Altair Engineering Inc.

Altair Grid Engine Documentation

Grid Engine Rest Service Guide

Author:
Altair Engineering

Version:
2025.1.0 (8.10.0)

January 10, 2025

Contents

1 Altair Grid Engine REST Service Overview

Altair Grid Engine REST Web Services gives access to a set of RESTful resources representing Altair Grid Engine internal data structures and functionality. They can be used to create, read, update, and delete various objects in Altair Grid Engine.

RESTful web services in general and their usage is explained in the following documentation. The underlying JSON data formats, the different API bindings and how to make use of the API for all communications with Altair Grid Engine is also shown in simple example code snippets here.

Representational State Transfer (REST) is a software architecture style consisting of guidelines and best practices for creating scalable web services.

REST has gained widespread acceptance across the Web as a simpler alternative to SOAP and WSDL-based Web services. RESTful systems typically, but not always, communicate over the Hypertext Transfer Protocol with the same HTTP verbs (GET, POST, PUT, DELETE, etc.) used by web browsers to retrieve web pages and send data to remote servers.

The REST architectural style was developed by W3C Technical Architecture Group (TAG) in parallel with HTTP 1.1, based on the existing design of HTTP 1.0.

The general architectural constraints that REST is built upon are:

- client-server
- stateless
- cacheable
- layered system
- code on demand (optional)
- uniform interface

Web service APIs that adhere to the REST architectural constraints are called RESTful APIs. HTTP-based RESTful APIs are defined with these aspects:

- base URI, such as http://example.com/resources/
- an Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g. XML, Atom, microformats, images, etc.)
- standard HTTP methods (e.g., GET, PUT, POST, or DELETE)
- hypertext links to reference state
- hypertext links to reference related resources

Applied to Altair Grid Engine REST Web Services this means

Table 1: REST methods, URLs and the services they provide

Method	URL	Description
GET	/adminhosts	get list of admin hosts
GET	/adminhosts/{id}	get admin host with {id}
GET	/adminhosts/{id}/{n}	get a range of {n} admin hosts starting with {id},

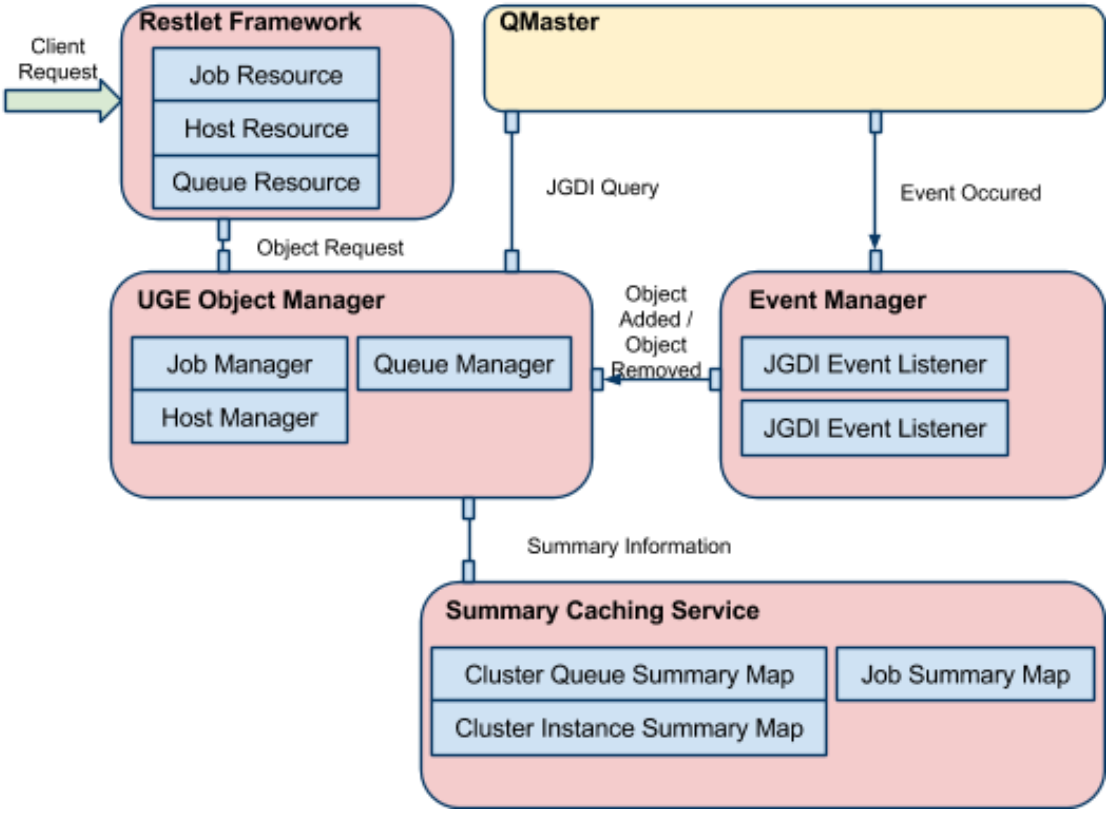
Method	URL	Description
		where {id} is kind of pattern matched
POST	/adminhosts	add a new admin host: post request contains a corresponding json data string
PUT	/adminhosts	modify an existing admin host: put request contains a corresponding json data string
DELETE	/adminhosts	delete an existing admin host: delete request contains a corresponding json data string

Altair Grid Engine REST Web Services provides RESTful web services for many resources. The methods and URLs available are documented in the [Resources, JSON Data Formats and usage with curl CLI] section.

2 General Overview of Altair Grid Engine REST Web Service API

The Altair Grid Engine REST Web Service is a web-based management, monitoring and submission API using modern web technologies. The server component is internally based on the Altair Grid Engine JGDI API and the Restlet development framework exposing Altair Grid Engine objects and functionality as REST resources to the application programmer.

The overall architecture is outlined in the following sketch:



The Altair Grid Engine REST Web Service runs as a standalone daemon on any host in the Altair Grid Engine cluster and connects via the JGDI API to the Altair Grid Engine master daemon. The web service registers for Altair Grid Engine JGDI events to update the corresponding REST resources.

The REST resources are exposed to client components or web browsers as a RESTful web service. In the sketch, Host, Queue and Job are shown as examples. The red boxes comprise the Altair Grid Engine REST Web Service with the JGDI API connections depicted as 'JGDI Query' and 'Event Occurred'; these are doing the actual communication with the Altair Grid Engine master.

3 Altair Grid Engine REST Service Installation and Setup

The Altair Grid Engine REST Service is an add-on to the Altair Grid Engine distributed resource management application that allows the access and manipulation of Altair Grid Engine resources. For a smooth installation process, the compute resources and the network infrastructure have to be prepared correctly. The following sections describe the necessary prerequisites, provide basic knowledge about the Altair Grid Engine REST Service, and ask questions that have to be answered by the Altair Grid Engine administrators during or before the installation process.

3.1 Planning the Installation

The following tasks and prerequisites have to be fulfilled for a successful installation of Altair Grid Engine REST Service.

Table 2: Tasks and prerequisites

Task	Explanation
Altair Grid Engine >8.3	Altair Grid Engine version greater than 8.3 needs to be installed. For detailed installation instructions see the Grid Engine Installation Guide
Altair Grid Engine REST Service host	Altair Grid Engine REST Service can be installed on any host that is part of the cluster. The host must be an admin and submit host though. If the service is started on the master host these conditions are usually fulfilled automatically
Java JDK >= 1.8	Java JDK greater than 1.8 must be installed to run the Altair Grid Engine REST Service due to restlet jar dependencies
(OPTIONAL) Python 2.7	For using the Altair Grid Engine REST Service Python Binding, Python 2.7 must be installed on Linux, Mac OS X and Solaris systems additionally.
(OPTIONAL) Node.js	For using the Altair Grid Engine REST Service Nodejs Binding node.js needs to be installed on Linux, Mac OS X and Solaris systems. NodeJs Blog NodeJS Zone for Solaris
(OPTIONAL) npm packages	Nodejs Package Manager packages that are required
(OPTIONAL) Meteor	Meteor system for JavaScript Meteor Binding
(OPTIONAL) Meteor packages	Meteor packages that are required

3.2 Sudo Configuration

The Altair Grid Engine REST Service runs under a special user e.g. user userrest. To be able to execute requests on behalf of a different user, so-called sudo requests have been introduced in Altair Grid Engine 8.3. They allow under special circumstances the submission of a request by user userrest for users user1, user2 etc. To guarantee that only privileged users

are allowed to change their identity the *sudomasters* access list has to be configured. In addition a *sudoers* access list must exist that contains all the abovementioned users for whom requests are executed. The following Altair Grid Engine commands have to be performed by the administrator:

- Where is the user under which the REST Service is running

  ```
  % qconf -au <userrest> sudomasters
  ```

- With , , the users for whom requests shall be executed

  ```
  % qconf -au <user1>,<user2> sudoers
  ```

To modify the access lists do the following

```
% qconf -mu sudoers
name    sudoers
type    ACL
fshare  0
oticket 0
entries user1,user2

% qconf -mu sudomasters
name    sudomaster
type    ACL
fshare  0
oticket 0
entries userrest,joe
```

It is also possible to use wildcards such as '*' and '?' in user entries to handle a whole pattern of user names.

To switch off sudo requests completely, delete the sudoers and sudomasters access list.

```
% qconf -dul sudomasters
% qconf -dul sudoers
```

3.3 Adjusting Logging

The default logging level is set to INFO messages and the handler used is the FileHandler for logging to /tmp/UGERestService%u.%g.log where %u and %g are unique numbers. To switch the ConsoleHandler on, uncomment the handler's line containing both handlers. For additional logging the logging level can be changed in general, or for special classes as you can see in the commented lines below.

```
% vi $SGE_ROOT/ugerest/conf/logging.properties

# Specify the handlers to create in the root logger
# (all loggers are children of the root logger)
# The following creates two handlers

# Per default we log to the console
# handlers = java.util.logging.ConsoleHandler java.util.logging.FileHandler

# Use FileHandler
handlers = java.util.logging.FileHandler

# ------------------------------------------------------------------------
#  Definition of log levels
# ------------------------------------------------------------------------
# Set the default logging level for the root logger
.level = INFO
#.level = FINE
#com.sun.grid.jgdi.JGDI.level = FINE
#com.sun.grid.jgdi.rmi.level = FINE
#com.sun.grid.jgdi.configuration.xml.XMLUtil.level = FINE
#com.sun.grid.jgdi.configuration.ClusterQueueTestCase.level = FINE

# ------------------------------------------------------------------------
#  Settings for ConsoleHandler
# ------------------------------------------------------------------------
# Set the default logging level for new ConsoleHandler instances
java.util.logging.ConsoleHandler.level = INFO

# Set the default formatter for new ConsoleHandler instances
java.util.logging.ConsoleHandler.formatter = com.sun.grid.jgdi.util.SGEFormatter

# ------------------------------------------------------------------------
#  Settings for FileHandler
# ------------------------------------------------------------------------
# Set the default logging level for new FileHandler instances
java.util.logging.FileHandler.level = ALL
java.util.logging.FileHandler.pattern=/tmp/UGERestService%u.%g.log
java.util.logging.FileHandler.formatter=com.sun.grid.jgdi.util.SGEFormatter
```

3.4 Unpacking and Installing Altair Grid Engine REST Service

To install the Altair Grid Engine REST Service you need a working Altair Grid Engine installation. The steps to set up Altair Grid Engine are outlined in detail in the Altair Grid Engine Installation Guide. In order to run the Altair Grid Engine REST Service successfully the master daemon must be running. This can be verified by issuing the following command:

```
% setenv $SGE_ROOT <installation dir>
```

```
% {source|.} $SGE_ROOT/{default|$SGE_CELL}/common/settings.(c)sh
% qstat -f
queuename                      qtype resv/used/tot. np_load  arch          states
---------------------------------------------------------------------------------
all.q@aleph                    BIP   0/0/8          0.18     darwin-x64
---------------------------------------------------------------------------------
all.q@amiata                   BIP   0/0/1          -NA-     -NA-          au
---------------------------------------------------------------------------------
all.q@vesuv                    BIP   0/0/1          -NA-     -NA-          au
```

UGERest can be automatically installed via the install_ugerest script. As the Altair Grid Engine admin user do the following:

```
% cd $SGE_ROOT
% pwd
% tar xvpzf ge-{version}-ugerest.tar.gz
```

This creates a subdirectory ugerest under $SGE_ROOT. Become super user and execute the following command:

```
# . $SGE_ROOT/$SGE_CELL/common/settings.sh
# cd $SGE_ROOT
# ./install_ugerest
```

Answer the questions as requested. This creates a startup/stop/status script under $SGE_ROOT/$SGE_CELL/common/ugerest with the parameters set according to your installation. In addition the file $SGE_ROOT/ugerest/conf/ugerest.properties is created according to your installation settings.

