**SAGA-Torque Adaptor**

**System Specification**

High Energy Accelerator Research Organization (KEK)

Computing Research Center

January 8, 2010

Index

[1 Introduction 4](#_Toc250716368)

[2 Overview of STA 4](#_Toc250716369)

[2.1 STA operations 4](#_Toc250716370)

[2.2 Scheme Definition to load STA 4](#_Toc250716371)

[2.3 JobID Format 5](#_Toc250716372)

[2.4 Status Notification 5](#_Toc250716373)

[3 How STA works with Torque commands 7](#_Toc250716374)

[3.1 Torque commands used in STA 7](#_Toc250716375)

[3.1.1 PBS script 7](#_Toc250716376)

[3.2 How to use Torque commands by SAGA API 7](#_Toc250716377)

[3.2.1 saga::job::service class 7](#_Toc250716378)

[3.2.2 saga::job::job class 9](#_Toc250716379)

[4 PBS script creation 12](#_Toc250716380)

[4.1 PBS script structure 12](#_Toc250716381)

[4.2 Attributes of the saga::job::description vs PBS directives 13](#_Toc250716382)

[4.2.1 Executable and Arguments 14](#_Toc250716383)

[4.2.2 Environment Variables 14](#_Toc250716384)

[4.2.3 Working Directory 15](#_Toc250716385)

[4.2.4 Interactive mode 17](#_Toc250716386)

[4.2.5 Standard output and error 17](#_Toc250716387)

[4.2.6 File staging 17](#_Toc250716388)

[4.2.7 Max Wall time 20](#_Toc250716389)

[4.2.8 JobContact 20](#_Toc250716390)

[4.3 Options saga::job::description does not support 21](#_Toc250716391)

[5 Job Status 22](#_Toc250716392)

[6 Adaptor Configuration File 24](#_Toc250716393)

[6.1 File name and location of Adaptor configuration file 24](#_Toc250716394)

[6.2 Configuration 24](#_Toc250716395)

[6.2.1 [saga.adaptors.torque\_job] section 24](#_Toc250716396)

[6.2.2 [saga.adaptors.torque\_job.cli] section 24](#_Toc250716397)

[6.2.3 [saga.adaptors.torque\_job.cli.description] section 24](#_Toc250716398)

[7 SAGA API specification by STA 25](#_Toc250716399)

[7.1 saga::job::service class 25](#_Toc250716400)

[7.1.1 service(rm) 25](#_Toc250716401)

[7.1.2 create\_job(job\_desc) 25](#_Toc250716402)

[7.1.3 run\_job(commandline, hostname, stdin\_stream, stdout\_stream, stderr\_stream) 26](#_Toc250716403)

[7.1.4 run\_job(commandline, hostname) 26](#_Toc250716404)

[7.1.5 list() 26](#_Toc250716405)

[7.1.6 get\_job(job\_id) 27](#_Toc250716406)

[7.1.7 get\_self() 27](#_Toc250716407)

[7.2 saga::job::job class 28](#_Toc250716408)

[7.2.1 get\_job\_id() 28](#_Toc250716409)

[7.2.2 run() 28](#_Toc250716410)

[7.2.3 wait(timeout) 28](#_Toc250716411)

[7.2.4 cancel(timeout) 29](#_Toc250716412)

[7.2.5 get\_state() 29](#_Toc250716413)

[7.2.6 get\_description() 29](#_Toc250716414)

[7.2.7 get\_stdin() 30](#_Toc250716415)

[7.2.8 get\_stdout() 30](#_Toc250716416)

[7.2.9 get\_stderr() 31](#_Toc250716417)

[7.2.10 suspend() 31](#_Toc250716418)

[7.2.11 resume() 31](#_Toc250716419)

[7.2.12 checkpoint() 32](#_Toc250716420)

[7.2.13 migrate(job\_desc) 32](#_Toc250716421)

[7.2.14 signal(signal) 32](#_Toc250716422)

[8 Source Files 34](#_Toc250716423)

[8.1 Source files related to Adaptor implementation 34](#_Toc250716424)

[8.2 Source files related to Torque commands 34](#_Toc250716425)

[9 Class Reference 35](#_Toc250716426)

[9.1 Namespace 35](#_Toc250716427)

[9.2 Class 35](#_Toc250716428)

[9.2.1 namespace torque\_job 35](#_Toc250716429)

[9.2.2 namespace torque\_job::cli 35](#_Toc250716430)

[9.2.3 namespace torque\_job::helper 36](#_Toc250716431)

[9.3 Functions 36](#_Toc250716432)

[9.3.1 namespace torque\_job::helper 36](#_Toc250716433)

# Introduction

This document is the system specification of the STA (SAGA-Torque Adaptor for Job Management).

