

![Image Portal](image-url)

**Figure 2: User Portal Images List**

8.2.1.3 **Launch Instance**

**Figure 3** shows the windows that appears when the user want to launch an instance from the Images page. In this case the user should give the instance a name and choose the flavor type he wants for his instance.
8.2.1.4 **Edit Security Groups Rule**

The cloud user may manage the Access and Security by defining different security groups and rules, keypairs and floating IPs. **Figure 4** shows the windows that appears when the user edits the security group rules for a chosen security group.
8.2.1.5 *Upload Object into Container*

The user can manage the object storage by creating containers and uploading object into a given container. [Figure 5](#) shows the window to upload an object into container.

Moreover a user can create volumes and attach an instance to them. He can also create snapshots out of existing Instances and Volumes.

8.2.1.6 *Admin page for Flavors*

When logged in as admin, one can see additional admin tab with the menu for the features available only to the admin. For example, only the admin can manage flavors, project, users and quotas. [Figure 6](#) shows the admin menu and the list of the current flavors. There are also the buttons to create and delete flavors.
8.2.1.7 **Language Settings**

For the moment in the Settings part, four different languages can be set up for the user portal - **Figure 7**.
8.3  Programmer Guide

The Self Service Interfaces GE is a frontend component, without backend interface. Therefore there is no need to provide a programmer guide. See underlying GEs for API-based interaction (it is not part of this GE).
9 Job Scheduler - User and Programmers Guide

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

9.1 Introduction

Welcome to the User and Programmers Guide of our ProActive Cloud Job Scheduler!

ProActive Cloud Job Scheduler is an open source implementation of the Job Scheduler Generic Enabler, whose interface, to interact with, is derived from ProActive Scheduling & Resourcing REST API. So, whenever possible, this guide will point to the appropriate on-line contents that have been created for this specific API. The on-line documents might be updated and improved, and so will be the most appropriate place to get the most up to date information on using this interface.

9.1.1 Recommended actions and assumptions before starting

- We assume that the ProActive Cloud Job Scheduler administrator has installed it according to its Installation and Administration Guide.

- If the on-line documentation is not available or out-of-synchronization, we let you notice that each ProActive Cloud Job Scheduler release contains the associate REST API documentation. In order to discover the exact endpoint, please check the http://localhost:8080/rest web page, otherwise ask your administrator.

- Since our ProActive Cloud Job Scheduler implementation is featured with a built-in and pre-configured AAA (Authentication, Authorization & Accounting) system where users can be profiled according two main groups (users and admin), we assume that both User and Programmer of the current guide belong to admin group, such as it happen for the pre-configured admin user. Doing so, let us cover all possible usage, since -as admin user- the User/Programmer can access all resources without any limitations.

- Following the previous assumption, we suggest to get familiar with our Baseline Assets, since that is really useful in case of describing/understanding what is going on behind the scene. Anyway, whenever possible, we will try to refer specifically to those sections might be of interest at this end.

- If problems may raise up while accessing resources, please check the default user group permissions, where you might belong to, at $JOB_SCHEDULER_GE_HOME/scheduling/config/security.java.policy-server. If not solved yet, then contact your ProActive Cloud Job Scheduler administrator which might have changed your profile.

- In order to give the User/Developer a consistent and understandable presentation of their usage, the API Operations order may change in this context.

- The access through the CLI is made using the curl program. Curl [1] is a client to get documents/files from or send documents to a server, using any of the supported protocols (HTTP, HTTPS, FTP, GOPHER, DICT, TELNET, LDAP or FILE) and therefore is also usable for ProActive Scheduling & Resourcing REST API. Use the curl command line tool or use libcurl from...
within your own programs in C. Curl is free and open software that compiles and runs under a wide variety of operating systems.

- To furnish a readable API operations output, we will let the operation request follow by "| python -mjson.tool" in order to achieve a well-done json format.

- As stated at the Job Scheduler GE API Operations introduction, the integration with Identity Management GE reasonably consists in storing in the sessionid parameter, required for each request against ProActive Scheduling & Resourcing REST API, the content of the id of the token returned after the successful user authentication, by providing username and password parameters. In such a way, we implicitly bind the Job Scheduler session id lifetime to the token one, by keeping the one-to-one mapping between them. Please, refer to login on the following subsections.

- For each operation, instead of worrying about getting the sessionid from the Job Scheduler each time, we will use a standalone invocation, which consists of "embedding" sessionid retrieval, through login usage, between backticks (`):

```shell
sessionid: `curl -v -X POST -d "username=admin&password=admin" 
http://localhost:8080/rest/rest/rm/login`
```

In what follows we will omit the related output of this embedded way for achieving the sessionid.

- Our examples that will follow are done by using a Unix system and the related bash command-line. In order to achieve the same result on Windows systems, it is just a matter of appending the suffix .bat to each Job Scheduler command, such as command.bat, and contained in $JOB_SCHEDULER_GE_HOME/scheduling/bin/windows folder.

- Finally, in order to fill properly the values related to parameters required to perform a given operation, it might be useful an URL decoding/enconding tool when URLs are required as parameters.

## 9.2 User Guide

Please refer to the following on-line documentation:

- ProActive Resource Manager User Guide and Admin Guide
- ProActive Scheduler User Guide and Admin Guide

## 9.3 Programmer Guide

### 9.3.1 Scheduler Service

#### 9.3.1.1 createCredential

create user credentials as function of his username, password and the public ssh key of the Job Scheduler GEi server. It is mapped with ProActive Parallel Suite /scheduler/createcredential operation
bash-4.2$ curl -v -X POST -F "username=admin" -F "password=admin" http://localhost:8080/rest/rest/scheduler/createcredential

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/createcredential HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
> libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
> libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Expect: 100-continue
> Content-Length: 250
> Content-Type: multipart/form-data; boundary=--------------------------eccdbecf3720
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Content-Type: */*
< Content-Length: 424
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0

U1NBCjEwMjQKULNBL0VDQi9QS0NTMVBhZGRpbmcKdy1l1x3+rxz7/90/+gdjYtk
wwpNgtBuo4LkLI790BDadftfM8wOdRIh59UX+OXCCSpHmxGwUEpH4LBHwIZzwvK
61bxSpPbg4GgXRb8XhaNeLj2zorzAcndtyhN9kyZAF63jDbp5EPePsL9/83Ak15F
00oaJLUUu/Cnjq1SK7H6kiIMZjFEcbeTWQfRcyl3jkmwvRa9KYNJ3E/kOWBKT
6t/pkI2NicY/ICS//Y4CswWa0ZWBRiGJeSa4jqj5M5kAQI1IzbZX+jIbE96OeN6
VgMwPCuUKGPa6CxPTMFSrARhWs592qeCztqOxJ5brPpulsZrDWUXDdosqhr9k
17Cw6DIJxdevexb/ot5aLNNk70CD6XbaxuxhLCBFGdoFYymRoAz262

In order to store the response in a dedicated file, perform the following command:

bash-4.2$ rm
$JOB_SCHEDULER_GE_HOME/scheduling/config/authentication/admin.cred # we remove any already existing 'admin.cred' file
bash-4.2$ touch
$JOB_SCHEDULER_GE_HOME/scheduling/config/authentication/admin.cred
9.3.1.2 login

enable the user to login to the scheduler using a form containing 2 fields: *username* and *password*. It is mapped with ProActive Parallel Suite `/scheduler/login` operation.

```bash
bash-4.2$ curl -v -X POST -F "username=admin" -F "password=admin"
http://localhost:8080/rest/rest/scheduler/login

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/login HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
400a658813dff1aba3477802a2034ac7a124c32400a658813dff1aba348000
```

9.3.1.3 loginWithCredential

enable the user to login to the scheduler by submitting a multipart including the credential file with field name *credential*. It is mapped with ProActive Parallel Suite `/scheduler/login` operation.

```bash
bash-4.2$ curl -v -X POST -F credential=@$JOB_SCHEDULER_GE_HOME/scheduling/config/authentication/admin.cred
```

D.4.2.3 FI-WARE User and Programmer Guide Page 79
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/login HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Content-Length: 622
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=-----------------------------6e6529f87d83
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
3f6fbcbl14117e8a8f818562a2034ac7a124c323f6fbcbl14117e8a8f88000

9.3.1.4 **disconnect**
disconnect user from the Scheduler. It is mapped with ProActive Parallel Suite
/scheduler/disconnect operation

bash-4.2$ curl -v -X PUT -H sessionid:`curl -v -X POST -d
"username=admin&password=admin"
http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/disconnect
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/disconnect HTTP/1.1
9.3.1.5 isConnected

test whether or not the user is connected to the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/isConnected operation


* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/isconnected HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:400a658813dff1aba3476e22a2034ac7a124c32400a658813dff laba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
9.3.1.6  **pauseScheduler**

pause the scheduler. It is mapped with ProActive Parallel Suite /scheduler/pause operation

```
bash-4.2$ curl -v -X PUT -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
hhttp://localhost:8080/rest/rest/scheduler/pause
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/pause HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:400a658813dflaba3475a52a2034ac7a124c32400a658813dflaba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```

9.3.1.7  **freezeScheduler**

freeze the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/freeze operation

```
bash-4.2$ curl -v -X PUT -H sessionid:`curl -v -X POST -d "username=admin&password=admin"
```

```
9.3.1.8 **resumeScheduler**

resume the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/resume operation

```bash
bash-4.2$ curl -v -X PUT -H "sessionid:400a658813dff1aba3474f92a2034ac7a124c32400a658813dff1aba348000"
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```
9.3.1.9 **stopScheduler**

stop the Scheduler. It is mapped with ProActive Parallel Suite [/scheduler/stop](http://localhost:8080/rest/rest/scheduler/stop) operation

```bash
```

- About to connect() to localhost port 8080 (#0)
- Trying 127.0.0.1...
- Connected to localhost (127.0.0.1) port 8080 (#0)

> PUT /rest/rest/scheduler/stop HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
```
> sessionid:400a658813dff1aba3475302a2034ac7a124c32400a658813dff
laba348000
```

> < HTTP/1.1 200 OK
> Content-Type: application/json
> Transfer-Encoding: chunked
> Server: Jetty(6.1.18)
>
> * Connection #0 to host localhost left intact
* Closing connection #0
9.3.1.10  **getSchedulerStatus**

Return the current Scheduler status. It has been mapped with ProActive Parallel Suite /scheduler/status operation.

```bash
* About to connect() to localhost port 8080 (#0)
 *   Trying 127.0.0.1... connected
 * Connected to localhost (127.0.0.1) port 8080 (#0)
 > GET /rest/rest/scheduler/status HTTP/1.1
 > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
 > Host: localhost:8080
 > Accept: */*
 >
 > sessionid:400a658813dfflab3472532a2034ac7a124c32400a658813dfflab348000
 >
 < HTTP/1.1 200 OK
 < Content-Type: application/json
 < Transfer-Encoding: chunked
 < Server: Jetty(6.1.18)
 <
 * Connection #0 to host localhost left intact
 * Closing connection #0
 "STOPPED"
```

9.3.1.11  **startScheduler**

Start the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/start operation.

```bash
* About to connect() to localhost port 8080 (#0)
```
9.3.1.12 **killScheduler**

kill the Scheduler (after that the operation is submitted, at the moment it is required to restart all service from scratch). It is mapped with ProActive Parallel Suite `/scheduler/kill` operation. It does not require advanced services configuration, since the `rmiregistry` service belongs to the RM Service, by default.

```
* About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1... connected
  * Connected to localhost (127.0.0.1) port 8080 (#0)
  > PUT /rest/rest/scheduler/kill HTTP/1.1
  > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
  > Host: localhost:8080
  > Accept: */*
  >
  < HTTP/1.1 200 OK
  < Content-Type: application/json
  < Transfer-Encoding: chunked
  < Server: Jetty(6.1.18)
  <
  * Connection #0 to host localhost left intact
  * Closing connection #0
  true
```
9.3.1.13  **getSchedulerStats**

returns statistics about the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/stats operation

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
% Total  % Received % Xferd Average Speed Time    Time    Current
          0       0  0.00%  0.00%  0.00%    0.00%        0 --:--:-- --:--:-- --:--:--
Dload  Upload Total Spent Left Speed
 0     0    0     0    0     0      0      0      0      0 0     0    0     0
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/stats HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813df1aba34702b2a2034ac7a124c32400a658813dff1aba348000
> < HTTP/1.1 200 OK
< Content-Type: application/json
```
9.3.1.14  **getMySchedulerStats**

return statistics about the Scheduler usage of the current user. It is mapped with ProActive Parallel Suite /scheduler/stats/myaccount operation

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
% Total    % Received % Xferd Average Speed Time   Time     Time  Current

* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/stats/myaccount HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> 
> sessionid:400a658813dff1aba346fb72a2034ac7a124c32400a658813dff
1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
  [data not shown]
100 104 0 104 0 0 622 0 --:-:-:- --:-:-:- --:-:-:- --:-:-:- --:-:-:- 626
* Connection #0 to host localhost left intact
* Closing connection #0
{
  "TotalJobCount": "34",
  "TotalJobDuration": "1203175",
  "TotalTaskCount": "268",
  "TotalTaskDuration": "2306514"
}

