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

D.4.4.2: 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.4.4.2
Project Document Date: 2013-04-30
Deliverable Type and Security: Public
Author: FI-WARE Consortium
Contributors: FI-WARE Consortium
1.1 Executive Summary
This document contains a guide for users and programmers that aim at working with Generic Enabler developed within in 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.2), 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 FIWARE.

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). Moreover, IaaS Service Management (SM) GE provides the means to host complex applications potentially comprising multiple virtual machines and other runtime components (as outlined above), by automated provisioning and life cycle management of such compound applications (also called services), including elasticity and auto-scaling based on metrics collected by the Monitoring GE. Lastly, PaaS Management GE uses the above capabilities to offer provisioning and management of complete PaaS environments, leveraging also the Software Deployment and Configuration (SDC) GE which offers a flexible framework for installation and customization of software products within individual virtual machines.

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.

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

The following resources were used to generate this document:

D.4.4.2_User_and_Programmers_Guide_front_page
IaaS Data Center Resource Management - User and Programmers Guide
IaaS Service Management - User and Programmers Guide
Self-Service Interfaces - User and Programmers Guide
Object Storage - User and Programmers Guide
Software Management And Configuration - User and Programmers Guide
Monitoring - User and Programmers Guide
PaaS Management - User and Programmers Guide
Job Scheduler - User and Programmers Guide
1.6 Typographical Conventions

Starting with October 2012 the FIWARE project improved the quality and streamlined the submission process for deliverables, generated out of the public and private FIWARE wiki. The project is currently working on the migration of as many deliverables as possible towards the new system.

This document is rendered with semi-automatic scripts out of a MediaWiki system operated by the FIWARE consortium.

1.6.1 Links within this document

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

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

1.6.2 Figures

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

\[
\text{[[Image:....|size|alignment|Caption]]}
\]

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

1.6.3 Sample software code

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

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

1.7 Acknowledgements

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


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 of deliverable submission generated</td>
<td>2013-04-22</td>
<td>TID</td>
</tr>
<tr>
<td>v2</td>
<td>Consolidated version</td>
<td>2013-05-10</td>
<td>IBM</td>
</tr>
<tr>
<td>v3</td>
<td>Version after revision</td>
<td>2013-05-13</td>
<td>IBM</td>
</tr>
<tr>
<td>v4</td>
<td>Version for delivery</td>
<td>2013-05-14</td>
<td>TID</td>
</tr>
</tbody>
</table>

1.10  Table of Contents

1.1  Executive Summary .......................................................... 2
1.2  About This Document .......................................................... 2
1.3  Intended Audience ............................................................. 2
1.4  Chapter Context ................................................................. 2
1.5  Structure of this Document .................................................. 3
1.6  Typographical Conventions ................................................... 4
   1.6.1  Links within this document ............................................. 4
   1.6.2  Figures ......................................................................... 4
   1.6.3  Sample software code ..................................................... 4
1.7  Acknowledgements .................................................................... 4
1.8  Keyword list .............................................................................. 5
1.9  Changes History ........................................................................ 5
1.10  Table of Contents ..................................................................... 5

2  IaaS Data Center Resource Management - User and Programmers Guide .......... 9
   2.1  Introduction .......................................................................... 9
   2.2  Programmer Guide ............................................................... 9
   2.2.1  Accessing DCRM from the CLI ............................................ 9
   2.2.2  Accessing DCRM from a Browser ........................................ 15
### 3 IaaS Service Management - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>16</td>
</tr>
<tr>
<td>3.2</td>
<td>User Guide</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Programmer Guide</td>
<td>16</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Accessing Service Manager from the CLI</td>
<td>17</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Accessing Service Manager from a Browser</td>
<td>27</td>
</tr>
</tbody>
</table>

### 4 Self-Service Interfaces - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>User Portal User Guide</td>
<td>34</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Examples of Use</td>
<td>35</td>
</tr>
</tbody>
</table>

### 5 Object Storage - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>User Guide</td>
<td>41</td>
</tr>
<tr>
<td>5.3</td>
<td>Programmer Guide</td>
<td>41</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Programming against the CDMI interface</td>
<td>42</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Using CLI commands</td>
<td>43</td>
</tr>
<tr>
<td>5.4</td>
<td>References</td>
<td>44</td>
</tr>
</tbody>
</table>

### 6 Software Management And Configuration - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>45</td>
</tr>
<tr>
<td>6.2</td>
<td>User Guide</td>
<td>45</td>
</tr>
<tr>
<td>6.3</td>
<td>Programmer Guide</td>
<td>45</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Accessing Service Deployment and Configuration from the CLI</td>
<td>46</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Accessing Service Manager from a Browser</td>
<td>49</td>
</tr>
</tbody>
</table>

### 7 Monitoring - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>52</td>
</tr>
<tr>
<td>7.2</td>
<td>User Guide</td>
<td>52</td>
</tr>
<tr>
<td>7.3</td>
<td>Programmer Guide</td>
<td>52</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Accessing Monitoring Systems from the CLI</td>
<td>53</td>
</tr>
</tbody>
</table>

### 8 PaaS Management - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>59</td>
</tr>
<tr>
<td>8.2</td>
<td>User Guide</td>
<td>59</td>
</tr>
<tr>
<td>8.3</td>
<td>Programmer Guide</td>
<td>59</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Accessing Service Manager from the CLI</td>
<td>60</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Accessing Service Manager from a Browser</td>
<td>69</td>
</tr>
</tbody>
</table>

### 9 Job Scheduler - User and Programmers Guide

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>73</td>
</tr>
</tbody>
</table>

D.4.4.2: FI-WARE User and Programmers Guide
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>73</td>
</tr>
<tr>
<td>9.1.1</td>
<td>Recommended actions and assumptions before starting</td>
<td>73</td>
</tr>
<tr>
<td>9.2</td>
<td>User Guide</td>
<td>74</td>
</tr>
<tr>
<td>9.3</td>
<td>Programmer Guide</td>
<td>74</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Resource Manager Service</td>
<td>74</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Node Source</td>
<td>89</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Nodes</td>
<td>136</td>
</tr>
<tr>
<td>9.3.4</td>
<td>Scheduler Service</td>
<td>142</td>
</tr>
<tr>
<td>9.3.5</td>
<td>Jobs</td>
<td>160</td>
</tr>
<tr>
<td>9.3.6</td>
<td>Tasks</td>
<td>207</td>
</tr>
</tbody>
</table>
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.

Note: 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).

2.2 Programmer Guide

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

2.2.1 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]

2.2.1.1 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.

**nova-credential authorization**

```
source openrc
```

If you dont already have openrc file, download it from Horizon (Openstack's dashboard service):

- Log in to the dashboard with your username & password
- Go to 'settings' in the top right corner
- On the left menu select 'Openstack credentials'
- Select a project and click the 'Download RC File' button

**REST authorization**

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

```
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"}], "version": 1}]}}
```

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:

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

D.4.4.2: FI-WARE User and Programmers Guide
### 2.2.1.2 ResourceManager Placement APIs

**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:

```bash
```

**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</td>
</tr>
<tr>
<td>mode</td>
<td>String</td>
<td>NULL</td>
<td>enable/disable</td>
</tr>
</tbody>
</table>

Start PutInMaintenance using nova-client:

```bash
$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:

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

### 2.2.1.3 ResourceManager Network APIs

Quantum API is supported and can be found here: [Quantum developer guide](#) and here [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>
<tr>
<td>Action</td>
<td>/actions</td>
<td>Action used in Policy definitions</td>
</tr>
<tr>
<td>Filter</td>
<td>/filters</td>
<td>Specification used in Action definitions</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.

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-b609-1493dd9dac0",
        "ttl": 360
    }
}
```

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:
### 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**

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

    { 
      "Filter": { 
        "name": "Filter1",
        "flow": "HTTP,FTP",
        "action": "DROP",
      }
    }
```

---

**Parameter** | **Type**  | **Default** | **Comment**
---|---|---|---
name | String | NULL | Action name
flow | String | NULL | Flow description
action | String | NULL | Action type (Connectivity, QoS, Waypoint, Security, ACL, etc.)

**Sample Request Body : JSON Request**

```json
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",
      }
    }
```
2.2.1.4  **Mgmt Fabric APIs**

**Evacuate**

Here we will describe how to perform instance evacuation from the cli. 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 has to have pre configured shared storage with down host. As well, you have to validate that the current vm host is down. Otherwise the evacuation will fail with error.

Evacuate server to specified host and preserve user data

This is relevant only if both the source host and the target host are on the same shared storage, 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 host. The instance will be booted from a new disk, but will preserve the configuration, e.g. id, name, uid, ip...etc. 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.

<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.

```
+-----------------------------+
| Property | Value        |
+-----------------------------+
| adminPass | kRAJpErnT4xZ |
+-----------------------------+
```

Using REST API:

```
curl -s -X POST -H 'X-Auth-Token:TOKEN' -d '{"evacuate":null, "host":target_host_id}' -H
```
"Content-type: application/json"

```http
http://ENDPOINT/servers/{server_id}/action
```

### 2.2.1.5 Resource Manager Capacity APIs

**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 throttle/unthrottle</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.

### 2.2.2 Accessing DCRM from a Browser

To send http commands to DCRM using browser, use:

- Firefox RESTClient add-ons [5].
3 IaaS Service Management - User and Programmers Guide

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

3.1 Introduction

Welcome the User and Programmer Guide for the Service Manager Generic Enabler. This generic enabler is built on a proprietary solution using standard interface to communicate with, like OpenStack Compute API, 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.

3.2 User Guide

The OpenStack API reference offers a [1] page, which describes how to use the OpenStack Compute API (plus Identity and extensions from third parties not all included in the implementation currently) in details. It includes a bunch of commands, which can be used to manipulate instance in the Cloud.

Since OpenStack Compute API is based on the HTTP protocol there are a rich variety of tools, which can be used to access the Service Manager interface.

The following sections will go into more detail on how to user and develop against OpenStack Compute API.

3.3 Programmer Guide

OpenStack Compute 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 Service Manager through OpenStack Compute API.

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

```plaintext
* About to connect() to 130.206.80.91 port 8774 (#0)
* Trying 130.206.80.91 (130.206.80.91) port 8774 (#0)
* Connected to 130.206.80.91 (130.206.80.91) port 8774 (#0)
> GET /v2.0/FIWARE/vdc/c8da25c7a373473f8e8945f5b0da8217/flavors HTTP/1.1
> User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8r zlib/1.2.3
> Host: 130.206.80.91:8774
> Access-Control-Request-Method: GET
> Access-Control-Request-Headers: Content-Type, X-Auth-Token
```
In this case the HTTP GET operation is used. Please note the X-Auth-Token, which is needed by Service Manager for authentication and assignment of user to a project. OpenStack Compute API uses the Create Retrieve Update Delete (CRUD) operations, which map almost to the HTTP verbs POST, GET, PUT and DELETE.

OpenStack compute 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 a json response and vice versa.

3.3.1 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 create a VM could be summarized in the following list:
3.3.1.1 **Get a valid token**

It gets a valid token for the tenant that we have (It is not a Service Manager Operation itself it is a OpenStack Keystone operation).

