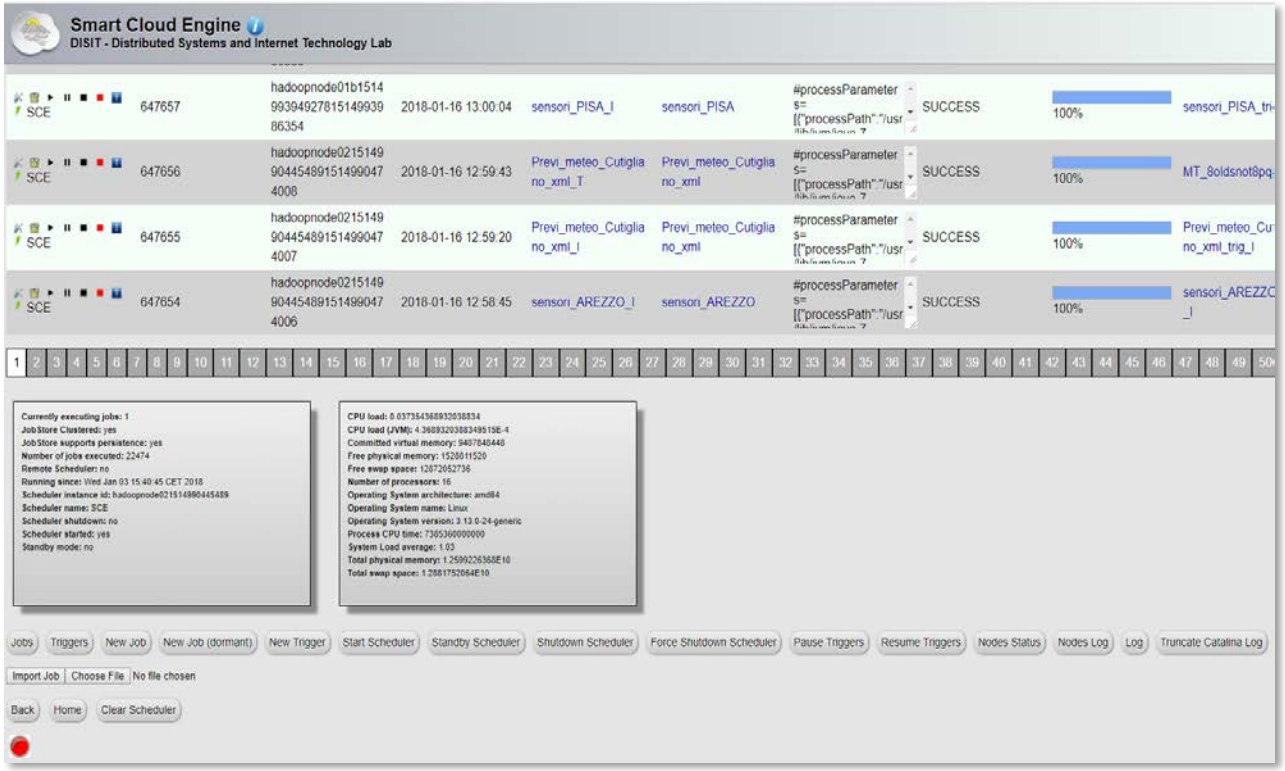
DISCES User Manual

Table of Contents

1. Introduction

A typical major requirement in a Smart City/Cloud environment consists of an engine for distributed task scheduling. In this context, DISIT lab developed an efficient solution for Smart management and scheduling, Distributed SCE Scheduler, DISCES. DISCES consists of a set of distributed instances of running agents performing concurrent tasks. DISCES engine with cluster functionality allows adding distributed nodes and defining jobs, without service downtime.



The DISCES is a core component:

- can be deployed on one or many Virtual Machines to create a distributed scheduler, where the single node automatically takes their jobs independently without any scheduling central services. The architecture is fully scalable and fault tolerant;
- can be used for cloud management, for smart city engine, and as cultural content management system;
- can be connected to a knowledge base (RDF stores), MySQL databases, NoSQL databases for gathering data to take decisions and for in/out processing;
- activate any inside or outside/detached processes, they can be direct executable on the operating system as well as called from REST invocations. Processes and can be classical ETL, verification and validation processes, Hadoop management, SLA management, etc.;
- where each scheduling job includes a name and a related group, a fire instance id, a repeat count, a start and an end time, a job data map, a job status (i.e., fired, running, completed, success, failed), and some associated triggers with their metadata (i.e., name and group, period and priority of execution);
- supports both concurrent a non-concurrent scheme for jobs, allows a direct monitoring of each job activity with a push interface, reporting the current status of the job, and the number of successes or failures in the last day or week, with relative percentages;

- variety of the hardware at disposal and the jobs to be scheduled require best practices for adaptive job scheduling. For example, a reconfiguration process written for a particular CPU architecture should be bounded to run on a certain set of scheduler nodes only; nodes with high CPU load could reject the execution of further tasks, until their computation capacity is fully restored at acceptable levels; more in general, there could be the need to assign certain selected tasks only to nodes with a certain level of processing capacity;
- mechanisms that allow scheduling tasks in a recursive way, based on the results obtained in previous tasks. For example, a reconfiguration strategy consisting of various steps could require taking different actions based on dynamical parameters evaluated at runtime.
- allows adaptive job execution (e.g., based on the physical or the logical status of the host), and conditional job execution, supporting both system and REST calls. The user can build an arbitrary number of job conditions that must be satisfied to trigger a new job or a set of jobs, or can even specify multiple email recipients to be notified in case of a particular jobs result. By combining an arbitrary number of conditions, it is possible to define complex flow chart job execution schemes, for the management of different cloud scenarios. A trigger associated to a conditional job execution is created at runtime and it is deleted upon completion. It is possible to define physical or virtual constraints (e.g., CPU type, number of CPU cores, operating system name and version, system load average, committed virtual memory, total and free physical memory, free swap space, CPU load, IP address), that bind a job to a particular scheduler node. Smart cloud best policies require services and tools to collect and analyze huge amount of data coming from different sources at periodic intervals. Virtual machines typically consist of hundreds of services and related metrics to be checked;
- SLAs often define bounds related to services or groups of services that consist of many applications, configurations, processing capacity or resources utilization. It is worth noting that collecting such a high number of data could lead to unmanageable systems, even if adopting the best practices of DMBS management or clustering, in a short period of time. For this purpose, it includes support for NoSQL, with the aim of allowing high performance in data retrieving and processing;
- includes event reporting and logging services, for a direct monitoring of the smart cloud infrastructure and the activity status of every cluster node, and notifications about the critical status of a system or service (e.g., sending of emails). Notifications can be conditioned or not to the results of execution;
- includes a web interface that allows monitoring the status of the cloud platform (i.e., hosts, virtual machines, applications, metrics, alerts and network interfaces), with details about the compliance of metrics with respect of the SLA, and a summary view of the global status of the cluster nodes (e.g., memory, disk, swap);
- provides graphs of all the relevant metrics to perform deep data analysis;
- performs SPARQL queries to the Knowledge Base to check the coherence of the services with respect to SLA and eventually instructs with a REST call the CM to take reconfiguration actions (e.g., increment of storage, computational resources or bandwidth);
- includes a logging service for registering every event related to the monitored services, and allows adjusting checking periods for each service;
- allows to define policies to apply in case of misfired events (e.g., reschedule a job with existing or remaining job count), and allows to produce detailed graphs for every metric (grouped per VM or not), with customizable time intervals;
- reports for each metric the total amount of times it was found to be out of scale, with respect to the total number of performed checks. Logged metrics report the list of SLA violations occurred in the selected time slot, with relevant data (e.g., the time at which the

violation occurred, the name of the metric, the registered value, the threshold, and the related business configuration, virtual machine and SLA);