To start up the Altair Grid Engine REST Service do:

```
# $SGE_ROOT/$SGE_CELL/common/ugerest start
```

To check for the service

```
# $SGE_ROOT/$SGE_CELL/common/ugerest status
```

To stop the service

```
# $SGE_ROOT/$SGE_CELL/common/ugerest stop
```

You can follow the behavior in the logging messages this way: ({n} and {m} usually default to 0)

```
% tail -f /tmp/UGERestService{n}.{m}.log
```

3.5 Jaas Module Configuration for Authentication

To enable PAM-based authentication the following configuration has to be set. This is the default authentication mode. In order to use a different authentication module such as LdapAuthentication or something else, comment out the original UGERestlet authenticator and replace it with another one. See the LDAP example below.

```
/*
** UGERestlet {
**      com.sun.security.auth.module.LdapLoginModule REQUIRED
**          userProvider="ldap://localhost:1389/ou=people,dc=example,dc=com"
**          userFilter="(&(uid={USERNAME})(objectClass=inetOrgPerson))"
**          debug=true
**          authzIdentity=controlRole
**          useSSL=false;
** };
*/

UGERestlet {
    com.sun.grid.security.login.UnixLoginModule requisite
        sge_root="${com.sun.grid.jgdi.sgeRoot}"
        debug=true
        auth_method="pam"
        pam_service="login";
};
```

It is also possible to write your own login module for the authentication process and plug this in as outlined above. What is required for the internal working of the REST service is the UNIX username and groupname and the uid and gid. These are used in so-called **sudo requests**. The user must be a valid user; this is verified against the masters user database.

3.6 Verification of the Installation

To verify the general working of the Altair Grid Engine REST Service you can test the following links in your browser and go from there using the different defined routes.

`http://{hostname}:{port}/test.html`

`https://{hostname}:{sport}/test.html`

`{hostname}` has to be replaced by the hostname of the machine that is running the Altair Grid Engine REST Service, `{port}` defaults to **8182**, and `{sport}` is usually `{port} + 1` i.e. by default **8183** .

Surfing the test page you can see links leading you to the different object routes and clicking them you'll see the raw JSON responses for a GET request.

When connecting the first time, the browser may request a confirmation that you trust the page in SSL mode, and your UNIX user credentials for the connection to the Altair Grid

Engine REST Service. This is needed for authenticating your requests. For further testing of other HTTP methods e.g. *curl* or *Postman, a REST Client* can be used. For *curl* you can find corresponding example requests below.

3.7 Prepare Key Material for https Mode

Adjust the following path settings (esp. port 5570) to the settings used by your installation. The CSP mode directories are used here to securely store the UGERest.jks keystore; this is not required. You can store the keystore in any secure place you choose. Do not forget to change the ugerest/conf/ugerest.properties file's keystore path and the key(store) password settings before starting up Altair Grid Engine REST Service. These settings are usually required for the Altair Grid Engine REST Service https mode.

ugerest/conf/ugerest.properties

```
...
default.keystore.type = JKS
default.keystore.path = /var/sgeCA/port5570/default/private/UGERest.jks
default.keystore.password = changeme
default.key.password = changeme
```

Creating Keys and a Self-signed Certificate

```
keytool -genkeypair\
 -v\
 -alias UGERest\
 -dname "CN=**UGERest**,OU=Engineering,O=Univa,C=DE"\
 -keypass password\
 -keystore /var/sgeCA/port5570/default/private/UGERest.jks\
 -storepass password\
 -keyalg "RSA"\
 -sigalg "MD5withRSA"\
 -keysize 2048\
 -validity 3650

The output should be:

Generating 2,048 bit RSA key pair and self-signed certificate
            (MD5withRSA) with a validity of 3,650 days
    for: CN=**UGERest**, OU=Engineering, O=Univa, C=DE
    [Storing /var/sgeCA/port5570/default/private/UGERest.jks]
```

Option	Explanation
-genkey	generate a pair of keys and a self-signed certificate
-v	display the output message
-alias	a unique name for the keys (does not need to be the name of the machine)

Option	Explanation
-dname	details of the machine where the keys and certificate will be used
-keypass	the password for the key pair identified by the '-alias' option
-keystore	the name of the operating system file where the keys/certificate will be saved
-storepass	the password for the keystore file
-keyalg	the encryption algorithm to use; "DSA" or "RSA"
-sigalg	the signature algorithm to use; "Sha1withDSA" and "MD5withRSA" are two
-keysize	the size of the key (larger = more secure; e.g. 512, 1024 or 2048)
-validity	the number of days before the certificate expires (3650 = approx. 10 years)

Make Self-signed Certificate Available

```
keytool -export\
 -v\
 -alias UGERest\
 -file UGERest.cer\
 -keystore UGERest.jks\
 -storepass password
```

As a privileged user use the following (depending on your JDK installation, please adjust example below)

Make Self-signed Certificate Available

```
keytool -import\
 -alias UGERest\
 -file UGERest.cer\
 -keystore "/Library/Java/.../security/cacerts"\
 -storepass "changeit"
```

Table 4: keytool switches and functionality

Option	Explanation
-import -export -alias -file -keystore -storepass	import the certificate export the certificate the name of the public key/certificate to export the name of a file where the certificate is to be saved the name of the keystore file containing the certificate the password for the keystore file

Further details can be found in the Restlet Technical Documentation

To look at the entries in the Java certificate database, execute the following command

```
% keytool -list -keystore /Library/Java/JavaVirtualMachines/jdk1.7.0_60.jdk/Contents/Home/jre/lib
Enter keystore password: changeit
```

For deleting an entry with the alias aleph for example execute the following command

```
become root
# keytool -delete -alias aleph -keystore /Library/Java/JavaVirtualMachines/jdk1.7.0_60.jdk/Conten
Enter keystore password: changeit
```

3.8 Installation of Altair Grid Engine REST Service Java Binding

In order to use the different language-specific bindings, the installation of additional packages might be necessary. The necessary jar file to make the Java Binding available is automatically contained in the Altair Grid Engine REST Service distribution package. No further action has to be taken. To make use of the corresponding Java Binding programming API, you have to include **$SGE_ROOT/ugerest/lib/ugerestsdkapi.jar** in the classpath of your application. Example applications and the corresponding command lines can be found in the Altair Grid Engine REST Service Java Binding section.

3.9 Installation of Altair Grid Engine REST Service Nodejs Binding

For the Node.js Binding the installation of the corresponding node.js packages and the installation of a node.js distribution can be achieved as outlined here. Many OS distributions allow the installation of pre-built node.js packages. If this is not possible follow the instructions at nodejs.org. For Mac OS X and Linux, installation packages are available. To install them, follow these steps:

- Mac OS X - MacPorts Package

  ```
  $ /opt/local/bin/port install nodejs nodejs-devel npm
  this will install node and npm (node package manager) for you
  $ /opt/local/bin/node --version
  ```

- Linux (like SUSE) - node.js Package

  ```
  # sudo zypper ar http://download.opensuse.org/repositories/devel:/languages:/nodejs/openSUS
  # sudo zypper in nodejs nodejs-devel    (contains npm too)
  # /usr/bin/node --version
  ```

- Solaris - Solaris x86 Packages

  ```
  $ pkgadd -d TRIBnode-0.10.36.pkg
  $ pkginfo | grep TRIB
  $ /opt/Node/bin/node --version
  ```

After installing the node for the nodejs binding, some special steps are necessary

- Install the urllib-sync npm package.

```
% npm install urllib-sync
```

- To install the Altair Grid Engine REST API Nodejs JavaScript Binding you have to unpack the package and install it with npm:

```
mkdir usdk
cd usdk
unzip $SGE_ROOT/ugerest/sdks/ugerest-jssdk.zip
cd ..
npm install usdk
```

To use ugerestsdk see the example here: Altair Grid Engine REST Service Nodejs Binding

3.10 Installation of Altair Grid Engine REST Service Meteor Binding

For Meteor installation follow the steps outlined at meteor.com

```
curl https://install.meteor.com/ | sh
```

To install the corresponding Altair Grid Engine REST API Meteor JavaScript Binding package follow these steps:

```
% mkdir ugerest-meteorsdk
% cd ugerest-meteorsdk
% unzip $SGE_ROOT/ugerest/sdks/ugerest-meteorsdk.zip
% export PACKAGE_DIRS=<whereever>/ugerest-meteorsdk   (required for newer meteor versions)
% cd <meteor-app>
% meteor add <whereever>/ugerest-meteorsdk            (old way to install from directory)
% meteor add ugerestsdk                               (required for newer meteor versions)
% meteor list | grep ugerest
```

To make use of the SDK in your code see Altair Grid Engine REST Service Meteor Binding.

3.11 Installation of Altair Grid Engine REST Service Python Binding

For Python installation use the following steps:

- Install python via MacPorts on Mac OS X

```
$ /opt/local/bin/port install python27
$ /opt/local/bin/python --version
```

or use the preinstalled version

```
$ /usr/bin/python --version
```

- On Linux (like Suse) install python

```
% sudo zypper in python
% python --version
```

- or on Solaris follow folloing steps

```
$ /opt/csw/bin/pkgutil -a python
$ /opt/csw/bin/pkgutil -i python27
$ /opt/csw/bin/pkgutil -i python27_dev
$ /opt/csw/bin/python --version
```

To install the corresponding package Altair Grid Engine REST API Python Binding package follow these steps:

```
% mkdir ugerest-pysdk
% cd ugerest-pysdk
% unzip $SGE_ROOT/ugerest/sdks/ugerest-pysdk.zip
% (sudo) python setup.py install
(On Mac OS e.g. ls /Library/Python/2.7/site-packages
 ...
 ugerestsdk/
 ugerestsdk-0.0.1-py2.7.egg-info
)
```

To make use of the sdk inside your code see: Altair Grid Engine REST Service Python Binding.

4 Resources, JSON Data Formats, and Usage with curl CLI

4.1 Overview

JSON (JavaScript Object Notation) is the data format which represents resources in the Altair Grid Engine REST Service. This format makes use of two main structures: collections of key/value pairs called objects and ordered lists of values called arrays. Objects are defined by using curly braces ({}), and arrays are defined by using square brackets ([]). A JSON object or array may contain several different types of values including numbers, booleans (true/false), strings, objects, arrays. For example, the ParallelEnvironment object might be defined as:

```
{
   "name": "bla",
   "slots": 5,
   "userList": ["arusers"],
   "xuserList": [],
   "startProcArgs": "/bin/startProc",
   "stopProcArgs": "/bin/stopProc",
   "allocationRule": "$pe_slots",
   "controlSlaves": false,
   "jobIsFirstTask": true,
   "urgencySlots": "min",
   "accountingSummary": false,
   "daemonForksSlaves": false,
   "masterForksSlaves": false
}
```

More general information on JSON is available under json.org.

Input data for an HTTP request with method *POST*, *PUT*, or *DELETE* must be in *JSON format*. The *Content-Type* header must be set to *application/json*. Output is in JSON format and always consists of an object with zero or more key/value pairs representing the corresponding Altair Grid Engine object or a list thereof; if this fails the output is a special JSON object representing an error of the form

```
{
    "errorMessage": "some error text",
    "errorCode" : 299
}
```

To read and display objects the general format of the request treats the curl command line as a GET Request URL scheme. To make a GET request for a special object the following schemes are used:

Curl GET command line structure

```
curl [-X GET] -H "Content-Type: application/json" {url} \
     -u {unix user}

with {url} e.g. http[s]//{hostname}:{port}/checkpoints
see the specific url reference for all objects below

This corresponds to `qconf -ckptl`


curl [-X GET] -H "Content-Type: application/json" {url} \
     -u {unix user}

with {url} e.g. http[s]//{hostname}:{port}/checkpoints/{id}
See the specific url reference for all objects below

This corresponds to `qconf -sckpt {id}`



curl [-X GET] -H "Content-Type: application/json" {url} \
     -u {unix user}

with {url} e.g. http[s]//{hostname}:{port}/checkpoints/{id}/{n}
See the specific url reference for all objects below

This corresponds to a range of `qconf -sckpt {id}` for {n}
checkpoint objects starting with {id}

usually {id} is a pattern matching the primary key of the object
that is the name or id of the object
```

The returned results are JSON-formatted strings containing in most cases the same information as the corresponding qconf command output. JSON String for PUT/POST/DELETE The following command examples illustrate how to POST/PUT/DELETE an object to the Altair Grid Engine system. They outline the JSON strings that have to be used to add/modify/delete an Altair Grid Engine object.