9.3.1.15 getConnectedUsers

return users currently connected to the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/users operation

bash-4.2$ curl -v -X GET -H "sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`"
http://localhost:8080/rest/rest/scheduler/users | python -mjson.tool
  * About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1...
  % Total  % Received % Xfered
Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
  0      0 0    0 0    0 0    0 0     0 --:--:-- --:--:----:--:--
  0        0 0    0 0    0 0    0 0     0 0    0 0    0 0    0 0    0
  * Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/users HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
>      libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
>      libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:400a658813df1aba3470c82a2034ac7a124c32400a658813df
>        laba348000
>
> < HTTP/1.1 200 OK
> < Content-Type: application/json
> < Transfer-Encoding: chunked
> < Server: Jetty(6.1.18)
> <
> {
> [data not shown]
> 100 2786 0 2786 0 0 21323 0 --:--:-- --:--:--
> 21596
> * Connection #0 to host localhost left intact
> * Closing connection #0
> [
>   {
>     "connectionTime": 1365787303252,
>     "hostName": "192.168.122.1:1099",
>     "lastSubmitTime": -1,
>     "submitNumber": 0,
>     "username": "admin"
>   },
>   {
>     "connectionTime": 1365788253368,
>     "hostName": "192.168.122.1:1099",
>     "lastSubmitTime": -1,
>     "submitNumber": 0,
>     "username": "admin"
>   }
> ]
"lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
},
{
  "connectionTime": 136579292006,
  "hostName": "192.168.122.1:1099",
  "lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
},
{
  "connectionTime": 1365788747091,
  "hostName": "192.168.122.1:1099",
  "lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
},
{
  "connectionTime": 1365788008171,
  "hostName": "192.168.122.1:1099",
  "lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
},
{
  "connectionTime": 1365788202425,
  "hostName": "192.168.122.1:1099",
  "lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
{
    "connectionTime": 1365788373756,
    "hostName": "192.168.122.1:1099",
    "lastSubmitTime": -1,
    "submitNumber": 0,
    "username": "admin"
},
{
    "connectionTime": 1365789297126,
    "hostName": "192.168.122.1:1099",
    "lastSubmitTime": -1,
    "submitNumber": 0,
    "username": "admin"
},
{
    "connectionTime": 1365787968219,
    "hostName": "192.168.122.1:1099",
    "lastSubmitTime": -1,
    "submitNumber": 0,
    "username": "admin"
},
{
    "connectionTime": 1365788388038,
    "hostName": "192.168.122.1:1099",
    "lastSubmitTime": -1,
    "submitNumber": 0,
    "username": "admin"
},
{
    "connectionTime": 1365788708895,
    "hostName": "192.168.122.1:1099",
    "lastSubmitTime": -1,
    "submitNumber": 0,
    "username": "admin"
},
{
    "connectionTime": 1365787981888,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
"username": "admin"
},
{
"connectionTime": 1365787537015,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
"username": "admin"
},
{
"connectionTime": 1365788115416,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
"username": "admin"
},
{
"connectionTime": 1365789154122,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
"username": "admin"
},
{
"connectionTime": 1365788416083,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
"username": "admin"
},
{
"connectionTime": 1365788247885,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitButton": 0,
```
"username": "admin"
},
{
"connectionTime": 1365788873565,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365789280466,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365784577260,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "watcher"
},
{
"connectionTime": 1365788150388,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365789168526,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
}
```
9.3.1.16 | **getJobs**

return jobs list. It is mapped with ProActive Parallel Suite /scheduler/jobs operation

```
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d
"username=admin&password=admin"
http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
lxcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813dff1aba346eb62a2034ac7a124c32400a658813dff
lab3a348000
> HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
["151","101","51","46","45","44","43","42","41","40","39","38","37","36","35","34","33","32","31","30","29","28","27","25","23","22","21","20","19","18","17","16","15","14","13","12","11","10","9","8","7","6","5","4","24","26"]
```

9.3.1.17 | **linkRM**

connect the Scheduler to a given the RM endpoint. It is mapped with ProActive Parallel Suite /scheduler/linkrm operation.

Whenever the RM Service crashes, an action to let the Scheduler be aware that the RM Service is again available is required. This re-binding allows the jobs, queued at the Scheduler level or under execution during the downtime, to be resumed afterwards.

In order to achieve a successful request, we assume the same scenario shown while shutdownRM operation was introduced, by additionally performing the shutdownRM operation and the RM Service restart (via rm-start script with the same options), as well. Then, if we submitted a job via submitJob operation, we could get
success, but thanks to **getJobState** the actual computation will not take place - job will be in the **PENDING** state-. That is because, the job is just queued. Only after we perform the following

```bash
http://localhost:8080/rest/rest/scheduler/linkrm
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/linkrm HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
``sessionid:42284ecb13e2181f4ef71c32a2034ac7a124c3242284ecb13e2181f4ef8000
> Content-Length: 41
> Content-Type: application/x-www-form-urlencoded
>
``< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```
job will start automatically without problem, changing its state in **RUNNING**.

9.3.1.18 **getSchedulerVersion**

return the current REST server API and Scheduler version. It has been mapped with ProActive Parallel Suite /scheduler/version operation

```bash
bash-4.2$ curl -v -X GET -H sessionid=`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
```

D.4.2.3 FI-WARE User and Programmer Guide Page 96
http://localhost:8080/rest/rest/scheduler/version | python -m json.tool

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... % Total % Received % Xfered
Average Speed Time Time Time Current
Dload Upload Total Spent
Left Speed
0 0 0 0 0 0 0 0 --:--:-- --:--:--
- --:--:-- 0 connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/version HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: */*
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
 [data not shown]
100 76 0 76 0 0 22405 0 --:--:-- --:--:--
- --:--:-- 38000
* Connection #0 to host localhost left intact
* Closing connection #0
{
  "rest": "FIWARE-2.2.3_(1.3.2+)",
  "scheduler": "FIWARE-2.2.3_(3.1.0+)"
}

9.3.2 Jobs
- **Recommended action:** read about [ProActive Scheduler Basics](#) and [ProActive Scheduler - User Guide](#), where it is described how is possible to define a job by an XML descriptor or via Java API. Our job of reference for this guide is the following "job_8_tasks.xml", taken from $JOB_SCHEDULER_GE_HOME/scheduling/samples/jobs_descriptors folder

```xml
<?xml version="1.0" encoding="UTF-8"?>
```
<job xmlns="urn:proactive:jobdescriptor:dev"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:proactive:jobdescriptor:dev
    ../../../src/scheduler/src/org/ow2/proactive/scheduler/common/xml
    /schemas/jobdescriptor/dev/schedulerjob.xsd"
    name="job_8_tasks" priority="normal">
  <description>Simple test of 8 tasks with variable durations</description>
  <taskFlow>
    <task name="task1">
      <!-- Task human description -->
      <description>This task will sleep 8s</description>
      <javaExecutable class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
        <parameters>
          <parameter name="sleepTime" value="8"/>
          <parameter name="number" value="1"/>
        </parameters>
      </javaExecutable>
    </task>
    <task name="task2">
      <!-- Task human description -->
      <description>This task will sleep 6s</description>
      <javaExecutable class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
        <parameters>
          <parameter name="sleepTime" value="6"/>
          <parameter name="number" value="2"/>
        </parameters>
      </javaExecutable>
    </task>
    <task name="task3">
      <!-- Task human description -->
      <description>This task will sleep 8s</description>
      <javaExecutable class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
        <parameters>
          <parameter name="sleepTime" value="8"/>
          <parameter name="number" value="3"/>
        </parameters>
      </javaExecutable>
    </task>
  </taskFlow>
</job>
<task name="task8" preciousResult="true">
  <![CDATA[
  -- Task human description -->
  <description>This task will sleep 20s</description>
  <javaExecutable
    class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
    <parameters>
      <parameter name="sleepTime" value="20"/>
      <parameter name="number" value="8"/>
    </parameters>
  </javaExecutable>
</task>

<task name="task4">
  <![CDATA[
  -- Task human description -->
  <description>This task will sleep 6s</description>
  <javaExecutable
    class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
    <parameters>
      <parameter name="sleepTime" value="6"/>
      <parameter name="number" value="4"/>
    </parameters>
  </javaExecutable>
</task>

<task name="task5">
  <![CDATA[
  -- Task human description -->
  <description>This task will sleep 6s</description>
  <javaExecutable
    class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
    <parameters>
      <parameter name="sleepTime" value="6"/>
      <parameter name="number" value="5"/>
    </parameters>
  </javaExecutable>
</task>

<task name="task6">
  <![CDATA[
  -- Task human description -->
  <description>This task will sleep 4s</description>
  <javaExecutable
    class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
    <parameters>
      <parameter name="sleepTime" value="4"/>
      <parameter name="number" value="6"/>
    </parameters>
  </javaExecutable>
</task>
9.3.2.1 submitJob

Submit a job to the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/submit operation.


* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
* Average Speed  Time  Time  Time  Current
<table>
<thead>
<tr>
<th>Dload</th>
<th>Upload</th>
<th>Total</th>
<th>Spent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>connected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/login HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
> ] [data not shown]
100 29 0 0 100 29 0 28 0:00:01
0:00:01 --:--:-- 28< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
} [data not shown]
100 91 0 62 100 29 37 17 0:00:01
0:00:01 --:--:-- 37
* Connection #0 to host localhost left intact
* Closing connection #0
% Total  % Received  % Xferd Average Speed Time Time
Time    Current  Dload  Upload   Total   Spent
Left   Speed
100 3442 0 39 100 3403 81 7080 --:--:-- --:--:--
--:--:-- 7089
} "id": 205
  "readableName": "job_8_tasks"

9.3.2.2 **pauseJob**

pause the job execution by means of **jobid**. It is mapped with ProActive Parallel Suite /scheduler/jobs/\{jobid\}/pause operation

bash-4.2# curl -v -X PUT -H "sessionid:`curl -v -X POST -d "username=admin&password=admin"
**Future Internet Core Platform**

http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/205/pause

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/jobs/205/pause HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
>   libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
>   libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813dff1aba343c532a2034ac7a124c32400a658813dff
   laba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true

### 9.3.2.3 `getJobState`

return the job state by providing `jobid`. It is mapped with ProActive Parallel Suite `/scheduler/jobs/{jobid}` operation

bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/205 | python -mjson.tool

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... % Total % Received % Xferd
  Average Speed  Time    Time     Time  Current
        Dload  Upload   Total   Spent
  Left  Speed
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
* Connected to localhost (127.0.0.1) port 8080 (#0)
```bash
> GET /rest/rest/scheduler/jobs/205 HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813dflaba343b8c2a2034ac7a124c32400a658813df
> laba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
  [data not shown]
  100 7235 0 7235 0 0 26413 0 --:--:-- --:--:--
  --:--:-- 26501
* Connection #0 to host localhost left intact

* Closing connection #0
{
  "cancelJobOnError": false,
  "description": "Simple test of 8 tasks with variable durations",
  "environment": {
    "containsJarFile": false,
    "crc": 0,
    "jobClasspath": null,
    "jobClasspathContent": null
  },
  "genericInformations": {},
  "inputSpace": null,
  "jobInfo": {
    "finishedTime": -1,
    "jobId": {
      "id": 205,
      "readableName": "job_8_tasks"
    },
    "modifiedTasks": null,
```
"numberOfFinishedTasks": 0,
"numberOfPendingTasks": 0,
"numberOfRunningTasks": 0,
"owner": "admin",
"priority": "NORMAL",
"removedTime": -1,
"startTime": -1,
"status": "PAUSED",
"submittedTime": 1365849531328,
"taskFinishedTimeModify": null,
"taskStatusModify": {
    "2050000": "PAUSED",
    "2050001": "PAUSED",
    "2050002": "PAUSED",
    "2050003": "PAUSED",
    "2050004": "PAUSED",
    "2050005": "PAUSED",
    "2050006": "PAUSED",
    "2050007": "PAUSED"
},
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
},
"maxNumberOfExecution": 1,
"name": "job_8_tasks",
"outputSpace": null,
"owner": "admin",
"priority": "NORMAL",
"projectName": "Not Assigned",
"restartTaskOnError": {
    "value": {
        "description": "Anywhere",
        "index": 1
    }
},
"tasks": {
    "2050000": {

"cScript": null,
"cancelJobOnError": false,
"dependenceIds": [],
"dependences": [],
"description": "This task will sleep 4s",
"flowBlock": "none",
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task7",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
  "value": {
    "description": "Anywhere",
    "index": 1
  }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskId": {
  "id": 2050000,
  "readableName": "task7"
"taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050001": {
"cScript": null,
"cancelJobOnError": false,
"dependenceIds": [],
"dependences": [],
"description": "This task will sleep 8s",
"flowBlock": "none",
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task3",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
"value": {
"description": "Anywhere",
"index": 1
}
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
"executionDuration": -1,
"executionHostName": null,
"finishedTime": -1,
"numberOfExecutionLeft": 1,
"numberOfExecutionOnFailureLeft": 2,
"progress": 0,
"startTime": -1,
"taskId": {
   "id": 2050001,
   "readableName": "task3"
},
"taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050002": {
   "cScript": null,
   "cancelJobOnError": false,
   "dependenceIds": [],
   "dependences": [],
   "description": "This task will sleep 6s",
   "flowBlock": "none",
   "flowScript": null,
   "genericInformations": {},
   "inputFiles": null,
   "maxNumberOfExecution": 1,
   "maxNumberOfExecutionOnFailure": 2,
   "name": "task4",
   "outputFiles": null,
   "parallelEnvironment": null,
   "postScript": null,
   "preScript": null,
   "preciousLogs": false,
   "preciousResult": false,
   "restartTaskOnError": {
      "value": {
         "description": "Anywhere",
         "index": 1
      }
   },
   "resultPreview": null,
   "runAsMe": false,
"sScripts": null,
"taskInfo": {
    "executionDuration": -1,
    "executionHostName": null,
    "finishedTime": -1,
    "numberOfExecutionLeft": 1,
    "numberOfExecutionOnFailureLeft": 2,
    "progress": 0,
    "startTime": -1,
    "taskId": {
        "id": 2050002,
        "readableName": "task4"
    },
    "taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050003": {
    "cScript": null,
    "cancelJobOnError": false,
    "dependenceIds": [],
    "dependences": [],
    "description": "This task will sleep 6s",
    "flowBlock": "none",
    "flowScript": null,
    "genericInformations": {},
    "inputFiles": null,
    "maxNumberOfExecution": 1,
    "maxNumberOfExecutionOnFailure": 2,
    "name": "task2",
    "outputFiles": null,
    "parallelEnvironment": null,
    "postScript": null,
    "preScript": null,
    "preciousLogs": false,
    "preciousResult": false,
    "restartTaskOnError": {
        "value": {
            "server": null,
            "taskId": null,
            "type": "none",
            "value": null
        }
    }
}
"description": "Anywhere",
"index": 1
}
,
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskId": {
  "id": 2050003,
  "readableName": "task2"
},
"taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050004": {
  "cScript": null,
  "cancelJobOnError": false,
  "dependenceIds": [],
  "dependences": [],
  "description": "This task will sleep 8s",
  "flowBlock": "none",
  "flowScript": null,
  "genericInformations": {},
  "inputFiles": null,
  "maxNumberOfExecution": 1,
  "maxNumberOfExecutionOnFailure": 2,
  "name": "task1",
  "outputFiles": null,
  "parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
    "value": {
        "description": "Anywhere",
        "index": 1
    }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
    "executionDuration": -1,
    "executionHostName": null,
    "finishedTime": -1,
    "numberOfExecutionLeft": 1,
    "numberOfExecutionOnFailureLeft": 2,
    "progress": 0,
    "startTime": -1,
    "taskId": {
        "id": 2050004,
        "readableName": "task1"
    },
    "taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050005": {
    "cScript": null,
    "cancelJobOnError": false,
    "dependenceIds": [],
    "dependences": [],
    "description": "This task will sleep 20s",
    "flowBlock": "none",
    "flowScript": null,
    "genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task8",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": true,
"restartTaskOnError": {
   "value": {
      "description": "Anywhere",
      "index": 1
   }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
   "executionDuration": -1,
   "executionHostName": null,
   "finishedTime": -1,
   "numberOfExecutionLeft": 1,
   "numberOfExecutionOnFailureLeft": 2,
   "progress": 0,
   "startTime": -1,
   "taskId": {
      "id": 2050005,
      "readableName": "task8"
   },
   "taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050006": {
   "cScript": null,
   "cancelJobOnError": false,
"dependenceIds": [],
"dependences": [],
"description": "This task will sleep 4s",
"flowBlock": "none",
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task6",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
  "value": {
    "description": "Anywhere",
    "index": 1
  }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
  "executionDuration": -1,
  "executionHostName": null,
  "finishedTime": -1,
  "numberOfExecution": 1,
  "numberOfExecutionOnFailure": 2,
  "progress": 0,
  "startTime": -1,
  "taskId": {
    "id": 2050006,
    "readableName": "task6"
  },
  "taskStatus": "PAUSED"
9.3.2.4 getTasks

return a list of the name of the tasks belonging to job, by providing jobid. It is mapped with ProActive Parallel Suite /scheduler/jobs/(jobid)/tasks operation