```
curl -d '"
/providers SWITCH"' -H "Content-type: application/json" -H "Accept: application/xml"
http://130.206.80.100:35357/v2.0/tokens
```

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 OpenStack Keystone, 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="c8da25c7a373473f8e8945f5b0da8217"name=$TENNANT>
```

3.3.1.2 **Get the list of available flavors.**

```
curl -v -H "Access-Control-Request-Method: GET" -H "Access-
Control-Request-Headers: Content-Type, X-Auth-Token" -H "Origin:
http://130.206.80.93" -H 'X-Auth-Token:
a9a861db6276414094bc1567f664084d' -H "Content-Type:
application/xml" -H "Accept: application/xml" -X GET
"http://$SERVICEMANAGER_IP:8774/v2.0/c8da25c7a373473f8e8945f5b0da8217/flavors"
```

$SERVICEMANAGER_IP will be the IP in which the OpenStack Compute API is installed.

We obtained the Id of the flavor, and taking one for example:

```
<ns3:flavor id="001" name="M1_TINY 100Mb">
```
More functionalities can be seen on the OpenStack Compute API [3].

3.3.1.3  **Get the list of available images.**

```bash
```

We obtained the list of available images, in our example only one:

```json
{
    "images": [
        {
            "id":"16e3fe14-ecf6-1be3-edfa-fe180ee2e6f2",
            "name":"cirros-0.3.0-x86_64-uec",
            "link": [
                {
                    "rel":"self",
                    "href": "http://$SERVICEMANAGER_IP:8774/v2.0/FIWARE/vdc/c8da25c7a373473f8e8945f5b0da8217/images/16e3fe14-ecf6-1be3-edfa-fe180ee2e6f2"
                }
            ]
        }
    ]
}
```
3.3.1.4 Instantiate resources.
The Service Manager provides the same functionality to instantiate a server as the OpenStack does. However, there is also the possibility to instantiate a virtual appliance following the extension of the OpenStack Compute API that is provided by this GE.

Instantiate servers.

Create a server.


$SERVICEMANAGER_IP will be the IP on which the OpenStack Compute API is installed.

We obtained the following server details:

```json
{
    "name": "Server1",
    "id": "2f55dd1b-f582-411e-b5d2-35951ae91e74",
    "adminPass": "welcome",
    "metadata": null,
    "personality": null
}
```

Check that the server was created.


$SERVICEMANAGER_IP will be the IP on which the OpenStack Compute API is installed.

We obtained the following server details:
Instantiate virtual appliances.

Create of a virtual appliance inside defined project.

We need to create the descriptor file in OVF format in order to describe the virtual appliance. For this example we decide to deploy a virtual appliance that only have a server on it with the following characteristics:

Virtual Hardware Requirements: 2048Mb, 2 CPU, 1 disk, 1 nic

- Two different network configuration:
  - management for public connectivity
  - service for private connectivity
- Memory of 2048Mb
- 2 virtual CPUs
<InstantiateOvfParams name="tomcat"
xsi:schemaLocation="http://schemas.tcloud.telefonica.com/tcloud/1 tcloud.xsd"
xmlns="http://schemas.tcloud.telefonica.com/tcloud/1"
xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:vssd=http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
xmlns:rsrvr="http://schemas.telefonica.com/claudia/ovf"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<ovf:Envelope xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
xmlns="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:ovfenvelope="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:rsrvr="http://schemas.telefonica.com/claudia/ovf"
xmlns:vssd=http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:rif="http://www.w3.org/2007/rif#"
xsi:schemaLocation="http://schemas.telefonica.com/claudia/ovf reservoir.xsd">

  <ovf:References>
    <ovf:File ovf:id="tomcat"
      ovf:href="file:///admin.template.snapshotpaas"
      rsrvr:digest="e56ee64a-700a-47c0-addb-edda3867b749"/>
  </ovf:References>

  <ovf:DiskSection>
    <ovf:Disk ovf:diskId="tomcat" ovf:fileRef="tomcat"
      ovf:capacity="10" ovf:format="http://www.gnome.org/~markmc/qcow-image-format.html"/>
  </ovf:DiskSection>

  <ovf:NetworkSection>
    <ovf:Info>Network</ovf:Info>
    <ovf:Network ovf:name="management"
      rsrvr:public="true"/>
  </ovf:NetworkSection>

</ovf:Envelope>

</InstantiateOvfParams>
<ovf:Description>Network
gestion</ovf:Description>

<ovf:Network>
<ovf:Description>Network

gestion</ovf:Description>
</ovf:Network>

<ovf:Network ovf:name="service"
rsvr:public="false">
<ovf:Description>Network

gestion</ovf:Description>
</ovf:Network>
</ovf:NetworkSection>

<ovf:VirtualSystemCollection ovf:id="fiwaretest">
<ovf:Info>VirtualSystemCollection</ovf:Info>
<ovf:VirtualSystem ovf:id="tomcat" rsvr:min="1"
rsvr:max="1" rsvr:initial="1">
<ovf:Info>tomcat</ovf:Info>
<ovf:OperatingSystemSection ovf:id="76">
<Info>Specifies the operating system
installed</Info>
<Description>file:///admin.template.snapshotpaas</Description>
</ovf:OperatingSystemSection>

<ovf:VirtualHardwareSection>
<Info>Virtual Hardware Requirements: 2048Mb,
1 CPU, 1 disk, 1 nic</Info>
<ovf:System>
<vssd:ElementName>Virtual Hardware
Family</vssd:ElementName>
<vssd:InstanceID>0</vssd:InstanceID>
<vssd:VirtualSystemIdentifier>deploy-
tcloud</vssd:VirtualSystemIdentifier>
<vssd:VirtualSystemType>kvm</vssd:VirtualSystemType>
</ovf:System>
<Item>
<rased:Description>Number of virtual
CPUs</rased:Description>
<rased:ElementName>1 virtual
CPU</rased:ElementName>
<rased:InstanceID>1</rased:InstanceID>
<rased:ResourceType>3</rased:ResourceType>
<rasd:VirtualQuantity>1</rasd:VirtualQuantity>
   </Item>
   <Item>
   <rasd:AllocationUnits>MegaBytes</rasd:AllocationUnits>
   <rasd:Description>Memory Size</rasd:Description>
   <rasd:ElementName>2048 MB of memory</rasd:ElementName>
   <rasd:InstanceID>2</rasd:InstanceID>
   <rasd:ResourceType>4</rasd:ResourceType>
   <rasd:VirtualQuantity>2</rasd:VirtualQuantity>
   </Item>
   <Item>
   <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
   <rasd:Connection>service</rasd:Connection>
   <rasd:ElementName>Ethernet adapter on admin_net network</rasd:ElementName>
   <rasd:InstanceID>3</rasd:InstanceID>
   <rasd:ResourceType>10</rasd:ResourceType>
   </Item>
   <Item>
   <rasd:ElementName>Harddisk 1</rasd:ElementName>
   <rasd:InstanceID>tomcat</rasd:InstanceID>
   <rasd:Parent>4</rasd:Parent>
   <rasd:ResourceType>17</rasd:ResourceType>
   </Item>
   </ovf:VirtualHardwareSection>
   </ovf:VirtualSystem>
   </ovf:VirtualSystemCollection>
After defined the file with the description of the virtual appliance we can invoke the IaaS SM with the following instruction:

```
```

We obtain a 200 Ok message with details of the service created with the following data in the content body:

```
<xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Service xmlns:ns2="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:ns3="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
xmlns:ns4="http://schemas.dmtf.org/wbem/wscim/1/common"
xmlns:ns5="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
xmlns:ns6="http://schemas.telefonica.com/claudia/ovf"
xmlns:ns7="http://schemas.tcloud.telefonica.com/tcloud/1"
name="tomcat"/>
```

Check that the server was created.

In order to check if service was deployed successfully, you can execute this following command to check details of service.

```
```

$SERVICEMANAGER_IP will be the IP in which the OpenStack Compute API is installed.

We obtained the following details:
To check if virtual machines were correctly instantiated, execute:


$SERVICEMANAGER_IP will be the IP in which the OpenStack Compute API is installed.

Response will look like this:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><servers xmlns:ns2="http://www.w3.org/2005/Atom"
xmlns:ns3="http://docs.openstack.org/compute/api/v1.1">
  <ns3:server name="tomcat" id="89095eece-bfcc-4b4a-b791-de5942777636" progress="100">
    <ns2:link rel="self" href="http://130.206.80.41:8774/v2.0/FIWARE/vdc/980ae4606f464bb8bc214999c596b158/services/fiwaretest/servers/89095eece-bfcc-4b4a-b791-de5942777636"/>
  </ns3:server>
</servers>
```
3.3.2 Accessing Service Manager from a Browser

We are using the Chrome browser [4] with the Simple REST Client plugin [5] in order to send http commands to the Service Manager. You can use it also in Firefox through RESTClient add-ons [6].

We follow the same sequence that we take previously in order to create a VM:

3.3.2.1 Get a valid token

Get a valid for the tenant that we have (It is not a Service Manager Operation itself it is a OpenStack Keystone operation.)
3.3.2.2 **Get the list of available flavors**

![Simple REST Client](image-url)
3.3.2.3  **Get the list of available images**

3.3.2.4  **Instantiate resources.**

The Service Manager provide the same functionality to instantiate server that the OpenStack do. Anyway there is the possibility to instantiate virtual appliance following the extension of the OpenStack Compute API that is provided by this GE.

*Instantiate servers.*

*Create a server.*
Check that the server was created.

Instantiate virtual appliances.

Create of a virtual appliance inside defined project.
Check that the virtual appliance was created.

Check that the server was created.
4 Self-Service Interfaces - User and Programmers Guide

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

4.1 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.

4.2 User Portal 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.

In order to facilitate initial testing of the User Portal functionalities, there is a web server test-page established that runs the client application and permits petitions over a local basic installation of the OpenStack infrastructure. A common log in user name: "admin" and password: "password" is fixed.

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 and run it locally or over a web server. However, additional fine-tuning is required in order for this to work properly, due to the lack of CORS implementation inside OpenStack Nova component, which prevent a direct access to Nova APIs from external domains.

After integration with the SM GE is finished, the user of the portal will be able to make the same operations to the infrastructure components, through the SM. Furthermore, additional functionalities will be leveraged by the SM for the user to inter-operate with the cloud resources and the PaaS GE.

4.2.1 Examples of Use

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

4.2.1.1 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://130.206.80.93
4.2.1.2 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.

![Figure 2: User Portal Images List](image)

4.2.1.3 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.
4.2.1.4 *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.
4.2.1.5 *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.

4.2.1.6 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.

Figure 6: Flavors Page

4.2.1.7 Language Settings
For the moment in the Settings part, four different languages can be set up for the user portal - Figure 7.
Figure 7: Language Settings
5 Object Storage - User and Programmers Guide

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

5.1 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.

5.2 User Guide

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

CDMI is a restful API and thus is accessed programmatically. This access is introduced in the Developer Guide below.

The complete CDMI specification is freely available online from the Storage Networking Industry Association (SNIA)[1].

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 complete OpenStack documentation is available online[2].

5.3 Programmer Guide

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.

Please note that with every CDMI call the version of CDMI being used must be provided.

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
```

will generally return a response similar to:

```
HTTP/1.1 200 OK
```
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 special command line utilities such as curl[9].