Curl POST command line structure

```
curl -X POST -H "Content-Type: application/json" {url} \
     -d {json data} -u {unix user}

with {url} e.g. http[s]//{hostname}:{port}/checkpoints
see the specific url reference for all objects below
```

Here is an overview of all GET operations, with the involved objects and the corresponding URLs. For POST, PUT and DELETE commands the *Get {object} List* line applies. Ranges are expressed as first element followed by number of following elements to return. The first element can be a string or a number. For example for hosts you can choose a special hostname as first element or 1 to start at the first host list entry. The range lists are ordered alphabetically and therefore the search starts with the first element alphabetically after the ID string, e.g. /hostsummary/s/2 starts with a hostname that starts with s or later.

Action	URL: http[s]://{host}/{port}
Get Job List	"/jobs"
Get Job {id}	"/jobs/{jobId}"
Get Job Range {jobId} {count}	"/jobs/{jobId}/{count}"
Get Jobs for {user} in {state}	"/jobs_for_user/{user}/{state}"
Get Host summary for {hostId}	"/hostsummary/{hostId}"
Get Host summary Range	"/hostsummary/{hostName}/{count}"
Get CQ summary for {cqName}	"/clusterqueuesummary/{cqName}"
Get CQ summary Range	"/clusterqueuesummary/{clusterQueueName}/{count}"
Get QI summary for {qiName}	"/queueinstances/{qiName}"
Get QI List	"/queueinstances"
Get QI Range	"/queueinstances/{queueInstanceName}/{count}"
Get JobClass {JobClassId}	"/jobclasses/{JobClassId}"
Get JobClass List	"/jobclasses"
Get JobClass Range	"/jobclasses/{name}/{count}"
Get Hostgroup {hgId}	"/hostgroups/{hgId}"
Get Hostgroup List	"/hostgroups"
Get Hostgroup Range	"/hostgroups/{name}/{count}"
Get SubmitHost {id}	"/submithosts/{id}"
Get SubmitHost List	"/submithosts"
Get SubmitHost Range	"/submithosts/{name}/{count}"
Get ExecHost {hostId}	"/exechosts/{hostId}"
Get ExecHost List	"/exechosts"
Get ExecHost Range	"/exechosts/{name}/{count}"
Get Manager {id}	"/managers/{id}"
Get Manager List	"/managers"
Get Manager Range	"/managers/{name}/{count}"
Get AdvanceReservation {arId}	"/advancereservations/{arId}"
Get AdavanceReservation List	"/advancereservations"
Get AdvanceReservation Range	"/advancereservations/{name}/{count}"
Get SchedConf {id}	"/schedconfs/{id}"
Get SchedConf List	"/schedconfs"
Get SchedConf Range	"/schedconfs/{name}/{count}"
Get Configuration {id}	"/configurations/{ConfigurationId}"
Get Configuration List	"/configurations"
Get Configuration Range	"/configurations/{name}/{count}"
Get Project {id}	"/projects/{id}"
Get Project List	"/projects"
Get Project Range	"/projects/{name}/{count}"
Get Sharetree {id}	"/sharetrees/{id}"
Get Sharetree List	"/sharetrees"

Action	URL: http[s]://{host}/{port}
Get Sharetree Range	"/sharetrees/{name}/{count}"
Get AdminHost {hostId}	"/adminhosts/{hostId}"
Get AdminHost List	"/adminhosts"
Get AdminHost Range	"/adminhosts/{name}/{count}"
Get Session {id}	"/sessions/{id}"
Get Sessions	"/sessions"
Get Session Range	"/sessions/{name}/{count}"
Get Checkpoint {id}	"/checkpoints/{id}"
Get Checkpoint List	"/checkpoints"
Get Checkpoint Range	"/checkpoints/{name}/{count}"
Get ParallelEnvironment {id}	"/parallelenvironments/{id}"
Get ParallelEnvironment List	"/parallelenvironments"
Get ParallelEnvironment Range	"/parallelenvironments/{name}/{count}"
Get ComplexEntry {id}	"/complexentries/{id}"
Get ComplexEntry List	"/complexentries"
Get ComplexEntry Range	"/complexentries/{name}/{count}"
Get ClusterQueue {id}	"/clusterqueues/{id}"
Get ClusterQueue List	"/clusterqueues"
Get ClusterQueue Range	"/clusterqueues/{name}/{count}"
Get UserSet {id}	"/usersets/{id}"
Get UserSet List	"/usersets"
Get UserSet Range	"/usersets/{name}/{count}"
Get Calendar {id}	"/calendars/{id}"
Get Calendar List	"/calendars"
Get Calendar Range	"/calendars/{name}/{count}"
Get Operator {id}	"/operators/{id}"
Get Operator List	"/operators"
Get Operator Range	"/operators/{name}/{count}"
Get ResourceQuotaSet {id}	"/resourcequotasets/{id}"
Get ResourceQuotaSet List	"/resourcequotasets"
Get ResourceQuotaSet Range	"/resourcequotasets/{name}/{count}"
Get User {id}	"/users/{id}"
Get User List	"/users"
Get User Range	"/users/{name}/{count}"

The following sections contain definitions of the corresponding json formats that are output and serve as input for *POST*, *PUT*, and *DELETE* requests, as outlined in the **curl** examples above. Every section always starts with the corresponding command line option of qconf, followed by the method options for **curl** and most importantly the json definition for the corresponding object.

4.2 Cluster Queue

```
-aq
-Aq
-mq
-Mq
-dq
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/clusterqueues
```


```
{"qname":"test.q",
"hostlist":["@allhosts"],
"seq_no":[{"hostgroup":"@/","value":0},
          {"hostgroup":"amiata","value":2}],
"load_thresholds":[{"hostgroup":"@/","value":[{"name":"np_load_avg","value":1.75}]}],
"suspend_thresholds":[{"hostgroup":"@/","value":[{"name":"np_load_avg", "value":1.25}]},
                      {"hostgroup":"amiata", "value":[{"name":"np_load_avg","value":1.35}]}],
"nsuspend":[{"hostgroup":"@/","value":1},
            {"hostgroup":"amiata", "value":5}],
"suspend_interval":[{"hostgroup":"@/","value":"00:05:00"},
                    {"hostgroup":"amiata","value":"00:01:00"}],
"priority":[{"hostgroup":"@/","value":"0"},
            {"hostgroup":"amiata","value":"10"}],
"min_cpu_interval":[{"hostgroup":"@/","value":"00:05:00"},
                    {"hostgroup":"amiata","value":"00:01:00"}],
"qtype":[{"hostgroup":"@/","value":"BATCH INTERACTIVE"}],
"ckpt_list":[{"hostgroup":"@/","value":[]}],
"pe_list":[{"hostgroup":"@/", "value":["make"]}],
"jc_list":[{"hostgroup":"@/", "value":["NO_JC","ANY_JC"]}],
"rerun":[{"hostgroup":"@/", "value":FALSE}],
"slots":[{"hostgroup":"@/", "value":1}],
"tmpdir":[{"hostgroup":"@/", "value":"/tmp"}],
"shell":[{"hostgroup":"@/", "value":"/bin/sh"}],
"prolog":[{"hostgroup":"@/", "value":"NONE"}],
"epilog":[{"hostgroup":"@/", "value":"NONE"}],
"shell_start_mode":[{"hostgroup":"@/", "value":"unix_behavior"}],
"starter_method":[{"hostgroup":"@/", "value":"NONE"}],
"suspend_method":[{"hostgroup":"@/", "value":"NONE"}],
"resume_method":[{"hostgroup":"@/", "value":"NONE"}],
"terminate_method":[{"hostgroup":"@/", "value":"NONE"}],
"notify":[{"hostgroup":"@/", "value":"00:00:60"}],
"owner_list":[{"hostgroup":"@/", "value":[]}],
"user_lists":[{"hostgroup":"@/", "value":[]}],
"xuser_lists":[{"hostgroup":"@/", "value":[]}],
"subordinate_list":[{"hostgroup":"@/", "value":[]}],
"complex_values":[{"hostgroup":"@/", "value":[]}],
"projects":[{"hostgroup":"@/", "value":[]}],
"xprojects":[{"hostgroup":"@/", "value":[]}],
"calendar":[{"hostgroup":"@/", "value":"NONE"}],
```

```
"initial_state":[{"hostgroup":"@/", "value":"default"}],
"s_rt":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_rt":[{"hostgroup":"@/", "value":"INFINITY"}],
"d_rt":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_cpu":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_cpu":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_fsize":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_fsize":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_data":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_data":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_stack":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_stack":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_core":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_core":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_rss":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_rss":[{"hostgroup":"@/", "value":"INFINITY"}],
"s_vmem":[{"hostgroup":"@/", "value":"INFINITY"}],
"h_vmem":[{"hostgroup":"@/", "value":"INFINITY"}]}
```

for delete

```
{"name":"all.q"}
```

4.3 Calendar

```
-acal
-Acal
-mcal
-Mcal
-dcal
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/calendars
```

```
{"name":"weekend_cal",
 "year":"NONE",
 "week":"sat-sun=suspended"}
```

for delete `{"name":"weekend_cal"}`

4.4 ComplexEntry

```
-mc
-Mc
-sc
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/complexentries
```

```
{"name":"arch",
 "shortcut":"a",
 "type":9,
 "relop":1,
 "requestable":2,
 "consumable":0,
 "default":"NONE",
 "urgency":"0"}
```

for delete

```
{"name":"arch"}
```

4.5 Checkpoint

```
-ackpt
-Ackpt
-dckpt
-mckpt
-Mckpt
-sckpt
-sckptl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/checkpoints
```

```
{"name":"abc",
"interface":"hibernator",
"clean_command":"dcmd",
"ckpt_command":"acmd",
"migr_command":"bcmd",
"rest_command":"ccmd",
"ckpt_dir":"/tmp",
"signal":"none",
"when":"xs"}
```

for delete

```
{"name":"abc"}
```

4.6 Configuration

```
-aconf host_list
-Aconf file_list
-dconf host_list
-mconf host_list|global
-Mconf file_list
```

```
-sconf host_list|global
-sconfl

-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/configurations
```


```
{"name":"global",
"entries":[{"name":"execd_spool_dir","value":"/Users/user/univa/clusters/UGE/default/spool"},
{"name":"mailer","value":"/usr/bin/mail"},
{"name":"xterm","value":"/usr/X11R6/bin/xterm"},
{"name":"load_sensor","value":"none"},
{"name":"prolog","value":"none"},
{"name":"epilog","value":"none"},
{"name":"shell_start_mode","value":"unix_behavior"},
{"name":"login_shells","value":"sh,bash,ksh,csh,tcsh"},
{"name":"min_uid","value":"0"},
{"name":"min_gid","value":"0"},
{"name":"user_lists","value":"none"},
{"name":"xuser_lists","value":"none"},
{"name":"projects","value":"none"},
{"name":"xprojects","value":"none"},
{"name":"default_jc","value":"none"},
{"name":"enforce_jc","value":"false"},
{"name":"enforce_project","value":"false"},
{"name":"enforce_user","value":"auto"},
{"name":"load_report_time","value":"00:00:40"},
{"name":"max_unheard","value":"00:05:00"},
{"name":"reschedule_unknown","value":"00:00:00"},
{"name":"loglevel","value":"log_warning"},
{"name":"administrator_mail","value":"none"},
{"name":"set_token_cmd","value":"none"},
{"name":"pag_cmd","value":"none"},
{"name":"token_extend_time","value":"none"},
{"name":"shepherd_cmd","value":"none"},
{"name":"qmaster_params","value":"none"},
{"name":"execd_params","value":"KEEP_ACTIVE=ERROR"},
{"name":"reporting_params","value":"accounting=true reporting=false \
                                   flush_time=00:00:15 joblog=false \
                                   sharelog=00:00:00"},
{"name":"finished_jobs","value":"100"},
{"name":"gid_range","value":"20000-20100"},
{"name":"qlogin_command","value":"builtin"},
{"name":"qlogin_daemon","value":"builtin"},
{"name":"rlogin_command","value":"builtin"},
{"name":"rlogin_daemon","value":"builtin"},
{"name":"rsh_command","value":"builtin"},
{"name":"rsh_daemon","value":"builtin"},
{"name":"max_aj_instances","value":"2000"},
{"name":"max_aj_tasks","value":"75000"},
```

```
{"name":"max_u_jobs","value":"0"},
{"name":"max_jobs","value":"0"},
{"name":"max_advance_reservations","value":"0"},
{"name":"auto_user_oticket","value":"0"},
{"name":"auto_user_fshare","value":"0"},
{"name":"auto_user_default_project","value":"none"},
{"name":"auto_user_delete_time","value":"86400"},
{"name":"delegated_file_staging","value":"false"},
{"name":"reprioritize","value":"0"},
{"name":"jsv_url","value":"none"},
{"name":"jsv_allowed_mod","value":"ac,h,i,e,o,j,M,N,p,w"},
{"name":"cgroups_params","value":"cgroup_path=none cpuset=false mount=false \
                                  freezer=false freeze_pe_tasks=false \
                                  killing=false forced_numa=false \
                                  h_vmem_limit=false m_mem_free_hard=false \
                                  m_mem_free_soft=false min_memory_limit=0"},
{"name":"libjvm_path","value":"/Library/Java/Home/../Libraries/libserver.dylib"},
{"name":"additional_jvm_args","value":"-Xmx256m"}]}
```

