OAR Documentation - REST API

Dedication

For users whishing to make programs interfaced with OAR

Abstract: OAR is a resource manager (or batch scheduler) for large clusters. By it's functionnalities, it's near of PBS, LSF, CCS and Condor. It's suitable for productive plateforms and research experiments.

BE CAREFULL : THIS DOCUMENTATION IS FOR OAR >= 2.4.0

PDF version : OAR-DOCUMENTATION-API.pdf

Table of Contents

1 Introduction

The OAR REST API is currently a cgi script being served by an http server (we recommend Apache) that allows the programming of interfaces to OAR using a REST library. Most of the operations usually done with the oar Unix commands may be done using this API from your favourite language.

2 Concepts

2.1 Access

A simple GET query to the API using wget may look like this:

```
# Get the list of resources
wget -O - http://www.mydomain.org/oarapi/resources.yaml?structure=simple
```

You can also access to the API using a browser. Make it point to http://www.myoarcluster.local/oarapi/index.html and you'll see a very simple HTML interface allowing you to browse the cluster resources and even post a job using a form. (of course, replace www.myoarcluster.local by a valid name allowing you to join the http service of the host where you installed the oar api)

But generally, you'll use a REST client or a REST library provided for your favorite language. You'll see examples using a ruby rest library in the next parts of this document.

2.2 Authentication

- **Set-up**:

  The API authentication relies on the authentication mechanism of the http server used to serve the CGI script. The API may be configured to use the IDENT protocol for authentication from trusted hosts, like a cluster frontend. In this case, a unix login is automatically used by the API. This only works for hosts that have been correctly configured (for which the security rules are trusted by the administrator). If IDENT is not used or not trusted, the API can use the basic HTTP authentication. You may also want to set-up https certificates. In summary, the API authentication is based on the http server's configuration. The API uses the **X_REMOTE_IDENT** http header variable, so the administrator has to set up

this variable inside the http server configuration. Look at the provided apache sample configuration files (API/apache2.conf of the OAR sources) for more details.

- **Usage**:

  Most of the time, you'll make requests that needs you to be authenticated. If the IDENT method is used, your unix login is automatically used. But as only a few hosts may be trusted, you'll probably have to open a tunnel to one of this host. You may use ssh to do this. For example, supposing access.mycluster.fr is a gateway host trusted by the api host:

  ```
  $ ssh -NL 8080:api.mycluster.fr:80 login@access.mycluster.fr
  ```

  Then, point your REST client to:

  ```
  # http://localhost:8080
  ```

2.3 Data structures and formats

The API currently can serve data into *YAML*, *JSON* or *HTML*. Posted data can also be coded into *YAML*, *JSON* or *x-www-form-urlencoded* (for HTML from posts). You may specify the requested format by 2 ways:

- giving an extension to resources: **.yaml**, **.json** or **.html**
- setting the **HTTP_ACCEPT** header variable to **text/yaml**, **application/json** or **text/html**

For the posted data, you have to correctly set the **HTTP_CONTENT_TYPE** variable to **text/yaml**, **application/json** or **application/x-www-form-urlencoded**.

Sometimes, the data structures returned (not the coding format, but the contents: array, hashes, array of hashes,...) may be changed. Currently, we have 2 available data structures: *simple* and *oar*. The structure is passed through the variable *structure* that you may pass in the url, for example: ?structure=simple

- The **simple** data structure tries to be as simple as possible, using simple arrays in place of hashes wherever it is possible
- The **oar** data structure serves data in the way oar does with the oarnodes/oarstat export options (-Y, -D, -J,...)

By default, we use the *simple* data structure.
Here are some examples, using the ruby restclient (see next section):

```
# Getting resources infos
  # in JSON
irb(main):004:0> puts get('/resources.json')
  # in YAML
irb(main):005:0> puts get('/resources.yaml')
  # Same thing
irb(main):050:0> puts get('/resources', :accept=>"text/yaml")
  # Specifying the "oar" data structure
irb(main):050:0> puts get('/resources.json?structure=oar')
  # Specifying the "simple" data structure
irb(main):050:0> puts get('/resources.json?structure=simple')
```

2.4 Errors and debug

When the API returns an error, it generally uses a standard HTTP return status (404 NOT FOUND, 406 NOT ACCEPTABLE, ...). But it also returns a body containing a hash like the following:

```
{
 "title" : "ERROR 406 - Invalid content type required */*",
 "message" : "Valid types are text/yaml, application/json or text/html",
 "code" : "200"
}
```

This error body is formated in the requested format. But if this format was not given, it uses JSON by default.

