UNIVA, AN ALTAIR COMPANY

GRID ENGINE DOCUMENTATION

Grid Engine Rest Service Guide

Author:
Altair Engineering

Version:
8.6.17

November 4, 2020

Contents

1 Univa Grid Engine REST Service Overview

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

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},

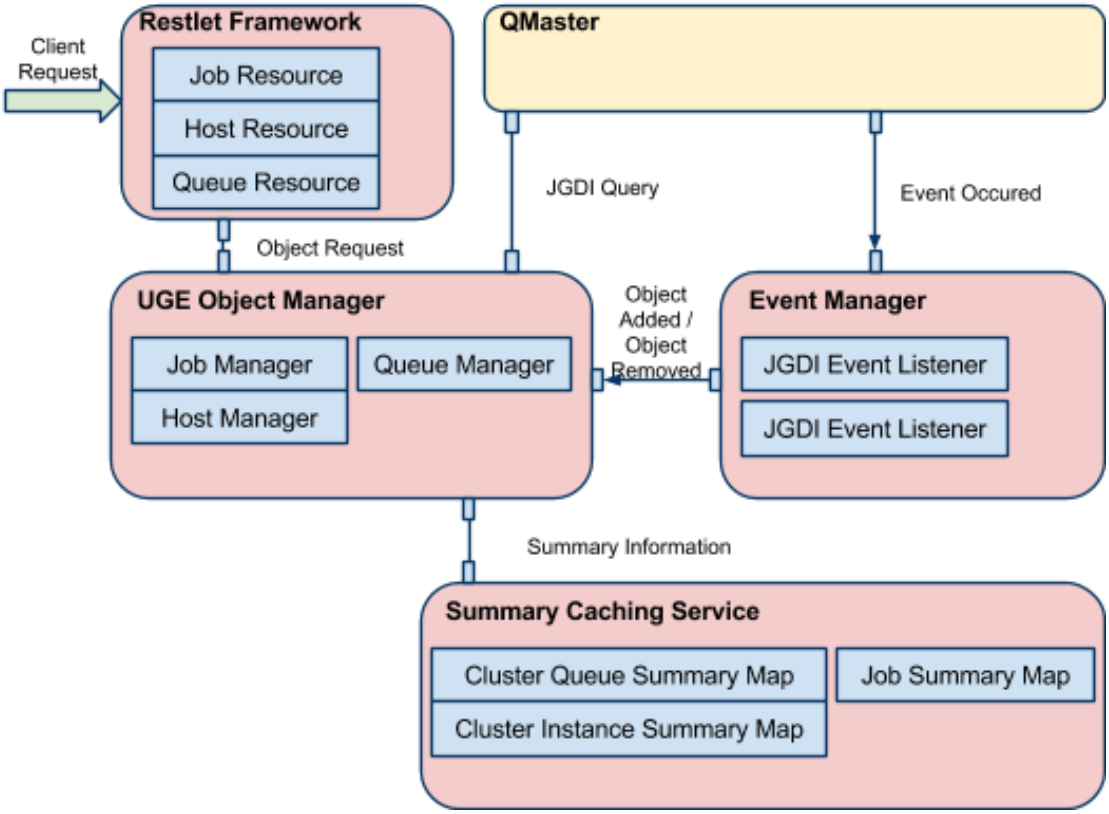
Method	URL	Description
		where {id} is kind of pattern matched
POST	/adminhosts	add a new admin host - the post request contains a corresponding json data string
PUT	/adminhosts	modify an existing admin host - the put request contains a corresponding json data string
DELETE	/adminhosts	delete an existing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

Univa 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 Univa Grid Engine REST Web Service API

Univa 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 Univa Grid Engine JGDI API and the Restlet development framework exposing Univa Grid Engine objects and functionality as REST resources to the application programmer.

The overall architecture is outlined in the following sketch:



The Univa Grid Engine REST Web Service is running as a standalone daemon on whatever host of the Univa Grid Engine cluster and connects via the JGDI API to the Univa Grid Engine master daemon and registers for Univa Grid Engine JGDI events to update the corresponding REST resources.