4.7 Hostgroup

```
-ahgrp group
-Ahgrp file
-dhgrp group
-mhgrp group
-Mhgrp file
-shgrp group
-shgrp_tree group
-shgrp_resolved group
-shgrpl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/hostgroups

{"name":"@allhosts",
 "hostlist":["aleph"]}
```

for delete `{"name":"@allhosts"}`

4.8 AdminHost

```
-ah host_list
-dh host_list
-sh
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/adminhosts

{"name":"aleph"}
```

4.9 SubmitHost

```
-as host_list
-ds host_list
-ss
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/submithosts
```

```
{"name":"aleph"}
```

4.10 ExecutionHost

```
-ae host
-me host
-se host
-sel
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/exechosts
```

```
{"hostname":"amiata",
"load_scaling":[],
"complex_values":[],
"user_lists":["arusers"],
"xuser_lists":[],
"projects":["prj1"],
"xprojects":[],
"usage_scaling":[{"load":"cpu","scaleFactor":1.2}],
"report_variables":["arch"]}
```

for delete

```
{"hostname":"amiata"}
```

4.11 Manager

```
-am user_list
-dm user_list
-sm
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/managers
```

```
{"name":"user"}
```

4.12 Operator

```
-ao user_list
-do user_list
-so
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/operators
```

```
{"name":"user"}
```

4.13 ParallelEnvironment

```
-ap pe-name
-Ap file
-dp pe-name
-mp pe-name
-Mp file
-sp pe-name
-spl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/parallelenvironments
```

```
{"name":"bla",
"slots": 5,
"userList":["arusers"],
"xuserList":[],
"startProcArgs":"/bin/startProc",
"stopProcArgs":"/bin/stopProc",
"allocationRule":"$pe_slots",
"controlSlaves":false,
"jobIsFirstTask":true,
"urgencySlots":"min",
"accountingSummary":false,
"daemonForksSlaves":false,
"masterForksSlaves":false}
```

for delete

```
{"name":"bla"}
```

4.14 ResourceQuotaSet

```
-arqs rqs_list
-Arqs file
-drqs rqs_list
-mrqs rqs_list
```

```
-Mrqs file
-srqs [rqs_list]
-srqsl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/resourcequotasets
```

```
{"name":"rqs1",
"description":"\"RQS number 1\"",
"enabled":false,
"limits":[ {"limits":[{"limit":[{"name":"slots","value":"2"}],
                        "users":["dag"],
                        "xqueues":[],
                        "xhosts":[],
                        "jclasses":[],
                        "projects":[],
                        "xprojects":[],
                        "pes":[],
                        "xusers":[],
                        "hosts":[],
                        "xpes":[],
                        "xjclasses":[],
                        "queues":["all.q"]
                       },
                       {"limit":[{"name":"slots","value":"1","type":1}],
                        "users":["dag"],
                        "xqueues":[],
                        "xhosts":[],
                        "jclasses":[],
                        "projects":[],
                        "xprojects":[],
                        "pes":[],
                        "xusers":[],
                        "hosts":[],
                        "xpes":[],
                        "xjclasses":[],
                        "queues":["testQueue"]
                        }],
              "enabled":true,
              "description":"NONE",
              "name":"test_max_per_queue"} ] }
```

for delete

```
{"name":"test_max_per_queue"}
```

4.15 Project

```
-aprj
```

```
-Aprj file
-dprj project_list
-mprj project
-Mprj file
-sprj project
-sprjl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/projects
```

```
{"name":"prj1",
"oticket":10,
"fshare":5,
"acls":["arusers"],
"xacls":[]}
```

for delete

```
{"name":"prj1"}
```

4.16 User

```
-auser
-Auser file
-duser user_list
-muser
-Muser file
-suser user_list
-suserl
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/users
```

```
{"name":"user",
"oticket":0,
"fshare":0,
"deleteTime":1405076192}
```

for delete

```
{"name":"user"}
```

4.17 UserSet

```
-au user_list listname_list
-Au file
-du user_list listname_list
-dul listname_list
```

```
-mu listname_list
-Mu file
-su listname_list
-sul

-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/usersets

{"name":"arusers",
"type":1,
"oticket":0
"fshare":0,
"entries":["user"]}
```

for delete

```
{"name":"arusers"}
```

4.18 ShareTree

```
-astree
-Astree file
-dstree
-Mstree file
-mstree
-sstree

-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/sharetree

{"id":0,
 "name":"Root",
 "type":0 ",
 shares":1,
 "childnodes"[{"id":1,
               "name":"prj1",
               "type":1,
               "shares":10,
               "childnodes":[]},
              {"id":2,
               "name":"prj2",
               "type":1,
               "shares":10,
               "childnodes":[]},
              {"id":3,
               "name":"prj3",
               "type":1,
```

```
                "shares":10,
                "childnodes":[],
               }]
}
```

4.19 SchedConf

```
-msconf
-Msconf file
-ssconf
-X POST -H "Content-Type: application/json"
http[s]://{host}:{port}/schedconf
```

4.20 Job

Job Support is based on DRMAA functionality and the options that are currently supported are using the DRMAA JobTemplate attributes:

```
{
    "remoteCommand":"/Users/user/bin/sleeper.sh",
    "args":["120"],
    "blockEmail": false,
    "email":["myname@domain.org"],
    "outputPath":":/dev/null",
    "errorPath":":/dev/null",
    "jobCategory":"fastJob",
    "jobEnvironment":[],
    "jobName":"testjob",
    "jobSubmissionState":0,
    "joinFiles":false,
    "startTime":"2014/11/12 21:39:22 +01:00",
    "nativeSpecification":"-l arch=sol-amd64"
}
```

or a minimal one:

```
{
    "remoteCommand":"/Users/user/bin/sleeper.sh",
    "args":["120"],
    "email":["user@example.com"],
    "jobName":"Snooze",
    "joinFiles":"true"
}
```

For request execution in a portal environment we introduced so-called **sudo requests** that are controlled via two special access lists: **sudoers/sudomasters** to allow running the WS/Restlet component as a user and perform requests on behalf of a different user. How these lists are set up is explained in Sudo Configuration.

4.21 Special Targets

For jobs and queues there exist in addition some special targets. The following routes map functionality of the *qstat and qhost* command variants.

Job Routes

Action	URL: http[s]://{host}/{port}
Get Job List	"/jobs"
Get Job {id}	"/jobs/{jobId}"
Get Job Range {jobId} {count}	"/jobs/{jobId}/{count}"
Get Jobs for {user} in {state}	"/jobs_for_user/{user}/{state}"

The first three routes are built following the general rules; the fourth one gives you a range of jobs that fulfill a user pattern {user} and {state}. The user pattern {user} can be a user name or a user pattern containing names consisting of or containing the wildcard * matching zero or more characters. The state pattern {state} describes the job's state in the same way as the *qstat -s* options.

As the qstat usage shows:

```
[-s {p|r|s|z|hu|ho|hs|hd|hj|ha|h|a}] show pending, running, suspended, zombie jobs,
                                     jobs with a user/operator/system/array-dependency hold,
                                     jobs with a start time in future, or any combination.
                                     h is an abbreviation for huhohshdhjha.
                                     a is an abbreviation for prsh.
```

Host Routes

Action	URL: http[s]://{host}/{port}
Get Host summary for {hostId}	"/hostsummary/{hostId}"
Get Host summary Range	"/hostsummary/{hostName}/{count}"
Get ExecHost {hostId}	"/exechosts/{hostId}"
Get ExecHost List	"/exechosts"
Get ExecHost Range	"/exechosts/{name}/{count}"

The first two routes correspond to the *qhost* command. The last three are built following the general rules specific for the execution host object.

Queue Routes

Action	URL: http[s]://{host}/{port}
Get CQ summary for {cqName}	"/clusterqueuesummary/{cqName}"
Get CQ summary Range	"/clusterqueuesummary/{clusterQueueName}/{count}"
Get QI summary for {qiName}	"/queueinstances/{qiName}"
Get QI List	"/queueinstances"

Action	URL: http[s]://{host}/{port}
Get QI Range	"/queueinstances/{queueInstanceName}/{count}"

The first two correspond to the *qstat -f -ext* command and the last three show the corresponding queue instances as a list of the attributes of a specific queue instance and a range thereof.

5 Altair Grid Engine REST Service Java Binding

Here you can find an example of how to use the Altair Grid Engine REST API Java Binding in order to write a simple client that makes use of the API functionality. The binding is contained in **ugerest/lib/ugerestsdkapi.jar**. The general mechanism is rather simple. There is a main class **UGEWSClientApi** that delivers access to the different model object classes such as **AdminHost**, **ClusterQueue**, etc.

Usually you have these functionalities for every object and some additional methods that deviate a little bit from the general pattern:

- `get<Object>List()`
- `get<Object>Range(pattern, count)`
- `get<Object>ById(id)`
- `add<Object>(data)`
- `modify<Object>(data)`
- `delete<Object>(id)`

In this simple test example below, the username and password are read for simplicity from a file */var/tmp/challenge.txt* in the test method **setUpClass()** using this code line:

```
File pwfile = new File("/var/tmp/challenge.txt");
...
```

The content of */var/tmp/challenge.txt* is two lines, one containing the username and one the password, like this:

```
username
password
```