To allow you to see the error body, you may find it useful to activate the **debug=1** variable. It will force the API to always return a 200 OK status, even if there's an error so that you can see the body with a simple browser or a rest client without having to manage the errors. For example:

```
wget -nv -O - "http://localhost:8080/oargridapi/sites/grenoble?debug=1"
```

Here is an example of error catching in ruby:

```
# Function to get objects from the api
# We use the JSON format
def get(api,uri)
  begin
    return JSON.parse(api[uri].get(:accept => 'application/json'))
  rescue => e
    if e.respond_to?('http_code')
      puts "ERROR #{e.http_code}:\n #{e.response.body}"
    else
      puts "Parse error:"
      puts e.inspect
    end
    exit 1
  end
end
```

3 Ruby REST client

One of the easiest way for testing this API is to use the rest-client ruby module:
http://rest-client.heroku.com/rdoc/

It may be used from ruby scripts (http://www.ruby.org/) or interactively. It is available as a rubygem, so to install it, simply install rubygems and do "gem install rest-client". Then, you can run the interactive client which is nothing else than irb with shortcuts. Here is an example irb session:

```
$ export PATH=$PATH:/var/lib/gems/1.8/bin
$ restclient http://localhost/oarapi
irb(main):001:0> puts get('/jobs.yaml')
---
```

```
- api_timestamp: 1246457384
  id: 551
  name: ~
  owner: bzizou
  queue: besteffort
  state: Waiting
  submission: 1245858042
  uri: /jobs/551
=> nil
irb(main):002:0>
```

or, if an http basic auth is required:

```
restclient http://localhost/api <login> <password>
...
```

4 REST requests description

Examples are given in the YAML format because we think that it is the more human readable and so very suitable for this kind of documentation. But you can also use the JSON format for your input/output data. Each resource uri may be postfixed by .yaml, .jso of .html.

In this section, we describe every REST resources of the OAR API. The authentication may be:

- public: everybody can query this resource
- user: only authenticated and valid users can query this resource
- oar: only the oar user can query this resource (administration usage)

4.1 GET /index

description: Home page for the HTML browsing

formats: html

authentication: public

output: ***example*:**

```
<HTML>
<HEAD>
<TITLE>OAR REST API</TITLE>
</HEAD>
<BODY>
<HR>
<A HREF=./resources.html>RESOURCES</A>   
<A HREF=./jobs.html>JOBS</A>   
<A HREF=./jobs/form.html>SUBMISSION</A>   
<HR>
Welcome on the oar API
```

note: Header of the HTML resources may be customized into the **/etc/oar/api_html_header.pl** file.

4.2 GET /version

description: Gives version informations about OAR and OAR API. Also gives the timezone of the API server.

formats: html , yaml , json

authentication: public

output: ***structure*:** hash

***yaml example*:**

```
---
api: 0.1.2
api_timestamp: 1245582255
api_timezone: CEST
apilib: 0.1.6
oar: 2.4.0
```

usage example: `wget -q -O - http://localhost/oarapi/version.yaml`

4.3 GET /timezone

description: Gives the timezone of the OAR API server

formats: html , yaml , json

authentication: public

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1245768107
timezone: CEST
```

usage example: `wget -q -O - http://localhost/oarapi/timezone.yaml`

4.4 GET /jobs

description: List currently running jobs

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes (a job is an array element described by a hash)

***yaml example*:**

```
---
- api_timestamp: 1245768256
  id: 547
  name: ~
  owner: bzizou
  queue: default
  resources_uri: /jobs/547/resources
  state: Running
  submission: 1245768249
```

```
  uri: /jobs/547
- api_timestamp: 1245768256
  id: 546
  name: ~
  owner: bzizou
  queue: default
  resources_uri: /jobs/546/resources
  state: Running
  submission: 1245768241
  uri: /jobs/546
```

note: You can make a GET on the *uri* value for more details about a given job.

usage example: `wget -q -O - http://localhost/oarapi/jobs.yaml`

4.5 GET /jobs/details

description: List the current jobs and some details like assigned resources (behaves like "oarstat -D")

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes (a job is an array element described by a hash)

***yaml example*:**