5.3.1 Programming against the CDMI interface
As mentioned using the CDMI interface is straightforward since it is built upon the HTTP protocol. The following sections illustrate how to create a client application using the popular Python programming language.

5.3.1.1 Authentication
Authentication is achieved using OpenStack's KeyStone service. The client application that wants to be authenticated needs to know a 'user', 'password' and a 'tenant' (which corresponds to a project name).

These values need to be encoded in a JSON rendered string which is then sent to the KeyStone service. The following Python code snippet illustrates how such a request could be made:

```python
conn = httplib.HTTPConnection('192.168.56.101:5000')
```
headers = {'Content-type':'application/json'}

body = '{"auth": {"tenantName": "" + tenant_name + ", "passwordCredentials": {"username": "" + username + ", "password": "" + password + "}}}'

conn.request("POST", '/v2.0/tokens', body, headers)
response = conn.getresponse()
if response.status not in [200]:
    print 'Failed to auth: ', response.reason,
    response.read()

heads = {}
for item in response.getheaders():
    heads[item[0]] = item[1]
body = response.read()
conn.close()
return heads, body

When authenticated some important values can be extracted from the 'body' that is returned. For example the "keystone id" in the body is needed for authentication. The "publicURL" is the URL that is used for subsequent CDMI calls.

5.3.1.2 Object Storage Manipulation
Once authenticated, the various other CDMI commands as defined in the CDMI specification[4] can be invoked. For example, the following Python code illustrates how to create a container:

conn = httplib.HTTPConnection(URL)
headers = {'X-Auth-Token': token,
            'content-type': 'application/directory',
            'content-length': '0'}
conn.request('PUT', URN '/' + container_name, None, headers)
res = conn.getresponse()

Please note that the Content-Type and the X-Auth-Token need to be sent within the headers of the requests.

5.3.2 Using CLI commands
For ad-hoc manipulation of the Object Storage generic enabler a command line interface can be used. The curl[5] tool allows HTTP commands to be sent and received from the command line.
5.3.2.1 **Authentication**
To authenticate using curl an X-Auth-Token is required. This is provided by the OpenStack Keystone system.

5.3.2.2 **Object Storage Manipulation**
Once an appropriate token is acquired, curl commands can be made against the Object Storage url. For example a basic GET against the top level of the object storage system:

```
curl -v -X GET -H 'X-Auth-Token: AUTH_tk56b01c82710b41328db7c9f953d3933d'
http://127.0.0.1:8080/v1/AUTH_test
```

The following command creates a new top level Container:

```
curl -v -X PUT -H 'X-Auth-Token: AUTH_tk56b01c82710b41328db7c9f953d3933d' -H 'Content-tType: application/directory' -H 'Content-Length: 0'
http://127.0.0.1:8080/v1/AUTH_test/<container_name>
```

The capabilities of a Container can be retrieved as follows:

```
curl -v -X GET -H 'X-Auth-Token: AUTH_tk56b01c82710b41328db7c9f953d3933d' -H 'X-CDMI-Specification-Version: 1.0.1'
http://127.0.0.1:8080/cdmi/cdmi_capabilities/AUTH_test/<container_name>
```

An Object (in this case with the text '<Some JSON>') can be stored in a Container using the following syntax:

```
http://127.0.0.1:8080/v1/AUTH_test/<container_name>/<object_name> -d '<Some JSON>'
```

5.4 **References**

1. ↑ http://www.snia.org/cdmi
2. ↑ http://docs.openstack.org/developer/horizon/
6 Software Management And Configuration - 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 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.

6.2 User 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.

The following sections will go into more detail on how to user and develop against SDC API.

6.3 Programmer Guide

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:

```bash
* 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
> Accept: application/xml
>```
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.

6.3.1 Accessing Service Deployment and Configuration from the CLI

The access through the CLI is made using the curl program. Curl\textsuperscript{[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

```bash
```

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.

```bash
```

```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).

```shell
```

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>
    <fqn>fqn</fqn>
    <osType>95</osType>
  </vm>
  <vdc>{vdc-id}</vdc>
  <product>
    <releaseNotes>Tomcat server 7</releaseNotes>
  </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).

```xml
<version>7</version>
<product>
  <name>tomcat</name>
  <description>tomcat J2EE container</description>
  <attributes>
    <key>port</key>
    <value>8080</value>
  </attributes>
</product>
</productInstance>
```

After that, it is possible to check that the product has been deployed, concretely in this case by [http://{NODE_IP}:8082](http://{NODE_IP}:8082)

6.3.2 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
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 Programmer Guide for the Monitoring 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.

7.2 User Guide

The Monitoring 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 manipulate instance in the Cloud.

Since Monitoring API is based on the HTTP protocol there are a rich variety of tools, which can be used to access the Monitoring interface.

The following sections will go into more detail on how to user and develop against the Monitoring API.

7.3 Programmer Guide

Monitoring 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 Monitoring Manager through Monitoring API.

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

```
* About to connect() to 109.231.69.204 port 8182 (#0)
* Trying 109.231.69.204... connected
* Connected to 109.231.69.204 (109.231.69.204) port 8182 (#0)
> GET /api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor 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: 109.231.69.204:8182
> Accept: */*
>
< HTTP/1.1 200 The request has succeeded
< Content-Type: text/xml; charset=ISO-8859-1
< Content-Length: 2024
```
In this case the HTTP GET operation is used. Monitoring API uses the Create Retrieve Update Delete (CRUD) operations, which map almost to the HTTP verbs POST, GET, PUT and DELETE.

Monitoring API deal with one media type, application/xml Content-Type and Accept, which means that we can send and receive a xml content.

7.3.1 Accessing Monitoring Systems 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 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.

There are different cloud resources which may be monitored. To provide an example, the links provided by virtual machine are followed.

1. Obtain URL for monitoring the VM

Getting information about the VM by its URL, we can obtain the URL for this measurable resource.


With the following result:

Response:
200 OK

Content-Type: application/vnd.telefonica.tcloud.vapp+xml
The previous request obtains the monitoring URL available for target VM.

http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor

2.- Obtain VM metrics
From the previous URL, it is possible to obtain the metrics associated to the VM

Request:


Response:

200 OK
Content-Type: application/vnc.telefonica.tcloud.measureDescriptorList+xml

<MeasureDescriptorList href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor">  
  <Link rel="up" type="application/vnd.telefonica.tcloud.vapp+xml" href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/"/>
  <MeasureDescriptor href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/memoryUsage" name="memoryUsage">  
    <Link rel="monitor:subscribe" type="application/vnd.telefonica.tcloud.monitoringCallback+plain" />
  </MeasureDescriptor>
</MeasureDescriptorList>
3.- Obtain information about a VM metric

In the case, we ask for the concrete metric information.

Request:

-H "X-Auth-Token: eaaa618-0fed-4b3a-81b4-663c99ec1cbb" -X GET "http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay"

Response:

200 OK
Content-Type: application/vnc.telefonica.tcloud.measureDescriptor+xml

<MeasureDescriptor href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/memoryUsage" />
  <Unit>byte * 2^20</Unit>
  <MinValue>0</MinValue>
  <MaxValue>2048</MaxValue>
  <Description>Amount of used memory</Description>
</MeasureDescriptor>

<MeasureDescriptor href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay" name="requestDelay">
  <Link rel="monitor:subscribe" type="application/vnd.telefonica.tcloud.monitoringCallback+plain"
href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay" />
  <Unit>milliseconds</Unit>
  <MinValue>0</MinValue>
  <MaxValue>10000</MaxValue>
  <Description>Amount of used memory</Description>
</MeasureDescriptor>
</MeasureDescriptorList>
4.- Obtain metric value

To obtain a measure value, the following request may be invoked.

Request:


Response:

200 OK
Content-Type: application/vnd.telefonica.tcloud.measure+xml

<Measure href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay">
  <Link rel="monitor:subscribe" type="application/vnd.telefonica.tcloud.monitoringCallback+plain" href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay" />
  <Unit>milliseconds</Unit>
  <MinValue>0</MinValue>
  <MaxValue>10000</MaxValue>
  <Description>Amount of used memory</Description>
</Measure>

In addition, it is possible to obtain several values for the same metric.

Request:
curl -v -H "Access-Control-Request-Method: GET" -H "Content-Type: application/xml"
-H "Accept: application/xml" -H "X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb"
-X GET
"http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay/values?samples=3"

Response:

```
200 OK
Content-Type: application/vnd.telefonica.tcloud.measure+xml

<Measure href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay">
  <Link rel="monitor:subscribe" type="application/x-url" href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay/subscription" />
  <Sample timestamp="2012-11-12T10:40:14" unit="miliseconds" value="100"/>
  <Sample timestamp="2012-11-12T10:40:04" unit="miliseconds" value="98.9"/>
  <Sample timestamp="2012-11-12T10:39:54" unit="miliseconds" value="97.0"/>
</Measure>
```

5.- **Register a subscription metric** It's possible to register a callback URL to obtain monitoring measures in a push manner.

Request:

curl -v "Access-Control-Request-Method: POST" -H "Content-Type: application/xml"
-H "Accept: application/xml" -H "X-Auth-Token: eaaafd18-0fed-4b3a-81b4-663c99ec1cbb"
-X POST
"http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay/subscription "

Response:

```
http://example.com/monitoring/do.ReceiveMeasure
```
200 OK

<MonitoringSubscription
href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay/subscription/1"
status="accepting">
  <Link rel="up"
href="http://cloud.telefonica.com/api/org/fiware/vdc/demo/vapp/service/tomcat/1/monitor/requestDelay" />
  <Callback
href="http://example.com/monitoring/do.ReceiveMeasure"
type="application/vnd.telefonicaa.tcloud.measure+xml" />
</MonitoringSubscription>
8  PaaS Management - User and Programmers Guide

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

8.1  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.