# Overview of STA

STA is the SAGA adaptor that is required to use a cluster system by Torque. STA enables SAGA applications to submit jobs to Torque cluster and to monitor the job statuses via SAGA API.

This chapter describes the STA operations and how to use STA.

## STA operations

Torque commands should be installed in the environment to use STA for the reason that STA required Torque commands to access Torque cluster. The following is the procedure that a SAGA application issues a Torque command via SAGA API.

1. Create an instance of saga::job::service class. The argument of the constructor has the SAGA URL including Torque scheme and the job submit host.
2. SAGA engine starts the initialization of STA. STA is initialized based on the adaptor configuration file.
3. SAGA application executes SAGA API.
4. SAGA engine calls suitable a STA function by the invoked SAGA API.
5. The STA function calls a suitable Torque command which accesses to PBS.

## Scheme Definition to load STA

The argument of the saga::job::service class constructor should have the following SAGA URL, in order to load STA by SAGA application.

torque://*localhost*/

|  |  |
| --- | --- |
| torque | Scheme name to load STA |
| *localhost* | FQDN of the host executing SAGA application |

This URL means that the SAGA application access Torque cluster by using Torque commands of the host, *localhost* , as backend commands. STA can accept the scheme “any” because SAGA specification defines that SAGA application can select any adaptor by using “any” scheme. It is possible that STA is not loaded if using “any” scheme when several adaptors are installed. The *localhost* is used as the hostname of the local file location to stage files, therefore, the FQDN of the host must be used instead of string “localhost”.

## JobID Format

SAGA defines JobID should be specified as the following.

‘[backend url] – [native id]’

Then, the SAGA JobID for Torque becomes like the following.

[torque://*localhost*/] – [*Torque JOB\_ID*]

|  |  |
| --- | --- |
| torque | Specify “torque” to use Torque commands to access Torque jobs. Not specify “any” here. |
| *localhost* | FQDN of the host executing SAGA application |
| *Torque JOB\_ID* | Torque JobID. This JobID should be specified as below.   * *sequence\_number* * *sequence\_number.server\_name*   There are other formats of Torque JobID but the current STA uses only above two types of formats. |

## Status Notification

STA uses Torque commands to acquire the job status but that requires some limitations.

One of the limitations is that “qstat” command cannot acquire the job status. Torque does not completely hold the information of completed jobs in the execution queue. It is possible that Torque returns the recent information of completed jobs in execution queue by the configuration of the “keep\_completed”, however, the information will be disregarded if the specified retention time of the keep\_completed is passed over.

The other limitation is that “qstat” command cannot acquire the error information if the job fails on the Torque server. In such a situation, the job becomes listed on the execution queue as “Wait” status.

The above limitations might cause problems to monitor job statuses correctly. Therefore, to avoid the situation that STA cannot get job statuses by the limitations, STA has the email notification function that the job statuses are informed to the use submitting the job. In the default configuration, STA always sends email whenever the job is aborted. The destination of the email is the user who executes the commands to submit a job.

The destination of the email can be specified as JobContact in the adaptor configuration file.

# How STA works with Torque commands

This chapter describes how STA works with Torque commands.

## Torque commands used in STA

STA uses the following Torque commands.

|  |  |
| --- | --- |
| qsub | To submit jobs.  Used in saga::job::service::run\_job() and saga::job::job::run() |
| qstat | To get job statuses and job lists.  Used in saga::job::job::get\_state() and saga::job::service::list() |
| tracejob | To get completed job statuses.  Used in saga::job::job::get\_state() |

Torque has other commands to control jobs but the current STA supports the above commands only.

### PBS script

STA creates PBS script and outputs the script to the standard input of the ‘qsub’ command without creating a file of the script. PBS script can include not only job execution command but also PBS directives as the options of qsub command. Further information of PBS directives in the PBS script by STA is described in the chapter 4.

## How to use Torque commands by SAGA API

This chapter describes each SAGA API methods to use Torque commands. The SAGA API implementation for STA is described in the chapter 7.

### saga::job::service class

#### run\_job(commandline, hostname)

This API executes the command specified by *commandline* as a job submission by using ‘qsub’ command. The following is the explanation about the arguments of this API.

|  |  |
| --- | --- |
| commandline | This string will be output to PBS script directly. |
| hostname | This argument is not supported in the current STA. |

STA converts the Torque JobID of the qsub standard output to SAGA JobID. Then, STA stores the SAGA JobID in the saga::job::attributes::jobid in saga::job::job object.