- Reports a global view of the cluster status and detailed views of each node. It is possible to monitor parameters such as last job execution time, number of jobs processed since the last restart, CPU load and utilization, time of last check, free physical memory and the total consumed computational capacity of the cluster (e.g., total CPU utilization, total capacity in terms of GHz and percentage of consumed capacity, total and free memory);
- DISCES is released in open source for the web user interface part.

2. User Interface

The web interface of DISCES offers various commands to monitor the scheduler.

The menu options at disposal are:

- *Jobs*, view the list of jobs;
- *Triggers*, view the list of triggers;
- *New Job*, create a new job with an associated trigger;
- *New Job (dormant)*, create a new job with no associated trigger;
- *New Trigger*, create a new trigger;
- *Start Scheduler*, start the scheduler;
- *Standby Scheduler*, put the scheduler in standby;
- *Shutdown Scheduler*, shutdown the scheduler;
- *Force Shutdown Scheduler*, forcibly shutdown the scheduler;
- *Pause Triggers*, pause all the triggers;
- *Resume Triggers*, resume all the triggers;
- *Nodes Status*, view the nodes status list;
- *Nodes Log*, view the nodes log;
- *Log*, view the log;
- *Truncate Catalina Log*, truncate the server's log;
- *Clear Scheduler*, delete all jobs and triggers from the Scheduler;
- *Import Job*, import a job from an external file;
- *Export Job,* export job to an external file;
- *Red button*, perform an automatic refresh of the page at regular intervals.

Fig. 1 - Menu buttons in the Home Page of DISCES

3. Jobs

In this page you can view the list of jobs installed on DISCES. Each job has the following fields:

- SCHED NAME, the name of the DISCES instance;
- JOB NAME, the name of the job;
- JOB GROUP, the name of the group;
- DESCRIPTION, the description of the job;
- FAILED 1D, the number of job failed executions, in the last day;
- SUCCESS 1D, the number of job successful executions, in the last day;
- FAILED 1D PERC, the percentage of job failed executions, in the last day;
- SUCCESS 1D PERC, the percentage of job successful executions, in the last day;
- FAILED 7D, the number of job failed executions, in the last week;
- SUCCESS 7D, the number of job successful executions, in the last week;
- FAILED 7D PERC, the percentage of job failed executions, in the last week;
- SUCCESS 7D PERC, the percentage of job successful executions, in the last week;
- NEXT FIRE TIME, the next time when the job will be executed;
- PREV FIRE TIME, the last time when the job executed;
- JOB CLASS NAME, the type of job;
- IS DURABLE, if a job is non-durable, it is automatically deleted from the scheduler once there are no longer any active triggers associated with it. In other words, non-durable jobs have a life span bounded by the existence of its triggers;
- IS NONCONCURRENT, tell DISCES not to execute multiple instances of a given job definition (that refers to the given job class) concurrently. The constraint is based upon an instance definition (JobDetail), not on instances of the job class;
- IS UPDATE DATA, enable JobDataMap to update data while execution and allows to re-store it after execution;
- REQUESTS RECOVERY, if a job "requests recovery", and it is executing during the time of a 'hard shutdown' of the DISCES (i.e. the process it is running within crashes, or the machine is shut off), then it is re-executed when the scheduler is started again;
- JOB DATA, list the data associated with the job.

Fig. 2 – Jobs' list

4. Triggers

In this page you can view the list of triggers installed on DISCES. Each trigger has the following fields:

- SCHED NAME, the name of the DISCES instance;
- TRIGGER NAME, the name of the trigger;
- TRIGGER GROUP, the group of the trigger;
- JOB NAME, the name of the job;
- JOB GROUP, the name of the group;
- DESCRIPTION, the description of the job;
- NEXT FIRE TIME, the next time when the trigger will be fired;
- PREV FIRE TIME, the last time when the trigger fired;
- PRIORITY, the priority of the trigger;
- TRIGGER STATE, the state of the trigger;
- TRIGGER TYPE, the type of the trigger;
- START TIME, the start time of the trigger;
- END TIME, the end time of the trigger;
- CALENDAR NAME, not used;
- MISFIRE INSTR, the instruction to be executed in case of misfire;
- JOB DATA, lists the data associated with the job.
- REPEAT COUNT, report how many time the trigger must execute (-1: forever);
- REPEAT INTERVAL, time interval to repeat the trigger;
- TIMES TRIGGERED, report how many time the trigger triggered.