8.2  User Guide

The PaaS 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 manipulate instance in the Cloud.

Since PaaS API is based on the HTTP protocol there are a rich variety of tools, which can be used to access the PaaS Manager interface.

The following sections will go into more detail on how to use and develop against PaaS API.

8.3  Programmer Guide

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: eaaafdl8-0fed-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.

8.3.1 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 version="1.0" encoding="UTF-8" standalone="yes"?>
<environmentDtoes>
  <environment>
    <environmentType>
```

D.4.4.2: FI-WARE User and Programmers Guide  Page 60
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:

```xml
<environmentDtoes>
  <environment>
    <environmentType>
      <id>24</id>
      <name>java_web_server</name>
      <description>An environment for Java applications deployed on a the web server</description>
    </environmentType>
    <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</description>
        </productType>
      </productReleases>
    </tiers>
  </environment>
</environmentDtoes>

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

```bash
```

With the following result:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environmentInstanceDtos>
   <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>
         </tiers>
      </environment>
   </environmentInstance>
</environmentInstanceDtos>
```
<minimum_number_instances>1</minimum_number_instances>
  <name>tomcat_tier</name>
  <productReleases>
    <product>tomcat</product>
    <version>7.0</version>
    <description>Tomcat 7.0</description>
  </productReleases>
  <productType>
    <id>6</id>
    <name>ApplicationWebServer</name>
    <description>Application Web Server</description>
  </productType>
</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>
  <fqn>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.replicas.1</fqn>
    <name>java_tomat_instance_tomcat</name>
    <productInstances>
      <date>2012-11-29T13:03:27.834+01:00</date>
    </productInstances>
    <name>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.replicas.1_tomcat_7.0</name>
      <status>INSTALLED</status>
      <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 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>
        </productRelease>
      </attributes>
    </artifact>
  </artifacts>
</applicationReleaseDto>
```
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.

"http://{PaaSManager_IP}:8080/paasmanager/rest/org/{org-id}/vdc/{vdc-id}/environmentInstance/{environmentInstanceId}/applicationInstance/{applicationInstanceId}"

Obtaining:

```
<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>
          
        </artifact>
        
        <artifact>
          
          <name>the properties file</name>
          
          <attributes>
            
            <key>properties_url</key><value>http://configuration.properties</value>
            
          </attributes>
          
        </artifact>
        
      </attributes>
      
    </artifact>
    
  </artifacts>
  
</applicationInstances>
```
<key>properties_name</key><value>flipper.properties</value>
  </attributes>
  <productRelease>
    <version>7.0</version>
    <product>tomcat</product>
  </productRelease>
</artifact>
</artifacts>
<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>
  </tierInstances>
8.3.2 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:

```xml
<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>

<fqdn>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.replicas.1</fqdn>
  <name>java_tomat_instance_tomcat</name>
  <productInstances>
    <date>2012-11-29T13:03:27.834+01:00</date>
    <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>
    <vm>
      <hostname>tomcat</hostname>
      <ip>130.206.80.114</ip>
    </vm>
  </productInstances>
</tierInstance>
</environmentInstance>
</applicationInstances>
```
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
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 for the Job Scheduler Generic Enabler!
This generic enabler is built on a open source solution, 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 has been created for this specific API. The on-line 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.

9.1.1 Recommended actions and assumptions before starting

- We assume that the Job Scheduler GE Administrator has operated in install Job Scheduler GE, according to Job Scheduler - Installation and Administration Guide.
- Since our Job Scheduler GE 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 Job Scheduler GE 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 Job Scheduler GE 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, GOPER, 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.
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, a possible integration with Identity Management GE -still under evaluation- might consist in replacing the sessionid, required for each request against ProActive Scheduling & Resourcing REST API, with the X-Auth-Token returned after the successful user authentication, by providing username and password parameters. 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 (`):

```
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/encoding 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 Resource Manager Service

9.3.1.1 login

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

```
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
1d871fb013dfdbb01357a912a2034ac7a124c321d871fb013dfdbb01358000

9.3.1.2 **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 -v -X POST -d
"username=admin&password=admin"
http://localhost:8080/rest/rest/rm/login`
http://localhost:8080/rest/rest/rm/disconnect

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/disconnect 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
9.3.1.3  **isActive**  
Test if the RM is operational. It has been mapped with ProActive Parallel Suite

```
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
```
9.3.1.4 shutdownRM

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

If the Resource Manger Service is launched according to default configuration, 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 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$ rm-start -Dproactive.rmi.port=1099
Starting the resource manager...

bash-4.2$ scheduler-start -Dproactive.rmi.port=2000 -u
rmi://192.168.122.1:1099/
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
```

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)
```
As administrator, you might check that, by typing `ps -eaf | grep RMStarter`, on the machine that hosts Job Scheduler GE, you will not get any process running.

### 9.3.1.5 `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.

```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)
```
9.3.1.6  **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

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...  % Total  % Received % Xferd Average Speed Time Time Current Dload Upload Total Spent
* Closing connection #0
```
* 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:41fa548513dfebd3eb7ee32a2034ac7a124c3241fa548513dfebd 
> f3eb8000
>
> 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":
    "eventType": null,
    "hostName": "192.168.122.1",
    "nodeInfo": "Node local-LocalNodes-1\nURL : rmi://192.168.122.1:1099/local-LocalNodes-1\nNode source : LocalNodes\nProvider : rm\nUsed by : nobody\nState : Free\nJMX RMI: service:jmx:rmi:///jndi/rmi://192.168.122.1:56987/rmnode",
    "nodeOwner": null,
    "nodeProvider": "rm",
    "nodeSource": "LocalNodes",
    "nodeState": "FREE",
    "nodeUrl": "rmi://192.168.122.1:1099/local-LocalNodes-1",
    "padname": "",
    "previousNodeState": null,
    "proactiveJMXUrl":
    "rmurl": null,
    "timeStamp": 1365778441281,
    "timeStampFormatted": "4/12/13 4:54 PM",
    "vmname": "rmi://192.168.122.1:1099/PA_JVM442274898",
    "vnName": null
},
{
    "counter": 0,
    "defaultJMXUrl":
    "eventType": null,
    "hostName": "192.168.122.1",
    "nodeInfo": "Node local-LocalNodes-0\nURL : rmi://192.168.122.1:1099/local-LocalNodes-0\nNode source : LocalNodes\nProvider : rm\nUsed by : nobody\nState : Free\nJMX RMI: service:jmx:rmi:///jndi/rmi://192.168.122.1:45792/rmnode",
    "nodeOwner": null,
"nodeProvider": "rm",
"nodeSource": "LocalNodes",
"nodeState": "FREE",
"nodeUrl": "rmi://192.168.122.1:1099/local-
LocalNodes-0",
  "padname": ",",
  "previousNodeState": null,
  "proactiveJMXUrl": 
  "rmurl": null,
  "timeStamp": 1365778441256,
  "timeStampFormatted": "4/12/13 4:54 PM",
  "vmname": 
  "rmi://192.168.122.1:1099/PAJvm993096019",
  "vnName": null
},
{
  "counter": 0,
  "defaultJMXUrl": 
  "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
JMX
RMI: 
RMI:
RO: 
nodeOwner": null,
"nodeProvider": "rm",
"nodeSource": "LocalNodes",
"nodeState": "FREE",
"nodeUrl": "rmi://192.168.122.1:1099/local-
LocalNodes-3",
  "padname": ",",
  "previousNodeState": null,
  "proactiveJMXUrl": 
  "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",
"nodeOwner": null,
"nodeProvider": "rm",
"nodeSource": "LocalNodes",
"nodeState": "FREE",
"nodeUrl": "rmi://192.168.122.1:1099/local-LocalNodes-2",
"padname": "",
"previousNodeState": null,
"rmurl": null,
"timeStamp": 1365778441264,
"timeStampFormatted": "4/12/13 4:54 PM",
"vmname": "rmi://192.168.122.1:1099/PA_JVM2004024026",
"vnName": null
9.3.1.7 **getRMStatHistory**

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

```

* 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  0.00+  0.00+  0.00+  0.00+  0.00+     0      0      0          0
* 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:41fa548513dfebdf3eb7d262a2034ac7a124c3241fa548513dfebdf3eb800
>
< 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": [1, 4, 4]
```
{"AverageActivity": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
"BusyNodesCount": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}