```

* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1...
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
Left  Speed
    0     0    0     0    0     0     0     0 --:-:-- --:-:-- --:-:--
    0     0    0     0     0     0     0     0
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/205/tasks HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:400a658813dfff1aba34392e2a2034ac7a124c32400a658813dff
> laba348000
>```
9.3.2.5  getTasksState

return the list of all the tasks state related to the job, by providing jobid. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/taskstates operation


* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
Average Speed   Time    Time     Time  Current
Dload  Upload   Total   Spent
Left  Speed
0     0    0     0    0     0      0      0
--:--:-- --:--:-- --:--:-- --:--:--
0connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/205/taskstates HTTP/1.1
```
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:400a658813dff1aba3438d72a2034ac7a124c32400a658813dff
  laba348000
>
> < HTTP/1.1 200 OK
> < Content-Type: application/json
> < Transfer-Encoding: chunked
> < Server: Jetty(6.1.18)
> <
> { [data not shown]
100  6145    0  6145    0     0  24926      0
--:--:-- --:--:-- --:--:-- 25081
* Connection #0 to host localhost left intact
* Closing connection #0
[
  {
    "cScript": null,
    "cancelJobOnError": false,
    "dependenceIds": [],
    "dependences": [],
    "description": "This task will sleep 6s",
    "flowBlock": "none",
    "flowScript": null,
    "genericInformations": {},
    "inputFiles": null,
    "maxNumberOfExecution": 1,
    "maxNumberOfExecutionOnFailure": 2,
    "name": "task2",
    "outputFiles": null,
    "parallelEnvironment": null,
    "postScript": null,
    "preScript": null,
    "preciousLogs": false,
    "preciousResult": false,
```
"restartTaskOnError": {
    "value": {
        "description": "Anywhere",
        "index": 1
    }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
    "executionDuration": -1,
    "executionHostName": null,
    "finishedTime": -1,
    "numberOfExecutionLeft": 1,
    "numberOfExecutionOnFailureLeft": 2,
    "progress": 0,
    "startTime": -1,
    "taskId": {
        "id": 2050003,
        "readableName": "task2"
    },
    "taskStatus": "PAUSED"
},
"wallTime": 0
},
{
    "cScript": null,
    "cancelJobOnError": false,
    "dependenceIds": [],
    "dependences": [],
    "description": "This task will sleep 6s",
    "flowBlock": "none",
    "flowScript": null,
    "genericInformations": {},
    "inputFiles": null,
    "maxNumberOfExecution": 1,
    "maxNumberOfExecutionOnFailure": 2,
    "name": "task4"
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
  "value": {
    "description": "Anywhere",
    "index": 1
  }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
  "executionDuration": -1,
  "executionHostName": null,
  "finishedTime": -1,
  "numberOfExecutionLeft": 1,
  "numberOfExecutionOnFailureLeft": 2,
  "progress": 0,
  "startTime": -1,
  "taskId": {
    "id": 2050002,
    "readableName": "task4"
  },
  "taskStatus": "PAUSED"
},
"wallTime": 0
},

{  
  "cScript": null,
  "cancelJobOnError": false,
  "dependenceIds": [],
  "dependences": [],
  "description": "This task will sleep 8s",
  "flowBlock": "none"}
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task3",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
   "value": {
      "description": "Anywhere",
      "index": 1
   }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
   "executionDuration": -1,
   "executionHostName": null,
   "finishedTime": -1,
   "numberOfExecutionLeft": 1,
   "numberOfExecutionOnFailureLeft": 2,
   "progress": 0,
   "startTime": -1,
   "taskId": {
      "id": 2050001,
      "readableName": "task3"
   },
   "taskStatus": "PAUSED"
},
"wallTime": 0
}
"cScript": null,
"cancelJobOnError": false,
"dependenceIds": [],
"dependences": [],
"description": "This task will sleep 4s",
"flowBlock": "none",
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task7",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
  "value": {
    "description": "Anywhere",
    "index": 1
  }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskId": {
  "id": 2050000,
  "readableName": "task7"
}
{"cScript": null,
"cancelJobOnError": false,
"dependenceIds": [],
"dependences": [],
"description": "This task will sleep 6s",
"flowBlock": "none",
"flowScript": null,
"genericInformations": {},
"inputFiles": null,
"maxNumberOfExecution": 1,
"maxNumberOfExecutionOnFailure": 2,
"name": "task5",
"outputFiles": null,
"parallelEnvironment": null,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
   "value": {
      "description": "Anywhere",
      "index": 1
   }
},
"resultPreview": null,
"runAsMe": false,
"sScripts": null,
"taskInfo": {
   "executionDuration": -1,
   "executionHostName": null,
   "finishedTime": -1,
   "numberOfExecutionLeft": 1,
"numberOfExecutionOnFailureLeft": 2,
"progress": 0,
"startTime": -1,
"taskId": {
  "id": 2050007,
  "readableName": "task5"
},
"taskStatus": "PAUSED"
},
"wallTime": 0
},
{
  "cScript": null,
  "cancelJobOnError": false,
  "dependenceIds": [],
  "dependences": [],
  "description": "This task will sleep 4s",
  "flowBlock": "none",
  "flowScript": null,
  "genericInformations": {},
  "inputFiles": null,
  "maxNumberOfExecution": 1,
  "maxNumberOfExecutionOnFailure": 2,
  "name": "task6",
  "outputFiles": null,
  "parallelEnvironment": null,
  "postScript": null,
  "preScript": null,
  "preciousLogs": false,
  "preciousResult": false,
  "restartTaskOnError": {
    "value": {
      "description": "Anywhere",
      "index": 1
    }
  },
  "resultPreview": null,
  "runAsMe": false,
```json
"sScripts": null,
"taskInfo": {
    "executionDuration": -1,
    "executionHostName": null,
    "finishedTime": -1,
    "numberOfExecutionLeft": 1,
    "numberOfExecutionOnFailureLeft": 2,
    "progress": 0,
    "startTime": -1,
    "taskId": {
        "id": 2050006,
        "readableName": "task6"
    },
    "taskStatus": "PAUSED"
},
"wallTime": 0
},
{
    "cScript": null,
    "cancelJobOnError": false,
    "dependenceIds": [],
    "dependences": [],
    "description": "This task will sleep 20s",
    "flowBlock": "none",
    "flowScript": null,
    "genericInformations": {},
    "inputFiles": null,
    "maxNumberOfExecution": 1,
    "maxNumberOfExecutionOnFailure": 2,
    "name": "task8",
    "outputFiles": null,
    "parallelEnvironment": null,
    "postScript": null,
    "preScript": null,
    "preciousLogs": false,
    "preciousResult": true,
    "restartTaskOnError": {
        "value": {  
```
"description": "Anywhere",
"index": 1
}
,"resultPreview": null,"runAsMe": false,"sScripts": null,"taskInfo": {
"executionDuration": -1,"executionHostName": null,"finishedTime": -1,"numberOfExecutionLeft": 1,"numberOfExecutionOnFailureLeft": 2,"progress": 0,"startTime": -1,"taskId": {
"id": 2050005,"readableName": "task8"
},
"taskStatus": "PAUSED"
},
"wallTime": 0
},
{
"cScript": null,"cancelJobOnError": false,"dependenceIds": [],"dependences": [],"description": "This task will sleep 8s","flowBlock": "none","flowScript": null,"genericInformations": {},"inputFiles": null,"maxNumberOfExecution": 1,"maxNumberOfExecutionOnFailure": 2,"name": "task1","outputFiles": null,"parallelEnvironment": null,"parallelity": null,"runAsMe": false,"runAsMeJson": null,"sScripts": null,"taskId": {
"id": 2050005,"readableName": "task8"
},
"taskStatus": "PAUSED"
},
"wallTime": 0}
9.3.2.6 **changeJobPriority**

It changes the priority of a job under execution. It can be achieved by distinguishing between *by-name* or *by-value* priority, according to [Configure users authentication](#) section of ProActive Scheduler Documentation). So, we have what follows:

- **changeJobPriorityByName** - change the previous job priority by providing `jobid` and priority `name`. It is mapped with ProActive Parallel Suite `/scheduler/jobs/{jobid}/priority/byname/{name}` operation
```
  * About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1... connected
  * Connected to localhost (127.0.0.1) port 8080 (#0)
  > PUT /rest/rest/scheduler/jobs/205/priority/byname/LOW
  HTTP/1.1
  > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
  > Host: localhost:8080
  > Accept: */*
  
  > sessionid:400a658813dff1aba34380a2a2034ac7a124c32400a658813dff
  laba348000
  >
  < HTTP/1.1 204 No Content
  < Server: Jetty(6.1.18)
  <
  * Connection #0 to host localhost left intact
  *
  Closing connection #0
```

- **changeJobPriorityByValue** - change the previous job priority by providing jobid and priority value. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/priority/byvalue/{value} operation

```
  * About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1... connected
  * Connected to localhost (127.0.0.1) port 8080 (#0)
  > PUT /rest/rest/scheduler/jobs/205/priority/byvalue/1
  HTTP/1.1
  > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
  > Host: localhost:8080
  > Accept: */*
```
9.3.2.7  **resumeJob**

resume the job execution by providing **jobid**. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/resume operation

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid: 4082375641d7103b700d856522d49f5729f348000

> < HTTP/1.1 200 OK
> Content-Type: application/json
> Transfer-Encoding: chunked
> Server: Jetty(6.1.18)

* Connection #0 to host localhost left intact
* Closing connection #0
true
```
9.3.2.8 `getServerLogs`

return job server logs by means of `jobid`. It is mapped with ProActive Parallel Suite `/scheduler/jobs/[jobid]/log/server` operation

```bash
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/205/log/server
* Trying to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/205/log/server HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
=============== Job 205 logs ===============
[2013-04-13 12:38:58,506 DEBUG] job 205 has just been paused !
[2013-04-13 13:36:24,420 INFO ] job 205 request to change the priority to Low
[2013-04-13 13:36:24,443 DEBUG] job 205 event [Job change piority]
[2013-04-13 13:37:16,695 INFO ] job 205 request to change the priority to Low
[2013-04-13 13:39:12,799 INFO ] job 205 request to change the priority to Normal
```
[2013-04-13 13:39:52,448 INFO ] job 205 request to change the priority to Normal
[2013-04-13 13:41:14,826 INFO ] job 205 request to change the priority to Normal
[2013-04-13 13:42:20,861 INFO ] job 205 request to change the priority to Highest
[2013-04-13 13:50:16,707 INFO ] job 205 request to change the priority to Lowest
[2013-04-13 14:10:01,139 DEBUG] job 205 has just been resumed!
[2013-04-13 14:10:01,161 DEBUG] job 205 event [Job resumed]
[2013-04-13 14:10:13,644 INFO ] job 205 listening logs
=============== Task 2050003 logs ===============
[2013-04-13 14:10:01,472 INFO ] task 2050003 scheduling
[2013-04-13 14:10:03,484 INFO ] task 2050003 scheduling
[2013-04-13 14:10:05,495 INFO ] task 2050003 scheduling
[2013-04-13 14:10:07,507 INFO ] task 2050003 scheduling
[2013-04-13 14:10:09,518 INFO ] task 2050003 scheduling
[2013-04-13 14:10:11,530 INFO ] task 2050003 scheduling
[2013-04-13 14:10:13,541 INFO ] task 2050003 scheduling
[2013-04-13 14:10:15,553 INFO ] task 2050003 scheduling
[2013-04-13 14:10:17,565 INFO ] task 2050003 scheduling
[2013-04-13 14:10:19,577 INFO ] task 2050003 scheduling
[2013-04-13 14:10:21,589 INFO ] task 2050003 scheduling
=============== Task 2050002 logs ===============

================ Task 2050001 logs =================

Cannot retrieve logs for task 2050000

================ Task 2050007 logs =================
Cannot retrieve logs for task 2050007

================ Task 2050006 logs =================
[2013-04-13 14:10:01,472 INFO ] task 2050006 scheduling
[2013-04-13 14:10:03,484 INFO ] task 2050006 scheduling
[2013-04-13 14:10:05,495 INFO ] task 2050006 scheduling
[2013-04-13 14:10:07,507 INFO ] task 2050006 scheduling
[2013-04-13 14:10:09,518 INFO ] task 2050006 scheduling
[2013-04-13 14:10:11,530 INFO ] task 2050006 scheduling
[2013-04-13 14:10:13,541 INFO ] task 2050006 scheduling
[2013-04-13 14:10:15,553 INFO ] task 2050006 scheduling
[2013-04-13 14:10:17,565 INFO ] task 2050006 scheduling
[2013-04-13 14:10:19,577 INFO ] task 2050006 scheduling
[2013-04-13 14:10:21,589 INFO ] task 2050006 scheduling

================ Task 2050004 logs =================

================ Task 2050005 logs =================

* Connection #0 to host localhost left intact
* Closing connection #0
9.3.2.9  **getLiveLogs**

return only the currently available logs of job identified by the id \( \text{jobid} \). It is mapped with ProActive Parallel Suite /scheduler/jobs/[jobid]/livelog operation.