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

3 Univa Grid Engine REST Service Installation and Setup

Univa Grid Engine REST Service is an add-on to Univa Grid Engine distributed resource management application that allows the access and manipulation of Univa 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 Univa Grid Engine REST Service, and ask questions that have to be answered by the Univa 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 Univa Grid Engine REST Service.

Task	Explanation
Univa Grid Engine 8.3	Univa Grid Engine version 8.3 needs to be installed For detailed installation instructions see the Grid Engine Installation Guide
Univa Grid Engine REST Service host	Univa 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 Univa Grid Engine REST Service due to restlet jar dependencies
OPTIONAL Python 2.7	For using the Univa Grid Engine REST Service Python Binding Python 2.7 must be installed on the system Linux: MacOs: Solaris:
OPTIONAL Python modules	The following Python modules are required: -
OPTIONAL Nodejs	For using the Univa Grid Engine REST Service Nodejs Binding node.js needs to be installed on the system Linux: MacOs:

Task	Explanation
	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
	-

http://blog.nodejs.org/release/1 http://www.petertribble.co.uk/Solaris/node.html

3.2 Sudo configuration

The Univa Grid Engine REST Service is running under a special user let's assume user userrest. To be able to execute requests on behalf of a different user so called sudo requests have been introduced in Univa 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 users for whom requests are executed, using the usernames mentioned above, the following Univa Grid Engine commands have to be performed by the administrator:

```
qconf -au userrest sudomasters      - where userrest is the user
                                      under which the REST Service is running
qconf -au user1,user2 sudoers       - with user1, user2, the users for
                                      whom requests shall be executed
```

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'?' inside 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

Per default the logging level is set to INFO messages and the handler used is the FileHandler logging to /tmp/UGERestService%u.%g.log where %u and %g are unique numbers. To switch the ConsoleHandler on in addition uncomment the handlers 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

```

```
31 # ------------------------------------------------------------------------
32 #  Settings for FileHandler
33 # ------------------------------------------------------------------------
34 # Set the default logging level for new FileHandler instances
35 java.util.logging.FileHandler.level = ALL
36 java.util.logging.FileHandler.pattern=/tmp/UGERestService%u.%g.log
37 java.util.logging.FileHandler.formatter=com.sun.grid.jgdi.util.SGEFormatter
```

3.4 Unpacking and installing Univa Grid Engine REST Service

To install the Univa Grid Engine REST Service you need a working Univa Grid Engine installation. The steps to setup Univa Grid Engine are outlined in detail in the Univa Grid Engine Installation Guide. In order to run the Univa 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
```

Installing UGERest is available by automatic installation with the install_ugerest script. As the Univa Grid Engine admin user do

```
% 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 startup the Univa 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
```

Second approach: Manual installation as the Univa Grid Engine admin user do

create the HTTPS credentials as outlined under Prepare key material for https mode

adjust the ugerest/conf/ugerest.properties file accordingly

start the Univa Grid Engine REST Service via

```
% ugerest/bin/ugerest.sh
```

you can follow the logging messages by ({n},{m} default usually 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 like LdapAuthentication or something else comment 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 like outlined above. What is required for the internal working of the REST service are the UNIX username and groupname and the uid and gid. These are used in so called **sudo requests**. There a prerequisite is that the user is a valid user which is verified against the masters user database.

3.6 Verification of the Installation

To verify the general working of the Univa Grid Engine REST Service you can test the following links in your browser and go from there to 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 Univa 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.

Connecting the first time with the browser will request possibly the confirmation that you trust the page in SSL mode and your Unix user credentials for the connection to Univa Grid Engine REST Service. This is needed for authenticating your requests. For further testing of other HTTP methods *curl* or something like *Postman - 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. port5570) to the settings of your installation. The CSP mode directories have been used here to securely store the UGERest.jks keystore which is not mandatory. 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 Univa Grid Engine REST Service. These settings are mandatory for the Univa Grid Engine REST Service https mode usually.