```json
{
  "DownNodesCount": [
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0
  ],
  "FreeNodesCount": [
    4,
    4,
    4,
    4
  ]
}
```
9.3.1.8 \textit{getRMState}

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


* 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
--:--:-- --:--:-- 0connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/state 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: */*
9.3.1.9  getRMVersion
return the current REST server API and RM version. It has been mapped with
ProActive Parallel Suite /rm/version operation


* 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/version HTTP/1.1
9.3.2 Node Source

9.3.2.1 `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.

* 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  --:--:-- --:--:-- --:--:-- 0connected
* 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:41fa548513dfebdf3eb7ba92a2034ac7a124c3241fa548513dfebdf3eb8000
>
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": "CREDENTIALE"
    },
    "name": "credentials",
    "value": ""
  }]}\]
"meta": {  
  "description": "Maximum number of nodes to be deployed on Resource Manager machine",  
  "type": "NONE"
}
},
  "name": "maxNodes",
  "value": "4"
}
},
  "meta": {  
  "description": "in ms. After this timeout expired, the node is considered to be lost",
  "type": "NONE"
}
},
  "name": "nodeTimeout",
  "value": "5000"
}
},
  "meta": {  
  "description": "Additional ProActive properties",
  "type": "NONE"
}
},
  "name": "paProperties",
  "value": ""
}
]
},
"defaultValues": {  
  "rmUrl": "rmi://192.168.122.1:1099/"
}
},
  "pluginDescription": "Deploys nodes on Resource Manager's machine",
  "pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.LocalInfrastructure"
"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 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": "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.CLI Infrastructure"
};
"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/"
   },
   {
      "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": "Absolute path of the java executable on the remote hosts",
        "type": "NONE"
    },
    "name": "javaPath",
    "value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
}
"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": ""
    },
    {
        "meta": {
            "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.SSH Infrastructure",

"configurableFields": [
{
"name": "rmUrl",
"value": "rmi://192.168.122.1:1099/"
},
{
"name": "javaPath",
"value": "/user/lcantelm/home/application/jdk1.6.0_43/bin/java"
},
{
"name": "sshOptions",
"value": ""
}]}
"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"
  },
  {
"description": "The batch system\nhead node name or IP adress",
"type": "NONE"
},
"name": "serverName",
"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": "CREDENTIAL"
  },
  "name": "logCredentialsPath",
  "value": ""
}
"pluginDescription": "[DEPRECATED] Infrastructure described in GCM deployment descriptor",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.GCM Infrastructure"

"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\nfor 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.GCM
CustomisedInfrastructure"
},
{
  "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"
    }
  ]
}
"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": "in ms. After this timeout expired the node is considered to be lost",
   "type": "NONE"
 },
"name": "nodeTimeOut",
"value": "300000"
 },
 {  
   "meta": {
      "description": "The batch system\nhead node name or IP adress",
      "type": "NONE"
   },
   "name": "serverName",
   "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": ""
 }
],
"defaultValues": {
   "rmUrl": "rmi://192.168.122.1:1099/" 
},
"pluginDescription": "Acquires nodes from a LSF resource manager.",

"pluginName": "org.ow2.proactive.resourcemanager.nodesource.infrastructure.LSF Infrastructure"
}

"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": ""
},
{
  "meta": {
    "description": "Hypervisor's user",
    "type": "NONE"
  }
}
{
  "meta": {
    "description": "Hypervisor's user's password",
    "type": "PASSWORD"
  },
  "name": "VMMUser",
  "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",
  "value": "0"}
}{
  "meta": {
    "description": "ProActive Configuration file path",
    "type": "FILEBROWSER"
  },
  "name": "PAConfig",
  "value": ""
},
{
  "meta": {
    "description": "Absolute path of the rm.cred file",
    "type": "CREDENTIAAL"
  },
  "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 executable 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) root directory on the remote hosts",
   "type": "NONE"
 },
   "name": "schedulingPath",
   "value": "/home/lcantelm/git/build/dist/scheduling"
 },

{ "meta": {
   "description": "Options for the java command launching 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\nhead node name or IP adress",
        "type": "NONE"
    },
    "name": "serverName",
    "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": "-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.PBS Infrastructure"
},
{"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": "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.EC2 Infrastructure"}
"meta": {
  "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": "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": "resourceManagerPath",
  "value": "/home/lcantelm/application/jdk1.6.0_43/jre/bin/resourceManager"
}
"type": "NONE"
},
"name": "rmPath",
"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\nlaunching the node on the remote hosts",
"type": "NONE"
},
"name": "javaOptions",
"value": ""
},
{
"meta": {
"description": "Additional classpath for the java command\nlaunching the node on the remote hosts",
"type": "NONE"
},
"name": "extraClassPath",
"value": ""
},
{
"meta": {
"description": "in ms. After this timeout expired\nthe node is considered to be lost",
"type": "NONE"
9.3.2.2 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.

* 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--:-:-- --:-:-- --:-:-- --:-:-- 0connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /rest/rest/rm/policies 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:41fa548513dfebdf3eb7a942a2034ac7a124c3241fa548513dfebdf3eb8000
HTTP/1.1 200 OK
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(6.1.18)

{
  "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": ""
    }]
}

```
"value": ""
},
{
"meta": {
    "description": "",
    "type": "CREDENTIAL"
},
"name": "schedulerCredentialsPath",
"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": "numberOfTasksPerNode",
"value": "10"
},
{
"meta": {
    "description": "total number of tasks",
    "type": "NONE"
},
"name": "totalNumberOfTasks",
"value": "10"
},
{
"meta": {
    "description": "total number of tasks",
    "type": "NONE"
},
"name": "totalNumberOfTasks",
"value": "10"
}
```

Future Internet Core Platform

D.4.4.2: FI-WARE User and Programmers Guide

```json
{
    "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": "nodeDeploymentTimeout",
        "value": "2400000"
    }
    ]
}
```

"type": "NONE"

"name": "loadFactor",
"value": "10"

"meta": {
    "description": "",
    "type": "NONE"
},
"name": "nodeDeploymentTimeout",
"value": "2400000"
}
"type": "NONE"
},
"name": "providerAccessType",
"value": "ME"
},
{
"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.",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.policy.TimeSlotPolicy"
},

{
    "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.resourcemanager.nodesource.policy.StaticPolicy"
}
```json
{
  "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": "ME|users=name1,name2;groups=group1,group2|ALL",
        "type": "NONE"
      },
      "name": "schedulerUrl",
      "value": ""
    },
    {
      "meta": {
        "description": "",
        "type": "CREDENTIAL"
      },
      "name": "schedulerCredentialsPath",
      "value": ""
    }
  ]
}```
{ "meta": {  
  "description": "refresh frequency (ms)",  
  "type": "NONE"  
},  
"name": "refreshTime",  
"value": "1000"  
},  
{ 
  "meta": {  
    "description": "",  
    "type": "NONE"  
  },  
  "name": "minNodes",  
  "value": "0"  
},  
{ 
  "meta": {  
    "description": "number of tasks per node",  
    "type": "NONE"  
  },  
  "name": "loadFactor",  
  "value": "10"  
},  
{ 
  "meta": {  
    "description": "",  
    "type": "NONE"  
  },  
  "name": "nodeDeploymentTimeout",  
  "value": ""  
}
"value": "10000"
},
{
    "meta": {
        "description": "Time since the nodes acquisition is allowed (crontab format)",
        "type": "NONE"
    },
    "name": "acquisitionAllowed",
    "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": [
    {
        "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": "CREDENTIAL"
   },
   "name": "schedulerCredentialsPath",
   "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"
}
],
"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": "userAccessType",
      "value": "ALL"
    },
  ]

```
"name": "providerAccessType",
"value": "ME"
},
{
"meta": {
"description": "",
"type": "NONE"
},
"name": "schedulerUrl",
"value": ""
},
{
"meta": {
"description": "",
"type": "CREDECIAL"
},
"name": "schedulerCredentialsPath",
"value": ""
},
{
"meta": {
"description": "ms",
"type": "NONE"
},
"name": "idleTime",
"value": "60000"
}
],
"defaultValues": {
"rmUrl": "rmi://192.168.122.1:1099/
}
,"pluginDescription": "Releases all resources when scheduler is idle for specified\ntime. Acquires them back on job submission.",
"pluginName": "org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.ReleaseResourcesWhenSchedulerIdle"}
{ "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": "CREDENTIAL"
   },
   "name": "schedulerCredentialsPath",
   "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|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": {
         "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": "Time of the nodes acquisition (crontab format)",
         "type": "NONE"
      },
      "name": "nodeAcquisition",
      "value": "* * * * *"
   }
]
{  
    "meta": {  
        "description": "Time of the nodes removal (crontab format)",
        "type": "NONE"
    },  
    "name": "nodeRemoval",
    "value": "* * * * *"
  },

  {
    "meta": {
        "description": "the mode how nodes are removed",
        "type": "NONE"
    },
    "name": "preemptive",
    "value": "false"
  },

  {
    "meta": {
        "description": "Start deployment immediately",
        "type": "NONE"
    },
    "name": "forceDeployment",
    "value": "false"
  }
],
"defaultValues": {
  "rmUrl": "rmi://192.168.122.1:1099/"
},
"pluginDescription": "Acquires and releases nodes at specified time.",
"pluginName": "org.ow2.proactive.resourcemanager.nodesource.policy.CronPolicy"
}
9.3.2.3 **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

Before proceeding by sending the related request, you need to create your credentials, which represent a shortcut to authenticate yourself. Thanks to `create-cred` script (check its usage by typing `create-cred -h`), those are achievable by processing your `username` and `password` with the public key of the Resource Manager. In our case we have already got them as follows:

```
```

Finally, by remarking that the order of the parameters at the query matters, we can submit successful our request:

```
```

* 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/create 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:425e53bb13e2b3f42e72be2a2034ac7a124c32425e53bb13e22b3f42e8000
> Content-Length: 857
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Content-Type: application/json
The increase of the computing nodes number at your disposal can be observed by `getRMState`.

### 9.3.2.4 `removeNodeSource`

remove a new node source from the RM. It is mapped with ProActive Parallel Suite `/rm/nodesource/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/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*.

### 9.3.3 Nodes

In order to manage the computing nodes, you should get the necessary information 
accessible by *getMonitoring* operation.

#### 9.3.3.1 lockNode

prevent other users from using a set of locked nodes. It is mapped with ProActive 
Parallel Suite */rm/node/lock* 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/lock 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:400a658813dff1aba347c082a2034ac7a124c32400a658813dff1aba348000
> 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
```
9.3.3.2 **unlockNode**
allow other users to use a set of nodes previously locked. It is mapped with ProActive Parallel Suite /rm/node/unlock operation

```bash
bash-4.2$ curl -v -X POST --data-urlencode
'nodeurls=rmi://192.168.122.1:1099/local-LocalNodes-1' --data-urlencode
'nodeurls=rmi://192.122.1:1099/local-LocalNodes-2' -H
sessionid:`curl -v -X POST -d "username=admin&password=admin"
http://localhost:8080/rest/rest/rm/login`
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: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
```

9.3.3.3 **releaseNode**
release a node, previously reserved for computation. It is mapped with ProActive Parallel Suite /rm/node/release operation "defaultJMXUrl":

```bash
curl -v -X POST -H sessionid:`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/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
> Host: localhost:8080
> Accept: */*
> sessionid:425e53bb13e22b3f42e6f7a2a2034ac7a124c32425e53bb13e22b3f42e8000
> Content-Length: 231
> 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

9.3.3.4 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

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.PAMRRemoteO
tector> 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.AMQPRemoteO
jectFactory> 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

bash-4.2$ curl -v -X POST -d
"nodeurl=rmi://192.168.122.1:1099/NEW-
NODE&nodesource=DefaultInfrastructure" -H sessionid:`curl -v -X
**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)
```

```
9.3.3.6  **removeNode**

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

9.3.4.1 login
enable user to access the Scheduler with his credentials. It is mapped with ProActive Parallel Suite /scheduler/login operation

bash-4.2$ curl -v -X POST -d "username=admin&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
9.3.4.2 **disconnect**

disconnect user from the Scheduler. It is mapped with ProActive Parallel Suite
/scheduler/disconnect 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/disconnect 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
true
```

9.3.4.3 **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)
true
```
9.3.4.4  **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/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:400a658813dff1aba3475a52a2034ac7a124c32400a658813dff1aba348000  
> < HTTP/1.1 200 OK  
< Content-Type: application/json
```
9.3.4.5  **resumeScheduler**
resume the Scheduler. It is mapped with ProActive Parallel Suite `/scheduler/resume` 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/resume 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:400a658813dfff1aba3475302a2034ac7a124c32400a658813dff1aba348000
>
< 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
```

9.3.4.6  **freezeScheduler**
freeze the Scheduler. It is mapped with ProActive Parallel Suite `/scheduler/freeze` 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/freeze 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:400a658813dff1aba3474f92a2034ac7a124c32400a658813dff1aba348000
>
< 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

9.3.4.7 startScheduler
start the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/start 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/start HTTP/1.1
9.3.4.8 **stopScheduler**

Stop the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/stop 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/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:400a658813dff1aba3473182a2034ac7a124c32400a658813dff1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked

true
9.3.4.9  **getSchedulerStatus**

return the current Scheduler status. It has been mapped with ProActive Parallel Suite \
/[scheduler/status](http://localhost:8080/rest/rest/scheduler/status) operation