```

* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/6/livelog HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> * Connection #0 to host localhost left intact
```
9.3.2.10 **removeLiveLogs**

return true if the live logs of job identified by the id `jobid` have been disabled. It is mapped with ProActive Parallel Suite `/scheduler/jobs/{jobid}/livelog` operation

```bash
bash-4.2$ curl -v -X GET -H sessionid:`curl -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/205/livelog
```

* About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1... connected
  * Connected to localhost (127.0.0.1) port 8080 (#0)
  > DELETE /rest/rest/scheduler/jobs/205/livelog HTTP/1.1
  > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  > libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  > libssh2/1.2.7
  > Host: localhost:8080
  > Accept: */*
  > sessionid:400a658813dff1aba3433b92a2034ac7a124c32400a658813dff
  > laba348000
  >
  < HTTP/1.1 200 OK
  < Content-Type: application/json
  < Transfer-Encoding: chunked
  < Server: Jetty(6.1.18)
  <
  * Connection #0 to host localhost left intact
  * Closing connection #0
  true
```

9.3.2.11 **getJobResult**

return the job result associated to `jobid`. It is mapped with ProActive Parallel Suite `/scheduler/jobs/{jobid}/result` operation

```bash
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
```

http://localhost:8080/rest/rest/scheduler/jobs/205/result | python -mjson.tool
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/205/result HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:400a658813df1aba3433b92a2034ac7a124c32400a658813df1aba348000
>
< HTTP/1.1 204 No Content
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
{
  "allResults": {
    "task1": {
      "flowAction": null,
      "id": {
        "id": 2050004,
        "readableName": "task1"
      },
      "jobClasspath": null,
      "output": {
        "loggerName": "logger.scheduler.205",
        "serializedAllEvents": "eNq1U8tu00AUvTR9pEI1mj5g61IhhFDs2Amtk0o0ErSikSLk24qIITH1DI4be8Ydj5t0w5oNf8CS2bdIIH6HLr4ve66KvdtAl2Xh49H1mTvnnDs++wkTgYS1I3J
CzEyFkemFvMeoF62q9n7Tevthw5c5EHSHweAHFJXhAxMkhC/y8xIBLujM01C0
xNBEPJg64Rx9fvXt1Vv+LKZg5kDWI6p/1wkp54QvSzZY8dZKBk9gKBn/mitBdM
qjF1bkTjxYNyniGCvn6sMc7y8LRCKy1s8NpK41EbyIkEskLBd/g7oWvQ2nWu
TRrKM1CaZZ3jYSbp4EKvGv+761dWv9Y21XkKMG LillyK0Y3gPOQ+KknHKUNwr1qy
kYBflGdWvjFAt7aIyhxXf filthyTs0tiV5WvUnOvkGQ10bvwSjhTvPHlkfDxUsIIttg0
kxxJ80iXFWF2gkv3g1o0ui19+LZdaF177mYGwzbZ1Jb6Dx1Z7a7mCvKTLs1ooN
ks6KpgEI/jljZ1LBlMy41jtBMeM9LvrcuKyMd0WqFmzZdce0l1zTdvCtoBDSp
v7ChxNhxElUq1IXrPbgf1VpiGuu16j9brfsKVwMWXcoq9UrjBjvVSaShK0Y45
ChcFPkpL2QpqgQi+wc/6qhWGE0Y80pBo5t+Suoa57veEgwdQ62MwjYbRYaky2"
"serializedAllEvents": "eNq1U01P1FAuvTigAwWOOn7otEmOMmXbaDkwZEplEITKpmDdjmjio312yrTvlsfAGTaa/RH+AuPwROpFceG/4D3DQQY2YMlnr7ent53zrvmn3/CRJbC0hE7YXqOYa570ehx3w0znHnUft66+F+AWCQ9GcBoEDUFZkG0kuY1+V6JIPakK4loewKiAhFsH3Cbf7+9W0V51c2cjb9AMux7z2VyakrZS9P9v1xHgbcP4B54XujtRZMYRjznRnU4cWHgYbgMz7usZgjLlhKoRExERhtTGMrTeJEklhFLvinUYIdUmacSbNGEoZJSrhXiHRRRSPnhCwC3oq18ZzlnWnRHuaCgAWSdgyzvoeBCEoXC5yTub8yFvDz8jR2UJ85Je28MkszV2VeCnrQ8v15L5yorjYe6NgcJRQu3njw0XhcRloq8dFTP6Es/j2hlr5kJBhv94IOaMLtYvfyw9Wwj++FmBsbB6Yjyfwd5gFMyd1Lkph1ZeQPkg2m2gBK/SJhama7bCJNPtLtcC0vC9ITsC+2lMt6VgSLmhmhuWbg47umnRHaEK+K31Rgw41jMqto15tZWzdU822pUbM9br9dp2Ha3KpbBq5dqpiVKvVJq2M0JiJEcn428iyXugjtKIfgCl8BFsayc6RgxRwRp31q362tWg9G861AnzdEaWodnqLU7z7b39xG2WpzWzyX8q2Sw2DGEIXMOsbnwy6jHFXVeoj3p86xDo/9x2YYyNScxy7mNkUEi8mOvVvVkpucjJyFbBHQV1BXMK5hUsKFhUsKrQgWchd6928fNW5ITd72t9uPlX+r5t7HU4iBf3mpq3JvLkLQ7N2ho8Af38WcW" },
"previewerClassName": null,
"propagatedProperties": null,
"serializedException": null,
"serializedValue": "r00ABXQ4AKU5vlJjggaKgZnJvbsBUXYNr1DgJlHnNZB0IGZvciaYMCBTZWNvbmRz",
"taskDuration": -1
},

"jobInfo": {
"finishedTime": 1366457036049,
"jobId": {
"id": 205,
"readableName": "NOT SET"
}
},

"modifiedTasks": null,
"numberOfFinishedTasks": 8,
"numberOfPendingTasks": 0,
"numberOfRunningTasks": 0,
"owner": "admin",
"priority": "NORMAL",
"removedTime": -1,
"startTime": 1366457014844,
"status": "FINISHED",
"submittedTime": 1366457011569,
"taskFinishedTimeModify": null,
"taskStatusModify": null,
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
},
"preciousResults": {
"task8": {
"flowAction": null,
"id": {
"id": 2050001,
"readableName": "task8"
},
"jobClasspath": null,
"output": {
"loggerName": "logger.scheduler.205",
"serializedAllEvents": "eNq1U01P1FAvTlqAwMO7ntEmOMxXbaDkw2Ep1E1kpmDDjhmjio312yrTv1faWG7au/RH+AwPWOPEfctG/4D3DQQYlMn7ent5zrvmv3/CRJbc0hE7YXqOYa570ehx3w0znHnUftN6++F+AWCQ9Mc0eEDUFZkG0kuYI1+V6JIPakZ4ioe7KIAhFsH3CBf7+9W0VSl+2CjB9AMux7z2VykzyS9P9vlxHqbcP4B54XujtRZYRjzNl4cWlGH8Dm7us2gjLlKoREEXRhtTGmrTeJElmFLvinURY0umacSnBGEOzSJrXhXIRRR5PnghCw3oq18Zl1nWRRrUacGAWGdzvleoBCOeXC5yTuBc8yFvdz8jR2U858Je28MksV2VcNrzQ8vl15L5yorjYe6NgcJRQu3njw0XhcR1ogd8FTP6Es/j2h1r5kJBB94OlALtYufyf69W0Wj++FmBsb6Yjyfwd5qFMydylKph12eQPkgZm2gBK/SJhma7bcCJNP1LtCC0VC91TsC+2imat6VGS1MmhuWbq47umnRHaEU+k31Rqwn4ljMgto15tWZadU822pUbM9br9dsp2Ha3KpbPq85dqPiKVvVjqZM0JijECn428iyXujjTKiFg1C8bFsaYc6RgRH9p131q362tWg4RS61AnzEaW0dnqLU7z7b39xG2GpSSeuWvYFXCGsBdZxxw2DFPExMOsbmWv6YhFFXeoj3p86xiDo/9x2YyyNScxy7mNKEi18emW0vVvKpCUjCyFbBH4QVlBXMK5hUsKfHUsKrgWcHd6928fNW5ITd72t9uP1X+r575H41bf3mpq3jvL4kLQ7N2h08Af30WcW"},

"previewerClassName": null,
"propagatedProperties": null,
"serializedException": null,
"serializedValue": "rO0ABXQAKU5vLjggaGkgZnJvbSBUYXNRdGJlHnsZXB0IGZvciAyMCBTZWNvb mRz",

"taskDuration": -1
}
}
}
9.3.2.12  **killJob**

kill the job execution by providing *jobid*. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/kill operation. In order to show how it works, we had submitted the same job of the beginning again (notice the *jobid* change)

```
bash-4.2$ curl -v -X PUT -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
 http://localhost:8080/rest/rest/scheduler/jobs/206/kill
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/jobs/206/kill HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:400a658813dff1aba3433432a2034ac7a124c32400a658813dff1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```

9.3.2.13  **deleteJob**

delete the job by providing *jobid*. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid} operation. In order to show how it works, we had submitted the same job of the beginning again (notice the *jobid* change)

```
bash-4.2$ curl -v -X DELETE -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/login`
 http://localhost:8080/rest/rest/scheduler/jobs/4
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
```
* Connected to localhost (127.0.0.1) port 8080 (#0)
> DELETE /rest/rest/scheduler/jobs/4 HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> 
> sessionid:400a658813dff1aba3433432a2034ac7a124c32400a658813dff
labasha48000
> 
> < HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
< 
> * Connection #0 to host localhost left intact
> * Closing connection #0
true

9.3.2.14 **getJobsInfo**

return a subset of the Scheduler state. It is mapped with ProActive Parallel Suite /scheduler/jobsinfo operation

bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin"
http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobsinfo | python -mjson.tool

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
  % Total  % Received % Xferd
  Average Speed  Time  Time  Time  Current
        --:--:-- --:--:-- --:--:-- --:--:--
 Left  Speed
  0 0 0 0 0 0 0 0 0--:--:-- --:--:--
  --:--:-- 0connected

* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobsinfo HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
Future Internet Core Platform

> Host: localhost:8080
> Accept: */*

> sessionid:400a658813dff1aba34f8f2ac7a124c32400a658813dff1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)

{
  [data not shown]

100 24936 0 24936 0 0 106k 0 --:--:-- --:--:--
--:--:-- 106k

* Connection #0 to host localhost left intact
* Closing connection #0

[
  {
    "jobOwner": "admin",
    "jobId": "205",
    "jobinfo": {
      "finishedTime": -1,
      "jobId": {
        "id": 205,
        "readableName": "job_8_tasks"
      },
      "modifiedTasks": null,
      "numberOfFinishedTasks": 0,
      "numberOfPendingTasks": 0,
      "numberOfRunningTasks": 0,
      "owner": "admin",
      "priority": "LOWEST",
      "removedTime": -1,
      "startTime": -1,
      "status": "PENDING",
      "submittedTime": 1365849531328,
      "taskFinishedTimeModify": null,
      "taskStatusModify": null,
  }
]
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
}
},
{
"jobOwner": "admin",
"jobid": "204",
"jobinfo": {
   "finishedTime": -1,
"jobId": {
     "id": 204,
     "readableName": "job_8_tasks"
   },
"modifiedTasks": null,
"numberOfFinishedTasks": 0,
"numberOfPendingTasks": 0,
"numberOfRunningTasks": 0,
"owner": "admin",
"priority": "NORMAL",
"removedTime": -1,
"startTime": -1,
"status": "PENDING",
"submittedTime": 1365849377892,
"taskFinishedTimeModify": null,
"taskStatusModify": null,
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
}
},
{
"jobOwner": "admin",
"jobid": "203",
"jobinfo": {
   "finishedTime": -1,
"jobId": {
     "id": 203,
"readableName": "job_8_tasks"
},
"modifiedTasks": null,
"numberOfFinishedTasks": 0,
"numberOfPendingTasks": 0,
"numberOfRunningTasks": 0,
"owner": "admin",
"priority": "NORMAL",
"removedTime": -1,
"startTime": -1,
"status": "PENDING",
"submittedTime": 1365849114526,
"taskFinishedTimeModify": null,
"taskStatusModify": null,
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
}
,
{
  "jobOwner": "admin",
  "jobid": "201",
  "jobinfo": {
    "finishedTime": -1,
    "jobId": {
      "id": 201,
      "readableName": "job_8_tasks"
    },
    "modifiedTasks": null,
    "numberOfFinishedTasks": 0,
    "numberOfPendingTasks": 0,
    "numberOfRunningTasks": 0,
    "owner": "admin",
    "priority": "NORMAL",
    "removedTime": -1,
    "startTime": -1,
    "status": "PENDING",
    "submittedTime": 1365848893330,
9.3.3 Tasks

9.3.3.1 getTaskState
get task state, identified by taskname. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/tasks/{taskname} operation. We assume here that we submit a new job and pause it, according submitJob and pauseJob operations. Then, thanks to getTasks, we fetch a task name, such as task8, and access its related information:

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
% Total    % Received % Xferd Average Speed   Time    Time     Time     Current
Dload  Upload   Total   Spent    Left  Speed
0     0     0     0     0     0     0     0     0     0 --:--:-- --:--:-- --:--:--
--:--:-- 0connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/207/tasks/task8 HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
> libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
> libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813dff1aba3417652a2034ac7a124c32400a658813dff1aba348000
```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.18)

{ [data not shown]
100 767 0    767 0    0    3149    0 --:--:-- --:--:--
--:--:--  3169
* Connection #0 to host localhost left intact
* Closing connection #0
{
  "cScript": null,
  "cancelJobOnError": false,
  "dependenceIds": [],
  "dependences": [],
  "description": "This task will sleep 20s",
  "flowBlock": "none",
  "flowScript": null,
  "genericInformations": {},
  "inputFiles": null,
  "maxNumberOfExecution": 1,
  "maxNumberOfExecutionOnFailure": 2,
  "name": "task8",
  "outputFiles": null,
  "parallelEnvironment": null,
  "postScript": null,
  "preScript": null,
  "preciousLogs": false,
  "preciousResult": true,
  "restartTaskOnError": { 
    "value": {
      "description": "Anywhere",
      "index": 1
    }
  },
  "resultPreview": null,
  "runAsMe": false,
  "sScripts": false,
}
As expected, its status is "PAUSED".