```
---
- Job_Id: 575
  api_timestamp: 1253103710
  array_id: 575
  array_index: 1
  assigned_network_address:
    - bart-3
  assigned_resources:
    - 3
  command: /bin/sleep 300
  cpuset_name: bzizou_575
  dependencies: []
  jobType: PASSIVE
  job_id: 575
  launchingDirectory: /home/bzizou
  message: FIFO scheduling OK
  name: Test_job
  owner: bzizou
  project: default
  properties: desktop_computing = 'NO'
  queue: default
  reservation: None
  resources_uri: /jobs/575/resources
  resubmit_job_id: 0
  startTime: 1253103705
```

```
  state: Running
  submissionTime: 1253103704
  types: []
  uri: /jobs/575
- Job_Id: 576
  api_timestamp: 1253103710
  array_id: 576
  array_index: 1
  assigned_network_address: []
  assigned_resources: []
  command: /bin/sleep 300
  cpuset_name: bzizou_576
  dependencies: []
  jobType: PASSIVE
  job_id: 576
  launchingDirectory: /home/bzizou
  message: FIFO scheduling OK
  name: Test_job
  owner: bzizou
  project: default
  properties: desktop_computing = 'NO'
  queue: default
  reservation: None
  resources_uri: /jobs/576/resources
  resubmit_job_id: 0
  startTime: 0
  state: Waiting
  submissionTime: 1253103707
  types: []
  uri: /jobs/576
```

note: You can make a GET on the *uri* value for more details about a given job.

usage example: `wget -q -O - http://localhost/oarapi/jobs/details.yaml`

4.6 GET /jobs/table

description: Dump the jobs table (only current jobs)

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes (a job is an array element described by a hash)

***yaml example*:**

```
---
- accounted: NO
  api_timestamp: 1253017554
  array_id: 566
  assigned_moldable_job: 566
  checkpoint: 0
```

```
  checkpoint_signal: 12
  command: ''
  exit_code: ~
  file_id: ~
  info_type: bart:33033
  initial_request: oarsub -I
  job_env: ~
  job_group: ''
  job_id: 566
  job_name: ~
  job_type: INTERACTIVE
  job_user: bzizou
  launching_directory: /home/bzizou/git/oar/git
  message: FIFO scheduling OK
  notify: ~
  project: default
  properties: desktop_computing = 'NO'
  queue_name: default
  reservation: None
  resubmit_job_id: 0
  scheduler_info: FIFO scheduling OK
  start_time: 1253017553
  state: Launching
  stderr_file: OAR.%jobid%.stderr
  stdout_file: OAR.%jobid%.stdout
  stop_time: 0
  submission_time: 1253017551
  suspended: NO
  uri: /jobs/566
- accounted: NO
  api_timestamp: 1253017554
  array_id: 560
  assigned_moldable_job: 0
  checkpoint: 0
  checkpoint_signal: 12
  command: /usr/bin/id
  exit_code: ~
  file_id: ~
  info_type: 'bart:'
  initial_request: oarsub --resource=/nodes=2/cpu=1 --use_job_key
  job_env: ~
  job_group: ''
  job_id: 560
  job_name: ~
  job_type: PASSIVE
  job_user: bzizou
  launching_directory: /home/bzizou
  message: Cannot find enough resources which fit for the job 560
  notify: ~
  project: default
```