```
bash-4.2$ curl -v -X GET -H "sessionid:`curl -v -X POST -d \n"username=admin&password=admin" \nhttp://localhost:8080/rest/rest/scheduler/login`" \nhttp://localhost:8080/rest/rest/scheduler/status
* About to connect() to localhost port 8080 (#0)
  * Connected to localhost (127.0.0.1) port 8080 (\n  > GET /rest/rest/scheduler/status HTTP/1.1
  > User-Agent: curl/7.21.7 (x86_64-redhat-linux-gnu) \nlibcurl/7.21.7 NSS/3.13.5.0 zlib/1.2.5 libidn/1.22 libssh2/1.2.7
  > Host: localhost:8080
  > Accept: */*
  > sessionid:400a658813dff1aba3472532a2034ac7a124c32400a658813dff1aba348000
  >
  < 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
  "STARTED"
```

9.3.4.10  **killScheduler**

kill the Scheduler. It is mapped with ProActive Parallel Suite [scheduler/kill](http://localhost:8080/rest/rest/scheduler/kill) operation. It does not require advanced services configuration, since the [rmiregistry](http://localhost:8080/rest/rest/scheduler/kill) service belongs to the RM Service, by default.
bash-4.2$ curl -v -X PUT -H "sessionid:`curl -v -X POST -d "username=admin&password=admin" http://localhost:8080/rest/rest/scheduler/killddluer/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) > 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: */* > sessionid:285d613e13e0cd6ba147f632a2034ac7a124c32285d613e13e0cd6 b148000 > < 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

9.3.4.11  

getSchedulerStats

return statistics about the Scheduler. It is mapped with ProActive Parallel Suite /scheduler/stats 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/stats | python -mjson.tool * About to connect() to localhost port 8080 (#0) * Trying 127.0.0.1... % Total % Received % Xferd Average Speed 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/schedulers/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: */*
>
> sesssionid:400a658813dflaba34702b2a2034ac7a124c32400a658813dff1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<

{ [data not shown]

100 468 0 468 0 2542 0 --:--:-- --:--:-- --:--:-- 2557

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

{

  "ConnectedUsersCount": "25",
  "FinishedJobsCount": "47",
  "FinishedTasksCount": "318",
  "FormattedJobSubmittingPeriod": "9h 36m 29s 109ms",
  "FormattedMeanJobExecutionTime": "35s 999ms",
  "FormattedMeanJobPendingTime": "2s 171ms",
  "JobSubmittingPeriod": "34589109",
  "MeanJobExecutionTime": "35999",
  "MeanJobPendingTime": "2171",
  "PendingJobsCount": "1",
  "PendingTasksCount": "2",
  "RunningJobsCount": "0",
  "RunningTasksCount": "0",
  "Status": "Started",
  "TotalJobsCount": "48",
  "TotalTasksCount": "324"
}
9.3.4.12  **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
     Dload  Upload   Total   Spent  Left  Speed
0     0    0     0    0     0  --:--:-- --:--:-- --:--:-- 0connected
* 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:400a658813dff1aba346fb72a2034ac7a124c32400a658813dff1ab348000
>
< 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"
}

9.3.4.13 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 % Xferd 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/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:400a658813d6f1aba3470c82a2034ac7a124c32400a658813d6f1aba34800
>
< 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"
  },
  {
    "connectionTime": 1365789292006,
    "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": 1365788472875,
    "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,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365787537015,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365788115416,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
"connectionTime": 1365789154122,
"hostName": "192.168.122.1:1099",
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
"lastSubmitTime": -1,
"submitNumber": 0,
"username": "admin"
},
{
  "connectionTime": 1365788416083,
  "hostName": "192.168.122.1:1099",
  "lastSubmitTime": -1,
  "submitNumber": 0,
  "username": "admin"
},
{
  "connectionTime": 1365788247885,
  "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,
9.3.4.14 **getJobs**

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

```
* 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) 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:400a658813dfe1aba346eb62a2034ac7a124c32400a658813dfe1aba348000
>
> < HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
```
9.3.4.15  **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 (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


* 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
job will start automatically without problem, changing its state in \textit{RUNNING}.

\subsection*{9.3.4.16 \texttt{getSchedulerVersion}}

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

```

* 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 --:--:-- --:--:-- 0connected
*   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
```
9.3.5 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/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>
  </task>
  <task name="task3">
    <!-- Task human description -->
    <description>This task will sleep 4s</description>
  </task>
  <task name="task4">
    <!-- Task human description -->
    <description>This task will sleep 2s</description>
  </task>
  <task name="task5">
    <!-- Task human description -->
    <description>This task will sleep 1s</description>
  </task>
  <task name="task6">
    <!-- Task human description -->
    <description>This task will sleep 0.5s</description>
  </task>
  <task name="task7">
    <!-- Task human description -->
    <description>This task will sleep 0.25s</description>
  </task>
  <task name="task8">
    <!-- Task human description -->
    <description>This task will sleep 0.125s</description>
  </task>
</taskFlow>
</job>
```
<javaExecutable class="org.ow2.proactive.scheduler.examples.WaitAndPrint">
  <parameters>
    <parameter name="sleepTime" value="6"/>
    <parameter name="number" value="2"/>
  </parameters>
</javaExecutable>
</task>
</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>
</task>
<task name="task8" preciousResult="true">
  <!-- 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>
<task name="task4">
  <!-- 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">
  <!-- 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">
  <!-- 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>

<task name="task7">
  <!-- 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="7"/>
    </parameters>
  </javaExecutable>
</task>
9.3.5.1 *submitJob*

submit a job to the Scheduler. It is mapped with ProActive Parallel Suite `/rm/infrastructures` operation.

```

* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1...
* 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)
>_accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
>
100 29 0 0 100 29 0 28 0:00:01 0:00:01 --:--:-- --:--:--

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

100 91 0 62 100 29 37 17 0:00:01 0:00:01 --:--:--

* Closing connection #0
```
### 9.3.5.2 pauseJob

pause the job execution by means of jobid. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/pause 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/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:400a658813df1aba343c532a2034ac7a124c32400a658813df1a ba348000
>
< 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
```
9.3.5.3 \textit{getJobState} \\
return the job state by providing \textit{jobid}. It is mapped with ProActive Parallel Suite \url{scheduler/jobs/[jobid]} operation


* 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 connected 
* Connected to localhost (127.0.0.1) port 8080 (#0) 
> 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:400a658813dff1aba343b8c2a2034ac7a124c32400a658813dff1a 
> <- 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",
"packageName": "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
            }
        }
    }
}
"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": 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": {
        "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": 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
    }
  }
}
"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": 2050006,
      "readableName": "task6"
   },
   "taskStatus": "PAUSED"
},
"wallTime": 0
},
"2050007": {
   "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
}
"type": "TASKSFLOW"
9.3.5.4 **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

```
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/tasks |
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 --:--:-- --:--:-- --:--:-- 0connected
* 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:400a658813dffa8a34392e2a2034ac7a124c32400a658813dff1a
ba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
{
  [data not shown]
100  65  0  65  0  0  280  0  --:--:-- --:--:--
   --:--:--  281
* Connection #0 to host localhost left intact
* Closing connection #0
[
  "task2",
  "task4",
  "task3",
```
### 9.3.5.5 `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

```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           0           0       0       0          0          0
0%  (0)   (0)  (0)   (0)       (0) (0) (0)       (0) (0)
% upload  (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/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: */*
>
```
<
< 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
```
Future Internet Core Platform

| Execution Duration: -1,        |
| Execution Host Name: null,     |
| Finished Time: -1,             |
| NumberOfExecutionLeft: 1,      |
| NumberOfExecutionOnFailureLeft: 2, |
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,
"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,
"taskInfo": {
    "executionDuration": -1,
    "executionHostName": null,
    "finishedTime": -1,
    "numberOfExecutionLeft": 1,
    "numberOfExecutionOnFailureLeft": 2,
    "progress": 0,
    "startTime": -1,
    "taskId": {
        "id": 2050000,
        "readableName": "task7"
    },
    "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": "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,
  "numberOfExecution": 1,
  "numberOfExecutionOnFailure": 2,
  "progress": 0,
  "startTime": -1,
  "taskId": {
    "id": 2050007,
    "readableName": "task5"
  },
  "taskStatus": "PAUSED"
},
"wallTime": 0


```json
{
  "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,
    "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": {"taskId": {
    "id": 2050006,
    "readableName": "task6"
},
    "taskStatus": "PAUSED"
},
    "wallTime": 0
},
"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,
"postScript": null,
"preScript": null,
"preciousLogs": false,
"preciousResult": false,
"restartTaskOnError": {
   "value": {
      "description": "Anywhere",
   }
}
9.3.5.6  **changeJobPriority**

change the priority of a job under execution. It can be achieved by distinguish 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)
```
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

```
```