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)
-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 (depending on your jdk installation, please adjust example below)
keytool -import\
 -alias UGERest\
 -file UGERest.cer\
 -keystore "/Library/Java/JavaVirtualMachines/jdk1.7.0_60.jdk/\
               Contents/Home/jre/lib/security/cacerts"\
 -storepass "changeit"
```

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

further details can be found in the Restlet Technical Documentation

For looking 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/security/cacerts 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/Contents/Ho Enter keystore password: changeit

3.8 Installation of Univa 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 Univa 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** into the classpath of your application.

Example application and the corresponding command lines can be found in the Univa Grid Engine REST Service Java Binding section.

3.9 Installation of Univa 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 prebuilt node.js packages, if this is not possible follow the instructions of nodejs.org. For Mac OS X and Linux installation packages are available, to install them, follow these steps:

```
[MacPorts](https://www.macports.org)
/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 Packages](https://github.com/joyent/node/wiki/installing-node.js-via-package-manager)
e.g. Suse
sudo zypper ar http://download.opensuse.org/repositories/devel:/languages:/nodejs/openSUSE_13.1/N
sudo zypper in nodejs nodejs-devel    (contains npm too)
/usr/bin/node --version

[Solaris x86 Packages](http://www.petertribble.co.uk/Solaris/node.html)
as root
pkgadd -d TRIBnode-0.10.36.pkg
pkginfo | grep TRIB
/opt/Node/bin/node --version
```

After installing node for the nodejs binding some special steps are necessary. You have to install the urllib-sync npm package in addition.

```
npm install urllib-sync
npm list |grep urllib-sync
-> installs urllib-sync to ~/node_modules/urllib-sync
```

To install the Univa 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
-> ugerestsdk@0.0.1 ../../../../../../node_modules/ugerestsdk
```

For using ugerestsdk see the example below. Univa Grid Engine REST Service Nodejs Binding

3.10 Installation of Univa Grid Engine REST Service Meteor Binding

For Meteor installation follow the steps outlined on meteor.com

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

To install the corresponding package Univa 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
```

For making use of the sdk inside your code see below Univa Grid Engine REST Service Meteor Binding.

3.11 Installation of Univa Grid Engine REST Service Python Binding

For Python installation follow the following steps:

```
[MacPorts](https://www.macports.org)
if there is no python installed
/opt/local/bin/port install python27
this will install python 2.7
/opt/local/bin/python2.7 --version

or preinstalled

/usr/bin/python --version

Linux Packages
e.g. Suse
sudo zypper in python
python --version

Solaris Packages
[Sunfreeware](www.sunfreeware.org)
[OpenCSW](www.opencsw.org
as root
/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 Univa 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
)
```

For making use of the sdk inside your code see below Univa 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 Univa 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 a 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 Univa Grid Engine object or a list thereof or 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 looks on the curl command line like 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 result are JSON formatted strings containing in most cases the same information as the corresponding qconf command output. JSON String for PUT/POST/DELETE To POST/PUT/DELETE an object to the UGE system the following command examples can be used. They outline the JSON strings that have to be used to add/modify/delete a UGE 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, 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 with 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"

Action	URL: http[s]://{host}/{port}
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"
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"

Action	URL: http[s]://{host}/{port}
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 a definition 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 important 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/aalefeld/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":"aalefeld"}
```

4.12 Operator

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

```
{"name":"aalefeld"}
```

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":"aalefeld",
"oticket":0,
"fshare":0,
"deleteTime":1405076192}
```

for delete

```
{"name":"aalefeld"}
```

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":["aalefeld"]}
```

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/aalefeld/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/aalefeld/bin/sleeper.sh",
    "args":["120"],
    "email":["aalefeld@univa.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 to run the WS/Restlet component as a user and perform requests on behalf of a different user. How these lists are setup is explained in Sudo configuration.

4.21 Special Targets

For job and queue there exist in addition some special targets. The following routes are mapping 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 build 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 or containing the wildcard * matching zero or more characters. The state pattern {state} describes the state the job has coded 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 only.
                                     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 build 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"
Get QI Range	"/queueinstances/{queueInstanceName}/{count}"

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

5 Univa Grid Engine REST Service Java Binding

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

Usually you have these functionalities for every object and some additional methods that are deviating 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 user name and password are read for simplicity from a file */var/tmp/challenge.txt* in the test method **setUpClass()** in code line:

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

The content of */var/tmp/challenge.txt* are 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/aalefeld/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("aalefeld@univa.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 Univa 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 user name 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 UGEWSClientApiTest.js, create the /var/tmp/challenge.txt file containing username and password and run after having followed the setup steps outlined above:

```
node UGEWSClientApiTest.js
```