UGEWSClientApiTest.java

```
1  package com.univa.ugerest.util;
2
3  import com.univa.client.ApiException;
4  import java.util.HashMap;
5  import com.univa.client.model.*;
6  ...
7
8  /**
9   * A UGEWSClient API Test
10  */
11 public class UGEWSClientApiTest {
12
13     private static String basePath = "http://localhost:8182";
14     private static String user;
15     private static String pw;
```

```
16      private static boolean withAuthentication = false;
17      private int sleeptime = 500;
18
19      @BeforeClass
20      public static void setUpClass() {
21          try {
22              File pwfile = new File("/var/tmp/challenge.txt");
23              BufferedReader br = new BufferedReader(new FileReader(pwfile));
24              user = br.readLine();
25              pw = br.readLine();
26              withAuthentication = true;
27          } catch (FileNotFoundException ex) {
28              Logger.getLogger(UGEWSClientApiTest.class.getName()).log(Level.SEVERE, null, ex);
29          } catch (IOException ex) {
30              Logger.getLogger(UGEWSClientApiTest.class.getName()).log(Level.SEVERE, null, ex);
31          }
32      }
33
34
35      @Test
36      public void testJob() {
37          String jobId = "0";
38          System.out.println("###################### Job ######################");
39          UGEWSClientApi apitester = new UGEWSClientApi(basePath, user, pw);
40          System.out.println("---------- Add Job ------------------");
41          try {
42              JobTemplate jt = new JobTemplate();
43              jt.setRemoteCommand("/Users/user/bin/sleeper.sh");
44              List<String> args = new ArrayList<String>();
45              args.add("120");
46              jt.setArgs(args);
47              Set<String> emailSet = new HashSet<String>();
48              emailSet.add("user@example.com");
49              jt.setEmail(emailSet);
50              jt.setJobName("MyJob");
51              jt.setJoinFiles(true);
52              // no.setName("amiata"); etc.
53              jobId = apitester.addJob(jt);
54              System.out.println(jobId);
55          } catch (ApiException e) {
56              System.out.println(e.getMessage());
57          }
58          System.out.println("---------- GetById Job ------------------");
59          try {
60              try {
61                  System.out.println("Wait " + sleeptime + "ms");
62                  Thread.sleep(sleeptime);
63              } catch (InterruptedException e) {
64                  // ignore
```

```
65              }
66              String json = apitester.getJobById(jobId);
67              System.out.println(json);
68          } catch (ApiException e) {
69              System.out.println(e.getMessage());
70          }
71          System.out.println("----------- GetList Job -------------------");
72          try {
73              List<String> result = apitester.getJobList();
74              System.out.println(result);
75          } catch (ApiException e) {
76              System.out.println(e.getMessage());
77          }
78          System.out.println("----------- GetRange Job -------------------");
79          try {
80              String result = apitester.getJobRange(1, 25);
81              System.out.println(result);
82          } catch (ApiException e) {
83              System.out.println(e.getMessage());
84          }
85          System.out.println("----------- Delete Job -------------------");
86          try {
87              boolean res = apitester.deleteJob(jobId);
88              if (res) {
89                  System.out.println("delete success");
90              } else {
91                  System.out.println("delete failed");
92              }
93          } catch (ApiException e) {
94              System.out.println(e.getMessage());
95          }
96      }
```

6 Altair Grid Engine REST Service Nodejs Binding

The following program tests the different object methods for the Parallel Environment object. For REST method execution BASIC authentication is needed for any request in authenticated mode.

In the readUserCredentials function the username and password are read from a file */var/tmp/challenge.txt* with the following content:

```
/var/tmp/challenge.txt

username
password
```

Then the UGEWSClientApiTest instance is created in main and the credentials are read in

and stored. In the testParallelEnvironment a UGEWSClientApi instance is created that exposes all the object manipulation functions to the user. The object helper classes are also contained in the ugerestsdk for easier handling of the diverse json strings via getter and setter methods.

To run the program, just save the following lines in a file called UGEWSClientApiTest.js, create the /var/tmp/challenge.txt file containing username and password, and run it after having followed the setup steps outlined above:

```
node UGEWSClientApiTest.js
```

Be sure that the Altair Grid Engine REST Service has been started before trying this example.

UGEWSClientApiTest.js

```
var uc = require("ugerestsdk");

/**
 * A tree displaying a set of email folders.
 */
function UGEWSClientApiTest() {

    this.basePath = "http://localhost:8182";
    this.user = null;
    this.pw = null;
    this.withAuthentication = false;

    this.getUser = function() {
        return this.user;
    },

    this.setUser = function(user) {
        this.user = user;
    },

    this.setPassword = function(pw) {
        this.pw = pw;
    },


    this.readUserCredentials = function() {
        var fs = require('fs');
        var array = fs.readFileSync('/var/tmp/challenge.txt').toString().split("\n");
        this.setUser(array[0]);
        this.setPassword(array[1]);
    },

    this.testParallelEnvironment = function() {
        'use strict';
```

```
35          console.log("###################### ParallelEnvironment ######################");
36          var apiclient = new uc.UGEWSClientApi(this.basePath, this.user, this.pw);
37          console.log("----------- Add ParallelEnvironment -------------------");
38          try {
39              var name = "xyz";
40              var no = new uc.ParallelEnvironment(name);
41              // no.setName("amiata"); etc.
42              var ao = apiclient.addParallelEnvironment(no);
43              console.log(ao.toJson());
44              console.log("Wait a while for UGERest to get data handed out");
45              var sleep = function(time, callback) {
46                  var stop = new Date().getTime();
47                  while(new Date().getTime() < stop + time) {
48                      ;
49                  }
50                  callback();
51              };
52
53              sleep(1000, function() {});
54          } catch(api_exception) {
55              console.log(api_exception);
56          }
57          console.log("----------- Modify ParallelEnvironment -------------------");
58          try {
59              var name = "xyz";
60              var no = new uc.ParallelEnvironment(name);
61              // no.setName("amiata"); etc.
62              var ao = apiclient.modifyParallelEnvironment(no);
63              console.log(ao);
64          } catch(api_exception) {
65              console.log(api_exception);
66          }
67          console.log("----------- GetById ParallelEnvironment -------------------");
68          console.log("----------- GetById ParallelEnvironment -------------------");
69          try {
70              var name = "xyz";
71              var go = apiclient.getParallelEnvironmentById(name);
72              console.log(go.toJson());
73          } catch(api_exception) {
74              console.log(api_exception);
75          }
76          console.log("----------- GetList ParallelEnvironment -------------------");
77          try {
78              var result = apiclient.getParallelEnvironmentList();
79              console.log(result);
80          } catch(api_exception) {
81              console.log(api_exception);
82          }
83          console.log("----------- GetRange ParallelEnvironment -------------------");
```

```
84          try {
85              var result = apiclient.getParallelEnvironmentRange(1, 25);
86              console.log(result);
87          } catch(api_exception) {
88              console.log(api_exception);
89          }
90          console.log("----------- Delete ParallelEnvironment ------------------");
91          try {
92
93              var name = "xyz";
94
95              var res = apiclient.deleteParallelEnvironment(name);
96              if (res) {
97                  console.log("delete success");
98              } else {
99                  console.log("delete failed");
100             }
101         } catch(api_exception) {
102             console.log("api_exception****");
103             console.log(api_exception);
104         }
105     }
106
107 }
108
109 var main = function(args) {
110     var apitest = new UGEWSClientApiTest();
111     var array = apitest.readUserCredentials();
112     console.log("Test User: %s", apitest.getUser());
113
114     apitest.testParallelEnvironment();
115 }
116
117 if (require.main === module) {
118     main();
119 }
```

7 Altair Grid Engine REST Service Meteor Binding

Here we present a simple Meteor application that makes use of the Altair Grid Engine REST Meteor Binding package. For a Meteor application of the stylesheet file myapp.css, it consists of myapp.html containing the layout, and myapp.js containing the JavaScript code that uses the Altair Grid Engine REST Meteor Binding package functionality.

The list of packages used is (meteor list):

```
accounts-base                   1.1.3  A user account system
accounts-facebook               1.0.3  Login service for Facebook accounts
```

```
accounts-google                1.0.3  Login service for Google accounts
accounts-password              1.0.6  Password support for accounts
accounts-ui                    1.1.4  Simple templates to add login widgets to an app
autopublish                    1.0.2  Publish the entire database to all clients
bootstrap                      1.0.1  Front-end framework from Twitter
http                           1.0.10  Make HTTP calls to remote servers
insecure                       1.0.2  Allow all database writes by default
iron:router                    1.0.7  Routing specifically designed for Meteor
mrt:crypto-hmac-sha256-base64  0.0.1  SHA256 HMAC and base64 from CryptoJS repackaged for meteor
npm-container                  1.0.0+ Contains all your npm dependencies
standard-app-packages          1.0.4  Moved to meteor-platform
ugerestsdk                     0.0.1+ \GEFullName{} Rest SDK
```

myapp.html

```
<head>
    <title>myapp</title>
</head>

<body>
    {{> navigation}}
</body>

<template name="home">
    {{#if currentUser}}
    {{> hello}}
    {{else}}
    {{> loggedOut}}
    {{/if}}
</template>

<template name="navigation">
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="navbar-inner">
            <div class="container">
                <button type="button" class="btn btn-navbar" data-toggle="collapse"
                                      data-target=".nav-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="brand" href="{{pathFor 'home'}}">UGE WS</a>
                <div class="nav-collapse collapse">
                    <ul class="nav">
                        <li class="active"><a href="{{pathFor 'home'}}">Home</a></li>
                        <li class="menu-item dropdown">
                            <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                               About<b class="caret"></b></a>
                            <ul class="dropdown-menu">
```

```
35                                  <li class="menu-item dropdown dropdown-submenu">
36                                      <a href="#" class="dropdown-toggle"
37                                         data-toggle="dropdown">Submenu</a>
38                                      <!--a href="/about">About</a-->
39                                      <ul class="dropdown-menu">
40                                          <li class="menu-item dropdown">
41                                              <a href="{{pathFor 'clusterqueues'}}">
42                                                  Cluster Queues</a></li>
43                                          <li class="menu-item dropdown">
44                                              <a href="{{pathFor 'hosts'}}">Hosts</a></li>
45                                      </ul>
46                                  </li>
47                                  <li class="menu-item dropdown">
48                                      <a href="{{pathFor 'clusterqueues'}}">
49                                                  Cluster Queues</a></li>
50                                  <li class="menu-item dropdown">
51                                      <a href="{{pathFor 'hosts'}}">Hosts</a></li>
52                                  <li class="menu-item dropdown">
53                                      <a href="{{pathFor 'jobs'}}">Jobs</a></li>
54                              </ul>
55                          </li>
56                          <li><a href="{{pathFor 'clusterqueues'}}">Cluster Queues</a></li>
57                          <li><a href="{{pathFor 'hosts'}}">Hosts</a></li>
58                          <li><a href="{{pathFor 'jobs'}}">Jobs</a></li>
59                      </ul>
60                      <ul class="nav navbar-nav navbar-right">
61                          <li>{{> loginButtons}}</li>
62                      </ul>
63                  </div> <!--/.nav-collapse -->
64              </div>
65          </div>
66      </div>
67  </template>
68
69  <template name="hello">
70      <h1>UGE WS App</h1>
71      <!-- input type="button" id="getClearPassword" value="Get Password" /-->
72  </template>
73
74  <template name="clusterqueues">
75      <h1>{{title}}</h1>
76      <input type="button" id="getCQS" value="Get CQS" />
77      <input type="button" id="getCQSList" value="Get CQS List" />
78      <input type="button" id="addCQ" value="Add CQ" />
79      <input type="button" id="deleteCQ" value="Delete CQ" />
80      <ul>
81      {{#each content}}
82      <li>{{name}}</li>
83      {{/each}}
```

```
84      </ul>
85      <div>
86      {{> clusterqueuelist }}
87      </div>
88  </template>
89
90
91  <template name="clusterqueuelist">
92      {{#each cqlist}}
93      <div class="col-md-3">
94      <h4>Name: {{name}}</h4>
95      Reserved Slots: {{reservedSlots}}
96      Error: {{error}}
97      Disabled by Calendar: {{disabledByCalendar}}
98      Disabled Manual: {{disabledManual}}
99      Total Slots: {{totalSlots}}
100     Suspend Thresholds: {{suspendThreshold}}
101     Unknown: {{unknown}}
102     Used Slots: {{usedSlots}}
103     Ambiguous: {{ambiguous}}
104     Suspend Manual: {{suspendManual}}
105     Orphaned: {{orphaned}}
106     Available Slots: {{availableSlots}}
107     Temporary Disabled: {{tempDisabled}}
108     Load: {{load}}
109     Load Alarm: {{loadAlarm}}
110     Manual Intervention: {{manualIntervention}}
111     Suspend On Subordinate: {{suspendOnSubordinate}}
112     Suspend by Calendar: {{suspendByCalendar}}
113     </div>
114     {{/each }}
115 </template>
116
117 <template name="hosts">
118     <h1>{{title}}</h1>
119     <input type="button" id="getHost" value="Get Hosts" />
120     <input type="button" id="getHostList" value="Get Host List" />
121     <p>{{content}}</p>
122     <p>{{> hostlist}}</p>
123 </template>
124
125 <template name="hostlist">
126     {{#each hostlist}}
127     <div class="col-md-3">
128     <h4>Hostname: {{hostname}}</h4>
129     Arch: {{arch}}
130     Total Memory: {{memTotal}}
131     Used Memory: {{memUsed}}
132     Total Swap: {{swapTotal}}
```

```
133     Used Swap: {{swapUsed}}
134     Queue Count: {{queueCount}}
135     Queue List: {{queueList}}
136     Load Avg: {{loadAvg}}
137     Number of Processors: {{numberOfProcessors}}
138     Job Count: {{jobCount}}
139     Job List: {{jobList}}
140     Dominance Set: {{dominanceSet}}
141     HostValueCount: {{hostValueCount}}
142     HostValueKeys: {{hostValueKeys}}
143     <!--:["swap_used","num_proc","mem_total","arch_string",
144           "mem_used","load_avg","swap_total"] -->
145     </div>
146     {{/each}}
147 </template>
148
149
150 <template name="jobs">
151     <h1>{{title}}</h1>
152     <input type="button" id="addJob" value="Submit Job" />
153     <input type="button" id="getJob" value="Get Jobs" />
154     <input type="button" id="getJobList" value="Get Job List" />
155     <input type="button" id="deleteJob" value="Delete Job" />
156     <input type="button" id="deleteJobViaApi" value="Delete Job Via Api" />
157     <input type="text" id="txtJob" value="JobId">
158     <p>{{content}}</p>
159     <p>{{> joblist}}</p>
160 </template>
161
162 <template name="joblist">
163     {{#each jobslist}}
164     <div class="col-md-3">
165     <h4>Name: {{name}}</h4>
166     <ul>
167     <li>Job Id: {{jobid}}</li>
168     <li>Tasks: {{tasks}}</li>
169     <li>Queue: {{queue}}</li>
170     <li>Command: {{command}}</li>
171     <li>Command Args: {{commandArgs}}</li>
172     <li>Hard Resources: {{resources.hard}}</li>
173     <li>Soft Resources: {{resources.soft}}</li>
174     <li>Submission Time: {{timeStamp.submit}}</li>
175     <li>Start Time: {{timeStamp.start}}</li>
176     <li>Submit Epoch: {{submitEpoch}}</li>
177     <li>Start Epoch: {{startEpoch}}</li>
178     <li>Mail Options: {{mailOpt}}</li>
179     <li>Department: {{department}}</li>
180     <li>State: {{state}}</li>
181     <li>Messages: {{messages}}</li>
```

```
    <li>Priority: {{priority}}</li>
    <li>Path: {{path}}</li>
    <li>Messages Global: {{messagesGlobal}}</li>
    <li>Restart: {{restart}}</li>
    <li>Host: {{host}}</li>
    <li>Context Vars: {{contextVars}}</li>
    <li>Queue Request: {{queueReq}}</li>
    <li>Time State: {{timeState}}</li>
    <li>Environment: {{env}}</li>
    <li>Shares: {{shares}}</li>
    <li>Slots: {{slots}}</li>
    <li>Account: {{account}}</li>
    <li>Usage: {{usage}}</li>
    <li>User: {{user}}</li>
    <li>Dependency Expression: {{dependencyExpr}}</li>
    </ul>
    </div>
    {{/each}}
</template>

<template name="about">
    <h1>About</h1>
    This is the about section.
</template>

<template name="loggedOut">
    <h1>Please sign in</h1>
</template>
```