* 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:400a658813dff1aba34380a2a2034ac7a124c32400a658813dff1aba348000
>
> HTTP/1.1 204 No Content
> Server: Jetty(6.1.18)
>
* Closing connection #0
9.3.5.7  **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)
> PUT /rest/rest/scheduler/jobs/205/resume 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:400a658813df1aba3434802a2034ac7a124c32400a658813df1aba348000
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
< Server: Jetty(6.1.18)
<
<
> true
```

9.3.5.8  **getServerLogs**
return job server logs by means of *jobid*. It is mapped with ProActive Parallel Suite
/scheduler/jobs/{jobid}/log/server 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/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: */*
> sessionid:400a658813df1aba34344b2a2134ac7a124c32400a658813df1aba348000
>
< 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 13:37:16,695 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 13:50:46,524 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 =================

[2013-04-13 14:10:01,472 INFO ] task 2050002 scheduling

[2013-04-13 14:10:03,484 INFO ] task 2050002 scheduling

[2013-04-13 14:10:05,495 INFO ] task 2050002 scheduling

[2013-04-13 14:10:07,507 INFO ] task 2050002 scheduling

[2013-04-13 14:10:09,518 INFO ] task 2050002 scheduling

[2013-04-13 14:10:11,530 INFO ] task 2050002 scheduling

[2013-04-13 14:10:13,541 INFO ] task 2050002 scheduling

[2013-04-13 14:10:15,553 INFO ] task 2050002 scheduling

[2013-04-13 14:10:17,565 INFO ] task 2050002 scheduling

[2013-04-13 14:10:19,577 INFO ] task 2050002 scheduling

[2013-04-13 14:10:21,589 INFO ] task 2050002 scheduling


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

[2013-04-13 14:10:01,472 INFO ] task 2050001 scheduling

[2013-04-13 14:10:03,484 INFO ] task 2050001 scheduling

[2013-04-13 14:10:05,495 INFO ] task 2050001 scheduling

[2013-04-13 14:10:07,507 INFO ] task 2050001 scheduling

[2013-04-13 14:10:09,518 INFO ] task 2050001 scheduling

[2013-04-13 14:10:11,530 INFO ] task 2050001 scheduling

[2013-04-13 14:10:13,541 INFO ] task 2050001 scheduling

[2013-04-13 14:10:15,553 INFO ] task 2050001 scheduling

[2013-04-13 14:10:17,565 INFO ] task 2050001 scheduling

[2013-04-13 14:10:19,577 INFO ] task 2050001 scheduling

[2013-04-13 14:10:21,589 INFO ] task 2050001 scheduling

getLiveLogs

return only the currently available logs of job identified by the id jobid. It is mapped with ProActive Parallel Suite /scheduler/jobs/{jobid}/liveLog operation

bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d "username=admin&password=admin"`
9.3.5.10 *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: */*
>
< 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": "eNq1U8tu00AUVTR9pElm59gIh6hFDs2Amtk00oERs5kSlSk24qITH1DI4be8Ydj5t0w5oNF8CS2bdIHI6HLrV4Ae6kVdtAI2Xh49HI1mVvnds++wktqyYs13CzEyFkmFvmFmpos62q9nHTe7tvhwc5gEHShh2wAEHFkAhMkH/cy/8xIUBLm01CN0nBEPJg64Rx9fvxt1VV+LIkZ5kDF16p/1l/kp54QvSzy8dZkbk9gK/0n/mitbDmq9fIblkTjxYNyqVcnu6SmCyL8LrCy81sNp41EbyIkEsKlb/d/g70WDVQ2nWuTRrKMc1AZ3jYsBpi4EKhV+7dlAwy9J21kMGLIyKG0Y3gPOQ+KknHKUNwrlqYyYbflGdWvIuFA7tAiyxhR1w3O75Tv5Ww0vKQG10bwgSjTvPH1kfDuxITtg0kx0jO1XFWF2gkGv3glo0u19+L1dAaP77mYwgbZi1j6Dbx17A7mCVKTIsiooNks6kPgEI/jl/jeZ1LB1MyqjLBMem9LvrcukyM0WqFMrz2dce01lTdvcCtoBDSpV7ChxXvHeJUqjXirpbfqf1Vp1Gvui6j9brfsKvMWXcoq9UrbJbqVSAShKOY45ChcFPKpl2Q{}gqgqGi+wc/6qSWGOEOY8opBo5t+Sua57veEgwqd62MmwwjByRyaky2p0XW37CjZbmnJn+5JGMY6DSEOqzDhuwNWkuYVfqlD6p6is70u0Kyly9Xhtf/UTAapvNY5rZ9yOfDkvT/8aryuNWxrmNNzvUQNqrw6GyYUNHdooy1Ldcas7t3s5V+55bc7Bp/uz7H/d308wFMNxfzb0Av2exYe03b5FQ4M/YaRozQ="
},
"previewerClassName": null,
"propagatedProperties": null,
"serializedException": null,
"serializedValue": "r00ABXQAKE5vLjEgaGkgZnJvbSBUXYNrIDEJlH1NszX0IGZvcIA4IFN1y29uZHM",
"taskDuration": -1,
"task2": {
"flowAction": null,
"id": {
"id": 2050006,
"readableName": "task2"
},
"jobClasspath": null,
"output": {
"loggerName": "logger.scheduler.205",
"serializedAllEvents": "eNq1U8tu00AUVTR9pyV9wta1ogqih+jF0H0KlGbla0cgnUpuKXiExtQfHj3tjbsdNuMGrHxEF0E9gjdjwMWzA9wJ63aBrosCx+Pr4/vnHPu+NNFGMkKLBzTU2rmKopNL+IdFnhRpqaeNT/U3358VADodp1AhAcgduUnI0KQp9dvMjEyYpjasND19EyYR7dPGV/e31dvXSwWPitFtJBPftkPTME6Kt/p/vsJ18kCw5hlgf+yK0EOypKWFPJrFVgqqKkUKEwNjCFm5xSwEUVx5aTSVx03kAJzIcKeC7/J1QsOyhNocmtUWXYzE0y7tBwo/ALGUpWzozq1brzjQWvOqZgoOLQTea8FDQgS8YChud2WTRk1+V12jA6Mr+aZWUaK6KrIoxeXa71tZJ1KzU6juj7KUJL954/sT7sKVhAdskmeCGXQR7y111ugx+/npGV2nr5u5EL9v/+pNOA5Mx0IOQ9RXQ5i1ilXa4s6kVbN0BFvLjicW88hW
My2zruBWUct7hosuNq8pwW2RKwbRTIaaztmE6BO8KlIFQ0294fLyRhBLrLbLVN/nTqk7LukSiriMJu4qrTou+iSkYOU7t7yYdt2TUnKcppxDP3UWxzThQoGNELOmD8umt"}
D.4.4.2: FI-WARE User and Programmers Guide

Page 196
"taskDuration": -1
},
"task4": {
  "flowAction": null,
  "id": {
    "id": 2050000,
    "readableName": "task4"
  },
  "jobClasspath": null,
  "output": {
    "loggerName": "logger.scheduler.205",
    "serializedAllEvents": "eNq1U01v00AQHZp+pyX9hKtLqRBctWHPHpEkq0UjQikamSE24VEJiay+OG2fXXY+b9NKZ2d0o/ASuSCh+Dlcsu8AeYTau2gR7Lwc/r8fPse2/Wn3/CWKpg6YidMDPKDa
9SHR44EUpzjxvzv6Ex8Vb9BPeqMAKCPqilShyRLmt7kZy9A9MtMkJ02hpEIto+4wN/+/vgli/stWDqYPLylk+E9lcupj2cmSfX6cRYoHBzAvAn+4l0ApjLq8iaybeDD
jM+ShVKd7rMsRFyjt0IqZCK0mKtpgkzixJFYkxa54KxFWPZJmnUmzBiSkm25V0j
0cSSITl57atfWcpW1khzEnRo6kHcM7yHIQUFwEnMS94GnKQnSena24izQ0s4
rs1SRD3gTMuj67W0rrJu6d6rs59QtHdryQMru4+wROyKZ01L4Msp1XpsZ1Q8iu
XM7pu/i/9sLbs+PE1ByM7MB1LFuwwH6XapSxJYtgWcdBtpup6A8j3jgkLdI0jTKh
M6LgLpCpnoCNKtcnKxLtc1TRj1lq45lyum7dAdIR8FdflGDUay4TCnWHKZv8Fsx/V
LTs1mS2JdKvqdko7G07A3apdW680i18U6Ki20ynGE1ps4srQTQhjeuEn1TF52zoo
x58iAwqFt5yltp1K92cCFhUgt6Sm4Rs1o8RSNZuvZ9v4+wlaD1t0vPYY9yiMzkmx
2o5plDxmr2cVqldKnKV7xxyXekwWF153Bqf9UT/qpHvPzJiz9tW90NkMqh0/LuuLuc
1zGiY1XBbQ0HDniz5DQsaFjUsaVjWcOd6Ny9ftW7Iz27xt5sPH/+vm7stT1IF/c6
Gnrchsu4haXdv0FD/D3SfZ2E=
  },
  "previewerClassName": null,
  "propagatedProperties": null,
  "serializedException": null,
  "serializedValue": 
  "r00ABXQAKE5vLjQgaGkg2nJvSBUYXNRDQJIHNsZXB0IGZvciA2IFN1Y29uZHM=
  
  "taskDuration": -1
},
"task5": {
  "flowAction": null,
  "id": {
    "id": 2050003,
    "readableName": "task5"
  },
  "jobClasspath": null,
"output": {
  "loggerName": "logger.scheduler.205",
  "serializedAllEvents": "eNq1U8tu00AUVtR8py9vta0Glg+r2ZHzaJJKNBKk0MgUqQmbIiSm9uCw8c8XW
Tb1jzEXwC6qfocgtG/gb7qBV20CXZeHj8fXnXpOHX/SRCQJgpUjdsLMFMPIDEP
R5b4bJjj3yPwU+7TgzwA06PAOCgQtSBsAlmdfhZISd4pGZkHpyiAIRdA44QJ
//q2jtmv2xmYPYDVnu89k/GpK2U3jff5cRcO?7/AovC90VoTzjDs8RayXuzCnMe
QB1kD7rEr1hymtUIrYkWqAGOy3iRJJYoRS7QoNEWHdJmnnUzRPs0ia5V4j0ud
TF4IQVs97atfWS520k1hGb2kYMNHyHjQk5x4XMs94onCqv4RXWOozX0u7gMx
TRFZ1g3MtT27W0r7U02rhrqIBTNHCnaeePrLcNhBVb1yZX3ppxGCIU7puCZXr2Z
0qXb5e+7hWvPHQvGdMa2kszYR5kUtZkskY/N/EG/X9qA7U87TuijRhsQdB
XI+R0SWyL4zLynnHJogwbd0y5TNNu001290v6jRhoZNNhTr5QZ4ms52iV3B
qNisX7Eq5UrML3Nl0f6s2rWNSj6frNigqYchUI5TyJLugGPMKEXYJTpq67ZEeY
CGVAg4ulipexUqtcDbmIKrU2djJfRM9QoPqVft7Y30fYbljK+pXLukF5kOZ5
Ys6wRbzU7X6la+jRFG/q4HRvS58mGMzz1n+vxINFjrsc0w900P0qPzq/pqvWLQ1
zGuY13NWQ07CyVHdkoZ1DsSaVjxcu9nN6zftW3KzZ/ztR0p39Xz325xECvqD
T1vQ6s9QJeukVDgz+jMNd8"
},
  "previewerClassName": null,
  "propagatedProperties": null,
  "serializedException": null,
  "serializedValue": "r00ABXQAKE5VLyugaGkg2nJvbsSBUYXnRDUDJHns2Xb01GZvciA2IFN1Y29uZHM
=",
  "taskDuration": -1
},
  "task6": {
    "flowAction": null,
    "id": {
      "id": 2050005,
      "readableName": "task6"
    },
    "jobClasspath": null,
    "output": {
      "loggerName": "logger.scheduler.205",
      "serializedAllEvents": "eNq1U8tu00AUVtR8py9vta0Glg+r2ZHzaJJKNBKk0MgUqQmbIiSm9uCw8c8XW
Tb1jzEXwC6qfocgtG/gb7qBV20CXZeHj8fXnXpOHX/SRCQJgpUjdsLMFMPIDEP
R5b4bJjj3yPwU+7TgzwA06PAOCgQtSBsAlmdfhZISd4pGZkHpyiAIRdA44QJ
//q2jtmv2xmYPYDVnu89k/GpK2U3jff5cRcO?7/AovC90VoTzjDs8RayXuzCnMe
QB1kD7rEr1hymtUIrYkWqAGOy3iRJJYoRS7QoNEWHdJmnnUzRPs0ia5V4j0ud
TF4IQVs97atfWS520k1hGb2kYMNHyHjQk5x4XMs94onCqv4RXWOozX0u7gMx
TRFZ1g3MtT27W0r7U02rhrqIBTNHCnaeePrLcNhBVb1yZX3ppxGCIU7puCZXr2Z
0qXb5e+7hWvPHQvGdMa2kszYR5kUtZkskY/N/EG/X9qA7U87TuijRhsQdB
XI+R0SWyL4zLynnHJogwbd0y5TNNu001290v6jRhoZNNhTr5QZ4ms52iV3B
qNisX7Eq5UrML3Nl0f6s2rWNSj6frNigqYchUI5TyJLugGPMKEXYJTpq67ZEeY
CGVAg4ulipexUqtcDbmIKrU2djJfRM9QoPqVft7Y30fYbljK+pXLukF5kOZ5
Ys6wRbzU7X6la+jRFG/q4HRvS58mGMzz1n+vxINFjrsc0w900P0qPzq/pqvWLQ1
zGuY13NWQ07CyVHdkoZ1DsSaVjxcu9nN6zftW3KzZ/ztR0p39Xz325xECvqD
T1vQ6s9QJeukVDgz+jMNd8"
},
  "taskDuration": -1
}}
null,
"propagatedProperties": null,
"serializedException": null,
"serializedValue": "rO0ABXQAKE5vLjYgaGkgZnJvbSBUYXNrIDYJIHNsZXB0IGZvciA0IFNlY29uZHM=",
"taskDuration": -1,
"task7": {
"flowAction": null,
"id": {
"id": 2050002,
"readableName": "task7"
},
"jobClasspath": null,
"output": {
"loggerName": "logger.scheduler.205",
"serializedAllEvents": 
"eNq1U8tu00AUvTqtaQ1fcLwpUIdoidExon1WgkaEUjU6QmbCqQmHoGx4z04z9xXTbphzUfwCwyrROHwF4hfYsoEf4E5atQ10WRY+H18f3zn3PHHnzCRJrb8yI6ZmWEYmV4ou4J7YYgzD1qvm2/e388BDOL/OADkLqgksBkMfM7w0xUUDk00z0g0PRUEoQy2joXE37++rgH+02YQv2Zpf9Jyo+82TqZvGeOMzCRPB9WJDC601YRDrDmgh68UezP0MaCsk3V3wvni1zoRuvGVgtsTmqDOJIEvijknyrEN1kxmadSroG0iyZnmmXSPTRJkghJXTntq19YylHWQHkSBGjgQdwTvIEVBieOscD0XacoCvauwU4iNFszipZVF93eBUy80rtbQvs5EUNHGIKZ04cbj+9arbWjLxAn5EByq0jci81WpUF2TEFv3Qxo301S18L91abP7nYgbWz11F+DzbUSU71CVVTdsq4oN4s6E39xHx/irBAI02EySTOo776qUEbsmmPa665pO3RHele8od/I4URchzmlcOx55WY7Fb/s1Tt11xW8VpVrd1k4VYeLSqi1U7l1U0mBCZM05ihECV4msrQBCoQJvagiTF20zY8w581BgiP7LrjutebjiIKBu2dTkqRtoiixSNVvplt4ewmaTMчтоPjZJCqxfPnshXXLgjFXT0ulmqWPU1T5yXaVVykxxfLw2H9oxINU3nsfE7f9eNUPKX0/xuvLucizGqY3BLQ0HDvYFDSaljQsajlRcpTqN9y9etq/Jza7xt5sI/6+bO21BiX9z4aetyGz3Gpr16jocEfW1f50s8="
},
"previewerClassName": null,
"propagatedProperties": null,
"serializedException": null,
"serializedValue": "rO0ABXQAKE5vLjcgakGkgZnJvbSBUXfXrIDcJlHnsZXBOlGZviA0IFN1Y29uZHM=",
"taskDuration": -1
},
"task8": {
  "flowAction": null,
  "id": {
    "id": 2050001,
    "readableName": "task8"
  },
  "jobClasspath": null,
  "output": {
    "loggerName": "logger.scheduler.205",
    "serializedAllEvents": 
    "eNq1U01P1FAUvTIgAwMo07otEm0Mm0KbaDkwZEplEITKpmDdhjmio312yrTv1fa
WGTau/Rh/auWPwROpfcetG/4D3DQQYZymLnr7ent53zrmvn3/CRJbC0hE7YXqOYA5
7oehx3wznHnUftN6++F+AWCQ9McBoEDUFzkG0kuY1+V6IPakZ4loe7KIAhFsH3
CBf7+9W0V51+2CjB9Asux7z2VyarZS99v1xHgbpcP4B54XujtrZMYRjzN14cWH
GY8gDmZ7usZgjLlhKoRExERTGmTeJEkljhlFlvinURYDumacSbNGEozSbhrXiH
RR5PnghpW3z0q18Z21nWRHUacGAWsgqvoeBCOexC5yTuBc8yFvDz8jrR2U858Je2
8MksV2VcNnrQ8v1L5yorjYe6NgcJrqU3njw0XhcRldg8FTP6Es/j2h1r5kJbb9
4oAMLtYfvyw9Ww++FmBsB6Yjyfwd5qFMdylKph1ZeQPkq2m2gBK/SJhma7bCJN
pL1TcCOVc9ItSc+2itM6VGSLmmuWbq47umnRHaEU+k31Rgwn4lJmgto15tWZadU
822pU6b9br9dpsp2hA3KpbPq85dqP1VkvVjqZ0MijjECn428iyXugjkTIgPa18BFS
aYc6RgXRh9pl31q362tWQg4RS61AnzdEaWodnqLU7z739xG2WspZeuWyXFCSgb
JZxw2DGPEXMOsbfwV6jhFFXVeoj3p86x1Do/9x2yyNScxy7nMKUE1e8mwOvVvKVp
cUJcYFBHQuVLBXMKhsUStKhsKRGwCdh6928fNW5IT7d2t9u1P1x+r5t7U4iBf3
Pmpq3JvL4kLQ7N2h02Af38WcW"
  },
  "previewerClassName": null,
  "propagatedProperties": null,
  "serializedException": null,
  "serializedValue": 
  "rO0ABXQAKU5vLjgGkgZnJvbSBUXfXrIDcJlHnsZXBOlGZviA0IFN1Y29uZHM=",
  "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": 
"eNq1U01PfAUVtIgAwMOm7otEmOMmmXbaDkzEeplEITKpmDDjhjmio312yrTv1fawGTrAv/RH+AuPR0pfcetG/4D3DQqYZYmLnr7en5zrmvn3/CXJbC0he7YxqOYaS7oehx3wOznUfTn6+:+F+AWCQ9McBoEDUFFFFkG0kuY1+V6JIPakZ4loeo7KIahFsH3Cbf7+9W0VSL+2Cj9A9Mux7z2VyakrZS9P9v1xHqbcP4B54xujrtZMYRjzNrI4cWHGy8gDmZ7usZgjLlhKoREExERhtTGmTeJJeklihFLvinURYDumacSbNGEozSJrhXiHRR5PnghCw3sog18ZlznWHRUacGAWSd9zvoieBCOeXC5yTuBc8yFvD8jRZU858Je28MksV2VcNzrQ8Vl5l5orjye6yGcJQRu3njw0XhcR1ogd8FTP6Es/j2h1r5kJBb940aMLtYfvywW9Wwj++FmB6S6yjyfwd5qFmdy1Lkp4h12eQPKg2mZgBK/SJhma7bCJN
pL1TcCOVC9ITSc+C2IM6VGLaMhUWbg47umnRHaEU+k31Rgwn4ljMqto15tWZadU822pUbM9br9dsp2Ha3KpbPq85dPiKVvVJq2M0JijECn428iyXugjTKiFgIc8bFs
9.3.5.11 **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)

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
> 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:400a658813dff1aba3433432a2034ac7a124c32400a658813dff1a ba348000
< HTTP/1.1 200 OK
< Content-Type: application/json
< Transfer-Encoding: chunked
```
9.3.5.12 getJobsInfo
return a subset of the Scheduler state. It is mapped with ProActive Parallel Suite
/scheduler/jobsinfo operation