9.3.3.2 **restartTask**

restart the task. It is mapped with ProActive Parallel Suite
/scheduler/jobs/{jobid}/tasks/{taskname}/restart operation

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/jobs/69/tasks/task8/restart
HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> 
sessionid:7827570013fd2d6fa434b4c2a2034ac7a124c327827570013fd2d6fa438000
```
**9.3.3.3 *preemptTask***

*preempt a task within a job. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/tasks/{taskname}/preempt operation*

```bash
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/jobs/67/tasks/task8/preempt
HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:7827570013fd2d6fa4354c52a2034a7124c327827570013fd2d6fa438000

>
< HTTP/1.1 200 OK
< Content-Type: */*
< Content-Length: 4
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```
9.3.3.4 **getTaskResult**

return the task result and related logs. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/tasks/{taskname}/result operation.

```bash
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin"
http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/209/tasks/task8/result | python -mjson.tool

> GET /rest/rest/scheduler/jobs/61/tasks/task8/result HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
>           libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
>           libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:7827570013fd2d6fa4364042a2034ac7a124c327827570013fd2d6fa438000
> < HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{ [data not shown]
100 128 0 128 0 0 808 0 --:--:-- --:--:-- --:--:--
--:--:-- 815
* Connection #0 to host localhost left intact
* Closing connection #0
{
  "id": {
    "id": 610000,
    "readableName": "task8"
  },
  "serializedValue": "rQ0ABXQAKU5vLjgggaGkgZnJvbSBUYXNrIDgJIHNsZXB0IGZvciAyMCBTZWNvbmRz"
}
```

Moreover, in order to access further details, such as its logs (standard error output, standard output, all of them), serialized ant not-serialized value, the following mapped sub-operations are considered, respectively:

**getTaskResultAsErrorLogs** with ProActive Parallel Suite

/scheduler/jobs/{jobid}/tasks/{taskname}/result/log/err sub-operation

```

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/61/tasks/task8/result/log/err HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
[task8@192.168.122.1;15:05:42] Task 8 : Test STDERR

* Connection #0 to host localhost left intact
* Closing connection #0
```

**getTaskResultAsOutputLogs** with ProActive Parallel Suite

/scheduler/jobs/{jobid}/tasks/{taskname}/result/log/out sub-operation

```

* About to connect() to localhost port 8080 (#0)
```
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/61/tasks/task8/result/log/out
HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:7827570013fd2d6fa434e762a2034ac7a124c327827570013fd2
d6fa438000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
[task8@192.168.122.1;15:05:42] Task 8 : Test STDOUT

[task8@192.168.122.1;15:06:02] Terminate task number 8

* Connection #0 to host localhost left intact
* Closing connection #0

* getTaskResultAsAllLogs with ProActive Parallel Suite
/scheduler/jobs/{jobid}/tasks/{taskname}/result/log/all sub-operation

bash-4.2$ curl -v -X GET -H sessionid: `curl -v -X POST -d
"username=admin&password=admin"
http://localhost:8080/rest/rest/scheduler/login`
http://localhost:8080/rest/rest/scheduler/jobs/209/tasks/task8
/result/log/all

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/61/tasks/task8/result/log/all
HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:7827570013fd2d6fa434eb92a2034ac7a124c327827570013fd2d6fa438000
> HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
< [task8@192.168.122.1;15:05:42] Task 8 : Test STDERR

[task8@192.168.122.1;15:05:42] Task 8 : Test STDOUT

[task8@192.168.122.1;15:06:02] Terminate task number 8

* Connection #0 to host localhost left intact
* Closing connection #0

- `getTaskResultAsSerializedValue` with ProActive Parallel Suite

```
```

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/61/tasks/task8/result/serializedvalue HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:7827570013fd2d6fa4350602a2034ac7a124c327827570013fd2d6fa438000
```

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/jobs/61/tasks/task8/result/value

HTTP/1.1
User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
Host: localhost:8080
Accept: */*

> sessionid:7827570013fd2d6fa434f232a2034ac7a124c327827570013fd2d6fa438000

> HTTP/1.1 200 OK
< Content-Type: */*
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0

No.8 hi from Task 8  slept for 20 Seconds

- **getTaskResultAsNotSerializedValue** with ProActive Parallel Suite
  /scheduler/jobs/[jobid]/tasks/[taskname]/result/value sub-operation
9.3.3.5 **killTask**

kill a task within a job. It is mapped with ProActive Parallel Suite `/scheduler/jobs/{jobid}/tasks/{taskname}/kill` operation

```bash

* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> PUT /rest/rest/scheduler/jobs/62/tasks/task8/kill HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:7827570013fd2d6fa435e0c2a2034ac7a124c327827570013fd2d6fa438000
>
< HTTP/1.1 200 OK
< Content-Type: */*
< Content-Length: 4
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
```

9.3.4 **Dataspaces**

Dataspaces are the place where users can push/pull/delete data to be processed. As described in the Dataspaces section, there are several type of them: each one with its own purpose, but all of them placed by default in your local filesystem (see [here](#)):  

- the **GLOBALSPACE** is a virtual place, under the Scheduler service control domain, shared among all the users, where anyone has the read and write permissions;
- the **USERSPACE** is a virtual place, under the Scheduler service control domain, whose access and manipulation is limited to the user in question only;
• the *INPUTSPACE* and *OUTPUTSPACE* are virtual, additional places where users can put/pull/delete data keeping them under their own control domain, which might be remote with respect to the Scheduler host location. Those add flexibility to fit the needs of the most exigent users, who cannot/do not want move their data from their premises.

Stated that, the access to the parametrized \( \{\text{spaceName}\} \) resource, we will see in a while, can be achieve by specifying a subset of those dataspaces: \( \{\text{GLOBALSPACE} \mid \text{USERSPACE}\} \).

9.3.4.1 **pushData**

pushes a file from the local file system into the given DataSpace. It is mapped with ProActive Parallel Suite \( /\text{scheduler/dataspace/\{spaceName\}/filePath} \) operation

```
```

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
     100       0    0  60  100    36      4      2  0:00:18
  00:14  00:04     5
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/scheduler/dataspace/GLOBALSPACE/input
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
> libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
> libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:7876921142e63f522a64852a2034ac7a124c327876921142e63f522a8000
> Content-Length: 25534
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=--------------------------a877f34009ad
>```
9.3.4.2 **pullData**

either pulls a file from the given DataSpace to the local file system or lists the content of a directory, if the path refers to a directory. It is mapped with ProActive Parallel Suite /scheduler/dataspace/[spaceName][filePath] operation. In the case the path to a file is given, the content of this file will be returned as an input stream:

```
```

---

D.4.2.3 FI-WARE User and Programmer Guide

Page 157
> sessionid:7876921142e63f522a63b12a2034ac7a124c327876921142e63f522a8000
>
< HTTP/1.1 200 OK
< Content-Type: application/octet-stream
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
  [data not shown]
  100 25226 0 25226 0 0 138k 0 --:--:-- --:--:--
  --:--:-- 139k
* Connection #0 to host localhost left intact
* Closing connection #0

otherwise the input stream returned will be a text stream where each line lists the content of the directory.


% Total % Received % Xfered Average Speed Time Time
Time  Current Dload Upload Total Spent
Left  Speed
100 96 0 60 100 36 4 2 0:00:18
0:00:13 0:00:05  6
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/scheduler/dataspace/GLOBALSPACE/output HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:7876921142e63f522a64352a2034ac7a124c327876921142e63f522a8000
>
< HTTP/1.1 200 OK
9.3.4.3 **deleteData**

deletes a file or recursively delete a directory from the given DataSpace. It is mapped with ProActive Parallel Suite `/scheduler/dataspace/{spaceName}/{filePath}` operation.

```bash

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> DELETE
/rest/rest/scheduler/dataspace/GLOBALSPACE/output/proactive.png HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> 
> sessionid:7876921142e63f522a635d2a2034ac7a124c327876921142e63f522a8000
>
< HTTP/1.1 200 OK
< Content-Type: */*
< Content-Length: 4
< Server: Jetty(6.1.18)
```
**9.3.5 Resource Manager Service**

**9.3.5.1 login**

enable user to access the RM with his credentials. It is mapped with ProActive Parallel Suite `/rm/login` operation

```
bash-4.2$ curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/rm/login

* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/rm/login HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
> 
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
1d871fb013dfdbb013557a912a2034ac7a124c321d871fb013dfdbb01358000
```
9.3.5.2 **loginWithCredential**

enable the user to login to the scheduler by submitting a multipart form including the credential file with field name `credential`. It is mapped with ProActive Parallel Suite /rm/login operation.

```
bash-4.2$ curl -v -X POST -F credential=@$JOB_SCHEDULER_GE_HOME/scheduling/config/authentication/admin.cred http://localhost:8080/rest/rest/rm/login

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/rm/login HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
> libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
> libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=--------------------------e4a4927fcla4
>
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
```

9.3.5.3 **disconnect**

allow the user to disconnect from the RM. It has been mapped with ProActive Parallel Suite /rm/disconnect operation.

```
bash-4.2$ curl -v -X POST -H sessionid:`curl -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/rm/login`
http://localhost:8080/rest/rest/rm/disconnect
```
9.3.5.4 *isActive*

test if the RM is operational. It has been mapped with ProActive Parallel Suite
/rm/isactive operation


* Connection #0 to host localhost left intact
* Closing connection #0
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/isactive HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
> Host: localhost:8080
> Accept: */*

< HTTP/1.1 204 No Content
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
9.3.5.5 shutdownRM

kill the RM. It has been mapped with ProActive Parallel Suite /rm/shutdown operation, which requires permissions as administrator.

If the Resource Manager Service has been launched via jrunscript tool, you can skip what just follows and jump in performing the curl command.

Otherwise, that is if you leveraged the default batch/shell scripts shipped with our implementation, this action will shut the Scheduler Service down, as well, since they share a common centralized rmi registry, owned by the RM. In order to avoid such a behaviour, an advanced configuration of rmi protocol, which assigns for each service a different rmiregistry, listening to a different port, is needed:

```
bash-4.2$ cd $JOB_SCHEDULER_GE_HOME/scheduling/bin/unix

bash-4.2$ ./rm-start -ln -Dproactive.rmi.port=1099 -Dpa.scheduler.home=$JOB_SCHEDULER_GE_HOME/scheduling -Dpa.rm.home=$JOB_SCHEDULER_GE_HOME/scheduling
Starting the resource manager...

RM URL : rmi://192.168.122.1:1099/
Starting the scheduler...
Connecting to the resource manager on rmi://192.168.122.1:1099/
The scheduler created on rmi://192.168.122.1:2000/
```
bash-4.2$ ./jetty-launcher -A
$JOB_SCHEDULER_GE_HOME/scheduling/dist/war/rest.war -r
Dproactive.rmi.port=2001 -
Dpa.scheduler.home=$JOB_SCHEDULER_GE_HOME/scheduling -
Dpa.rm.home=$JOB_SCHEDULER_GE_HOME/scheduling

Deploying REST Server in /tmp/tmp.DbhIq3im8H/rest/
Jetty Launcher logs to /tmp/tmp.B236uTUSWi
Deployed application: http://localhost:8080/rest

Afterwards, we can proceed by submitting our operation request:

bash-4.2$ curl -v -X GET -H sessionid:curl -v -X POST -d
"username=admin&password=admin"
http://localhost:8080/rest/rest/rm/login
http://localhost:8080/rest/rest/rm/shutdown

* Connection #0 to host localhost left intact
* Closing connection #0
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/shutdown HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:1d871fb013dfdbb0135778d2a2034ac7a124c321d871fb013dfd
bb0135b000
>
> < HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
*
* Connection #0 to host localhost left intact
* Closing connection #0
true
As administrator, you might check that, by typing `ps -eaf | grep RMStarter`, on the machine that hosts the Job Scheduler GE, you will not get any process running. We remark that such action will not affect the Scheduler Service, since it will be still available (check it via `ps -eaf | grep SchedulerStarter`).

9.3.5.6 **getRMInfo**

retrieve specific RM information by furnishing the *name* of available resource. It has been mapped with ProActive Parallel Suite `/rm/info/(name)` operation. Since accessible information depend on the technology is used at back-end, how to query is a matter of formatting the parameters properly. In our case, the choice falls on *JMX* and *MBeans* technology as stated in ProActive Resource Manager.