SCHED NAME	TRIGGER NAME	TRIGGER GROUP	JOB NAME	JOB GROUP	DESCRIPTION	NEXT FIRE TIME	PREV FIRE TIME	PRIORITY	TRIGGER STATE	TRIGGER TYPE
SCE	08623d91-78f7-4c02-9421-23f55e14634a	83ee877a-7869-4d10-bba7-a27ced8e6c7d	Parcheggi_Grosseto_RT	Parcheggi		2018-01-18 08:56:00	2018-01-18 08:41:00	5	WAITING	SIMPLE
SCE	38cccd79-0741-499a-8b2c-53962fb5e3f5	1001e069-86ff-4bed-be99-9787564b322f	Arpat_DailyDispatch_OZONO_I	Arpat		2018-01-19 13:00:00	1970-01-01 01:00:00	5	WAITING	SIMPLE
SCE	5fbdce2b-4d92-4abc-a24e-4c12e7831874	TPL_trigger	TPL_NM_Bus_acvbus_I	TPL	TPL_NM_Bus_acvbus_trigger	2018-01-18 09:30:00	2018-01-17 09:30:00	5	WAITING	SIMPLE
SCE	88b91993-9cb2-46e5-a10a-126f23993d61	d6043bcf-22aa-4311-878d-274fd0435a20	TPL_NM_Bus_amvbus_I	TPL		2018-01-19 06:00:00	2018-01-18 06:00:00	5	WAITING	SIMPLE
SCE	Arpat_DailyDispatch_ARIA_I_trigger	ARPAT_trigger	Arpat_DailyDispatch_ARIA_I	ARPAT		2018-01-18 14:00:00	2018-01-18 08:00:00	5	WAITING	SIMPLE
SCE	Arpat_DailyDispatch_ICQA_trigger	ARPAT_trigger	Arpat_DailyDispatch_ICQA_I	ARPAT		2018-01-18 12:33:00	2018-01-18 06:33:00	5	WAITING	SIMPLE
SCE	avm_linea17_trig_I	AVM_trig	avm_linea17_I	avm_linea17		2018-11-05 10:02:31	1970-01-01 01:00:00	5	WAITING	SIMPLE
SCE	avm_linea31_trig_I	AVM_trig	avm_linea31_I	avm_linea31		2018-07-07 07:10:00	1970-01-01 01:00:00	5	WAITING	SIMPLE
SCE	avm_linea4_trig_I	AVM_trig	avm_linea4_I	avm_linea4		2018-04-28 05:24:04	1970-01-01 01:00:00	5	WAITING	SIMPLE
SCE	avm_linea6_trig_I	AVM_trig	avm_linea6_I	avm_linea6		2018-11-05 10:02:32	1970-01-01 01:00:00	5	WAITING	SIMPLE
SCE	Bike_Pisa_trigger	Bike_trigger	Bike_Pisa	Bike		2018-01-18 08:45:00	2018-01-18 08:30:00	5	WAITING	SIMPLE
SCE	Bike_Siena_trigger	Bike_trigger	Bike_Siena	Bike		2018-01-18 08:45:00	2018-01-18 08:30:00	5	WAITING	SIMPLE
SCE	check_RT	check_RT	check_RT	check_RT		2018-01-18 09:27:46	2018-01-17 09:27:46	5	WAITING	SIMPLE
SCE	CKAN_POI_trigger	CKAN_trigger	CKAN_POI	CKAN		2018-01-18 09:44:00	2018-01-18 08:44:00	5	WAITING	SIMPLE

Fig. 3 – Triggers' list

5. New Job

Click on this button to create a new job click. A new view will open with these details to fill up:

5.1 Job Data

- *Store Durably (checkbox)*, set whether or not the Job should remain stored after it is orphaned. If a job is non-durable, it is automatically deleted from the scheduler once there are no longer any active triggers associated with it. In other words, non-durable jobs have a life span bounded by the existence of its triggers;
- *Non-concurrent (checkbox)*, set the job to disallow to execute concurrently (new triggers that occur before the completion of the current running job will be delayed);
- *Request recovery (checkbox)*, in clustering mode, this parameter must be set to true to ensure job fail-over. If a job 'requests recovery', and it is executing during the time of a 'hard shutdown' of the scheduler (i.e. the process it is running within crashes, or the machine is shut off), then it is re-executed when the scheduler is started again;
- *Job Name*, the name of the job;
- *Job Group*, the name of the group;
- *Description*, the description of the job;
- *Job Type*, the type of the job;
- *URL*, the URL to be called, in case of a REST type job;
- *Process Path*, the path of the process to be executed, in case of a ProcessExecutor type job;

5.2 Trigger Data

- *Start At*, the starting time of the trigger;
- *End At*, the end time of the trigger;
- *Calendar Name*, not used;
- *Trigger Name*, the name of the trigger;
- *Trigger Group*, the group of the trigger;
- *Priority*, the priority of the trigger;
- *Repeat Count*, specify how many times the trigger has to be fired (0 = forever);
- *Interval (s)*, time interval to fire the trigger;
- *Misfire Instruction*, misfire instruction in case of a misfiring;
- *Email*, email where to send a notification upon completion of the job;

5.3 Add Data Map

This section allows the user to add customized parameters (must be coded in the DISCES project, though).

One useful parameter already at disposal is job's timeout (note that this function could work only for ProcessExecutor type jobs). Example of usage:

Click on *Add Data Map* and enter "#jobTimeout" as the key, and a numerical interger (seconds) for the value.

5.4 Add Next Job

This section allows the user to define job(s) to started upon completion of the actual job. You must select a boolean operator and fill the result to be checked upon job completion, and then the next job to be fired, that must previously exist. You can alternatively choose not to trigger a new job, but to notify some users by email; in that case it is sufficient to write the emails (comma separated) in the job name field, and a blank space in the job group field.

5.5 Add Process Parameter

These are additional arguments that are concatenated to the process path (for ProcessExecutor type jobs). The key field is only a label, the value field contains the actual value to be concatenated to the process path.

5.6 Job Constraints

You can define job constraints, such as OS Architecture, Available Processors, OS Names and so on. If these conditions are matched on the DISCES node that wants to execute the job, then the job is effectively put in execution, otherwise it is skipped and made available to be executed y another DISCES node.

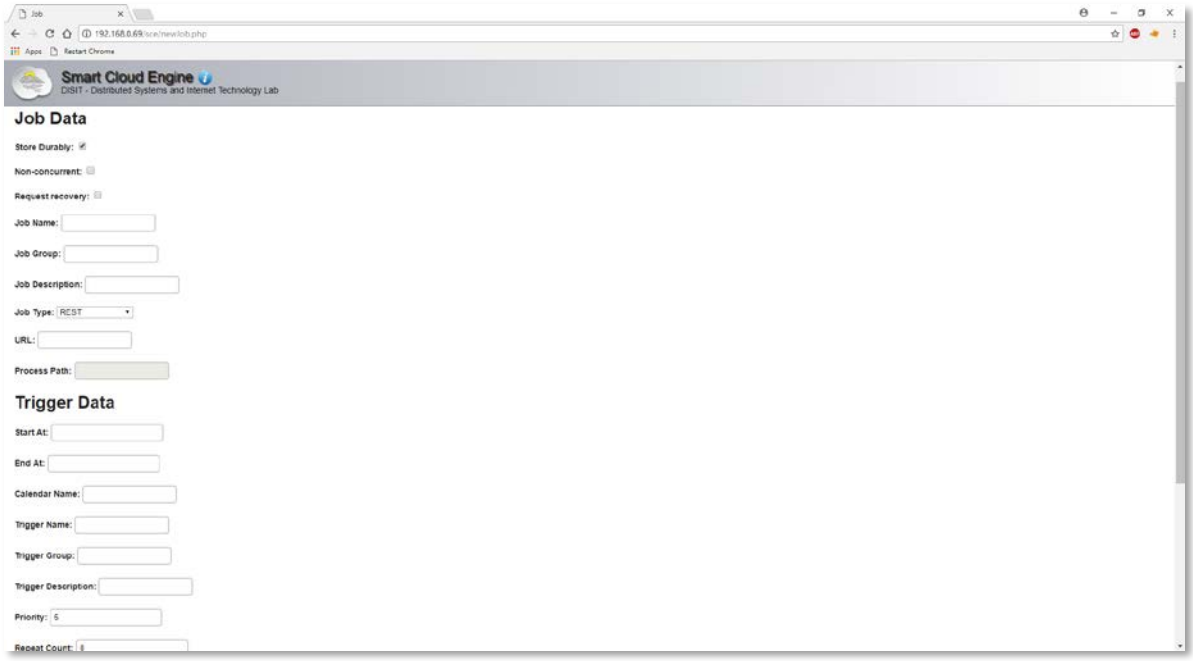


Fig. 4 – New Job view

6. New Job (dormant)

It has the same functionality as the New Job page, expect for the fact that in this case the resulting job created has no associated trigger.