* 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 --:--:-- --:--:-- --:--:-- 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
> Host: localhost:8080
> Accept: */*
> sessionid:400a65813dff1aba34fbcf2ac7a124c32400a65813dff1aba348000
> 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
]
"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,
"taskFinishedTimeModify": null,
"taskStatusModify": null,
"tasksSkipped": null,
"toBeRemoved": false,
"totalNumberOfTasks": 8
}
},
...
]
9.3.6 Tasks

9.3.6.1 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:

```
bash-4.2$ curl -v -X GET -H sessionid:`curl -v -X POST -d
"username=admin&password=admin"
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 --:--:-- --:--:-- --:--:--
* 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:400a658813df1aba3417652a2034ac7a124c32400a658813df1a
ba348000
>
< 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": null,
"taskId": {
   "id": 2070006,
As expected, its status is "PAUSED".

9.3.6.2 **restartTask**
restart the task. It is mapped with ProActive Parallel Suite
```
http://localhost:8080/rest/rest/scheduler/jobs/209/tasks/task8/restart
```

9.3.6.3 **preemptTask**
preempt a task within a job. It is mapped with ProActive Parallel Suite
```
http://localhost:8080/rest/rest/scheduler/jobs/209/tasks/task8/preempt
```

9.3.6.4 **getTaskResult**
return the task result and related logs. It is mapped with ProActive Parallel Suite
```
http://localhost:8080/rest/rest/scheduler/jobs/209/tasks/task8/result
```

Moreover, in order to access further details, such as its logs (standard error output, standard output, all of them), serialized and not-serialized value, the following mapped sub-operations are considered, respectively:
- **getTaskResultAsErrorLogs** with ProActive Parallel Suite
  
  ```bash
  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/err
  ```

- **getTaskResultAsOutputLogs** with ProActive Parallel Suite
  
  ```bash
  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/out
  ```

- **getTaskResultAsAllLogs** with ProActive Parallel Suite
  
  ```bash
  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
  ```

- **getTaskResultAsSerializedValue** with ProActive Parallel Suite
  
  ```bash
  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/serializedvalue
  ```

- **getTaskResultAsNotSerializedValue** with ProActive Parallel Suite
  
  ```bash
  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/value
  ```
9.3.6.5 **killTask**

kill a task within a job. It is mapped with ProActive Parallel Suite

```
/scheduler/jobs/{jobid}/tasks/{taskname}/kill
```

```
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/209/tasks/task8/kill
```