```
```

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/info/ProActiveResourceManager:name=Management?attr=AccountingCacheValidityTimeInSeconds HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:41fa548513dfebdf3eb7f5e2a2034ac7a124c3241fa548513dfb3eb8000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
{
    "name": "AccountingCacheValidityTimeInSeconds",
```
9.3.5.7 **getMonitoring**

get the initial state of the RM, included current deployed Node Sources, Node and Policies info. It has been mapped with ProActive Parallel Suite `/rm/monitoring` operation

```bash

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
% Total  % Received  % Xferd Average Speed Time Time  Time Current
               0      0  0.00+0.00+0.00+0.00 0.00+0.00+0.00+0.00-
--:--:-- --:--:-- connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/monitoring HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:41fa548513dfebd5ebf3eb7ee32a2034ac7a124c3241fa548513dfebd5eb8000
>
> < HTTP/1.1 200 OK
> < Content-Type: application/json
> < Transfer-Encoding: chunked
> < Server: Jetty(6.1.18)
> <
> { [data not shown]}
```
100 3672 0 3672 0 0 245k 0 --:--:-- --:--:-- --:--:-- 275k

* Connection #0 to host localhost left intact
* Closing connection #0

{  
  "nodeSource": [
    {
      "counter": 0,
      "eventType": null,
      "nodeSourceAdmin": "rm",
      "rmurl": null,
      "sourceDescription": "Infrastructure: org.ow2.proactive.resourcemanager.nodesource.infrastructure.LocalInfrastructure@6479b43f, Policy: Restart Down Nodes Policy user access type [ALL], provider access type [ALL]",
      "sourceName": "LocalNodes",
      "timeStamp": 0,
      "timeStampFormatted": "1/1/70 1:00 AM"
    }
  ],
  "nodesEvents": [
    {
      "counter": 0,
      "defaultJMXUrl": "service:jmx:rmi:///jndi/rmi://192.168.122.1:56987/rmnode",
      "eventType": null,
      "hostName": "192.168.122.1",
      "nodeOwner": null,
      "nodeProvider": "rm",
      "nodeSource": "LocalNodes",
      "nodeState": "FREE",
      "nodeUrl": "rmi://192.168.122.1:1099/local-LocalNodes-1",
      "padname": "",
      "previousNodeState": null,
    }
  ]
}
    "rmurl": null,
    "timeStamp": 1365778441281,
    "timeStampFormatted": "4/12/13 4:54 PM",
    "vmname": "rmi://192.168.122.1:1099/PA_JVM442274898",
    "vnName": null
 },
{ "counter": 0,
    "eventType": null,
    "hostName": "192.168.122.1",
    "nodeOwner": null,
    "nodeProvider": "rm",
    "nodeSource": "LocalNodes",
    "nodeState": "FREE",
    "nodeUrl": "rmi://192.168.122.1:1099/local-LocalNodes-0",
    "padname": "",
    "previousNodeState": null,
    "rmurl": null,
    "timeStamp": 1365778441256,
    "timeStampFormatted": "4/12/13 4:54 PM",
    "vmname": "rmi://192.168.122.1:1099/PA_JVM993096019",
    "vnName": null
 },
{ "counter": 0,
{"defaultJMXUrl": "service:jmx:rmi:///jndi/rmi://192.168.122.1:38458/rmnode",
"eventType": null,
"hostName": "192.168.122.1",
"nodeInfo": "Node local-LocalNodes-3
URL: rmi://192.168.122.1:1099/local-LocalNodes-3
Node source: LocalNodes
Provider: rm
Used by: nobody
State: Free
"nodeOwner": null,
"nodeProvider": "rm",
"nodeSource": "LocalNodes",
"nodeState": "FREE",
"nodeUrl": "rmi://192.168.122.1:1099/local-LocalNodes-3",
"padname": "",
"previousNodeState": null,
"rmurl": null,
"timeStamp": 1365778441272,
"timeStampFormatted": "4/12/13 4:54 PM",
"vmname": "rmi://192.168.122.1:1099/PA_JVM1749982671",
"vnName": null
},
{
 "counter": 0,
 "defaultJMXUrl": "service:jmx:rmi:///jndi/rmi://192.168.122.1:41539/rmnode",
 "eventType": null,
 "hostName": "192.168.122.1",
 "nodeInfo": "Node local-LocalNodes-2
Node source: LocalNodes
Provider: rm
Used by: nobody
State: Free
 "nodeOwner": null,
 "nodeProvider": "rm",
 "nodeSource": "LocalNodes",
}
9.3.5.8  **getRMStatHistory**

return the statistic history. It has been mapped with ProActive Parallel Suite /rm/stathistory operation

```bash
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/stathistory?range=s HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
```
> sessionid:41fa548513dfebf3eb7d262a2034ac7a124c3241fa548513dfeb3eb8000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<

{ [data not shown]
100 262 0 262 0 0 12831 0 --:--:-- --:--:-- --:--:-- --:--:-- 13789
* Connection #0 to host localhost left intact
* Closing connection #0
{
    "AvailableNodesCount": [
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
        4,
    ],
    "AverageActivity": [
        0,
        0,
        0,
        0,
        0,
    ]
}
<table>
<thead>
<tr>
<th>BusyNodesCount</th>
<th>DownNodesCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</td>
</tr>
</tbody>
</table>
9.3.5.9 getRMState

return the current RM state. It has been mapped with ProActive Parallel Suite /rm/state operation

* About to connect() to localhost port 8080 (#0)
9.3.5.10 \textbf{getRMVersion} \\
return the current REST server API and RM version. It has been mapped with ProActive Parallel Suite /rm/version operation

9.3.6 Node Source

9.3.6.1 `getSupportedInfrastructures`

return the list of supported node source infrastructures descriptors. It is mapped with ProActive Parallel Suite `/rm/infrastructures` operation.

In order to dig into available infrastructures, we suggest to refer to Organizing Your Nodes section of ProActive Resource Manager documentation.
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d 
"username=admin&password=admin"
http://localhost:8080/rest/rest/rm/login`
http://localhost:8080/rest/rest/rm/infrastructures | python -m json.tool
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
% Total    % Received % Xferd
Average Speed  Time   Time  Time  Current
Dload  Upload  Total  Spent  Left  Speed
0     0    0     0    0     0      0      0
--:--:-- --:--:-- --:--:-- 0 connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/infrastructures HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
sessionid:41fa548513dfebdf3eb7ba92a2034ac7a124c3241fa548513dfe
bdf3eb8000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
    [data not shown]
100 15050 0 15050 0 0 29236 0 --:--:-- --:--:--
--:--:-- 29337

* Connection #0 to host localhost left intact
* Closing connection #0

[

    "configurableFields": [

        {
            "meta": {
                "description": "The URL of the resource manager",
                "type": "NONE"
            }
        }
    ]
]
},
  "name": "rmUrl",
  "value": "rmi://192.168.122.1:1099/"
},
{
  "meta": {
    "description": "Absolute path to credentials file\nused to add the node to the Resource Manager",
    "type": "CREDENTIAL"
  },
  "name": "credentials",
  "value": ""
},
{
  "meta": {
    "description": "Maximum number of nodes\nto\nde deployed on Resource Manager machine",
    "type": "NONE"
  },
  "name": "maxNodes",
  "value": "4"
},
{
  "meta": {
    "description": "in ms. After this timeout\nexpired\nthe node is considered to be lost",
    "type": "NONE"
  },
  "name": "nodeTimeout",
  "value": "5000"
},
{
  "meta": {
    "description": "Additionnal ProActive\nproperties",
    "type": "NONE"
  },
  "name": "paProperties",
  "value": ""
}
"meta": {
  "description": "The URL of the resource manager",
  "type": "NONE"
},
"name": "rmUrl",
"value": "rmi://192.168.122.1:1099/"
},
{
  "meta": {
    "description": "Absolute path of the file containing the list of remote hosts",
    "type": "FILEBROWSER"
  },
  "name": "hostsList",
  "value": ""
},
{
  "meta": {
    "description": "in ms. After this timeout expired the node is considered to be lost",
    "type": "NONE"
  },
  "name": "nodeTimeOut",
  "value": ""
"value": "60000"
],
{
"meta": {
"description": "Maximum number of failed attempt to deploy on a host before discarding it",
"type": "NONE"
},
"name": "maxDeploymentFailure",
"value": "5"
},
{
"meta": {
"description": "An interpreter that executes the script",
"type": "NONE"
},
"name": "interpreter",
"value": "bash"
},
{
"meta": {
"description": "A script that deploys a node on host (parameters: host, node, ns names and rm url).",
"type": "FILEBROWSER"
},
"name": "deploymentScript",
"value": ""
},
{
"meta": {
"description": "A script that removes a node (parameters: host name and node url",
"type": "FILEBROWSER"
},
"name": "removalScript",
"value": ""
}]}
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Creates remote runtimes using custom scripts",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.CLIInfrastructure"
},

"configurableFields": [
{
  "meta": {
    "description": "The URL of the resource manager",
    "type": "NONE"
  },
  "name": "rmUrl",
  "value": "rmi://192.168.122.1:1099/"
}
],
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Default infrastructure",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.DefaultInfrastructureManager"
},

"configurableFields": [
{
  "meta": {
    "description": "The URL of the resource manager",
    "type": "NONE"
  },
  "name": "rmUrl",
  "value": "rmi://192.168.122.1:1099/"
```json
{
    "meta": {
        "description": "Absolute path of the file containing the list of remote hosts",
        "type": "FILEBROWSER"
    },
    "name": "hostsList",
    "value": ""
},
{
    "meta": {
        "description": "in ms. After this timeout expired the node is considered to be lost",
        "type": "NONE"
    },
    "name": "nodeTimeOut",
    "value": "60000"
},
{
    "meta": {
        "description": "Maximum number of failed attempt to deploy on a host before discarding it",
        "type": "NONE"
    },
    "name": "maxDeploymentFailure",
    "value": "5"
},
{
    "meta": {
        "description": "Options for the ssh command to log in the remote hosts",
        "type": "NONE"
    },
    "name": "sshOptions",
    "value": ""
},
{
    "meta": {
        "description": "
```


```
{
  "meta": {
    "description": "Absolute path of the java\nexecutable on the remote hosts",
    "type": "NONE"
  },
  "name": "javaPath",
  "value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
},
{
  "meta": {
    "description": "Absolute path of the Resource Manager (or Scheduler)\nroot directory on the remote hosts",
    "type": "NONE"
  },
  "name": "schedulingPath",
  "value": "/home/lcantelm/git/build/dist/scheduling"
},
{
  "meta": {
    "description": "Linux, Cygwin or Windows depending on\nthe operating system of the remote hosts",
    "type": "NONE"
  },
  "name": "targetOs",
  "value": "Linux"
},
{
  "meta": {
    "description": "Options for the java command\nlaunching the node on the remote hosts",
    "type": "NONE"
  },
  "name": "javaOptions",
  "value": ""
}}
```
"description": "Absolute path of the credential file",
        "type": "CREDENTIAL"
    },
    "name": "rmCredentialsPath",
    "value": ""
},
],
"defaultValues": {
    "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Creates remote runtimes using SSH",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.SSHInfrastructure"
},
{
    "configurableFields": [
    {
        "meta": {
            "description": "The URL of the resource manager",
            "type": "NONE"
        },
        "name": "rmUrl",
        "value": "rmi://192.168.122.1:1099/"
    },
    {
        "meta": {
            "description": "Absolute path of the java\nexecutable on the remote hosts",
            "type": "NONE"
        },
        "name": "javaPath",
        "value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
    },
    {
        "meta": {
            "description": "Absolute path of the credential file",
            "type": "CREDENTIAL"
        },
        "name": "rmCredentialsPath",
        "value": ""
    }
]
"description": "Options for the ssh command used to log in the batch system head node",
"type": "NONE"
},
"name": "sshOptions",
"value": ""
},
{
"meta": {
"description": "Absolute path of the Resource Manager (or Scheduler) directory on the remote hosts",
"type": "NONE"
},
"name": "schedulingPath",
"value": "/home/lcantelm/git/build/dist/scheduling"
},
{
"meta": {
"description": "Options for the java command on launching the node on the remote hosts",
"type": "NONE"
},
"name": "javaOptions",
"value": ""
},
{
"meta": {
"description": "The maximum number of nodes to be requested to the batch system",
"type": "NONE"
},
"name": "maxNodes",
"value": "1"
},
{
"meta": {
"description": "in ms. After this timeout expired the node is considered to be lost",
"type": "NONE"
},
"name": "nodeTimeout",
"value": "600000"
}
"type": "NONE"
),
"name": "nodeTimeOut",
"value": "300000"
},
{
  "meta": {
    "description": "The batch system\nnode name or IP address",
    "type": "NONE"
  },
  "name": "nodeName",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the credential file",
    "type": "CREDENTIAL"
  },
  "name": "rmCredentialsPath",
  "value": ""
},
{
  "meta": {
    "description": "Options for the\njob submission command",
    "type": "NONE"
  },
  "name": "submitJobOpt",
  "value": ""
},
{
  "meta": {
    "description": "Fully qualified classname\nof the implementation",
    "type": "NONE"
  },
  "name": "implementationClassname",
"value": ""
}
{
  "meta": {
    "description": "Absolute path to the\nclass file of the implementation",
    "type": "FILEBROWSER"
  },
  "name": "implementationFile",
  "value": ""
}
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Acquires nodes from a GENERIC
resource manager."
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.GenericBatchJobInfrastructure"
}
{
  "configurableFields": [
    {
      "meta": {
        "description": "The URL of the resource
manager",
        "type": "NONE"
      },
      "name": "rmUrl",
      "value": "rmi://192.168.122.1:1099/"
    },
    {
      "meta": {
        "description": "",
        "type": "FILEBROWSER"
      },
      "name": "descriptor",
      "value": ""
    }
  ]
{ "defaultValues": { "rmUrl": "rmi://192.168.122.1:1099/" },

"pluginDescription": "[DEPRECATED] Infrastructure described in GCM deployment descriptor",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMInfrastructure"
 },

"configurableFields": [
  {
    "meta": {
      "description": "The URL of the resource manager",
      "type": "NONE"
    },
    "name": "rmUrl",
    "value": "rmi://192.168.122.1:1099/"
  },
  {
    "meta": {
      "description": "",
      "type": "FILEBROWSER"
    },
    "name": "descriptor",
    "value": ""
  },
  {
    "meta": {
      "description": "List of host to use for the deployment",
      "type": "FILEBROWSER"
    },
    "name": "hostsList",
    "value": ""
  }
]}
{  
  "meta": {  
    "description": "Timeout after which one  
    the node\nis considered to be lost",  
    "type": "NONE"
  },  
  "name": "timeout",  
  "value": "60000"
},  
"defaultValues": {  
  "rmUrl": "rmi://192.168.122.1:1099/"
},  
"pluginDescription": "[DEPRECATED] Handles hosts from  
    the list using specified gcm deployment descriptor\ntemplate  
    with HOST java variable contract (see proactive  
    documentation)",  
  "pluginName":  
    "org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCMCustomisedInfrastructure"
},  
  "configurableFields": [  
  
  {  
    "meta": {  
      "description": "The URL of the resource  
      manager",
      "type": "NONE"
    },  
    "name": "rmUrl",  
    "value": "rmi://192.168.122.1:1099/"
  },  
  {  
    "meta": {  
      "description": "Absolute path of the  
      java\nexecutable on the remote hosts",
      "type": "NONE"
    },  
    "name": "javaPath",
  },
"value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
},
{
  "meta": {
    "description": "Options for the ssh command used to log in the batch system head node",
    "type": "NONE"
  },
  "name": "sshOptions",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the Resource Manager (or Scheduler)\nroot directory on the remote hosts",
    "type": "NONE"
  },
  "name": "schedulingPath",
  "value": "/home/lcantelm/git/build/dist/scheduling"
},
{
  "meta": {
    "description": "Options for the java command\nlaunching the node on the remote hosts",
    "type": "NONE"
  },
  "name": "javaOptions",
  "value": ""
},
{
  "meta": {
    "description": "The maximum number of nodes\nto be requested to the batch system",
    "type": "NONE"
  },
  "name": "maxNodes",
  "value": "1"


```
{
  "meta": {
    "description": "After this timeout expired, the node is considered to be lost",
    "type": "NONE"
  },
  "name": "nodeTimeOut",
  "value": "300000"
},
{
  "meta": {
    "description": "The batch system head node name or IP address",
    "type": "NONE"
  },
  "name": "serverName",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the credential file",
    "type": "CREDENTIAL"
  },
  "name": "rmCredentialsPath",
  "value": ""
},
{
  "meta": {
    "description": "Options for the job submission command",
    "type": "NONE"
  },
  "name": "submitJobOpt",
  "value": ""
}
}
```


"rmUrl": "rmi://192.168.122.1:1099/",

"pluginDescription": "Acquires nodes from a LSF resource manager."