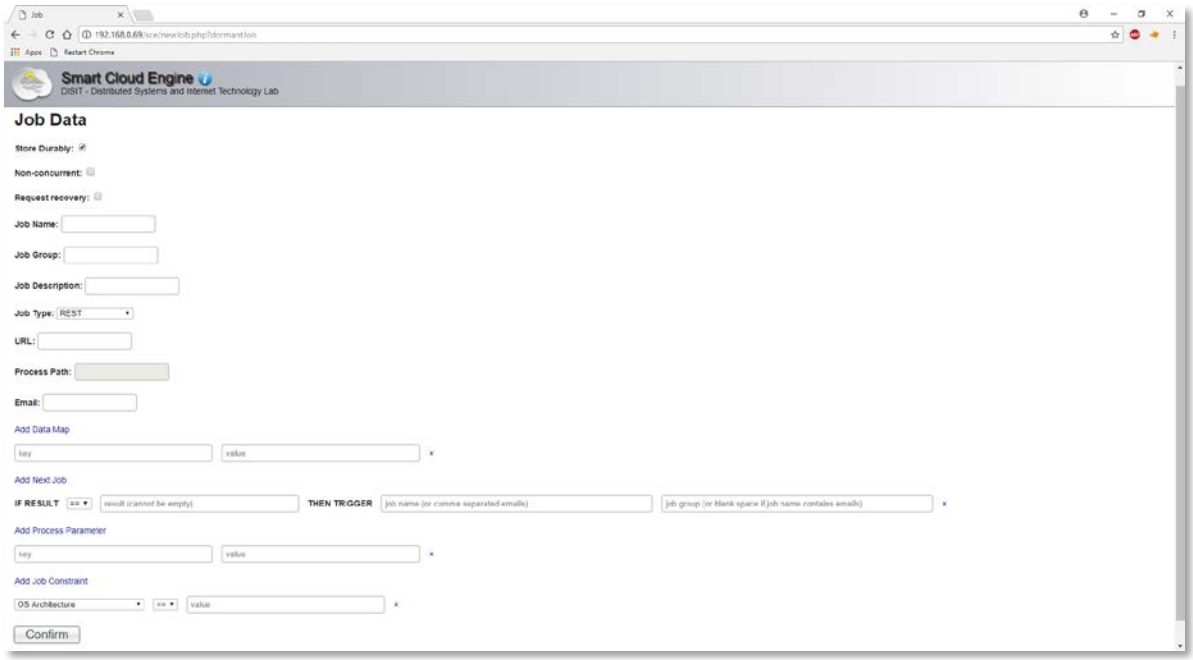


Fig. 5 – New Job (dormant) view

7. Start Scheduler

Click this button to start the DISCES node. Note that DISCES automatically runs at Tomcat startup. In any case this button only tries to start the actual DISCES node.

8. Shutdown Scheduler

Click this button to shut down the DISCES node. Note that the process will wait for the completion of every running job before shutting down the DISCES node. In any case this button only tries to shut down the actual DISCES node.

9. Force Shutdown Scheduler

Click this button to forcibly shut down the DISCES node. Note that the process will not wait for the completion of every running job before shutting down the DISCES node. In any case this button only tries to shut down the actual DISCES node.

10. Pause Triggers

Click this button to pause all the triggers of DISCES nodes.

11. Resume Triggers

Click this button to resume all the triggers of DISCES nodes.

12. Nodes Status

Click this button to view the most recent status of every DISCES node. Each node has the following fields:

- ID, incremental id of the DISCES node;
- DATE, date of the check;
- IP ADDRESS, ip address of the DISCES node;
- SCHEDULER INSTANCE ID, DISCES node id;
- CPU LOAD, cpu load;
- FREE PHYSICAL MEMORY, free physical memory;
- JOBS EXECUTED, number of jobs executed by DISCES node since last restart;
- SCHEDULER NAME, scheduler's name;
- RUNNING SINCE, starting date of DISCES node;
- CLUSTERED, if the DISCES node is clustered;
- PERSISTENCE, if DISCES node supports data persistence;
- REMOTE SCHEDULER, if DISCES node supports remote scheduling;
- CURRENTLY EXECUTING JOBS, number of currently executing jobs on the DISCES node;
- CPU LOAD JVM, cpu load of the Java Virtual Machine of the DISCES node;
- SYSTEM LOAD AVERAGE, system load average of the DISCES node;
- OPERATING SYSTEM VERSION, operating system version of the DISCES node;
- COMMITTED VIRTUAL MEMORY, committed virtual memory of the DISCES node;
- OPERATING SYSTEM NAME, operating system name of the DISCES node;
- FREE SWAP SPACE, free swap space of the DISCES node;
- PROCESS CPU TIME, process cpu time of the DISCES node;
- TOTAL PHYSICAL MEMORY, total physical memory of the DISCES node;
- NUMBER OF PROCESSORS, number of processors of the DISCES node;
- OPERATING SYSTEM ARCHITECTURE, operating system architecture of the DISCES node;

- TOTAL SWAP SPACE, total swap space of the DISCES node;
- IS SCHEDULER STANDBY, if the DISCES node is in standby;
- IS SCHEDULER SHUTDOWN, if the DISCES node is shutdown;
- IS SCHEDULER STARTED, if DISCES node is started;
- TOTAL DISK SPACE, total disk space of DISCES node;
- UNALLOCATED DISK SPACE, unallocated space of DISCES node;
- USABLE DISK SPACE, usable disk space of DISCES node.