myapp.css

```
/* CSS declarations go here */
body {
  padding-top: 60px;
}

/* START NAV MENU */
nav {
  background-color:#2C5463;
  height:40px;
}


nav ul {
  font-family: Arial, Verdana;
  font-size: 20px;
  margin: 0;
  padding: 0;
  list-style: none;
```

```
}

nav ul li {
  display: block;
  position: relative;
  float: left;

}

nav li ul {
  display: none;
}

nav ul li a {
  display: block;
  text-decoration: none;
  padding: 7px 15px 3px 15px;
  background: #2C5463;
  color: #ffffff;
  margin-left: 1px;
  white-space: nowrap;
  height:30px; /* Width and height of top-level nav items */
  width:90px;
  text-align:center;

}

nav ul li a:hover {
  background: #617F8A;
}

nav li:hover ul {
  display: block;
  position: absolute;
  height:30px;
}

nav li:hover li {
  float: none;
  font-size: 11px;

}

nav li:hover a {
  background: #3A464F;
  height:30px; /* Height of lower-level nav items is shorter than main level */
}

nav li:hover li a:hover {
```

```
  background: #95A9B1;
}

nav ul li ul li a {
    text-align:left; /* Top-level items are centered, but nested list items are left-aligned */
}
/* END NAV MENU */
.dropdown-submenu {
    position:relative;
}
.dropdown-submenu>.dropdown-menu {
    top:0;
    left:100%;
    margin-top:-6px;
    margin-left:-1px;
    -webkit-border-radius:0 6px 6px 6px;
    -moz-border-radius:0 6px 6px 6px;
    border-radius:0 6px 6px 6px;
}
.dropdown-submenu:hover>.dropdown-menu {
    display:block;
}
.dropdown-submenu>a:after {
    display:block;
    content:" ";
    float:right;
    width:0;
    height:0;
    border-color:transparent;
    border-style:solid;
    border-width:5px 0 5px 5px;
    border-left-color:#cccccc;
    margin-top:5px;
    margin-right:-10px;
}
.dropdown-submenu:hover>a:after {
    border-left-color:#ffffff;
}
.dropdown-submenu.pull-left {
    float:none;
}
.dropdown-submenu.pull-left>.dropdown-menu {
    left:-100%;
    margin-left:10px;
    -webkit-border-radius:6px 0 6px 6px;
    -moz-border-radius:6px 0 6px 6px;
    border-radius:6px 0 6px 6px;
}
```