Be sure that the Univa Grid Engine REST Service has been started before trying the 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 Univa Grid Engine REST Service Meteor Binding

Here we present a simple Meteor application that makes use of the Univa Grid Engine REST Meteor Binding package. It consists as usual for a Meteor application of the stylesheet file myapp.css, myapp.html containing the layout and myapp.js containing the javascript code that uses the Univa 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+ Univa Grid Engine 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="
                    <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 cl
                            <ul class="dropdown-menu">
                                <li class="menu-item dropdown dropdown-submenu">
                                    <a href="#" class="dropdown-toggle" data-toggle="dropdown">Su
```

```
35                                      <!--a href="/about">About</a-->
36                                      <ul class="dropdown-menu">
37                                          <li class="menu-item dropdown"><a href="{{pathFor 'cluste
38                                          <li class="menu-item dropdown"><a href="{{pathFor 'hosts'
39                                      </ul>
40                                  </li>
41                                  <li class="menu-item dropdown"><a href="{{pathFor 'clusterqueues'
42                                  <li class="menu-item dropdown"><a href="{{pathFor 'hosts'}}">Host
43                                  <li class="menu-item dropdown"><a href="{{pathFor 'jobs'}}">Jobs<
44                              </ul>
45                          </li>
46                          <li><a href="{{pathFor 'clusterqueues'}}">Cluster Queues</a></li>
47                          <li><a href="{{pathFor 'hosts'}}">Hosts</a></li>
48                          <li><a href="{{pathFor 'jobs'}}">Jobs</a></li>
49                      </ul>
50                      <ul class="nav navbar-nav navbar-right">
51                          <li>{{> loginButtons}}</li>
52                      </ul>
53                  </div> <!--/.nav-collapse -->
54              </div>
55          </div>
56      </div>
57  </template>
58
59  <template name="hello">
60      <h1>UGE WS App</h1>
61      <!-- input type="button" id="getClearPassword" value="Get Password" /-->
62  </template>
63
64  <template name="clusterqueues">
65      <h1>{{title}}</h1>
66      <input type="button" id="getCQS" value="Get CQS" />
67      <input type="button" id="getCQSList" value="Get CQS List" />
68      <input type="button" id="addCQ" value="Add CQ" />
69      <input type="button" id="deleteCQ" value="Delete CQ" />
70      <ul>
71      {{#each content}}
72      <li>{{name}}</li>
73      {{/each}}
74      </ul>
75      <div>
76      {{> clusterqueuelist }}
77      </div>
78  </template>
79
80
81  <template name="clusterqueuelist">
82      {{#each cqlist}}
83      <div class="col-md-3">
```

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

```
133     <!--:["swap_used","num_proc","mem_total","arch_string","mem_used","load_avg","swap_total"] --
134     </div>
135     {{/each}}
136 </template>
137
138
139 <template name="jobs">
140     <h1>{{title}}</h1>
141     <input type="button" id="addJob" value="Submit Job" />
142     <input type="button" id="getJob" value="Get Jobs" />
143     <input type="button" id="getJobList" value="Get Job List" />
144     <input type="button" id="deleteJob" value="Delete Job" />
145     <input type="button" id="deleteJobViaApi" value="Delete Job Via Api" />
146     <input type="text" id="txtJob" value="JobId">
147     <p>{{content}}</p>
148     <p>{{> joblist}}</p>
149 </template>
150
151 <template name="joblist">
152     {{#each jobslist}}
153     <div class="col-md-3">
154     <h4>Name: {{name}}</h4>
155     <ul>
156     <li>Job Id: {{jobid}}</li>
157     <li>Tasks: {{tasks}}</li>
158     <li>Queue: {{queue}}</li>
159     <li>Command: {{command}}</li>
160     <li>Command Args: {{commandArgs}}</li>
161     <li>Hard Resources: {{resources.hard}}</li>
162     <li>Soft Resources: {{resources.soft}}</li>
163     <li>Submission Time: {{timeStamp.submit}}</li>
164     <li>Start Time: {{timeStamp.start}}</li>
165     <li>Submit Epoch: {{submitEpoch}}</li>
166     <li>Start Epoch: {{startEpoch}}</li>
167     <li>Mail Options: {{mailOpt}}</li>
168     <li>Department: {{department}}</li>
169     <li>State: {{state}}</li>
170     <li>Messages: {{messages}}</li>
171     <li>Priority: {{priority}}</li>
172     <li>Path: {{path}}</li>
173     <li>Messages Global: {{messagesGlobal}}</li>
174     <li>Restart: {{restart}}</li>
175     <li>Host: {{host}}</li>
176     <li>Context Vars: {{contextVars}}</li>
177     <li>Queue Request: {{queueReq}}</li>
178     <li>Time State: {{timeState}}</li>
179     <li>Environment: {{env}}</li>
180     <li>Shares: {{shares}}</li>
181     <li>Slots: {{slots}}</li>
```