##### Example:

The following example shows the case that Torque JobID is 179.kek-sna131.soum.co.jp.

|  |
| --- |
| 179.kek-sna131.soum.co.jp |

#### list()

This API gets a list of Torque JobID by using ‘qstat’ command. The argument is not specified. STA takes information of Torque JobID in the job list of the output by the qstat standard output. STA converts all of the Torque JobID information to SAGA JobID and returns them in form of std::vector.

##### Example:

For example, if qstat returns like the flowing output,

|  |
| --- |
| Job id Name User Time Use S Queue  ------------------------- ---------------- --------------- -------- - -----  66.kek-sna131 kkk takando 00:00:00 C workq  67.kek-sna131 kkk takando 00:00:00 C workq |

STA returns the following SAGA JobID.

|  |
| --- |
| [torque://kek-sna.soum.co.jp/] – [66.kek-sna131]  [torque://kek-sna.soum.co.jp/] – [67.kek-sna131] |

#### get\_job(jobid)

This API executes ‘qstat *Torque\_JobID’* by using the Torque JobID converted by the SAGA JobID specified in the argument jobid. This API uses the ‘qstat’ command in order to check the availability of job information but not to check job statues.

### saga::job::job class

#### run()

This API creates a PBS script based on saga::job::description that is specified by the arguments of saga::job::service::create\_job(), and then submits a job with the PBS script by using ‘qsub’ command. STA executes the ‘qsub’ command without arguments. Instead, SPI inputs the created PBS script to the standard input of the qsub process. Also this API gets the Torque JobID from the qsub standard output, converts Torque JobID to SAGA JobID, and then stores the Torque JobID to saga::job::attoributes::jobid of the saga::job::job object.

#### get\_state()

This API executes ‘qstat –f’ in order to get a job status. Torque JobID will be specified in the argument of ‘qstat –f’. The Torque JobID will be created based on the saga::job::attributes::jobid of the saga::job::job object. The command ‘qstat –f’ outputs the job information like the following. STA checks the information of job\_state and exit\_status in the job information. Further information how to check their values is described in the chapter 5.

|  |
| --- |
| Job Id: 80.kek-sna131.soum.co.jp  Job\_Name = cmd1.txt  Job\_Owner = takando@kek-sna.soum.co.jp  resources\_used.cput = 00:00:00  resources\_used.mem = 0kb  resources\_used.vmem = 0kb  resources\_used.walltime = 00:00:00  job\_state = C  queue = workq  server = kek-sna131.soum.co.jp  Checkpoint = u  ctime = Thu Mar 12 13:32:45 2009  Error\_Path = kek-sna.soum.co.jp:/home/takando/SB/sta-trunk/test/saga/cmd1.  txt.e80  exec\_host = kek-sna132.soum.co.jp/0  Hold\_Types = n  Join\_Path = n  Keep\_Files = n  Mail\_Points = a  Mail\_Users = takando@soum.co.jp  mtime = Thu Mar 12 13:32:45 2009  Output\_Path = kek-sna.soum.co.jp:/home/takando/SB/sta-trunk/test/saga/cmd1  .txt.o80  Priority = 0  qtime = Thu Mar 12 13:32:45 2009  Rerunable = True  Resource\_List.host = kek-sna131.soum.co.jp  Resource\_List.nodect = 1  Resource\_List.nodes = 1  session\_id = 11410  substate = 59  Variable\_List = PBS\_O\_HOME=/home/takando,PBS\_O\_LANG=en\_US.UTF-8,  PBS\_O\_LOGNAME=takando,  PBS\_O\_PATH=/usr/local/torque/bin:/usr/local/torque/sbin:/usr/naregi/b  in:/usr/kerberos/bin:/usr/java/default/bin:/usr/local/globus-4.0.8/bin  :/usr/local/globus-4.0.8/sbin:/opt/condor-7.0.4/bin:/opt/condor-7.0.4/  sbin:/usr/local/apache-ant-1.7.1/bin:/usr/local/bin:/bin:/usr/bin:/hom  e/takando/bin:/home/takando/local/bin,  PBS\_O\_MAIL=/var/spool/mail/takando,PBS\_O\_SHELL=/bin/bash,  PBS\_SERVER=kek-sna.soum.co.jp,PBS\_O\_HOST=kek-sna.soum.co.jp,  PBS\_O\_WORKDIR=/home/takando/SB/sta-trunk/test/saga,PBS\_O\_QUEUE=workq  comment = Job started on Thu Mar 12 at 13:32  etime = Thu Mar 12 13:32:45 2009  exit\_status = 0  submit\_args = ./cmd1.txt  start\_time = Thu Mar 12 13:32:45 2009  start\_count = 1 |

# PBS script creation

There are two ways to submit a job by SAGA applications;

* Create saga::job::job object by saga::job::service::create\_job() and then execute run()
* Execute saga::job::service::run\_job()

In the former way, SAGA application should configure the job information in the saga::job::description object. STA creates a PBS script based on the saga::job::description object.