myapp.js

```
Router.configure({
//layoutTemplate: 'ApplicationLayout',
    templateNameConverter: 'upperCamelCase'
});
Router.map(function() {
    this.route('clusterqueues', {
        path: '/clusterqueues',
        template: 'clusterqueues',
        content: function() {
            Meteor.call("checkCQS", function(error, results) {
                Session.set("UGE_CQS", results.data);
                return results.data;
            });
            return Session.get("UGE_CQS");
        }
    });

    this.route('hosts', {
        path: '/hosts',
        template: 'hosts',
        content: function() {
            Meteor.call("checkHosts", function(error, results) {
                Session.set("UGE_Hosts", results.data);
                return results.data;
            });
            return Session.get("UGE_Hosts");
        }
    });

    this.route('jobs', {
        path: '/jobs',
        template: 'jobs',
        content: function() {
            Meteor.call("checkJobs", function(error, results) {
                Session.set("UGE_Jobs", results.data);
                return results.data;
            });
            return Session.get("UGE_Jobs");
        }
    });

    this.route('about', {
        path: '/about',
        template: 'about'
    });

    this.route('home', {path: '/', template: 'home'});
```

```
48
49      this.route('contact', 'contact');
50
51      this.route('page.one', {
52          path: '/one'
53      });
54
55      this.route('page.two', {
56          // if the template is different from the name we can specify it directly
57          // like this
58          template: 'PageTwo',
59          // if the path can't be inferred from the name we can provide it here
60          path: '/two'
61      });
62
63      this.route('downloadfile', {
64          // server route
65          where: 'server',
66          action: function() {
67              this.response.end('SERVER ROUTE');
68          }
69      });
70  });
71
72
73  if (Meteor.isClient) {
74      var cqcontents = null;
75      var cqcontentsDep = new Deps.Dependency();
76      var cqlcontents = null;
77      var cqlcontentsDep = new Deps.Dependency();
78      var hostcontents = null;
79      var hostcontentsDep = new Deps.Dependency();
80      var hostlistcontents = null;
81      var hostlistcontentsDep = new Deps.Dependency();
82      var jobcontents = null;
83      var jobcontentsDep = new Deps.Dependency();
84      var joblcontents = null;
85      var joblcontentsDep = new Deps.Dependency();
86
87      // subscribe additional fields that are exported by Meteor server (see below)
88      // Meteor.subscribe("userData"); // not necessary if autopublish package present
89
90      Accounts.ui.config({passwordSignupFields: 'USERNAME_ONLY'});
91
92      Template.hello.greeting = function() {
93          return "Welcome to myapp.";
94      };
95
96      var getCQS = function() {
```

```
97          console.log("User: " + Meteor.user().username);
98          console.log("Meteor.user(): %j", Meteor.user());
99          Meteor.call("checkCQS", function(error, results) {
100             console.log("Status code: " + results.statusCode);
101             console.log("Content: " + results.content);
102             console.log("Headers: ", results.headers)
103             cqcontents = results.data;
104             cqcontentsDep.changed();
105         });
106     };
107
108     var getCQSList = function() {
109         console.log("User: " + Meteor.user().username);
110         console.log("Meteor.user(): %j", Meteor.user());
111         Meteor.call("checkCQSList", function(error, results) {
112             console.log("Status code: " + results.statusCode);
113             console.log("Content: " + results.content);
114             console.log("Headers: ", results.headers)
115             cqlcontents = results.data;
116             cqlcontentsDep.changed();
117         });
118     };
119
120     var getHost = function() {
121         console.log("User: " + Meteor.user().username);
122         console.log("Meteor.user(): %j", Meteor.user());
123         Meteor.call("checkHosts", function(error, results) {
124             console.log("Status code: " + results.statusCode);
125             console.log("Content: " + results.content);
126             console.log("Headers: ", results.headers)
127             hostcontents = results.content;
128             hostcontentsDep.changed();
129         });
130     };
131
132     var getHostList = function() {
133         console.log("User: " + Meteor.user().username);
134         console.log("Meteor.user(): %j", Meteor.user());
135         Meteor.call("checkHostList", function(error, results) {
136             console.log("Status code: " + results.statusCode);
137             console.log("Content: " + results.content);
138             console.log("Headers: ", results.headers)
139             hostlistcontents = results.data;
140             hostlistcontentsDep.changed();
141         });
142     };
143
144     var getJob = function() {
145         console.log("User: " + Meteor.user().username);
```

```
146         console.log("Meteor.user(): %j", Meteor.user());
147         Meteor.call("checkJobs", function(error, results) {
148             console.log("Status code: " + results.statusCode);
149             console.log("Content: " + results.content);
150             console.log("Headers: ", results.headers)
151             jobcontents = results.content;
152             jobcontentsDep.changed();
153         });
154     };
155 
156     var getJobList = function() {
157         console.log("User: " + Meteor.user().username);
158         console.log("Meteor.user(): %j", Meteor.user());
159         Meteor.call("checkJobList", function(error, results) {
160             console.log("Status code: " + results.statusCode);
161             console.log("Content: " + results.content);
162             console.log("Headers: ", results.headers)
163             joblcontents = results.data;
164             joblcontentsDep.changed();
165         });
166     };
167 
168 
169     Template.clusterqueues.events({
170         'click #getCQS': function() {
171             getCQS();
172         },
173         'click #getCQSList': function() {
174             getCQSList();
175         },
176         'click #deleteCQ': function() {
177             console.log("User: " + Meteor.user().username);
178             console.log("Meteor.user(): %j", Meteor.user());
179             Meteor.call("deleteCQ", function(error, results) {
180                 console.log("Status code: " + results.statusCode);
181                 console.log("Content: " + results.content);
182                 console.log("Headers: ", results.headers)
183             });
184             setTimeout(function() {
185                 getCQS();
186                 getCQSList();
187             }, 1000);
188         },
189         'click #addCQ': function() {
190             console.log("User: " + Meteor.user().username);
191             console.log("Meteor.user(): %j", Meteor.user());
192             Meteor.call("addCQ", function(error, results) {
193                 console.log("Status code: " + results.statusCode);
194                 console.log("Content: " + results.content);
```

```
195                 console.log("Headers: ", results.headers)
196             });
197             setTimeout(function() {
198                 getCQS();
199                 getCQSList();
200             }, 1000);
201         }
202     });
203
204     Template.hosts.events({
205         'click #getHost': function() {
206             getHost();
207         },
208         'click #getHostList': function() {
209             getHostList();
210         }
211     });
212
213     Template.hello.events({
214         'click #getClearPassword': function() {
215             Meteor.call("getClearPassword", function(error, results) {
216                 console.log(results); //results.data should be a JSON object
217             });
218         }
219     });
220
221     Template.jobs.events({
222         'click #addJob': function() {
223             console.log("User: " + Meteor.user().username);
224             console.log("Meteor.user(): %j", Meteor.user());
225             Meteor.call("addJob", function(error, results) {
226                 console.log("Status code: " + results.statusCode);
227                 console.log("Content: " + results.content);
228                 console.log("Headers: ", results.headers)
229             });
230             setTimeout(function() {
231                 getJob();
232                 getJobList();
233             }, 1000);
234         },
235         'click #deleteJob': function() {
236             console.log("User: " + Meteor.user().username);
237             console.log("Meteor.user(): %j", Meteor.user());
238             var jobid = document.getElementById("txtJob").value;
239             Meteor.call("deleteJob", jobid, function(error, results) {
240                 console.log("Status code: " + results.statusCode);
241                 console.log("Content: " + results.content);
242                 console.log("Headers: ", results.headers)
243             });
```

```
244             setTimeout(function() {
245                 getJob();
246                 getJobList();
247             }, 1000);
248         },
249         'click #deleteJobViaApi': function() {
250             console.log("User: " + Meteor.user().username);
251             console.log("Meteor.user(): %j", Meteor.user());
252             var jobid = document.getElementById("txtJob").value;
253             Meteor.call("deleteJobViaApi", jobid, function(error, results) {
254                 console.log("Status code: " + results.statusCode);
255                 console.log("Content: " + results.content);
256                 console.log("Headers: ", results.headers)
257             });
258             setTimeout(function() {
259                 getJob();
260                 getJobList();
261             }, 1000);
262         },
263         'click #getJob': function() {
264             getJob();
265         },
266         'click #getJobList': function() {
267             getJobList();
268         }
269     });
270 
271     Template.clusterqueues.helpers({
272         content: function() {
273             Meteor.call("checkCQS", function(error, results) {
274                 console.log("data1: " + results.data);
275                 console.log("template call: %j", results);
276                 Session.set("UGE-CQS", results.data);
277             });
278             cqcontents = Session.get("UGE-CQS");
279             cqcontentsDep.depend();
280             //var cqs = cqcontents.split(',');
281             var qs = new Array();
282             for (i = 0; i < cqcontents.length; i++) {
283                 var q = new Object();
284                 q.name = cqcontents[i];
285                 qs[i] = q;
286             }
287             console.log(qs);
288             return qs;
289         },
290         title: function() {
291             return "Cluster Queue";
292         }
```

```
293     });
294
295     Template.clusterqueuelist.helpers({
296         cqlist: function() {
297             cqlcontentsDep.depend();
298             return cqlcontents;
299         }
300     });
301
302     Template.hosts.helpers({
303         content: function() {
304             Meteor.call("checkHosts", function(error, results) {
305                 console.log("data1: " + results.data);
306                 Session.set("UGE_Hosts", results.data);
307             });
308             hostcontents = Session.get("UGE_Hosts");
309             hostcontentsDep.depend();
310             return hostcontents;
311         },
312         title: function() {
313             return "Hosts";
314         }
315     });
316
317     Template.jobs.helpers({
318         content: function() {
319             Meteor.call("checkJobs", function(error, results) {
320                 Session.set("UGE_Jobs", results.data);
321             });
322             jobcontents = Session.get("UGE_Jobs");
323             jobcontentsDep.depend();
324             return jobcontents;
325         },
326         title: function() {
327             return "Jobs";
328         }
329     });
330
331     Template.joblist.helpers({
332         jobslist: function() {
333             joblcontentsDep.depend();
334             return joblcontents;
335         }
336     });
337
338     Template.hostlist.helpers({
339         hostlist: function() {
340             hostlistcontentsDep.depend();
341             return hostlistcontents;
```

```
342         }
343     });
344
345 }
346
347 if (Meteor.isServer) {
348     Meteor.startup(function() {
349         user = "";
350         pw = "";
351         var fs = Npm.require('fs');
352         readUserCredentials = function() {
353             var array = fs.readFileSync('/var/tmp/challenge.txt').toString().split("\n");
354             user = array[0];
355             pw = array[1];
356         };
357         readUserCredentials();
358         var basePath = "http://localhost:8182";
359         apiclient = new UGEWSClientApi(basePath, user, pw);
360         // code to run on server at startup
361     });
362
363     // export additional field services.password.clear in users collection
364     // (see Meteor.users() doc)
365     /*
366      Meteor.publish("userData", function () {
367      if (this.userId) {
368      return Meteor.users.find({_id: this.userId}, {fields: {'services.password.clear': 1}});
369      } else {
370      this.ready();
371      }
372      });
373      */
374
375     // var creds = "Basic ";
376     // Meteor.call("getClearPassword", function(result) {
377     //   creds = creds + result;
378     // });
379     // var creds = "Basic YWFsZWZlbGQ6OHVuZzl1bmcK";
380     var urlbase = "http://localhost:8182";
381     Meteor.methods({
382         getClearPassword: function() {
383             return "Basic " + Meteor.user().services.password.clear;
384         },
385         checkCQS: function() {
386             var response;
387             try {
388                 this.unblock();
389                 // var creds = getClearPassword();  does not work why ???
390                 // console.log("CheckCQS Clear " + creds);
```

```
391                 var creds = "Basic " + Meteor.user().services.password.clear;
392                 // return HTTP.call("GET", urlbase + "/clusterqueues",
393                 //                  { auth : cleartextcreds});
394                 response = HTTP.call("GET", urlbase + "/clusterqueues",
395                                      {headers: {"Authorization": creds}});
396             } catch (e) {
397                 console.log("Exception: %j", e.response);
398                 response = e.response;
399             } finally {
400                 return response;
401             }
402         },
403         checkCQSList: function() {
404             var response;
405             try {
406                 this.unblock();
407                 // var creds = getClearPassword();  does not work why ???
408                 // console.log("CheckCQS Clear " + creds);
409                 var creds = "Basic " + Meteor.user().services.password.clear;
410                 // return HTTP.call("GET", urlbase + "/clusterqueues",
411                 // { auth : cleartextcreds});
412                 response = HTTP.call("GET", urlbase + "/clusterqueuesummary/1/25",
413                                      {headers: {"Authorization": creds}});
414             } catch (e) {
415                 console.log("Exception: %j", e.response);
416                 response = e.response;
417             } finally {
418                 return response;
419             }
420         },
421         checkHosts: function() {
422             var response;
423             try {
424                 this.unblock();
425                 var creds = "Basic " + Meteor.user().services.password.clear;
426                 response = HTTP.call("GET", urlbase + "/exechosts",
427                                      {headers: {"Authorization": creds}});
428             } catch (e) {
429                 console.log("Exception: %j", e.response);
430                 response = e.response;
431             } finally {
432                 return response;
433             }
434         },
435         checkHostList: function() {
436             var response;
437             try {
438                 this.unblock();
439                 var creds = "Basic " + Meteor.user().services.password.clear;
```

```
440                 response = HTTP.call("GET", urlbase + "/hostsummary/1/100",
441                                      {headers: {"Authorization": creds}});
442             } catch (e) {
443                 console.log("Exception: %j", e.response);
444                 response = e.response;
445             } finally {
446                 return response;
447             }
448         },
449         checkJobs: function() {
450             var response;
451             try {
452                 this.unblock();
453                 var creds = "Basic " + Meteor.user().services.password.clear;
454                 response = HTTP.call("GET", urlbase + "/jobs",
455                                      {headers: {"Authorization": creds}});
456             } catch (e) {
457                 console.log("Exception: %j", e.response);
458                 response = e.response;
459             } finally {
460                 return response;
461             }
462         },
463         checkJobList: function() {
464             var response;
465             try {
466                 this.unblock();
467                 var creds = "Basic " + Meteor.user().services.password.clear;
468                 response = HTTP.call("GET", urlbase + "/jobs/1/100",
469                                      {headers: {"Authorization": creds}});
470             } catch (e) {
471                 console.log("Exception: %j", e.response);
472                 response = e.response;
473             } finally {
474                 return response;
475             }
476         },
477         addCQ: function() {
478             var response;
479             try {
480                 this.unblock();
481                 var creds = "Basic " + Meteor.user().services.password.clear;
482                 response = HTTP.call("POST", urlbase + "/clusterqueues",
483                         {headers: {"content-type": "application/json", "Authorization": creds},
484                             data: {"qname": "test.q", "hostlist": ["@allhosts"],
485                                    "seq_no": [{"hostgroup": "@/", "value": 0},
486                                    {"hostgroup": "amiata", "value": 2}],
487                                    "load_thresholds": [{"hostgroup": "@/",
488                                    "value": [{"name": "np_load_avg",
```

```
 "value": 1.75}]}], "suspend_thresholds":
[{"hostgroup": "@/", "value": [{"name": "np_load_avg",
  "value": 1.25}]}, {"hostgroup": "amiata",
  "value": [{"name": "np_load_avg", "value": 1.35}]}],
  "nsuspend": [{"hostgroup": "@/", "value": 1},
  {"hostgroup": "amiata", "value": 5}],
  "suspend_interval": [{"hostgroup": "@/",
  "value": "00:05:00"}, {"hostgroup": "amiata",
  "value": "00:01:00"}], "priority":
  [{"hostgroup": "@/", "value": "0"},
  {"hostgroup": "amiata", "value": "10"}],
  "min_cpu_interval": [{"hostgroup": "@/",
  "value": "00:05:00"}, {"hostgroup": "amiata",
  "value": "00:01:00"}], "qtype":
  [{"hostgroup": "@/", "value": "BATCH INTERACTIVE"}],
  "ckpt_list": [{"hostgroup": "@/", "value": []}],
  "pe_list": [{"hostgroup": "@/", "value": ["make"]}],
  "jc_list": [{"hostgroup": "@/", "value": ["NO_JC",
  "ANY_JC"]}], "rerun": [{"hostgroup": "@/",
  "value": "FALSE"}], "slots": [{"hostgroup": "@/",
  "value": 1}], "tmpdir": [{"hostgroup": "@/",
  "value": "/tmp"}], "shell": [{"hostgroup": "@/",
  "value": "/bin/sh"}], "prolog": [{"hostgroup": "@/",
  "value": "NONE"}], "epilog": [{"hostgroup": "@/",
  "value": "NONE"}], "shell_start_mode":
  [{"hostgroup": "@/", "value": "unix_behavior"}],
  "starter_method": [{"hostgroup": "@/", "value": "NONE"}],
  "suspend_method": [{"hostgroup": "@/", "value": "NONE"}],
  "resume_method": [{"hostgroup": "@/", "value": "NONE"}],
  "terminate_method": [{"hostgroup": "@/", "value": "NONE"}],
  "notify": [{"hostgroup": "@/", "value": "00:00:60"}],
  "owner_list": [{"hostgroup": "@/", "value": []}],
  "user_lists": [{"hostgroup": "@/", "value": []}],
  "xuser_lists": [{"hostgroup": "@/", "value": []}],
  "subordinate_list": [{"hostgroup": "@/", "value": []}],
  "complex_values": [{"hostgroup": "@/", "value": []}],
  "projects": [{"hostgroup": "@/", "value": []}],
  "xprojects": [{"hostgroup": "@/", "value": []}],
  "calendar": [{"hostgroup": "@/", "value": "NONE"}],
  "initial_state": [{"hostgroup": "@/", "value": "default"}],
  "s_rt": [{"hostgroup": "@/", "value": "INFINITY"}],
  "h_rt": [{"hostgroup": "@/", "value": "INFINITY"}],
  "d_rt": [{"hostgroup": "@/", "value": "INFINITY"}],
  "s_cpu": [{"hostgroup": "@/", "value": "INFINITY"}],
  "h_cpu": [{"hostgroup": "@/", "value": "INFINITY"}],
  "s_fsize": [{"hostgroup": "@/", "value": "INFINITY"}],
  "h_fsize": [{"hostgroup": "@/", "value": "INFINITY"}],
  "s_data": [{"hostgroup": "@/", "value": "INFINITY"}],
  "h_data": [{"hostgroup": "@/", "value": "INFINITY"}],
```

```
                                    "s_stack": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "h_stack": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "s_core": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "h_core": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "s_rss": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "h_rss": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "s_vmem": [{"hostgroup": "@/", "value": "INFINITY"}],
                                    "h_vmem": [{"hostgroup": "@/", "value": "INFINITY"}]}
                        });
            } catch (e) {
                console.log("Exception: %j", e.response);
                response = e.response;
            } finally {
                return response;
            }
        },
        deleteCQ: function() {
            var response;
            try {
                this.unblock();
                var creds = "Basic " + Meteor.user().services.password.clear;
                response = HTTP.call("DELETE", urlbase + "/clusterqueues",
                        {headers: {"content-type": "application/json", "Authorization": creds},
                            data: {"qname": "test.q"}
                        });
            } catch (e) {
                console.log("Exception: %j", e.response);
                response = e.response;
            } finally {
                return response;
            }
        },
        addJob: function() {
            var response;
            try {
                this.unblock();
                var creds = "Basic " + Meteor.user().services.password.clear;
                response = HTTP.call("POST", urlbase + "/jobs",
                        {headers: {"content-type": "application/json", "Authorization": creds},
                            data: {"remoteCommand": "/Users/user/bin/sleeper.sh",
                                   "args": ["120"], "email": ["user@example.com"],
                                   "jobName": "Snooze", "joinFiles": "true"}
                        });
            } catch (e) {
                console.log("Exception: %j", e.response);
                response = e.response;
            } finally {
                return response;
            }
```

```
587         },
588         deleteJob: function(jobid) {
589             var response;
590             try {
591                 this.unblock();
592                 var creds = "Basic " + Meteor.user().services.password.clear;
593                 response = HTTP.call("DELETE", urlbase + "/jobs",
594                         {headers: {"content-type": "application/json", "Authorization": creds},
595                             data: {"jobid": jobid}
596                         });
597             } catch (e) {
598                 console.log("Exception: %j", e.response);
599                 response = e.response;
600             } finally {
601                 return response;
602             }
603         },
604         deleteJobViaApi: function(jobid) {
605             var response = true;
606             if (Meteor.user() != null) {
607             // try {
608                 // var user = Meteor.user().username;
609                 // var pw = "secret";
610                 console.log(UGEWSClientApi);
611                 // console.log("apiclient");
612                 //console.log(apiclient);
613                 //console.log("---------");
614                 //console.log(apiclient.getBasePath());
615                 //console.log(jobid);
616                 response = apiclient.deleteJob(jobid);
617                 console.log(response);
618 
619             // } catch(e) {
620             //     console.log("Exception: %j", e.response);
621             // }
622             }
623             return response;
624         }
625     });
626 }
```