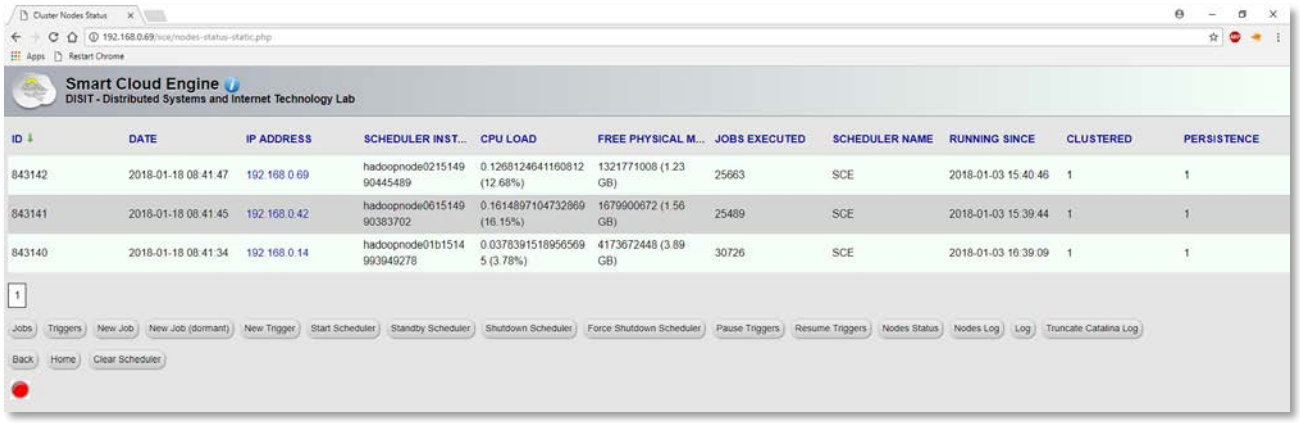


Fig. 6 – Nodes status view

13. Nodes Log

Click this button to view the nodes log list. Each node has the same fields as the previous menu button, for each date/time.

Fig. 7 – Nodes log view

14. Log

Click this button to view the activity log of the DISCES nodes. Each event has the following fields:

- ID, incremental id of the DISCES event;
- DATE, date of the event;
- JOB NAME, the name of the job;
- JOB GROUP, the name of the group;
- JOB DATA, list the data associated with the job.
- TRIGGER NAME, the name of the trigger;
- TRIGGER GROUP, the group of the trigger;
- STATUS, the status of the event;
- RESULT, the result of the job;
- PREV FIRE TIME, the last time when the job executed;
- NEXT FIRE TIME, the next time when the job will be executed;
- REFIRE COUNT, how many times the job will be executed;
- SCHEDULER INSTANCE ID, DISCES node id;
- SCHEDULER NAME, scheduler's name;
- FIRE INSTANCE ID, fire instance id;
- IP ADDRESS, ip address of the DISCES node;
- LOGGER, the logger's class;
- LEVEL, the log's level;
- MESSAGE, the message produced by the DISCES node upon job completion.

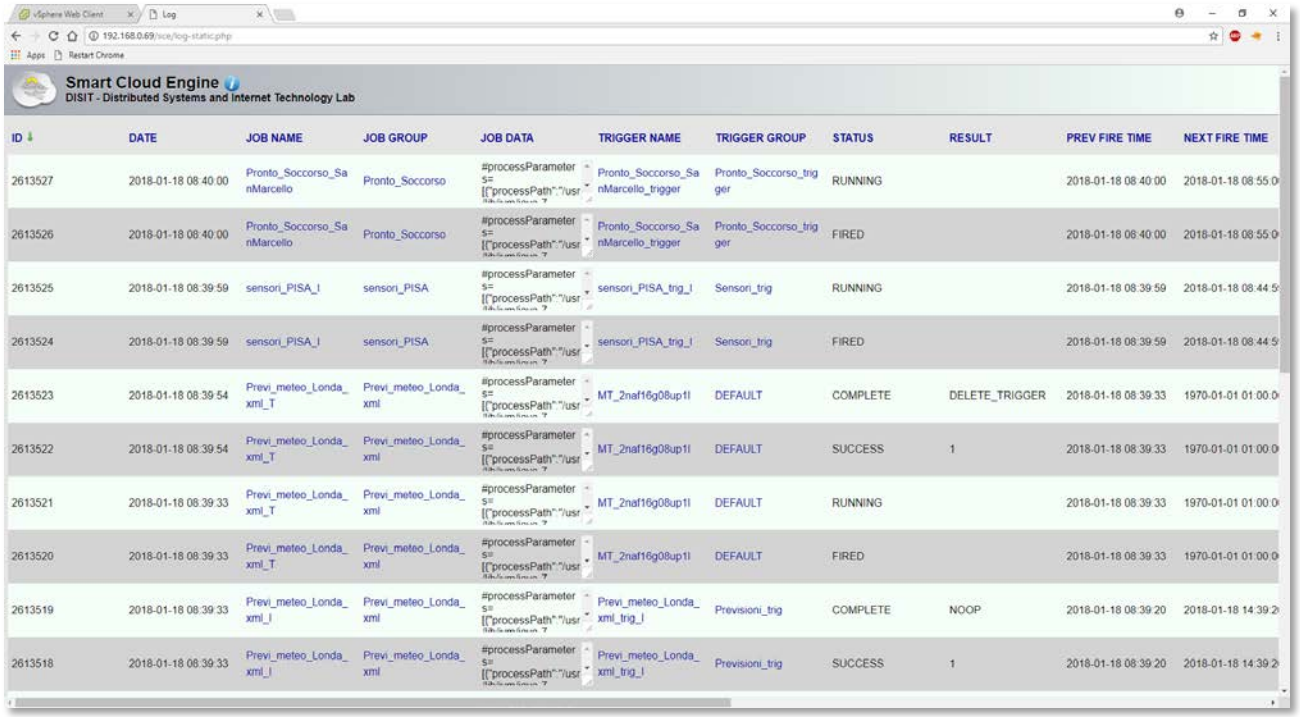


Fig. 8 – Log view

15. Truncate Catalina Log

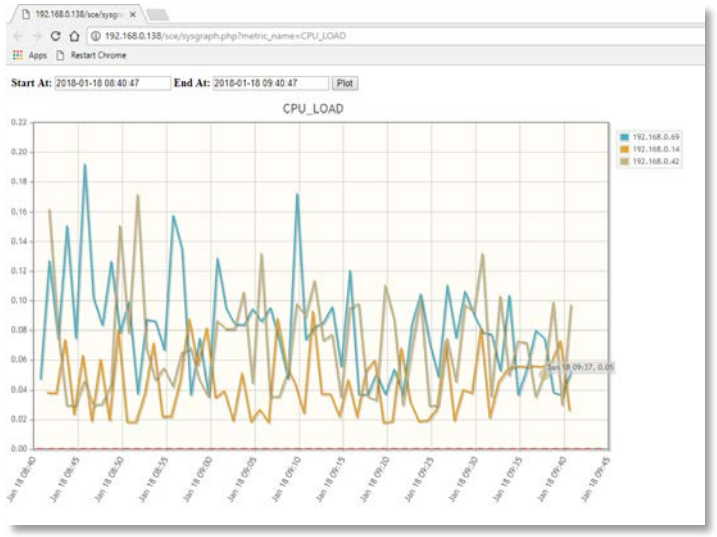
This button removes all jobs and triggers from the DISCES nodes.