```
properties: desktop_computing = 'NO'
queue_name: default
reservation: None
resubmit_job_id: 0
scheduler_info: Cannot find enough resources which fit for the
start_time: 0
state: Waiting
stderr_file: OAR.%jobid%.stderr
stdout_file: OAR.%jobid%.stdout
stop_time: 0
submission_time: 1246948570
suspended: NO
uri: /jobs/560
```

note: You can make a GET on the *uri* value for more details about a given job.

note: Field names may vary from the other job lists because this query results more like a dump of the jobs table.

usage example: `wget -q -O - http://localhost/oarapi/jobs/table.yaml`

4.7 GET /jobs/<id>

description: Get details about the given job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
Job_Id: 547
id: 547
uri: /jobs/547
array_id: 547
array_index: 1
assigned_network_address:
  - liza-2
assigned_resources:
  - 6
command: ''
cpuset_name: bzizou_547
dependencies: []
events:
  - date: 1245775464
    description: User root requested to frag the job 547
    event_id: 1315
    job_id: 547
    to_check: NO
    type: FRAG_JOB_REQUEST
```

```
  - date: 1245775464
    description: '[sarko] Job [547] from 1245768251 with 7200; cu
    event_id: 1316
    job_id: 547
    to_check: NO
    type: WALLTIME
  - date: 1245775464
    description: '[Leon] Send kill signal to oarexec on liza-2 fo
    event_id: 1318
    job_id: 547
    to_check: NO
    type: SEND_KILL_JOB
  - date: 1245775469
    description: '[bipbip 547] Ask to change the job state'
    event_id: 1320
    job_id: 547
    to_check: NO
    type: SWITCH_INTO_ERROR_STATE
exit_code: ~
initial_request: ''
jobType: INTERACTIVE
job_uid: ~
job_user: bzizou
launchingDirectory: /home/bzizou
message: FIFO scheduling OK
name: ~
owner: bzizou
project: default
properties: desktop_computing = 'NO'
queue: default
reservation: None
resources_uri: /jobs/547/resources
resubmit_job_id: 0
scheduledStart: ~
startTime: 1245768251
state: Error
submissionTime: 1245768249
types: []
walltime: 7200
wanted_resources: "-l \"{type = 'default'}/resource_id=1,walltime
```


usage example: `wget --user test --password test -q -O - http://localhost/oa`

4.8 GET /jobs/<id>/resources

description: Get resources reserved or assigned to a job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: public

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253279408
assigned_nodes:
  - node: liza-1
    node_uri: /resources/nodes/liza-1
assigned_resources:
  - resource_id: 4
    resource_uri: /resources/4
  - resource_id: 5
    resource_uri: /resources/5
job_id: 622
job_uri: /jobs/622
reserved_resources: []
```

usage example: `wget -q -O - http://localhost/oarapi/jobs/547/resources.yaml`

4.9 POST /jobs/<id>/deletions/new

description: Deletes a job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253025331
cmd_output: |
  Deleting the job = 567 ...REGISTERED.
  The job(s) [ 567 ] will be deleted in a near future.
id: 567
status: Delete request registered
```

usage example: `irb(main):148:0> puts post('/jobs/567/deletions/new.yaml',''`

4.10 POST /jobs/<id>/checkpoints/new

description: Send the checkpoint signal to a job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253025555
cmd_output: |
```

```
  Checkpointing the job 568 ...DONE.
  The job 568 was notified to checkpoint itself.
id: 568
status: Checkpoint request registered
```

usage example: `irb(main):148:0> puts post('/jobs/568/checkpoints/new.yaml',`

4.11 POST /jobs/<id>/holds/new

description: Asks to hold a waiting job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253025718
cmd_output: "[560] Hold request was sent to the OAR server.\n"
id: 560
status: Hold request registered
```

usage example: `irb(main):148:0> puts post('/jobs/560/holds/new.yaml','')`

4.12 POST /jobs/<id>/rholds/new

description: Asks to hold a running job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: oar

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253025868
cmd_output: "[569] Hold request was sent to the OAR server.\n"
id: 569
status: Hold request registered
```

usage example: `irb(main):148:0> puts post('/jobs/560/rholds/new.yaml','')`

4.13 POST /jobs/<id>/resumptions/new

description: Asks to resume a holded job

parameters:
- **id**: the id of a job

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253026081
cmd_output: "[569] Resume request was sent to the OAR server.\n"
id: 569
status: Resume request registered
```

usage example: `irb(main):148:0> puts post('/jobs/560/resumptions/new.yaml',`

4.14 POST /jobs/<id>/signals/<signal>

description: Asks to resume a holded job

parameters:
- **id**: the id of a job
- **signal**: the number of a signal (see kill -l)

formats: html , yaml , json

authentication: user

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1253102493
cmd_output: |
  Signaling the job 574 with 12 signal.
  DONE.
  The job 574 was notified to signal itself with 12.
id: 574
status: Signal sending request registered
```

usage example: `irb(main):148:0> puts post('/jobs/560/signals/12.yaml','')`

4.15 POST /jobs

description: Creates (submit) a new job

formats: html , yaml , json

authentication: user

input: Only [resource] and [command] are mandatory

structure: hash with possible arrays (for options that may be passed multiple times)

***fields*:**

- **resource** (*string*): the resources description as required by oar (example: "/nodes=1/cpu=2")
- **command** (*string*): a command name or a script that is executed when the job starts
- **workdir** (*string*): the path of the directory from where the job will be submited
- **All other option accepted by the oarsub unix command**: every long option that may be passed to the oarsub command is known as a key of the input hash. If the option is a toggle (no value), you just have to set it to "1" (for example: 'use-job-key' => '1'). Some options may be arrays (for example if you want to specify several 'types' for a job)

***yaml example*:**