"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.LSFInfrastructure"

"configurableFields": [

  {
    "meta": {
      "description": "The URL of the resource manager",
      "type": "NONE"
    },
    "name": "rmUrl",
    "value": "rmi://192.168.122.1:1099/"
  },
  {
    "meta": {
      "description": "Virtual Infrastructure Type: nxenserver, virtualbox, vmware, hyperv-winrm or hyperv-wmi",
      "type": "NONE"
    },
    "name": "infrastructure",
    "value": ""
  },
  {
    "meta": {
      "description": "Hypervisor's url",
      "type": "NONE"
    },
    "name": "VMMUrl",
    "value": ""
  }
]
"description": "Hypervisor's user",
    "type": "NONE"
},
"name": "VMMUser",
"value": ""
},
{
    "meta": {
        "description": "Hypervisor's user's password",
        "type": "PASSWORD"
    },
    "name": "VMMPwd",
    "value": ""
},
{
    "meta": {
        "description": "Template virtual machine's name",
        "type": "NONE"
    },
    "name": "VMTemplate",
    "value": ""
},
{
    "meta": {
        "description": "The maximum number of vm",
        "type": "NONE"
    },
    "name": "VMMax",
    "value": "0"
},
{
    "meta": {
        "description": "The number of node per virtual machine",
        "type": "NONE"
    },
    "name": "hostCapacity",
```json
{
  "meta": {
    "description": "ProActive Configuration file path",
    "type": "FILEBROWSER"
  },
  "name": "PAConfig",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the rm.cred file",
    "type": "CREDENTIAL"
  },
  "name": "RMCredentials",
  "value": ""
},
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Virtualized Infrastructure node acquisition",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.VirtualInfrastructure"
}
{
  "configurableFields": [
    {
      "meta": {
        "description": "The URL of the resource manager",
        "type": "NONE"
      },
      "name": "rmUrl",
```
"value": "rmi://192.168.122.1:1099/",
,
{
  "meta": {
    "description": "Absolute path of the java\nexecutable on the remote hosts",
    "type": "NONE"
  },
  "name": "javaPath",
  "value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
},
{
  "meta": {
    "description": "Options for the ssh command used\nto log in the batch system head node",
    "type": "NONE"
  },
  "name": "sshOptions",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the Resource Manager (or Scheduler)\nroot directory on the remote hosts",
    "type": "NONE"
  },
  "name": "schedulingPath",
  "value": "/home/lcantelm/git/build/dist/scheduling"
},
{
  "meta": {
    "description": "Options for the java command\nlaunching the node on the remote hosts",
    "type": "NONE"
  },
  "name": "javaOptions",
  "value": ""
}
},
{
"meta": {
    "description": "The maximum number of nodes\nto be requested to the batch system",
    "type": "NONE"
},
"name": "maxNodes",
"value": "1"
},
{
"meta": {
    "description": "in ms. After this timeout expired\nthe node is considered to be lost",
    "type": "NONE"
},
"name": "nodeTimeOut",
"value": "300000"
},
{
"meta": {
    "description": "The batch system\nnode name or IP adress",
    "type": "NONE"
},
"name": "serverName",
"value": ""
},
{
"meta": {
    "description": "Absolute path of the credential file",
    "type": "CREDENTIAL"
},
"name": "rmCredentialsPath",
"value": ""
}
"description": "Options for the `submitJob` submission command",
    "type": "NONE"
},
    "name": "submitJobOpt",
    "value": "-l \"nodes=1:ppn=1\"
  }
]
,"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Acquires nodes from a PBS resource manager.",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.PBSInfrastructure"
},
{
  "configurableFields": [
    {
      "meta": {
        "description": "The URL of the resource manager",
        "type": "NONE"
      },
      "name": "rmUrl",
      "value": "rmi://192.168.122.1:1099/"
    },
    {
      "meta": {
        "description": "Absolute path of EC2 configuration file",
        "type": "FILEBROWSER"
      },
      "name": "configurationFile",
      "value": ""
    },
    {
      "meta": {
        "description": "The URL of the resource manager",
        "type": "NONE"
      },
      "name": "rmUrl",
      "value": "rmi://192.168.122.1:1099/"
    }
"description": "Absolute path of the credential file",
  "type": "CREDENTIAL"
},
"name": "RMCredentialsPath",
"value": ""
},
{
  "meta": {
    "description": "The communication protocol the remote node",
    "type": "NONE"
  },
  "name": "communicationProtocol",
  "value": "pamr"
},
{
  "meta": {
    "description": "Additional JVM options \n Ex: -Dproperty1=value1 -Dproperty2=value2",
    "type": "NONE"
  },
  "name": "additionalJVMOptions",
  "value": ""
}
],
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/
},
"pluginDescription": "Handles nodes from the Amazon Elastic Compute Cloud Service.",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.EC2Infrastructure"
}
"description": "The URL of the resource manager",
    "type": "NONE"
},
"name": "rmUrl",
"value": "rmi://192.168.122.1:1099/
},
{
    "meta": {
        "description": "Maximum number of nodes to deploy",
        "type": "NONE"
    },
    "name": "maxNodes",
    "value": "1"
},
{
    "meta": {
        "description": "Url of the WinHPC web service",
        "type": "NONE"
    },
    "name": "serviceUrl",
    "value": "https://<computerName>/HPCBasicProfile"
},
{
    "meta": {
        "description": "Username for windows scheduler connection",
        "type": "NONE"
    },
    "name": "userName",
    "value": ""
},
{
    "meta": {
        "description": "Password for windows scheduler connection",
        "type": "NONE"
    },
    "name": "password",
    "value": ""
}
"type": "PASSWORD"
},
"name": "password",
"value": ""
},
{
"meta": {
   "description": "Name of the trustStore",
   "type": "FILEBROWSER"
},
"name": "trustStore",
"value": ""
},
{
"meta": {
   "description": "Password for the trustStore",
   "type": "PASSWORD"
},
"name": "trustStorePassword",
"value": ""
},
{
"meta": {
   "description": "Absolute path of the java\nexecutable on the remote hosts",
   "type": "NONE"
},
"name": "javaPath",
"value": "/home/lcantelm/application/jdk1.6.0_43/jre/bin/java"
},
{
"meta": {
   "description": "Absolute path of the Resource Manager\nroot directory on the remote hosts",
   "type": "NONE"
},
"name": "rmPath",
"value": ""
"value": "/home/lcantelm/git/build/dist/scheduling",
}
{
  "meta": {
    "description": "Absolute path of the credential file",
    "type": "CREDENTIAL"
  },
  "name": "RMCredentialsPath",
  "value": ""
},
{
  "meta": {
    "description": "Options for the java command\n launching the node on the remote hosts",
    "type": "NONE"
  },
  "name": "javaOptions",
  "value": ""
},
{
  "meta": {
    "description": "Additional classpath for the java command\n launching the node on the remote hosts",
    "type": "NONE"
  },
  "name": "extraClassPath",
  "value": ""
},
{
  "meta": {
    "description": "in ms. After this timeout expired\n the node is considered to be lost",
    "type": "NONE"
  },
  "name": "timeout",
  "value": "60000"}
getSupportedPolicies

return the list of supported node source policies descriptors. It is mapped with ProActive Parallel Suite /rm/policies operation. About available types of policies, check the Node Source Policies link.
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
  [data not shown]
100 9562 0 9562 0 0 18634 0 --:--:-- --:--:-- --:--:-- 18639
* Connection #0 to host localhost left intact
* Closing connection #0
[

    "configurableFields": [
      {
        "meta": {
          "description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
          "type": "NONE"
        },
        "name": "userAccessType",
        "value": "ALL"
      },
      {
        "meta": {
          "description": "ME|users=name1,name2;groups=group1,group2|ALL",
          "type": "NONE"
        },
        "name": "providerAccessType",
        "value": "ME"
      },
      {
        "meta": {
          "description": "",
          "type": "NONE"
        },
        "name": "schedulerUrl",
        "value": ""
      }
    ]
}
{{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "nodeDeploymentTimeout",
    "value": "2400000"
}
],
"defaultValues": {
    "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Allocates as many resources as scheduler required according to loading factor. Releases resources smoothly.",
"pluginName": "org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.EC2Policy"
},
{
    "configurableFields": [
    {
        "meta": {
            "description": "ME\|users=name1,name2;groups=group1,group2;tokens=t1,t2\|ALL",
            "type": "NONE"
        },
        "name": "userAccessType",
        "value": "ALL"
    },
    {
        "meta": {
            "description": "ME\|users=name1,name2;groups=group1,group2\|ALL",
            "type": "NONE"
        },
        "name": "providerAccessType",
        "value": "ME"
    }
}
Future Internet Core Platform

```json
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "acquireTime",
    "value": "4/12/13 5:38:46 PM CEST"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "releaseTime",
    "value": "4/12/13 6:38:46 PM CEST"
},
{
    "meta": {
        "description": "ms (1 day by default)",
        "type": "NONE"
    },
    "name": "period",
    "value": "86400000"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "preemptive",
    "value": "true"
}
},
"defaultValues": {
    "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Acquires and releases nodes at specified time."
}```
"configurableFields": [ 
  
  "meta": {
    "description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
    "type": "NONE"
  },
  
  "name": "userAccessType",
  "value": "ALL"
},

{ 
  
  "meta": {
    "description": "ME|users=name1,name2;groups=group1,group2|ALL",
    "type": "NONE"
  },
  
  "name": "providerAccessType",
  "value": "ME"
}
],
  
  "defaultValues": {
    
    "rmUrl": "rmi://192.168.122.1:1099/"
  },

  "pluginDescription": "Static nodes acquisition.",
  "pluginName": 
  "org.ow2.proactive.resourcemanage
r.nodesource.policy.StaticPolicy"
},

{ 
  
  "configurableFields": [ 
    
    "meta": {
      
      "description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
      "type": "NONE"
    }
  ]
{
    "name": "userAccessType",
    "value": "ALL"
},
{
    "meta": {
        "description": "ME|users=name1,name2;groups=group1,group2|ALL",
        "type": "NONE"
    },
    "name": "providerAccessType",
    "value": "ME"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "schedulerUrl",
    "value": ""
},
{
    "meta": {
        "description": "refresh frequency (ms)",
        "type": "NONE"
    },
    "name": "refreshTime",
    "value": "1000"
}
"meta": {
    "description": "",
    "type": "NONE"
  },
  "name": "minNodes",
  "value": "0"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "maxNodes",
    "value": "10"
},
{
    "meta": {
        "description": "number of tasks per node",
        "type": "NONE"
    },
    "name": "loadFactor",
    "value": "10"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "nodeDeploymentTimeout",
    "value": "10000"
},
{
    "meta": {
        "description": "Time since the nodes acquisition is allowed (crontab format)",
        "type": "NONE"
    },
    "name": "acquisitionAllowed",
    "value": ""
"value": "* * * * *
},
{
  "meta": {
    "description": "Time since the nodes acquisition is forbidden (crontab format)",
    "type": "NONE"
  },
  "name": "acquisitionForbidden",
  "value": "* * * * *
},
{
  "meta": {
    "description": "the mode how nodes are removed",
    "type": "NONE"
  },
  "name": "preemptive",
  "value": "false"
},
{
  "meta": {
    "description": "If true acquisition will be immediately allowed",
    "type": "NONE"
  },
  "name": "allowed",
  "value": "false"
}
],
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Triggers new nodes acquisition when scheduler is overloaded within a time slot defined in crontab syntax."
"pluginName": "org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.CronLoadBasedPolicy"
{  
  "configurableFields": [  
  
  
   
  
   
  
  
  
   
  
  
  ]  
}


"description": "refresh frequency (ms)",
"type": "NONE"
},
"name": "refreshTime",
"value": "1000"
},
{
"meta": {
"description": "",
"type": "NONE"
},
"name": "minNodes",
"value": "0"
},
{
"meta": {
"description": "",
"type": "NONE"
},
"name": "maxNodes",
"value": "10"
},
{
"meta": {
"description": "number of tasks per node",
"type": "NONE"
},
"name": "loadFactor",
"value": "10"
},
{
"meta": {
"description": "",
"type": "NONE"
},
"name": "nodeDeploymentTimeout",
"value": "10000"
}
"defaultValues": {
   "rmUrl": "rmi://192.168.122.1:1099/"
},

"pluginDescription": "Allocates as many resources as scheduler required according into loading factor. Releases resources smoothly.",

"pluginName": "org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.SchedulerLoadingPolicy"
},

{ "configurableFields": [
   {
      "meta": {
         "description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
         "type": "NONE"
      },
      "name": "userAccessType",
      "value": "ALL"
   },
   {
      "meta": {
         "description": "ME|users=name1,name2;groups=group1,group2|ALL",
         "type": "NONE"
      },
      "name": "providerAccessType",
      "value": "ME"
   },
   {
      "meta": {
         "description": "",
         "type": "NONE"
      },
      "name": "schedulerUrl",
      "value": ""
   }
],
"meta": {  
    "description": "ME|users=name1,name2;groups=group1,group2|ALL",
    "type": "NONE"
},
"name": "providerAccessType",
"value": "ME"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "schedulerUrl",
    "value": ""
},
{
    "meta": {
        "description": "refresh frequency (ms)",
        "type": "NONE"
    },
    "name": "refreshTime",
    "value": "1000"
},
{
    "meta": {
        "description": "",
        "type": "NONE"
    },
    "name": "minNodes",