8 Altair Grid Engine REST Service Python Binding

The following example uses the Altair Grid Engine REST Service Python Binding to write a small test program operating on the Hostgroup object. In the testHostgroup function the different http requests for adding (POST), modifying (PUT), deleting (DELETE) and the get list, get range and get by ID (all GET) are executed. First a setup is performed for the test reading

in again the username and password from /var/tmp/challenge.txt, creating a UGEWSClientApi instance and then working with it. For the creation of the UGEWSClientApi instance basePath which is the base, a URL like `http://localhost:8182`, username, and password are needed. These variables are initialized in the **init**() and readUserCredential() functions. To make the classes of the Altair Grid Engine REST Service Python Binding available `from ugerestsdk import *` is used.

In order to run the example copy it to a file and do:

```
python UGEWSClientApiTest.py
```

UGEWSClientApiTest.py

```
import random
import unittest
import logging
import time
from ugerestsdk import *


class UGEWSClientApiTest(unittest.TestCase):

    def __init__(self, *args, **kwargs):
        super(UGEWSClientApiTest, self).__init__(*args, **kwargs)
        self.basePath = "http://localhost:8182"
        self.user = None
        self.pw = None
        self.withAuthentication = False
        self.timeout = 0.500 # unit is second

    def readUserCredentials(self):
        try:
            pwfile = open('/var/tmp/challenge.txt', 'r')
            self.user = pwfile.readline().strip("\n")
            self.pw = pwfile.readline().strip("\n")
            self.withAuthentication = True
        except OSError:
            print("usercred file not found")
        except IOError:
            print("usercred file read error")

    def setUp(self):
        print("Setup test...")
        if self.withAuthentication == False:
            print("reading creds...")
            self.readUserCredentials()

    def tearDown(self):
        print("Teardown")
```

```
37
38      # Hostgroup Test is enabled
39
40      def testHostgroup(self):
41          print("###################### Hostgroup ######################")
42          apitester = UGEWSClientApi(self.basePath, self.user, self.pw)
43          print("----------- Add Hostgroup -------------------")
44          try:
45
46              no = Hostgroup("@xyz")
47
48              print(no)
49              print(no.dump())
50              print("no __class__.__name__", no.__class__.__name__)
51              # no.setName("amiata"); etc.
52              ao = apitester.addHostgroup(no)
53              print(ao)
54              print("ao.__class__.__name__", ao.__class__.__name__)
55          except Exception as e:
56              print(e)
57
58          print("----------- Modify Hostgroup -------------------")
59          try:
60
61              no = Hostgroup("@xyz")
62
63              print(no)
64              print(no.dump())
65              print("no __class__.__name__", no.__class__.__name__)
66              # no.setName("amiata"); etc.
67              ao = apitester.modifyHostgroup(no)
68              print(ao)
69              print("ao.__class__.__name__", ao.__class__.__name__)
70          except ApiException as e:
71              print(e)
72
73          print("----------- GetById Hostgroup -------------------")
74          try:
75              print("Wait {} s", self.timeout)
76              time.sleep(self.timeout)
77
78              name = "@xyz"
79
80              go = apitester.getHostgroupById(name)
81              print(go.toJson())
82          except ApiException as e:
83              print(e)
84
85          print("----------- GetList Hostgroup -------------------")
```

```
86          try:
87              result = apitester.getHostgroupList()
88              print(result)
89          except ApiException as e:
90              print(e)
91
92          print("----------- GetRange Hostgroup -------------------")
93          try:
94              result = apitester.getHostgroupRange(1, 25)
95              print(result)
96          except ApiException as e:
97              print(e)
98
99          print("----------- Delete Hostgroup -------------------")
100         try:
101
102             name = "@xyz"
103
104             res = apitester.deleteHostgroup(name)
105             if res == True:
106                 print("delete success")
107             else:
108                 print("delete failed")
109         except ApiException as e:
110             print(e)
```

9 Troubleshooting

9.1 Missing keystore file

When you see the following exception upon starting up the Altair Grid Engine REST Service, the most likely cause is a missing or unreadable keystore file. You have to perform the setup under Prepare key material for https mode and make sure that the file is readable and a valid keystore. Also check the related entries in your $SGE_ROOT/ugerest/conf/ugerest.properties as described. The other way to avoid this problem is to start the Altair Grid Engine REST Service with the **-ns** option, which disables the https connector, but this is only useful for testing purposes.

```
[java] 2015-03-06 11:18:47.847:INFO::main: Logging initialized @530ms
[java] java.lang.NullPointerException
[java]     at org.eclipse.jetty.server.AbstractConnector.<init>(AbstractConnector.java:185)
[java]     at org.eclipse.jetty.server.AbstractNetworkConnector.<init>
                                    (AbstractNetworkConnector.java:44)
[java]     at org.eclipse.jetty.server.ServerConnector.<init>(ServerConnector.java:227)
[java]     at org.restlet.ext.jetty.JettyServerHelper.createConnector
                                                    (JettyServerHelper.java:404)
[java]     at org.restlet.ext.jetty.JettyServerHelper.createServer(JettyServerHelper.java:466)
```

```
[java]     at org.restlet.ext.jetty.JettyServerHelper.getWrappedServer
                                                        (JettyServerHelper.java:806)
[java]     at org.restlet.ext.jetty.JettyServerHelper.start(JettyServerHelper.java:824)
[java]     at org.restlet.Server.start(Server.java:579)
[java]     at org.restlet.Component.startServers(Component.java:642)
[java]     at org.restlet.Component.start(Component.java:567)
[java]     at com.univa.ugerest.UgeRestMain.main(UgeRestMain.java:88)
```

9.2 Solaris basic authentication

When the login on a solaris machine fails with the following jaas.conf configuration

```
UGERestlet {
   com.sun.grid.security.login.UnixLoginModule requisite
       sge_root="${com.sun.grid.jgdi.sgeRoot}"
       debug=false
       auth_method="pam"
       pam_service="login";
};
```

If the Altair Grid Engine REST Service is configured to use basic authentication fetching data usually requires a single authentication from the web browser, and in the bad behavior it is recurring endlessly. In this case proceed with the following steps.

Verify whether the login works for the following command (is internally used by the JaasVerifier):

```
SGE_ARCH=`$SGE_ROOT/util/arch`
$SGE_ROOT/utilbin/$SGE_ARCH/authuser pam -s login
```

Enter the username and corresponding password and check whether the command succeeds. If not, the reason can be the following check in /etc/pam.d/login:

```
auth required    pam_dial_auth.so.1
```

It is usually safe to comment this line out. Then reverify using the command above.

9.3 Exception during basic authentication

If there is not enough memory available e.g. in a virtual machine setup, the startup script might need some tweaking. Look for JVMARGS and limit the setting -Xmx5120m to a smaller value otherwise the following exception might show up in /tmp/UGERestService*.log Solution:

```
$SGE_ROOT/default/common/ugerest stop
vi $SGE_ROOT/default/common/ugerest
   reduce JVMARGS="... -Xmx5120m ..." to e.g. -Xmx2048m
$SGE_ROOT/default/common/ugerest start
curl http://localhost:8182/adminhosts -u andre
["<hostname>"...]

16/07/2016 10:37:09|48|sun.grid.security.login.UnixLoginModule.initialize|W|Can not
                              determine gridengine arch: {0}
                              java.io.IOException: Cannot run program
                                "/home/andre/univa/clusters/UGE/util/arch": error=12,
                                Not enough space
                                java.lang.ProcessBuilder.start(ProcessBuilder.java:1047)
                                java.lang.Runtime.exec(Runtime.java:617)
                                java.lang.Runtime.exec(Runtime.java:528)
                                com.sun.grid.util.expect.Expect.exec(Expect.java:161)
                                com.sun.grid.util.SGEUtil.getArch(SGEUtil.java:164)
                                com.sun.grid.security.login.UnixLoginModule.initialize
                                                        (UnixLoginModule.java:153)
                                sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                                sun.reflect.NativeMethodAccessorImpl.invoke
                                                        (NativeMethodAccessorImpl.java:57)
                                sun.reflect.DelegatingMethodAccessorImpl.invoke
                                                        (DelegatingMethodAccessorImpl.java:43)
                                java.lang.reflect.Method.invoke(Method.java:606)
                              Caused by java.io.IOException: error=12, Not enough space
                                java.lang.UNIXProcess.forkAndExec(Native Method)
                                java.lang.UNIXProcess.<init>(UNIXProcess.java:137)
                                java.lang.ProcessImpl.start(ProcessImpl.java:130)
                                java.lang.ProcessBuilder.start(ProcessBuilder.java:1028)
                                java.lang.Runtime.exec(Runtime.java:617)
                                java.lang.Runtime.exec(Runtime.java:528)
                                com.sun.grid.util.expect.Expect.exec(Expect.java:161)
                                com.sun.grid.util.SGEUtil.getArch(SGEUtil.java:164)
                                com.sun.grid.security.login.UnixLoginModule.initialize
                                                        (UnixLoginModule.java:153)
                                sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

Another possible problem might be insufficient file permissions for the utilbin/*/authuser binary. Some pam frameworks require the authuser to be setuid root. Solution:

```
# chmod 4555 utilbin/*/authuser
```

9.4 Logging Settings

Usually stack traces are not shown in the logging file. To enable this you have to disable the following lines in $SGE_ROOT/ugerest/conf/logging.properties and possibly restart the Altair Grid Engine REST Service.

```
vi conf/logging.properties
...
#java.util.logging.ConsoleHandler.formatter = com.sun.grid.jgdi.util.SGEFormatter
...
#java.util.logging.FileHandler.formatter=com.sun.grid.jgdi.util.SGEFormatter
```

or better:

```
#
# Possible columns:
#
#   time    timestamp of the log message
#   host    hostname of the log message
#   name    name of the logger
#   thread  id of the thread
#   level   log level (short form)
#   source  class and method name
#   level_long log_level long form
#
com.sun.grid.jgdi.util.SGEFormatter.columns = time thread source level message

#
#  Print the stacktrace of the log record
#
com.sun.grid.jgdi.util.SGEFormatter.withStacktrace=true

#
#  Delimiter between columns
#
com.sun.grid.jgdi.util.SGEFormatter.delimiter = |
```