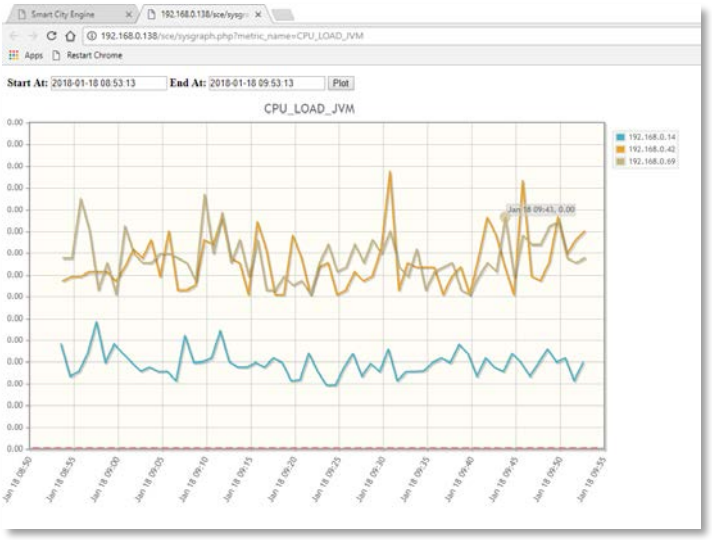
16. Notes

When present in the page, the red button at the buttom automatically refresh the page, updating all the field values.

17. Node Statistics

The statistics page allows the user to visualize various metrics, related to the DISCES node workload. For example, the following figure reports the historical CPU usage, free physical memory, CPU load JVM.





18. API

This section reports the REST API provided by DISCES, with some example of usage.

Exposed Protocol

The module exposes a REST interface on HTTP protocol for access to scheduling functions.

Nome: checkExistJob	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to check the existence of a job on the scheduler. Mode: GET, POST	
Required parameters	
id	The id of the required action
jobName	The job name
jobGroup	The job group
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22checkExistJob%22,%22jobName%22:%22job1%22,%22jobGroup%22:%22group1%22}	

JSON Result:

{"0":["response"],"1":"false"}

Nome: checkExistTrigger
URL: http://hostname:8080/SmartCloudEngine

Description

API to check the existence of a trigger on the scheduler.

Mode: GET, POST

Required Parameters

id	The id of the required action
triggerName	The job name
triggerGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22checkExistTrigger%22,%22jobName%22:%22trigger1%22,%22triggerGroup%22:%22group1%22}

JSON Result:

{"0":["response"],"1":"false"}

Name: clear, clearScheduler

URL: http://hostname:8080/SmartCloudEngine

Description

API to delete all jobs and triggers on the scheduler

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22clear%22}

Result:

true

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: deleteCalendar

URL: http://hostname:8080/SmartCloudEngine

Description

API to delete a calendar on the scheduler

Mode: GET, POST

Required Parameters

id	The id of the required action
calendarName	The calendar name

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22 calendarName %22:%22 %22}

Result:

true

Name: deleteJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to delete a job from the scheduler, and all associated triggers

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22 jobName%22:%22job1%22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: getCurrentlyExecutingJobs

URL: http://hostname:8080/SmartCloudEngine

Description
API to ask for a list of running jobs Mode: GET, POST

Required Parameters	
id	The id of the required action
jobName	The job name
jobGroup	The job group

Example
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

JSON Result:

{"0":["response"],"1":"{\"0\":[\"job1\",\"group1\",\"Sat Mar 12 14:07:44 CET 2016\"]}"}

Name: getTriggersOfJob

URL: http://hostname:8080/SmartCloudEngine

Description
API to ask for a job trigger list Mode: GET, POST

Required Parameters	
id	The id of the required action
jobName	The job name
jobGroup	The job group

Example
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

JSON Result:

{"0":["response"],"1":"{\"0\":[Ljava.lang.String;@2257088e}"}

Name: getJobDetail

URL: http://hostname:8080/SmartCloudEngine

Description	
API to ask for job details Mode: GET, POST	
Required Parameters	
id	The id of the required action
jobName	The job name
jobGroup	The job group
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}	

JSON Result:

{"#isNonConcurrent":"[{\"processPath\":\"\\\/opt\\\/jdk1.8.0_51\\\/bin\\\/java\"},{\"cp\":\"-classpath\"},{\"lib\":\":\\\/var\\\/www\\\/html\\\/sce\\\/test\\\/lib\\\/*\"},{\"InsertTweets\":\"inserttweets.InsertTweets\"},{\"insertTweets.properties\":\"\\\/var\\\/www\\\/html\\\/sce\\\/test\\\/lib\\\/insertTweets.properties\"}]"}

Name: getJobGroupNames	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to ask for a list of groups Mode: GET, POST	
Required Parameters	
id	The id of the required action
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}	

JSON Result:

{"0":"InsertTweetsRecommender","1":"TwitterVigilanceIndexing"}

Name: getPausedTriggerGroups
URL: http://hostname:8080/SmartCloudEngine
Description

API to ask for a list of paused triggers

Mode: GET, POST

Required Parameters	
id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getPausedTriggerGroups%22}

JSON Result:

{"0":"InsertTweetsRecommender","1":"TwitterVigilanceIndexing"}

Name: getSchedulerInstanceId

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for the scheduler's instance id

Mode: GET, POST

Required Parameters	
id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

Result:

debian1456734408633

Name: getSchedulerName

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for the scheduler's instance name

Mode: GET, POST

Required Parameters	
id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getSchedulerName%22}

Result:

SCE

Name: getTriggerGroupNames

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a list of all triggers

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getTriggerGroupNames%22}

JSON Result:

{"0":"9cc52dfb-8360-4f75-9ed2-1c0c61530f46","1":"4b2cf714-cd6c-4265-8b2e-702a80150736"}

Name: getTriggerKeys

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a list of all triggers in a group

Mode: GET, POST

Required Parameters

id	The id of the required action
triggerGroup	The group name

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getTriggerKeys%22,%22triggerGroup%22:%22group1%22}

JSON Result:

{"0":"9cc52dfb-8360-4f75-9ed2-1c0c61530f46","1":"4b2cf714-cd6c-4265-8b2e-702a80150736"}

Name: getTriggersOfJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a list of all triggers of a job

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getTriggersOfJob%22,%22jobName%22:%22job1%22,%22jobGroup%22:%22group1%22}

JSON Result:

{"0":["response"],"1":"{\"0\":[Ljava.lang.String;@4710c620}"}

Name: getTriggerState

URL: http://hostname:8080/SmartCloudEngine

Description

API to request a trigger status

Mode: GET, POST

Required Parameters

id	The id of the required action
triggerName	The job name
triggerGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getTriggerState%22,%22triggerName%22:%22trigger1%22,%22triggerGroup%22:%22group1%22}

Result:

NONE

Name: interruptJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to request the interruption of all instances of a job

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22jobName%22:%22job1%22,%22jobGroup%22:%22group1%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: interruptJobInstance

URL: http://hostname:8080/SmartCloudEngine

Description

API to request the interruption of an instance of a job

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group
fireInstanceId	The job's instance id

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22jobName%22:%22job1%22,%22jobGroup%22:%22group1%22, %22fireInstanceId%22:%22debian145673440863314567344102 34%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: interruptJobs

URL: http://hostname:8080/SmartCloudEngine

Description

API to request the interruption of all running jobs

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22interruptJobs%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: interruptFireInstanceId

URL: http://hostname:8080/SmartCloudEngine

Description

API to request the interruption of an instance of a job

Mode: GET, POST

Required Parameters

id	The id of the required action
fireInstanceId	The job's instance id

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22interruptFireInstanceId%22,%22fireInstanceId%22:%22debian145673440863314567344102 34%22}

Result:

true

Name: isInStandbyMode

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask if the scheduler is on standby

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22isInStandbyMode%22}

JSON Result:

{"0":["response"],"1":"false"}

Name: isShutdown

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask if the scheduler is shutdown

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22isShutdown%22}

JSON Result:

{"0":["response"],"1":"false"}

Name: isStarted

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask if the scheduler is running

Mode: GET, POST

Required Parameters	
id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22isStarted%22}

JSON Result:

{"0":["response"],"1":"true"}

Name

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask you to pause all triggers (including future triggers)

Mode: GET, POST

Required Parameters	
id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22pauseAll%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: pauseJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to prompt you to pause a job

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22pauseJob%22,%22jobName%22:%22job1%22,%22jobGroup%22:%22group1%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: pauseJobs

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask you to pause a group's jobs

Mode: GET, POST

Required Parameters

id	The id of the required action
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

Result:

true

Name: pauseTrigger

URL: http://hostname:8080/SmartCloudEngine

Description

API to prompt you to pause a trigger

Mode: GET, POST

Required Parameters

id	The id of the required action
triggerName	The trigger name
triggerGroup	The trigger group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22%,%22triggerName%22:%22 %22,%22triggerGroup%22:%22 %22}

Result:

{"0":["response"],"1":"true"}

Name: pauseTriggers

URL: http://hostname:8080/SmartCloudEngine

Description

API to prompt you to pause triggers in a group

Mode: GET, POST

Required Parameters

id	The id of the required action
groupName	The group name

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22groupName%22:%22 %22}

Result:

true

Name: rescheduleJob

URL: http://hostname:8080/SmartCloudEngine

Description

API for requesting job rescheduling

Mode: GET, POST

Required Parameters

id	The id of the required action
withIdentityNameGroup	The job and group names separated by a dot (.)

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22with IdentityNameGroup%22:%22job1. %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: resumeAll

URL: http://hostname:8080/SmartCloudEngine

Description

API to request resume of all jobs and triggers

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: resumeJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to request a job resume

Mode: GET, POST

Required Parameters

id	The id of the required action

jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: resumeJobs

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a group's job resume

Mode: GET, POST

Required Parameters

id	The id of the required action
groupName	The group name

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22groupName%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: resumeTrigger

URL: http://hostname:8080/SmartCloudEngine

Description

API to request a trigger resume

Mode: GET, POST

Required Parameters

id	The id of the required action
triggerName	The trigger name
triggerGroup	The trigger group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22triggerName%22:%22 %22,%22triggerGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: resumeTriggers

URL: http://hostname:8080/SmartCloudEngine

Description

API to request resume of triggers of a group

Mode: GET, POST

Required Parameters

id	The id of the required action
groupName	The group name

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22groupName%22:%22 %22}

Result:

true

Name: shutdownScheduler

URL: http://hostname:8080/SmartCloudEngine

Description

API for requesting scheduler shutdowns

Mode: GET, POST

Required Parameters

id	The id of the required action
waitForJobsToComplete	If true, the scheduler is shutdown when jobs are completed, otherwise it will be shutdown immediately.

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%2
2waitForJobsToComplete%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: standbyScheduler

URL: http://hostname:8080/SmartCloudEngine

Description

API to request standby of the scheduler

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22standbyScheduler%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: startScheduler

URL: http://hostname:8080/SmartCloudEngine

Description

API for requesting the scheduler start

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22startScheduler%22}

JSON Result:

{"0":["response"],"1":"true"}

Name: startDelayed

URL: http://hostname:8080/SmartCloudEngine

Description

API for requesting delayed scheduler start

Mode: GET, POST

Required Parameters

id	The id of the required action
seconds	The seconds of delay

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22,%22seconds%22:%22 %22}

Result:

true

Name: truncateCatalinaLog

URL: http://hostname:8080/SmartCloudEngine

Description

API to request for Catalina log truncate (Tomcat)

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: triggerJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to request for a job execution

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: unscheduleJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to request removal of a job from the scheduler

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: listJobTriggers

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a job trigger list

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Esempio

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["Trigger Name","Trigger Group","Calendar Name","Description","End Time","Final Fire Time","Misfire Instruction","Next Fire Time","Previous Fire Time","Priority","Start Time","May Fire Again"],"1":["7e369086-851f-4191-927f-229327d0f44c","4b2cf714-cd6c-4265-8b2e-702a80150736","","","","","0","Thu Mar 17 12:00:00 CET 2016","Thu Mar 17 06:00:00 CET 2016","5","Thu Dec 10 12:00:00 CET 2015","true"]}

Name: updateJobDataMap

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for a job trigger list

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group
jobDataMap	The job data map in JSON format

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22, %22jobDataMap%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: addJob

URL: http://hostname:8080/SmartCloudEngine

Description

API to request to add a job to the scheduler

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group
jobClass	The job class

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22, %22jobClass%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: getNotificationEmail

URL: http://hostname:8080/SmartCloudEngine

Description

API to ask for the notification email associated with the job completion

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22 %22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["response"],"1":"true"}

Name: getJobDataMap

URL: http://hostname:8080/SmartCloudEngine

Description

API to request for a job data map in JSON format

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getJobDataMap%22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"#processParameters":"[{\"processPath\":\"\\\/opt\\\/jdk1.8.0_51\\\/bin\\\/java\"},{\"cp\":\"-classpath\"},{\"lib\":\":\\\/var\\\/www\\\/html\\\/sce\\\/test\\\/lib\\\/*\"},{\"TwitterIndexing\":\"twitterindexing.TwitterIndexing\"},{\"twitter.properties\":\"\\\/var\\\/www\\\/html\\\/sce\\\/test\\\/lib\\\/twitter.properties\"}]","#isNonConcurrent":"true"}