In the latter way, SAGA application needs to specify the job infromation in the argument of the saga::job::service::run\_job(). STA creates a saga::job::description object based on the API arguments, and then creates a PBS script by the object.

This chapter describes how to create a PBS script by STA.

## PBS script structure

PBS script is a shell script to input for the ‘qsub’ command. The following is an example.

|  |
| --- |
| #! /bin/sh  #PBS option  #PBS option  ...  executable argument ...  executable argument ...  executable argument ... |

In that example, the portion ‘executable argument …’ means the executable command and its arguments on the job execution host. The portion ‘#PBS …’ means PBS directives. The PBS directives are used as the arguments of the ‘qsub’ command.

## Attributes of the saga::job::description vs PBS directives

The following table shows the corresponding table the saga::job::description attributes and PBS directives. STA does not care about the attribute that is “Ignore” in the requirement column in the table. The “Not implemented” attributes are planned to be supported in the future version. The attribute names beginning with “description\_ …” are defined in the namespace saga::job::attributes. They should be “saga::job::attributes::description\_ …” to be exact but the tables uses only “description\_ …” here to avoid redundancies.

|  |  |  |
| --- | --- | --- |
| **saga::job::attributes** | **PBS directives** | **Requirement** |
| description\_executable | (executable) | Required |
| description\_arguments | (argument) | Option |
| description\_environment | -v variable list[,...] | Option |
| description\_working\_directory | -d path | Option |
| description\_interactive | -I or -W interactive=true | Not Implemented |
| description\_input | - | Not Implemented |
| description\_output | -o path | Option |
| description\_error | -e path | Option |
| description\_file\_transfer | -W stagein=file\_list and -W stageout=file\_list | Option |
| description\_cleanup | - | Not Implemented |
| description\_job\_start\_time | -a date time | Not Implemented |
| Description\_totall\_cpu\_time | -l cput=seconds | Not Implemented |
| description\_wall\_time\_limit | -l walltime=seconds | Option |
| description\_total\_physical\_memory | -l pmem=size | Not Implemented |
| description\_cpu\_architecture | -l arch=string | Not Implemented |
| description\_operating\_system\_type | -l opsys=string | Not Implemented |
| description\_candidate\_hosts | -l host=string | Not Implemented |
| description\_queue | -q destination | Not Implemented |
| description\_job\_contact | -M user list | Option |
| description\_job\_project | - | Ignore |
| description\_spmd\_variation | - | Ignore |
| description\_total\_cpu\_count | -l nodes | Not Implemented |
| description\_number\_of\_proceses | -l nodes | Not Implemented |
| description\_processes\_per\_host | -l nodes | Not Implemented |
| description\_threads\_per\_process | - | Ignore |

### Executable and Arguments

The values of description\_executable and description\_arguments are written in the end of the created PBS script. The description\_executable must be specified. If the description\_executable is not specified, the exceptions are happen in the saga::job::service::create\_job(). According to SAGA specification, two or more description\_executable values cannot be specified in one saga::job::description even if a PBS script itself can accept several command lines. Therefore, the PBS script that is created by STA can have only one executable command.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::description jd;  jd.set\_attribute(sja::description\_executable, "/usr/bin/ci");  std::vector<std::string> args;  args.push\_back("-m\"add include\"");  args.push\_back("sample.c");  jd.set\_vector\_attribute(sja::description\_arguments, args); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  /usr/bin/ci -m"add include" sample.c |

### Environment Variables

The environment variables should be specified in description\_environment by std::vector object. Each entry is a string in the form of “*name*=*value*”. STA combines their entries by commas and puts the combined string as the –v option value.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::description jd;  std::vector<std::string> env;  env.push\_back("FOO=HOGE");  env.push\_back("BAR=FUGA");  jd.set\_vector\_attribute(sja::description\_environment, env); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  #PBS -v FOO=HOGE,BAR=FUGA  ... |

### Working Directory

The working directory defined in description\_working\_directory is specified by the ‘-d’ option of the ‘qsub’ command. If the description\_working\_directory is not specified, STA will not create PBS directives.