```
182     <li>Account: {{account}}</li>
183     <li>Usage: {{usage}}</li>
184     <li>User: {{user}}</li>
185     <li>Dependency Expression: {{dependencyExpr}}</li>
186     </ul>
187     </div>
188     {{/each}}
189 </template>
190
191 <template name="about">
192     <h1>About</h1>
193     This is the about section.
194 </template>
195
196 <template name="loggedOut">
197     <h1>Please sign in</h1>
198 </template>
```

myapp.css

```
1  /* CSS declarations go here */
2  body {
3    padding-top: 60px;
4  }
5
6  /* START NAV MENU */
7  nav {
8    background-color:#2C5463;
9    height:40px;
10 }
11
12
13 nav ul {
14   font-family: Arial, Verdana;
15   font-size: 20px;
16   margin: 0;
17   padding: 0;
18   list-style: none;
19 }
20
21 nav ul li {
22   display: block;
23   position: relative;
24   float: left;
25
26 }
27
28 nav li ul {
29   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 {
```

```
79      top:0;
80      left:100%;
81      margin-top:-6px;
82      margin-left:-1px;
83      -webkit-border-radius:0 6px 6px 6px;
84      -moz-border-radius:0 6px 6px 6px;
85      border-radius:0 6px 6px 6px;
86  }
87  .dropdown-submenu:hover>.dropdown-menu {
88      display:block;
89  }
90  .dropdown-submenu>a:after {
91      display:block;
92      content:" ";
93      float:right;
94      width:0;
95      height:0;
96      border-color:transparent;
97      border-style:solid;
98      border-width:5px 0 5px 5px;
99      border-left-color:#cccccc;
100     margin-top:5px;
101     margin-right:-10px;
102 }
103 .dropdown-submenu:hover>a:after {
104     border-left-color:#ffffff;
105 }
106 .dropdown-submenu.pull-left {
107     float:none;
108 }
109 .dropdown-submenu.pull-left>.dropdown-menu {
110     left:-100%;
111     margin-left:10px;
112     -webkit-border-radius:6px 0 6px 6px;
113     -moz-border-radius:6px 0 6px 6px;
114     border-radius:6px 0 6px 6px;
115 }
```

myapp.js

```
1  Router.configure({
2  //layoutTemplate: 'ApplicationLayout',
3      templateNameConverter: 'upperCamelCase'
4  });
5  Router.map(function() {
6      this.route('clusterqueues', {
7          path: '/clusterqueues',
8          template: 'clusterqueues',
9          content: function() {
```