```
---
stdout: /tmp/outfile
command: /usr/bin/id;echo "OK"
resource: /nodes=2/cpu=1
workdir: ~bzizou/tmp
type:
- besteffort
- timesharing
use-job-key: 1
```

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1245858042
id: 551
status: submitted
uri: /jobs/551
```

note: more informations about the submited job may be obtained with a GET on the provided *uri*.

usage example:

```
# Submitting a job using ruby rest client
irb(main):010:0> require 'json'
irb(main):012:0> j={ 'resource' => '/nodes=2/cpu=1', 'command' => '/
irb(main):015:0> job=post('/jobs' , j.to_json , :content_type => 'ap

# Submitting a job with a provided inline script
irb(main):024:0> script="#!/bin/bash
irb(main):025:0" echo \"Hello world\"
irb(main):026:0" whoami
irb(main):027:0" sleep 300
irb(main):028:0" "
irb(main):029:0> j={ 'resource' => '/nodes=2/cpu=1', 'script' => scr
irb(main):030:0> job=post('/jobs' , j.to_json , :content_type => 'ap
```

4.16 POST /jobs/<id>

description: Updates a job. In fact, as some clients (www browsers) doesn't support the DELETE method, this POST resource has been created mainly to workaround this and provide another way to delete a job. It also provides *checkpoint*, *hold* and *resume* methods, but one should preferably use the /checkpoints, /holds and /resumptions resources.

formats: html , yaml , json

authentication: user

input: *structure*: hash {"action" => "delete"}

***yaml example*:**

```
---
method: delete
```

output: *structure*: hash

***yaml example*:**

```
---
api_timestamp: 1245944206
cmd_output: |
  Deleting the job = 554 ...REGISTERED.
  The job(s) [ 554 ] will be deleted in a near future.
id: 554
status: Delete request registered
```

usage example:

```
# Deleting a job in the ruby rest client
puts post('/jobs/554.yaml','{"method":"delete"}',:content_type => "a
```

4.17 DELETE /jobs/<id>

description: Delete or kill a job.

formats: html , yaml , json

authentication: user

output: *structure*: hash returning the status

***yaml example*:**

```
---
api_timestamp: 1245944206
cmd_output: |
  Deleting the job = 554 ...REGISTERED.
  The job(s) [ 554 ] will be deleted in a near future.
id: 554
status: Delete request registered
```

usage example:

```
# Deleting a job in the ruby rest client
puts delete('/jobs/554.yaml')
```

note: Not all clients support the DELETE method, especially some www browsers. So, you can do the same thing with a POST of a {"method":"delete"} hash on the /jobs/<id> resource.

4.18 GET /jobs/form

description: HTML form for posting (submiting) new jobs from a browser

formats: html

authentication: user

output: ***example*:**

```
<HTML>
 <HEAD>
 <TITLE>OAR REST API</TITLE>
 </HEAD>
```

```
<BODY>
<HR>
<A HREF=../resources.html>RESOURCES</A>   
<A HREF=../jobs.html>JOBS</A>   
<A HREF=../jobs/form.html>SUBMISSION</A>   
<HR>

<FORM METHOD=post ACTION=../jobs.html>
<TABLE>
<CAPTION>Job submission</CAPTION>
<TR>
  <TD>Resources</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=resource VALUE="/nodes=1/cpu
</TR><TR>
  <TD>Name</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=name VALUE="Test_job"></TD>
</TR><TR>
  <TD>Properties</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=property VALUE=""></TD>
</TR><TR>
  <TD>Program to run</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=command VALUE='"/bin/sleep 3
</TR><TR>
  <TD>Types</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=type></TD>
</TR><TR>
  <TD>Reservation dates</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=reservation></TD>
</TR><TR>
  <TD>Directory</TD>
  <TD><INPUT TYPE=text SIZE=40 NAME=directory></TD>
</TR><TR>
  <TD></TD><TD><INPUT TYPE=submit VALUE=SUBMIT></TD>
</TR>
</TABLE>
</FORM>
```