Name: getJobFireTimes

URL: http://hostname:8080/SmartCloudEngine

Description

API to request job firing times

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getJobFireTimes%22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"startTime":"Thu Dec 10 12:00:00 CET 2015","state":"BLOCKED","previousFireTime":"Thu Dec 10 12:00:00 CET 2015"}

Name: getJobTriggers

URL: http://hostname:8080/SmartCloudEngine

Description

API to request job triggers

Mode: GET, POST

Required Parameters

id	The id of the required action
jobName	The job name
jobGroup	The job group

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getJobTriggers%22, %22jobName%22:%22 %22, %22jobGroup%22:%22 %22}

JSON Result:

{"0":["7e369086-851f-4191-927f-229327d0f44c","4b2cf714-cd6c-4265-8b2e-702a80150736"]}

Name: getSchedulerMetadata

URL: http://hostname:8080/SmartCloudEngine

Description

API to request scheduler metadata

Mode: GET, POST

Required Parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getSchedulerMetadata%22}

JSON Result:

{"Scheduler instance id":["debian1456734408633","Reports the instance id of the scheduler"],"Running since":["Mon Feb 29 09:26:48 CET 2016","Reports the date at which the scheduler started running"],"Number of jobs executed":[1826,"Reports the number of jobs executed since the scheduler

started"],"Scheduler started":["yes","Reports whether the scheduler has been started"],"JobStore supports persistence":["yes","Reports whether or not the scheduler's JobStore instance supports persistence"],"Remote Scheduler":["no","Reports whether the scheduler is being used remotely (via RMI)"],"Scheduler shutdown":["no","Reports whether the scheduler has been shutdown"],"Standby mode":["no","Reports whether the scheduler is in standby mode"],"JobStore Clustered":["yes","Reports whether or not the scheduler's JobStore is clustered"],"Scheduler name":["SCE","Reports the name of the scheduler"]}

Name: getSystemStatus

URL: http://hostname:8080/SmartCloudEngine

Description

API to request the scheduler node status

Mode: GET, POST

Required parameters

id	The id of the required action

Example

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={%22id%22:%22getSystemStatus%22}

JSON Result:

{"Operating System version":["3.16.0-4-amd64","Reports the operating system version"],"Operating System architecture":["amd64","Returns the operating system architecture"],"CPU load (JVM)":["6.318334157463876E-4","Returns the recent cpu usage for the Java Virtual Machine process. This value is a double in the [0.0, 1.0] interval. A value of 0.0 means that none of the CPUs were running threads from the JVM process during the recent period of time observed, while a value of 1.0 means that all CPUs were actively running threads from the JVM 100% of the time during the recent period being observed. Threads from the JVM include the application threads as well as the JVM internal threads. All values between 0.0 and 1.0 are possible depending of the activities going on in the JVM process and the whole system. If the Java Virtual Machine recent CPU usage is not available, the value reports a negative value"],"Number of processors":["8","Reports the number of processors available to the Java virtual machine"],"Process CPU time":["5379360000000","Returns the cpu time used by the process on which the Java virtual machine is running in nanoseconds. The returned value is of nanoseconds precision but not necessarily nanoseconds accuracy. This value reports -1 if the platform does not support this operation"],"Total physical memory":["5.0756255744E10","Returns the total amount of physical memory in bytes"],"CPU load":["0.129306082083402","Returns the recent cpu usage for the whole system. This value is a double in the [0.0, 1.0] interval. A value of 0.0 means that all CPUs were idle during the recent period of time observed, while a value of 1.0 means that all CPUs were actively running 100% of the time during the recent period being observed. All values between 0.0 and 1.0 are possible depending of the activities going on in the system. If the system recent cpu usage is not available, the value reports a negative value"],"Free physical memory":["32348704768","Reports the amount of free physical memory in bytes"],"Committed virtual memory":["6919090176","Reports the amount of virtual memory that is guaranteed to be available to the running process in bytes, or -1 if this operation is not supported"],"Free

swap space":["7839490048","Reports the amount of free swap space in bytes"],"Total swap space":["8.585736192E9","Returns the total amount of swap space in bytes"],"System Load average":["1.31","Reports the system load average for the last minute. The system load average is the sum of the number of runnable entities queued to the available processors and the number of runnable entities running on the available processors averaged over a period of time. The way in which the load average is calculated is operating system specific but is typically a damped time-dependent average.\n\nIf the load average is not available, a negative value is returned.\nThis value is designed to provide a hint about the system load and may be queried frequently. The load average may be unavailable on some platform where it is expensive to implement this method"],"Operating System name":["Linux","Reports the operating system name"]}

Name: getConnectionPoolInfo	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API for requesting connection pool information for the database	
Mode: GET, POST	
Required Parameters	
id	The id of the required action
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={“id”:”getConnectionPoolInfo”} **JSON Result:** {"MaxIdle":8,"MinIdle":0,"MaxActive":10,"NumActive":0,"TestOnReturn":false,"TestWhileIdle":false,"SoftMinEvictableIdleTimeMillis":-1,"TestOnBorrow":false,"TimeBetweenEvictionRunsMillis":-1,"MaxWait":-1,"WhenExhaustedAction":2,"NumIdle":1,"Lifo":true,"MinEvictableIdleTimeMillis":1800000,"NumTestsPerEvictionRun":3}	

Name: buildTriggerForJob	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to request the creation of a trigger for a job	
Mode: GET, POST	
Required Parameters	
id	The id of the required action
jobName	The job name

jobGroup	The job group
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={“id”:”buildTriggerForJob”,”jobname”:”job1”,”jobGroup”:”group1”} **Result:** true	

Name: setJobProgress	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to set the percentage progress of a job	
Mode: GET, POST	
Required Parameters	
id	The id of the required action
fire_instance_id	The job’s fire instance id
progress	The progress percentage of a job
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={“id”:”setJobProgress”,”fire_instance_id”:”job1”,”progress”:”30”} **Result:** true	

Name: getJobProgress	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to request the percentage progress of a job	
Mode: GET, POST	
Required Parameters	
id	The id of the required action
fire_instance_id	The job’s fire instance id
Example	

http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={“id”:”getJobProgress”,”fire_instance_id”:”Job1”}

Result:

30

Name: getJobProgress	
URL: http://hostname:8080/SmartCloudEngine	
Description	
API to request the progress percentage of a job	
Mode: GET, POST	
Required Parameters	
id	The id of the required action
fire_instance_id	The job’s fire instance id
Example	
http://192.168.0.23:8080/SmartCloudEngine/index.jsp?json={“id”:”getJobProgress”,”fire_instance_id”:”job1”} **Result:** 10	