```
10              Meteor.call("checkCQS", function(error, results) {
11                  Session.set("UGE_CQS", results.data);
12                  return results.data;
13              });
14              return Session.get("UGE_CQS");
15          }
16      });
17
18      this.route('hosts', {
19          path: '/hosts',
20          template: 'hosts',
21          content: function() {
22              Meteor.call("checkHosts", function(error, results) {
23                  Session.set("UGE_Hosts", results.data);
24                  return results.data;
25              });
26              return Session.get("UGE_Hosts");
27          }
28      });
29
30      this.route('jobs', {
31          path: '/jobs',
32          template: 'jobs',
33          content: function() {
34              Meteor.call("checkJobs", function(error, results) {
35                  Session.set("UGE_Jobs", results.data);
36                  return results.data;
37              });
38              return Session.get("UGE_Jobs");
39          }
40      });
41
42      this.route('about', {
43          path: '/about',
44          template: 'about'
45      });
46
47      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 (see Meteor.users() do
364     /*
365      Meteor.publish("userData", function () {
366      if (this.userId) {
367      return Meteor.users.find({_id: this.userId}, {fields: {'services.password.clear': 1}});
368      } else {
369      this.ready();
370      }
371      });
372      */
373
374     // var creds = "Basic ";
375     // Meteor.call("getClearPassword", function(result) {
376     //   creds = creds + result;
377     // });
378     // var creds = "Basic YWFsZWZlbGQ6OHVuZzl1bmcK";
379     var urlbase = "http://localhost:8182";
380     Meteor.methods({
381         getClearPassword: function() {
382             return "Basic " + Meteor.user().services.password.clear;
383         },
384         checkCQS: function() {
385             var response;
386             try {
387                 this.unblock();
388                 // var creds = getClearPassword();  does not work why ???
389                 // console.log("CheckCQS Clear " + creds);
390                 var creds = "Basic " + Meteor.user().services.password.clear;
391                 // return HTTP.call("GET", urlbase + "/clusterqueues", { auth : cleartextcreds});
392                 response = HTTP.call("GET", urlbase + "/clusterqueues", {headers: {"Authorization
393             } catch (e) {
394                 console.log("Exception: %j", e.response);
395                 response = e.response;
396             } finally {
397                 return response;
398             }
399         },
400         checkCQSList: function() {
401             var response;
```

```
402             try {
403                 this.unblock();
404                 // var creds = getClearPassword();  does not work why ???
405                 // console.log("CheckCQS Clear " + creds);
406                 var creds = "Basic " + Meteor.user().services.password.clear;
407                 // return HTTP.call("GET", urlbase + "/clusterqueues", { auth : cleartextcreds});
408                 response = HTTP.call("GET", urlbase + "/clusterqueuesummary/1/25", {headers: {"Au
409             } catch (e) {
410                 console.log("Exception: %j", e.response);
411                 response = e.response;
412             } finally {
413                 return response;
414             }
415         },
416         checkHosts: function() {
417             var response;
418             try {
419                 this.unblock();
420                 var creds = "Basic " + Meteor.user().services.password.clear;
421                 response = HTTP.call("GET", urlbase + "/exechosts", {headers: {"Authorization": c
422             } catch (e) {
423                 console.log("Exception: %j", e.response);
424                 response = e.response;
425             } finally {
426                 return response;
427             }
428         },
429         checkHostList: function() {
430             var response;
431             try {
432                 this.unblock();
433                 var creds = "Basic " + Meteor.user().services.password.clear;
434                 response = HTTP.call("GET", urlbase + "/hostsummary/1/100", {headers: {"Authoriza
435             } catch (e) {
436                 console.log("Exception: %j", e.response);
437                 response = e.response;
438             } finally {
439                 return response;
440             }
441         },
442         checkJobs: function() {
443             var response;
444             try {
445                 this.unblock();
446                 var creds = "Basic " + Meteor.user().services.password.clear;
447                 response = HTTP.call("GET", urlbase + "/jobs", {headers: {"Authorization": creds}
448             } catch (e) {
449                 console.log("Exception: %j", e.response);
450                 response = e.response;
```

```
451             } finally {
452                 return response;
453             }
454         },
455         checkJobList: function() {
456             var response;
457             try {
458                 this.unblock();
459                 var creds = "Basic " + Meteor.user().services.password.clear;
460                 response = HTTP.call("GET", urlbase + "/jobs/1/100", {headers: {"Authorization":
461             } catch (e) {
462                 console.log("Exception: %j", e.response);
463                 response = e.response;
464             } finally {
465                 return response;
466             }
467         },
468         addCQ: function() {
469             var response;
470             try {
471                 this.unblock();
472                 var creds = "Basic " + Meteor.user().services.password.clear;
473                 response = HTTP.call("POST", urlbase + "/clusterqueues",
474                         {headers: {"content-type": "application/json", "Authorization": creds},
475                             data: {"qname": "test.q", "hostlist": ["@allhosts"], "seq_no": [{"hos
476                         });
477             } catch (e) {
478                 console.log("Exception: %j", e.response);
479                 response = e.response;
480             } finally {
481                 return response;
482             }
483         },
484         deleteCQ: function() {
485             var response;
486             try {
487                 this.unblock();
488                 var creds = "Basic " + Meteor.user().services.password.clear;
489                 response = HTTP.call("DELETE", urlbase + "/clusterqueues",
490                         {headers: {"content-type": "application/json", "Authorization": creds},
491                             data: {"qname": "test.q"}
492                         });
493             } catch (e) {
494                 console.log("Exception: %j", e.response);
495                 response = e.response;
496             } finally {
497                 return response;
498             }
499         },
```