note: This form may be customized in the **/etc/oar/api_html_postform.pl** file

4.19 GET /resources

description: Get the list of resources and their state

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes

***yaml example*:**

```
---
- api_timestamp: 1253201950
```

```
  jobs_uri: /resources/4/jobs
  network_address: liza-1
  node_uri: /resources/nodes/liza-1
  resource_id: 4
  state: Alive
  uri: /resources/4
- api_timestamp: 1253201950
  jobs_uri: /resources/5/jobs
  network_address: liza-1
  node_uri: /resources/nodes/liza-1
  resource_id: 5
  state: Alive
  uri: /resources/5
- api_timestamp: 1253201950
  jobs_uri: /resources/6/jobs
  network_address: liza-2
  node_uri: /resources/nodes/liza-2
  resource_id: 6
  state: Alive
  uri: /resources/6
- api_timestamp: 1253201950
  jobs_uri: /resources/7/jobs
  network_address: liza-2
  node_uri: /resources/nodes/liza-2
  resource_id: 7
  state: Alive
  uri: /resources/7
```

note: More details about a resource can be obtained with a GET on the provided *uri*. The list of all the resources of the same node may be obtained with a GET on *node_uri*. The list of running jobs on a resource can be obtained with a GET on the jobs_uri resource.

usage example: `wget -q -O - http://localhost/oarapi/resources.yaml`

4.20 GET /resources/full

description: Get the list of resources and all the details about them

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes

***yaml example*:**

```
---
- api_timestamp: 1253202216
  available_upto: 0
  besteffort: YES
  cluster: 0
  cpu: 3
  cpuset: 0
  deploy: YES
```

```
  desktop_computing: NO
  expiry_date: 0
  finaud_decision: NO
  jobs_uri: /resources/4/jobs
  last_available_upto: 0
  last_job_date: 1245825515
  licence: ~
  network_address: liza-1
  next_finaud_decision: NO
  next_state: UnChanged
  node_uri: /resources/nodes/liza-1
  resource_id: 4
  scheduler_priority: 4294967289
  state: Alive
  state_num: 1
  suspended_jobs: NO
  test: ~
  type: default
  uri: /resources/4
- api_timestamp: 1253202216
  available_upto: 0
  besteffort: YES
  cluster: 0
  cpu: 4
  cpuset: 1
  deploy: YES
  desktop_computing: NO
  expiry_date: 0
  finaud_decision: NO
  jobs_uri: /resources/5/jobs
  last_available_upto: 0
  last_job_date: 1240244422
  licence: ~
  network_address: liza-1
  next_finaud_decision: NO
  next_state: UnChanged
  node_uri: /resources/nodes/liza-1
  resource_id: 5
  scheduler_priority: 4294967293
  state: Alive
  state_num: 1
  suspended_jobs: NO
  test: ~
  type: default
  uri: /resources/5
- api_timestamp: 1253202216
  available_upto: 0
  besteffort: YES
  cluster: 0
  cpu: 5
```

```
  cpuset: 0
  deploy: NO
  desktop_computing: NO
  expiry_date: 0
  finaud_decision: NO
  jobs_uri: /resources/6/jobs
  last_available_upto: 0
  last_job_date: 1253198104
  licence: ~
  network_address: liza-2
  next_finaud_decision: NO
  next_state: UnChanged
  node_uri: /resources/nodes/liza-2
  resource_id: 6
  scheduler_priority: 0
  state: Alive
  state_num: 1
  suspended_jobs: NO
  test: ~
  type: default
  uri: /resources/6
- api_timestamp: 1253202216
  available_upto: 0
  besteffort: YES
  cluster: 0
  cpu: 6
  cpuset: 1
  deploy: NO
  desktop_computing: NO
  expiry_date: 0
  finaud_decision: NO
  jobs_uri: /resources/7/jobs
  last_available_upto: 0
  last_job_date: 1245671780
  licence: ~
  network_address: liza-2
  next_finaud_decision: NO
  next_state: UnChanged
  node_uri: /resources/nodes/liza-2
  resource_id: 7
  scheduler_priority: 0
  state: Alive
  state_num: 1
  suspended_jobs: NO
  test: ~
  type: default
  uri: /resources/7
```