##### Note:

In the case that SAGA application specifies the working directory, specifying absolute directory paths of local host and remote host are recommended. The directory specified in description\_working\_direcoty is used not only as a job working directory on the remote host but also as a working directory on the local host. Therefore, the working directories on the local host and remote host are influenced by existence or nonexistence of the ‘-d’ option and absolute or relative path of the specified path.

* Working directory on Local host
  + The working directory becomes the current directory if description\_working\_directory is not specified.
  + The working directory becomes the relative directory to the current directory if description\_working\_directory is specified as a relative directory.
  + The ‘qsub’ command returns errors and the job is not created if the directory specified in description\_working\_directory does not exist on the local host.
* Working directory on Remote host
  + The working directory becomes the home directory if description\_working\_directory is not specified.
  + The working directory becomes the same path as the working directory on the local host if description\_working\_directory is specified.
  + The ‘exit\_status’ value of the job becomes “-2” and sends a email with the abort notification to the user, if the directory specified in description\_working\_directory does not exist on the remote host.

Then, STA handles the specified path as below.

* The working directory is specified as Relative path
  + STA converts the relative path to a absolute path. In this case, the path of both home directories on the local and remote hosts should be same.
* The working directory is specified as Absolute path
  + The specified path is used as the working directory directly.

STA does not check whether the working directory does exist or not. The users should create working directories before to submit a job.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::description jd;  jd.set\_vector\_attribute(sja::description\_working\_directory, "/tmp"); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  #PBS –d /tmp  ... |

### Interactive mode

The current STA does not support the interactive mode. STA can accept the only “false” value of the description\_interactive. If the specified value is “true”, STA returns the exception, “Not Implemented”.

### Standard output and error

STA supports the standard output/error. TBD.

### File staging

STA converts the value of the file transfer directive specified in the description\_file\_transfer to the argument of the ‘qsub’ command as the ‘-W’ option.

#### Format and Limitation of File transfer directive

The following is the format to specify the file transfer directive but there are some limitations.

|  |
| --- |
| *local\_file operator remote\_file* |

|  |  |
| --- | --- |
| *local\_file* | Only absolute or relative path can be specified. URL can NOT be specified. |
| *operator* | Only ‘>’ or ‘<’ can be specified. Existing files will be overwritten according to Torque specification. If other characters are specified here, STA returns exceptions. |
| *remote\_file* | Only absolute or relative path can be specified. URL can NOT be specified. |

#### Format of the ‘-W’ option

* Stage in option to transfer files to Remote host before job execution

-W stagein=*file\_list*

* Stage out option to transfer files to Local host after job execution

-W stageout=*file\_list*

The *file\_list* format is the following.

*Local\_file@hostname:remote\_file[,...]*

#### Conversion to ‘-W’ option

##### Transfer files to a job execution host

The operator ‘>’ of the file transfer directive converts to ‘-W statein=…’ option to transfer files to a job execution host.

**Source file**  In use of STA, the only files on SAGA application execution host can be specified as the source files (left hand side of *operator*). The relative path is assumed as a relative path to the current directory if the source file is specified with a relative path, and then the relative path is converted to the absolute path. The *hostname* uses the host name of the URL specified in the arguments of the saga::job::service constructor.

**Target file** The target is the job execution host. The relative path is assumed as a relative path to the working directory if the target file (right hand side of *operator*) is specified with a relative path, and then, the relative path is converted to the absolute path.

##### Transfer files from the job execution host

The operator ‘<’ of the file transfer directive converts to ‘-W stateout=…’ option to transfer files from the job execution host.

**Source file** The source is the job execution host. The relative path is assumed as a relative path to the working directory if the target file (right hand side of *operator*) is specified with a relative path, and then, the relative path is converted to the absolute path.

**Target file** In use of STA, the only files on SAGA application execution host can be specified as the target files (left hand side of *operator*). The relative path is assumed as a relative path to the current directory if the target file is specified with a relative path, and then the relative path is converted to the absolute path. The *hostname* uses the host name of the URL specified in the arguments of the saga::job::service constructor.