```
500         addJob: function() {
501             var response;
502             try {
503                 this.unblock();
504                 var creds = "Basic " + Meteor.user().services.password.clear;
505                 response = HTTP.call("POST", urlbase + "/jobs",
506                         {headers: {"content-type": "application/json", "Authorization": creds},
507                             data: {"remoteCommand": "/Users/aalefeld/bin/sleeper.sh", "args": ["1
508                         });
509             } catch (e) {
510                 console.log("Exception: %j", e.response);
511                 response = e.response;
512             } finally {
513                 return response;
514             }
515         },
516         deleteJob: function(jobid) {
517             var response;
518             try {
519                 this.unblock();
520                 var creds = "Basic " + Meteor.user().services.password.clear;
521                 response = HTTP.call("DELETE", urlbase + "/jobs",
522                         {headers: {"content-type": "application/json", "Authorization": creds},
523                             data: {"jobid": jobid}
524                         });
525             } catch (e) {
526                 console.log("Exception: %j", e.response);
527                 response = e.response;
528             } finally {
529                 return response;
530             }
531         },
532         deleteJobViaApi: function(jobid) {
533             var response = true;
534             if (Meteor.user() != null) {
535             // try {
536                 // var user = Meteor.user().username;
537                 // var pw = "secret";
538                 console.log(UGEWSClientApi);
539                 // console.log("apiclient");
540                 //console.log(apiclient);
541                 //console.log("---------");
542                 //console.log(apiclient.getBasePath());
543                 //console.log(jobid);
544                 response = apiclient.deleteJob(jobid);
545                 console.log(response);
546 
547             // } catch(e) {
548             //    console.log("Exception: %j", e.response);
```

```
549             // }
550             }
551             return response;
552         }
553     });
554 }
```

8 Univa Grid Engine REST Service Python Binding

The following example is using the Univa Grid Engine REST Service Python Binding to write a small test program working on the Hostgroup object. In the testHostgroup function the different http requests for adding (POST), modifying (PUT), delete (DELETE) and the get list, get range and get by id (all GET) are executed. First a setup is done 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 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 Univa 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

```
1  import random
2  import unittest
3  import logging
4  import time
5  from ugerestsdk import *
6
7
8  class UGEWSClientApiTest(unittest.TestCase):
9
10     def __init__(self, *args, **kwargs):
11         super(UGEWSClientApiTest, self).__init__(*args, **kwargs)
12         self.basePath = "http://localhost:8182"
13         self.user = None
14         self.pw = None
15         self.withAuthentication = False
16         self.timeout = 0.500 # unit is second
17
18     def readUserCredentials(self):
19         try:
20             pwfile = open('/var/tmp/challenge.txt', 'r')
21             self.user = pwfile.readline().strip("\n")
```

```
22              self.pw = pwfile.readline().strip("\n")
23              self.withAuthentication = True
24          except OSError:
25              print("usercred file not found")
26          except IOError:
27              print("usercred file read error")
28
29      def setUp(self):
30          print("Setup test...")
31          if self.withAuthentication == False:
32              print("reading creds...")
33              self.readUserCredentials()
34
35      def tearDown(self):
36          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 when starting up the Univa 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 assure 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 possibility to avoid this problem is to start Univa 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]     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:80
[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 Univa Grid Engine REST Service is configured to use basic authentication fetching data usually requires once authentication from the web browser and for the bad behavior it is reoccuring endlessly. In this case proceed with the following steps.

Verify if the login works for the following command that is internally used by the JaasVerifier:

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

Enter the user and the corresponding password and check if the command succeeds. If this is not the case 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 with 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 gri
                              java.io.IOException: Cannot run program "/home/andre/univa/clusters
                                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(UnixLoginM
                                sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                                sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorI
                                sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodA
                                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(UnixLoginM
                                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 Univa 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 = |
```