usage example: `wget -q -O - http://localhost/oarapi/resources/full.yaml`

4.21 GET /resources/<id>

description: Get details about the resource identified by *id*

formats: html , yaml , json

authentication: public

output: *structure*: 1 element array of hash

***yaml example*:**

```
---
api_timestamp: 1253202322
available_upto: 0
besteffort: YES
cluster: 0
cpu: 20
cpuset: 0
deploy: NO
desktop_computing: NO
expiry_date: 0
finaud_decision: NO
jobs_uri: /resources/1/jobs
last_available_upto: 0
last_job_date: 1253201845
licence: ~
network_address: bart-1
next_finaud_decision: NO
next_state: UnChanged
node_uri: /resources/nodes/bart-1
resource_id: 1
scheduler_priority: 0
state: Alive
state_num: 1
suspended_jobs: NO
test: ~
type: default
uri: /resources/1
```

usage example: `wget -q -O - http://localhost/oarapi/resources/1.yaml`

4.22 GET /resources/nodes/<network_address>

description: Get details about the resources belonging to the node identified by *network_address*

formats: html , yaml , json

authentication: public

output: *structure*: array of hashes

***yaml example*:**

```
---
- api_timestamp: 1253202379
  jobs_uri: /resources/4/jobs
```

```
  network_address: liza-1
  node_uri: /resources/nodes/liza-1
  resource_id: 4
  state: Alive
  uri: /resources/4
- api_timestamp: 1253202379
  jobs_uri: /resources/5/jobs
  network_address: liza-1
  node_uri: /resources/nodes/liza-1
  resource_id: 5
  state: Alive
  uri: /resources/5
```

usage example: `wget -q -O - http://localhost/oarapi/resources/nodes/liza-1.`

4.23 POST /resources

description: Creates a new resource

formats: html , yaml , json

authentication: oar

input: A [hostname] or [network_address] entry is mandatory

structure: hash describing the resource to be created

***fields*:**

- **hostname** alias **network_address** (*string*): the network address given to the resource
- **properties** (*hash*): an optional hash defining some properties for this new resource

***yaml example*:**

```
---
hostname: test2
properties:
  besteffort: "NO"
  cpu: "10"
```

output: *structure*: hash returning the id of the newly created resource and status

***yaml example*:**

```
---
api_timestamp: 1245946199
id: 32
status: ok
uri: /resources/32
warnings: []
```

usage example:

```
# Adding a new resource with the ruby rest client (oar user only)
irb(main):078:0> r={ 'hostname'=>'test2', 'properties'=> { 'besteffo
irb(main):078:0> puts post('/resources', r.to_json , :content_type =
```

4.24 POST /resources/<id>/state

description: Change the state

formats: html , yaml , json

authentication: oar

input: A [state] entry is mandatory and must be “Absent”, “Alive” or “Dead”

structure: hash of state

***fields*:**

- **state**: Alive, Absent or Dead

***yaml example*:**

```
---
state: Dead
```

output: *structure*:

***yaml example*:**

```
---
api_timestamp: 1253283492
id: 34
status: Change state request registered
uri: /resources/34
```

usage example: `irb`

4.25 DELETE /resources/<id>

description: Delete the resource identified by *id*

formats: html , yaml , json

authentication: oar

output: *structure*: hash returning the status

***yaml example*:**

```
---
api_timestamp: 1245946801
status: deleted
```

usage example:

```
# Deleting a resource with the ruby rest client
puts delete('/resources/32.yaml')
```

note: If the resource could not be deleted, returns a 403 and the reason into the message body.

4.26 DELETE /resources/<node>/<cpuset_id>

description: Delete the resource corresponding to *cpuset_id* on node *node*. It is useful when you don’t know about the ids, but only the number of cpus on physical nodes.

formats: html , yaml , json

authentication: oar

output: *structure*: hash returning the status

***yaml example*:**

```
---
api_timestamp: 1246459253
status: deleted
=> nil
```

usage example:

```
# Deleting a resource with the ruby rest client
puts delete('/resources/test/0.yaml')
```

note: If the resource could not be deleted, returns a 403 and the reason into the message body.

5 Some equivalences with oar command line

OAR command	**REST request**
oarstat	GET /jobs.html
oarstat -Y	GET /jobs/details.yaml?structure=oar
oarstat -Y -fj <id>	GET /jobs/<id>.yaml
oardel <id>	DELETE /jobs/<id>.yaml
oardel <id> *(alternative way)*	POST /jobs/deletions/<id>/new.yaml
oarnodes -Y	GET /resources/full.yaml?structure=oar
oarnodes -Y -r1	GET /resources/1.yaml?structure=oar