```
{"value": "0"},
{
"meta": {
    "description": ",",
    "type": "NONE"
},
"name": "maxNodes",
"value": "10"
},
{
"meta": {
    "description": "number of tasks per node",
    "type": "NONE"
},
"name": "loadFactor",
"value": "10"
},
{
"meta": {
    "description": "",
    "type": "NONE"
},
"name": "nodeDeploymentTimeout",
"value": "10000"
},
{
"meta": {
    "description": "Time when all nodes are deployed (crontab format)",
    "type": "NONE"
},
"name": "deployAllAt",
"value": "* * * * *"
},
{
"meta": {

```
"description": "Time when all nodes are removed and the policy starts watching the scheduler loading",
"type": "NONE"
},
"name": "undeployAllAt",
"value": "* * * * *"
},
{
"meta": {
"description": "the mode how nodes are removed",
"type": "NONE"
},
"name": "preemptive",
"value": "false"
},
{
"meta": {
"description": "If true the policy will acquire all nodes immediately",
"type": "NONE"
},
"name": "acquireNow",
"value": "false"
}
],
"defaultValues": {
"rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Keeps all nodes up and running within specified time slot and acquires node on demand when scheduler is overloaded at another time.",
"pluginName": "org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.CronSlotLoadBasedPolicy"
},
{
"configurableFields": [
{
"meta": {

"description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
   "type": "NONE"
 },
 "name": "userAccessType",
 "value": "ALL"
 },
 {   "meta": {
 "description": "ME|users=name1,name2;groups=group1,group2;tokens=t1,t2|ALL",
 "type": "NONE"
 },
 "name": "providerAccessType",
 "value": "ME"
 },
 {   "meta": {
 "description": "ms (30 mins by default)",
 "type": "NONE"
 },
 "name": "checkNodeStateEach",
 "value": "1800000"
 }
],
 "defaultValues": {
   "rmUrl": "rmi://192.168.122.1:1099/"
 },
 "pluginDescription": "Static nodes acquisition. If node becomes down policy tries to restart it."
 "pluginName": "org.ow2.proactive.resourcemanager.nodesource.policy.RestartDownNodesPolicy"
 }
},
 {   "configurableFields": [
 {   "meta": {

9.3.6.3 `createNodeSource`

create a new node source in the RM, specifying infrastructure and policy, with related parameters. It is mapped with ProActive Parallel Suite `/rm/nodesource/create` operation.

In order to successfully perform the nodes source instantiation, we will leverage the admin credentials, previously generated and stored in `$JOB_SCHEDULER_GE_HOME/scheduling/config/authentication/admin.cred` file:

```bash
bash-4.2$ curl -v -X POST -H "sessionid:\`curl -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/rm/login`\" -d "nodeName=LocalNSTest" -d "infrastructureType=org.ow2.proactive.resourcemanager.nodesource.policy.CronPolicy"
```
The increase of the computing nodes number at your disposal can be observed by `getRMState`. 
9.3.6.4 **removeNodeSource**
remove a new node source from the RM. It is mapped with ProActive Parallel Suite /rm/nodesource/remove operation

```bash
bash-4.2$ curl -v -X POST -d "name=LocalNSTest&preemp=true" -H sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/rm/login`
http://localhost:8080/rest/rest/rm/nodesource/remove
* About to connect() to localhost port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/rm/nodesource/remove HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:425e53bb13e22b3f42e71fa2a2034ac7a124c32425e53bb13e22b3f42e8000
> Content-Length: 23
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
> * Connection #0 to host localhost left intact
* Closing connection #0
true
```

The decrease of the computing nodes number at your disposal can be observed by getRMState.

9.3.7 **Nodes**
In order to manage the computing nodes, you should get the necessary information accessible by getMonitoring operation, such as the *nodeUrl*.

9.3.7.1 **lockNode**
prevent other users from using a set of locked nodes. It is mapped with ProActive Parallel Suite /rm/node/lock operation
9.3.7.2 **unlockNode**

allow other users to use a set of nodes previously locked. It is mapped with ProActive Parallel Suite `/rm/node/unlock` operation

http://localhost:8080/rest/rest/rm/node/unlock

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest/rm/node/unlock HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu)
  libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22
  libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
> sessionid:400a658813dff1aba347c082a2034ac7a124c32400a658813dff
  laba348000
> Content-Length: 125
> Content-Type: application/x-www-form-urlencoded
> < HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest.rm/node/unlock HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*
>
> sessionid:400a658813dff1aba347bb12a2034ac7a124c32400a658813dff1aba348000
> Content-Length: 125
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true

9.3.7.3 **releaseNode**

release a node, previously reserved for computation. If the node is locked, it will force to unlock it. It is mapped with ProActive Parallel Suite /rm/node/release operation "defaultJMXUrl":

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /rest/rest.rm/node/release HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
9.3.7.4 **addNode**

add a node to a particular node source. If not specified, add it to the default node source of the RM. It is mapped with ProActive Parallel Suite `/rm/node` operation. In order to accomplish that, we first launch a new local *free* node, thanks to the embedded *ProActive Node Agent*.

```
bash-4.2$ ./rm-start-node -nodeName NEW-NODE
--- StartNode ---------------------------------------------

Detecting a network interface to bind the node
Using default value for the number of add node attempts: 10
Using default value for the add node attempts delay: 5000
Using default value for the rm ping delay: 30000
Using default value for the wait on join: 60000
Reconfigured log4j using
file:$JOB_SCHEDULER_GE_HOME/scheduling/config/log4j/log4j-defaultNode
Rank is not set. Previous URLs will not be stored
Logging to org.slf4j.impl.Log4jLoggerAdapter(org.mortbay.log) via org.mortbay.log.Slf4jLog
jetty-6.1.18
Started SelectChannelConnector@0.0.0.0:49828
```
Remote Object Factory provider <pamr, class org.objectweb.proactive.extensions.pamr.remoteobject.PAMRRemoteObjectFactory> found
Remote Object Factory provider <pnp, class org.objectweb.proactive.extensions.pnp.PNPRemoteObjectFactory> found
Remote Object Factory provider <pnps, class org.objectweb.proactive.extensions.pnpssl.PNPSslRemoteObjectFactory> found
Remote Object Factory provider <pamrd, class org.objectweb.proactive.extensions.pamrd.PAMRDRemoteObjectFactory> found
Remote Object Factory provider <amqp, class org.objectweb.proactive.extensions.amqp.remoteobject.AMQPRemoteObjectFactory> found
Remote Object Factory provider <amqp-federation, class org.objectweb.proactive.extensions.amqp.federation.AMQPFederationRemoteObjectFactory> found
Remote Object Factory provider <rmissl, class org.objectweb.proactive.extra.rmissl.RmiSslRemoteObjectFactory> found
Detected an existing RMI Registry on port 1099
You don't seem to be running the latest released version of ProActive
Version you are using: 2013-03-19, latest version: 5.3.2
To download the latest release, please visit
http://www.activeeon.com/community-downloads
To disable this check, set the proactive.runtime.ping property to false
... ...
URL of this node rmi://192.168.122.1:1099/NEW-NODE

then, by opening another terminal, type the following in order to bind the node to the Default Infrastructure (designed to be used with ProActive agent):


* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
Only, now the node is *registered*, that means *visible* from the RM perspective. Test it via `isNodeAvailable` as confirmation.

### 9.3.7.5 `isNodeAvailable`

test if a node is registered to the RM. It is mapped with ProActive Parallel Suite `/rm/node/isavailable` operation

```
   * About to connect() to localhost port 8080 (#0)
   * Trying 127.0.0.1...
   *   connected
   * Connected to localhost (127.0.0.1) port 8080 (#0)
   > GET /rest/rest/rm/node/isavailable?nodeurl=rmi://192.168.122.1:1099/NEW-NODE HTTP/1.1
   > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
   > Host: localhost:8080
```
9.3.7.6 **removeNode**

remove a node from the RM. It is mapped with ProActive Parallel Suite /rm/node/remove operation

```bash

* About to connect() to localhost port 8080 (#0)
  * Trying 127.0.0.1... connected
  * Connected to localhost (127.0.0.1) port 8080 (#0)

> POST /rest/rest/rm/node/remove HTTP/1.1
> User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) libcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
> Host: localhost:8080
> Accept: */*

> sessionid:400a658813dff1aba347c6d2a2034ac7a124c32400a658813dff1aba348000
> Content-Length: 51
> Content-Type: application/x-www-form-urlencoded
>
> HTTP/1.1 200 OK
> Content-Type: application/json
> Transfer-Encoding: chunked
```
< Server: Jetty(6.1.18)
<
* Connection #0 to host localhost left intact
* Closing connection #0
true
10 Edgelets - User and Programmers Guide

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

10.1 Introduction

This guide will give you a step by step procedure in order to add a new device to the edgelet networks and run one edgelet test instance on it.

10.1.1 Background and Detail

This User and Programmers Guide relates to the Edgelets GE which is part of the Cloud Hosting chapter. Please find more information about this Generic Enabler in the following Open Specification.

10.2 User Guide

This GE provides a portal that you can navigate to with a modern browser. By default it runs on port 8080 or the edgelet master.

Navigate to http://<masterip>:8080/portal to get a login form.

Depending if you are a user or a platform administrator you will get the following screens.

As an administrator:
- List of users
- Add a user
- Edgelets management (from the infrastructure manager point of view)

As a user:
- List of nodes and status
- Edgelet management (from a user point of view)

These panels are quite simple and mostly reflect the different API commands in a user friendly way.

10.3 Programmer Guide

10.3.1 Prerequisites

- You must be sure to have the master installed and running, see the Installation and administration guide

- You need a GNU/Linux server (preferably in the form of a VM) in order to act as a slave node:
  - python 2.6 or 2.7 installed

10.3.2 Slave node installation

We will first configure a new server in the master, then install it and connect it to the network. Please refer to the API Open Specifications document for a full list of the currently available APIs.
10.3.2.1 **API access**

10.3.2.2 **Login**
- Retrieve Edgelet master IP (or use localhost from within the master VM) and get the list of available servers. For convenience, we will use masterip as a shorthand.
- The use the login url to retrieve a token to be used with the next requests (token parameter to pass as X-Auth-Token HTTP header - it will not be shown in the next requests)

```
curl http://masterip:8080/login --date "user=<user>&pwd=<password>
```

10.3.2.3 **List the available servers**
- Retrieve Edgelet master IP (or use localhost from within the master VM) and get the list of available servers. For convenience, we will use masterip as a shorthand.

```
curl http://masterip:8080/servers
```

which will give you a list of currently available servers (a dummy one is already configured)

```
[{
  "title": "slapmasterclient1test",
  "reference": "COMP-0",
  "id": "20130709-2E91"
}]
```

10.3.2.4 **Register a new node**
- We will create a new server with the name "myserver"

```
curl --data "name=mynewnode" http://localhost:8080/servers
```

The return gives you a key and a certificate that you should saved for later. It will allow the node to authenticate against the master.

10.3.2.5 **Check**
- Re-query the list of servers, you server should be available now in the list
- It is now time to install it
10.3.2.6 **Installation of the node**
- Use a distribution supported for easier installation, and avoid compiling it from the sources. A .deb package is available for ubuntu 12.04

Retrieve the .deb package at <address to be updated with R3.3> and install it

```
# dpkg -i file.deb
```

Then install the dependencies

```
# apt-get install -f
```

10.3.2.7 **Configuration of the node**

10.3.2.7.1 **Store key and certificate**

You will need (as root) to create the configuration directory, then paste the key and the certificate you can see in the web page (opened in the previous step) into correct files. Here we use vim, but feel free to use any other editor (nano, gedit, ...):

```
# mkdir -p /etc/opt/slapos/ssl/partition_pki
# vim /etc/opt/slapos/ssl/computer.crt
# vim /etc/opt/slapos/ssl/computer.key
```

Then you can check that everything is okay:

```
/opt/slapos/parts/openssl/bin/openssl x509 -noout -in
/etc/opt/slapos/ssl/computer.crt
```

Should give you nothing as a result.

```
/opt/slapos/parts/openssl/bin/openssl rsa -noout -in
/etc/opt/slapos/ssl/computer.key -check
```

Should tell you everything is okay

10.3.2.7.2 **Configuration**

```
# cd /etc/opt/slapos/
# wget -O slapos.cfg
http://git.erp5.org/gitweb/slapos.core.git/blob_plain/HEAD:/slapos.cfg.example
# nano slapos.cfg
```

Then change "computer_id" and "interface_name".
It is now time to configure SlapOS. For this purpose, we create a `slapos.cfg` file with all configuration located in the previously created configuration directory. You will need to change the `computer_id` from the COMP-12345 to whichever value was supplied to you during certificate generation.

Set the "interface_name" option to your real interface name that you are using to connect to the internet. It should be eth0, eth1, wlan0... for most distribution, but you should still check by using the command:

```bash
# ifconfig
```

Like for the master installation, change the `local_ipv4_network` configuration if needed.

Specify how many computer partitions you need. 10 is a good start.

Then, restart the `slapos-node` daemon Ubuntu/Debian:

```bash
# /etc/init.d/slapos-node restart
```

### 10.3.3 Instantiation of a test edgelet

In this guide we give indication on how to instantiate a test edgelet via the API. Refer to the API open documentation to get a full list of available commands.

#### 10.3.3.1 Installation of the software release

We will first install the software release on the node, that is all the code needed to run an edgelet. This will get automatically downloaded and compiled from source.

For this release, specific edgelets SDK and API are not available, and we will install a dummy LAMP server that will a simple php page. Please note that this will no more be the case for the next releases, and also that the API should be improved, as the current behaviour is due to technical limitations. More precisely, the edgelets catalog is not available yet, which requires the use of not-so-user-friendly references and ids.

First, be sure to have the reference to the server we have installed ("COMP-XXX" - replace XXX accordingly). The id for our Software Release will be [https://raw.github.com/lbesson/slapos/master/software/simple-example2/software.cfg](https://raw.github.com/lbesson/slapos/master/software/simple-example2/software.cfg)

We ask for the installation of the software release using the API:

```bash
```

Check that the installation has been correctly requested, using the server id that you should replace in the following code (not the reference, aka COMP-XXX)

```bash
curl http://localhost:8080/server/20130709-2E91
```
The installation will now occur, which can take quite a long time, depending on network conditions and computer velocity. You can check the installation process by watching the logs

```
# tail -f /opt/slapos/slapos-node-software.log
```

Due to a bug, the software release stays with the "Installation requested" state.

### 10.3.3.2 Instanciation of the service

We can now request to instantiate the service on our server.

```
```

Again, you can check the advancement of the process via the logs

```
# tail -f /opt/slapos/slapos-node-instance.log
```

You can then get the endpoints from your edgelet instance. First retrieve the id

```
curl http://localhost:8080/services
```

Then retrieve the details (use a proper id)

```
curl http://localhost:8080/service/20130711-4F16
```

You can use the backend url from the parameters, and check that it correctly displays the page (a dummy IPv6 address is given here)

```
Trying 2001:470:1f14:169:75f0:da22:9afc:509b...
Escape character is '^[].
GET /
<html>
<head>
<title>Simple php test</title>
</head>
<body>
This is a simple php test
</body>
```


You can also of course use your browser to navigate to this webpage.