#### Files after Job execution

The files that are staged in before the job execution will be removed from the job execution host after the job execution.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::service js("naregi://example.com/");  saga::job::description jd;  std::vector<std::string> ft;  ft.push\_back("/home/user/tiger.eps > /tmp/tiger.eps");  ft.push\_back("/home/user/tiger.pdf < /tmp/tiger.pdf");  jd.set\_vector\_attribute(sja::description\_file\_transfer, ft); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  #PBS -W stagein=/tmp/tiger.eps@example.com:/home/user/tiger.eps  #PBS -W stageout=/tmp/tiger.pdf@example.com:/home/user/tiger.pdf  ... |

### Max Wall time

STA uses the value of the description\_wall\_time\_limit as the *walltime*.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::description jd;  jd.set\_attribute(sja::description\_wall\_time\_limit, "300"); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  #PBS -l walltime=300  ... |

### JobContact

STA uses the email address specified in the description\_job\_contact as the ‘-M’ option of the ‘qsub’ command. That enables that users can receive the status notification emails from Torque server when the job is aborted. The format of the description\_job\_contact value is URI as the following. The current SAGA C++ API ver. 1.1.1 supports to specify only one address as JobContact even if the SAGA specification defines the JobContact as vector string.

|  |
| --- |
| mailto:*<mail address>* |

The saga::job::description will not be referred when the job is executed by the saga::job::server::run\_job(). In this case, Torque server tries to send an abort notification to the default address that is the user executing the ‘qsub’ command.

In the case of submitting a job by using the saga::job::server::run\_job(), Torque server tries to send an abort notification to the user address that executes the “qsub” command on the job execution host as the default JobContact. However, it is possible that the host executing the SAGA application cannot receive emails. To avoid such a situation, the default JobContact address can be specified in the adaptor ini file. Also in the case of submitting a job by using the saga::job::server::create\_job() and the description\_job\_contact is not specified, the JobContact address specified in the adaptor ini file is used.

##### Example: SAGA application example

|  |
| --- |
| namespace sja = saga::job::attributes;  saga::job::description jd;  jd.set\_attribute(sja::description\_job\_contact, "mailto:kek-sna@soum.co.jp"); |

##### Example: PBS script sample

|  |
| --- |
| #! /bin/sh  ...  #PBS -M kek-sna@soum.co.jp  ... |

## Options saga::job::description does not support

The saga::job::description does not support the following options. STA uses fixed values because SAGA applications cannot specify the values.

|  |  |
| --- | --- |
| -N *name* | *name* is shown in the Name column of the ‘qstat’ command output. The fixed value is “saga-app” in STA. |

# Job Status

The following table shows the comparison between PBS job state and the saga::job::state.

|  |  |  |
| --- | --- | --- |
| **Torque job\_state** | | **saga::job::state** |
| - | (Right after a job object is created) | saga::job::New |
| C | Job is completed after having run. | saga::job::Done, saga::job::Failed |
| E | Job is exiting after having run. | saga::job::Running |
| H | Job is held. |
| Q | Job is queued, eligible to run or routed. |
| R | Job is running. |
| S | (Unicos only) Job is suspended. | saga::job::Suspend |
| T | Job is being moved to new location. | saga::job::Running |
| W | Job is waiting for its execution time (-a option) to be reached. |
| - | (Cancel after the job execution) | saga::job::Canceled |

#### saga::job::New

This state, New, is set in the saga::job::job object created by the saga::job::service::create\_job() before submitting the job. PBS does not have this job state because this state is the state before submitting the job.

#### saga::job::Running

This state, Running, is set the saga::job::job object when submitting the job by the saga::job::job::run() or the saga::job::service::run\_job(). This state does not change unless the saga::job::job::get\_state() detects that the job is completed or fails. The job in the ‘W’ PBS job state is also this Running state. That is because the ‘W’ state means “submitted job” that is no different from the ‘R’ state from a viewpoint of the SAGA.

#### saga::job::Suspended

The current STA does not support the Suspended state.

#### saga::job::Done

This state, Done, is set the saga::job::job object when the saga::job::job::get\_state() returns the following results. The get\_state() uses the ‘qstat’ command to get the “job\_status” and the “exit\_status”.

* The “job\_status” is “C”
* The “exit\_status” is 0.

#### saga::job::Failed

This state, Failed, is set the saga::job::job object when the saga::job::job::get\_state() returns the following results. The get\_state() uses the ‘qstat’ command to g get the “job\_status” and the “exit\_status”.

* The “job\_status” is “C”
* The “exit\_status” is NOT 0.

#### saga::job::Canceled

The current STA does not support the Canceled state.

# Adaptor Configuration File

The adaptor configuration file is used to specify STA configuration. Users can modify STA default configuration as they need.

## File name and location of Adaptor configuration file

The file name of the STA adaptor configuration file is “saga\_adaptor\_torque\_job.ini”. The ini file is typically installed in the directory, $SAGA\_LOCATION/share/saga.

## Configuration

### [saga.adaptors.torque\_job] section

|  |  |
| --- | --- |
| name | Specified as “torque\_job”. No change in typical use. |
| path | Specified as “$[saga.location]/lib”. No change in typical use. |
| enabled | Specified as “false” when STA is disabled. No change in typical use |

### [saga.adaptors.torque\_job.cli] section

Reserved.

### [saga.adaptors.torque\_job.cli.description] section

|  |  |
| --- | --- |
| JobContact | Specifies the email address to receive from Torque server. The format is *mailto*:*user*@*host*. This JobContact is used as the default value when the description\_job\_contact is not specified. Also, this JobContact is always used in the case of jobs submitted by using saga::job::service::run\_job().  Specifying this JobContact is mandatory to load STA. If JobContact is not specified, STA returns errors in being loaded. |

# SAGA API specification by STA

This chapter describes the specification of the saga::job::service and saga::job::job in the case of using STA.

## saga::job::service class

### service(rm)

|  |  |
| --- | --- |
| **Purpose** | |
| Constructor of the saga::job::service class. | |
| **Inputs** | |
| rm | Specify the SAGA URL. (Refer to 2.2) |
| **Outputs** | |
| n/a |  |
| **Exceptions** | |
| BadParameter | Occurs if the URL is not correct. |

### create\_job(job\_desc)

|  |  |
| --- | --- |
| **Purpose** | |
| Creates a saga::job::job object. This API checks following attributes of the saga::job::description.   * description\_executable * description\_interactive | |
| **Inputs** | |
| job\_desc | Specify the saga::job::description to be submitted. |
| **Outputs** | |
| Returns a saga::job::job. The job status becomes saga::job::New. | |
| **Exceptions** | |
| BadParameter | Occurs if the mandatory attribute, descriptin\_executable, is not specified or null. |
| Not Implemented | Occurs if the description\_interactive is ‘True’. |

### run\_job(commandline, hostname, stdin\_stream, stdout\_stream, stderr\_stream)

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### run\_job(commandline, hostname)

|  |  |
| --- | --- |
| **Purpose** | |
| Submits a job without a saga::job::description | |
| **Inputs** | |
| commandline | Specifies a command to be executed. |
| hostname | The current STA does not support |
| **Outputs** | |
| Returns the saga::job::job of the submitted job. The job status becomes saga::job::Running, saga::job::Done, or saga::job::Failed. | |
| **Exceptions** | |
| NoSuccess | Occurs when executing the ‘qsub’ command has problems. |

### list()

|  |  |
| --- | --- |
| **Purpose** | |
| Gets the job list that Torque server controls. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns SAGA JobID in the std::vector<std::string> type | |
| **Exceptions** | |
| NoSuccess | Occurs when executing the ‘qstat’ command has problems. |

### get\_job(job\_id)

|  |  |
| --- | --- |
| **Purpose** | |
| Gets a saga::job::job object by specifying SAGA JobID. | |
| **Inputs** | |
| job\_id | Specify the SAGA JobID |
| **Outputs** | |
| Returns the saga::job::job if the specified job exists. | |
| **Exceptions** | |
| BadPrameter | Occurs when the SAGA JobID is specified in wrong format. |
| DoesNotExist | Occurs when the specified job does not exist. |
| NoSuccess | Occurs when executing the ‘qstat’ command has problems. |

### get\_self()

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

## saga::job::job class

### get\_job\_id()

|  |  |
| --- | --- |
| **Purpose** | |
| Returns the SAGA JobID of this object. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns the SAGA JobID.  Returns empty string if the job status is saga::job::New. | |
| **Exceptions** | |
| n/a | No exception occurs by this API |

### run()

|  |  |
| --- | --- |
| **Purpose** | |
| Submits the job whose status is saga::job::New | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| BadPrameter | Occurs when the attribute values in the saga::job::description are wrong. |
| IncorrectState | Occurs when the job state is not saga::job::New. |
| NotImplemented | Occurs when the saga::job::description has wrong attributes. |
| NoSuccess | Occurs when executing the ‘qsub’ command has problems. |

### wait(timeout)

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### cancel(timeout)

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### get\_state()

|  |  |
| --- | --- |
| **Purpose** | |
| Gets the state of this job. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns the saga::job::state | |
| **Exceptions** | |
| NoSuccess | Occurs when executing the ‘qstat’ command has problems. |

### get\_description()

|  |  |
| --- | --- |
| **Purpose** | |
| Returns the saga::job::description object of this job if the saga::job::job object corresponds to either of the following.   * The object is given by saga::job::service::run\_job() . * The object is created by saga::job::service::create\_job(). | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns a saga::job::description | |
| **Exceptions** | |
| DoesNotExist | Occurs if the saga::job::job object does not correspond to the above cases. |

### get\_stdin()

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### get\_stdout()

|  |  |
| --- | --- |
| **Purpose** | |
| Returns standard output strings as job outputs | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns standard output strings as job outputs in the std::string type. | |
| **Exceptions** | |
| IncorrectState | Occurs when the job state is not saga::job::Done. |

### get\_stderr()

|  |  |
| --- | --- |
| **Purpose** | |
| Returns standard error strings as job errors | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| Returns standard error strings as job errors in the std::string type. | |
| **Exceptions** | |
| IncorrectState | Occurs when the job state is not saga::job::Done. |

### suspend()

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### resume()

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### checkpoint()

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### migrate(job\_desc)

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### signal(signal)

|  |  |
| --- | --- |
| **Purpose** | |
| The current STA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

# Source Files

## Source files related to Adaptor implementation

The following files are using templates created by adaptors/generator/generator.pl SAGA provides. The *italic files* are directly using the templates without modifications.

* torque\_job\_adaptor.cpp
* torque\_job\_adaptor.hpp
* torque\_job\_service.cpp
* torque\_job\_service.hpp
* torque\_job.cpp
* torque\_job.hpp
* torque\_job\_adaptor.ini
* *torque\_job\_async.cpp*
* *torque\_job\_service\_async.cpp*
* *torque\_job\_istream.hpp*
* *torque\_job\_ostream.hpp*
* *torque\_job\_stream.hpp*

## Source files related to Torque commands

The following files are newly created to implement STA.

* debug.hpp
* directives.hpp
* directives\_impl.cpp
* directives\_impl.hpp
* script.cpp
* script.hpp
* staging.hpp
* torque\_cli.cpp
* torque\_cli.hpp
* torque\_cli\_staging.cpp
* torque\_cli\_staging.hpp
* torque\_helper.cpp
* torque\_helper.hpp

# Class Reference

## Namespace

STA uses the following namespace.

|  |  |
| --- | --- |
| torque\_job | Contains whole STA |
| torque\_job::cli | Contains the classes and functions related to Torque command executions. |
| torque\_job::helper | Contains helper functions. |

## Class

The section describes main classes in each namespace shown in the section 9.1.

### namespace torque\_job

|  |  |
| --- | --- |
| adaptor (struct) | The adaptor implementation inherited from the saga::adaptor |
| job\_cpi\_impl | The STA implementation for the saga::job::job. |
| job\_srvice\_cpi\_impl | The STA implementation for the saga::job::service |

### namespace torque\_job::cli

|  |  |
| --- | --- |
| directives | The interface that configures the PBS directives. |
| directives\_checker | The interface that checks the PBS directives. |
| directives\_builder | The interface that builds the PBS directives. |
| directives\_impl | The class that configures the PBS directives. |
| directives\_checker\_impl | The class that checks the PBS directives. |
| directives\_builder\_impl | The class that builds the PBS directives. |
| job\_script | The PBS script file class. |
| job\_script\_builder | The class that builds the PBS script files. |
| \_directives\_checker\_impl | This class that checks the PBS directives is used by the job\_script\_builder class. |
| file\_transfer | The class that defines the file transfer. |
| file\_transfer\_parser | The interface that parses the file transfer directives. |
| output\_parser | The class that parses the Torque command outputs. |
| jobstat | The class that contains the job attributes of the ‘qstat –f’ command outputs. |
| jobstat\_builder | The class that builds jobstat. |
| qsub | The ‘qsub’ command class. |
| qstat | The ‘qstat’ command class. |
| file\_transfer\_impl | The class that implements the file\_transfer. |
| file\_transfer\_parser\_impl | The class that implements the file\_transfer\_parser\_impl. |
| staging\_path\_builder | The class that builds path names for the workflow file staging. |

### namespace torque\_job::helper

|  |  |
| --- | --- |
| jobid\_converter | The class that converts JobID formats between Torque JobID and SAGA JobID. |

## Functions

This section describes the functions belonging to no class.

### namespace torque\_job::helper

|  |  |
| --- | --- |
| convert\_saga\_job\_state(torque\_status) | The function that converts a Torque Job string to a SAGA state string. |
| create\_saga\_job\_description(jd, cmd, host) | The function that builds a saga::job::description for the saga::job::service::run\_job(). |
| split\_command\_line(cmd, executable, options) | The function that splits command line strings for the saga::job::service::run\_job(). |