

# **SAGA-PBSPro Adaptor**

## **System Specification**

High Energy Accelerator Research Organization (KEK)  
Computing Research Center

January 8, 2010

## Index

1	Introduction.....	4
2	Overview of SPA.....	4
2.1	SPA operations .....	4
2.2	Scheme Definition to load SPA.....	4
2.3	JobID Format .....	5
2.4	Status Notification .....	5
3	How SPA works with PBSPro commands .....	7
3.1	PBSPro commands used in SPA.....	7
3.1.1	PBS script.....	7
3.2	How to use PBSPro commands by SAGA API .....	7
3.2.1	saga::job::service class .....	7
3.2.2	saga::job::job class .....	9
4	PBS script creation .....	12
4.1	PBS script structure .....	12
4.2	Attributes of the saga::job::description vs PBS directives .....	13
4.2.1	Executable and Arguments .....	14
4.2.2	Environment Variables.....	14
4.2.3	Working Directory.....	15
4.2.4	Interactive mode .....	17
4.2.5	Standard output and error .....	17
4.2.6	File staging.....	17
4.2.7	Max Wall time .....	20
4.2.8	JobContact .....	20
4.3	Options saga::job::description does not support .....	21
5	Job Status.....	22
6	Adaptor Configuration File.....	24
6.1	File name and location of Adaptor configuration file .....	24
6.2	Configuration .....	24
6.2.1	[saga.adaptors.pbspro_job] section.....	24
6.2.2	[saga.adaptors.pbspro_job.cli] section.....	24
6.2.3	[saga.adaptors.pbspro_job.cli.description] section .....	24
7	SAGAAPI specification by SPA.....	25
7.1	saga::job::service class.....	25
7.1.1	service(rm).....	25
7.1.2	create_job(job_desc) .....	25

7.1.3	run_job(commandline, hostname, stdin_stream, stdout_stream, stderr_stream) .....	26
7.1.4	run_job(commandline, hostname) .....	26
7.1.5	list() .....	26
7.1.6	get_job(job_id) .....	27
7.1.7	get_self() .....	27
7.2	saga::job::job class .....	28
7.2.1	get_job_id() .....	28
7.2.2	run() .....	28
7.2.3	wait(timeout) .....	28
7.2.4	cancel(timeout) .....	29
7.2.5	get_state() .....	29
7.2.6	get_description() .....	29
7.2.7	get_stdn() .....	30
7.2.8	get_stdout() .....	30
7.2.9	get_stderr() .....	31
7.2.10	suspend() .....	31
7.2.11	resume() .....	31
7.2.12	checkpoint() .....	32
7.2.13	migrate(job_desc) .....	32
7.2.14	signal(signal) .....	32
8	Source Files .....	34
8.1	Source files related to Adaptor implementation .....	34
8.2	Source files related to PBSPro commands .....	34
9	Class Reference .....	35
9.1	Namespace .....	35
9.2	Class .....	35
9.2.1	namespace pbspro_job .....	35
9.2.2	namespace pbspro_job::cli .....	35
9.2.3	namespace pbspro_job::helper .....	36
9.3	Functions .....	36
9.3.1	namespace pbspro_job::helper .....	36

# 1 Introduction

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

## 2 Overview of SPA

SPA is the SAGA adaptor that is required to use a cluster system by PBSPro (Portable Batch System Professional Edition). SPA enables SAGA applications to submit jobs to PBS and to monitor the job statuses via SAGA API.

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

### 2.1 SPA operations

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

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

### 2.2 Scheme Definition to load SPA

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

`pbspro://localhost/`

<code>pbspro</code>	Scheme name to load SPA
<code>localhost</code>	FQDN of the host executing SAGA application

This URL means that the SAGA application access PBS by using PBSPro commands of the host, `localhost`, as backend commands. SPA can accept the scheme “any” because SAGA specification defines that SAGA application can select any adaptor by using “any” scheme. It is possible that SPA 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”.

## 2.3 JobID Format

SAGA defines JobID should be specified as the following.

‘[backend url] – [native id]’

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

[pbspro://localhost] – [PBS JOB\_ID]

<code>pbspro</code>	Specify “pbspro” to use PBSPro commands to access PBS jobs. Not specify “any” here.
<code>localhost</code>	FQDN of the host executing SAGA application
<code>PBS JOB_ID</code>	<p>PBS JobID. This JobID should be specified as below.</p> <ul style="list-style-type: none"> <li>➤ <code>sequence_number</code></li> <li>➤ <code>sequence_number.server_name</code></li> </ul> <p>There are other formats of PBS JobID but the current SPA uses only above two types of formats.</p>

## 2.4 Status Notification

SPA uses PBSPro commands to acquire the job status but that requires some limitations.

One of the limitations is that “qstat” command cannot acquire the job status. PBSPPro ver.9.xx does not hold the information of completed jobs in the execution queue. To solve that, SPA uses “tracejob” & “grep” command to parse the log data to get only “exit\_state” value, instead of using “qstat”.

The other limitation is that “qstat” command cannot acquire the error information if the job fails on the PBS 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 SPA cannot get job statuses by the limitations, SPA has the email notification function that the job statuses are informed to the user submitting the job. In the default configuration, SPA 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.

## 3 How SPA works with PBSPro commands

This chapter describes how SPA works with PBSPro commands.

### 3.1 PBSPro commands used in SPA

SPA uses the following PBSPro commands.

qsub	To submit jobs. Used in <code>saga::job::service::run_job()</code> and <code>saga::job::job::run()</code>
qstat	To get job statuses and job lists. Used in <code>saga::job::job::get_state()</code> and <code>saga::job::service::list()</code>
tracejob	To get completed job statuses. Used in <code>saga::job::job::get_state()</code>

PBSPro has other commands to control jobs but the current SPA supports the above commands only.

#### 3.1.1 PBS script

SPA 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 SPA is described in the chapter 4.

### 3.2 How to use PBSPro commands by SAGA API

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

#### 3.2.1 `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 SPA.

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

### Example:

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

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

### list()

This API gets a list of PBS JobID by using ‘qstat’ command. The argument is not specified. SPA takes information of PBS JobID in the job list of the output by the qstat standard output. SPA converts all of the PBS 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

SPA returns the following SAGA JobID.



```
[pbspro://kek-sna.soum.co.jp/] - [66.kek-sna131]
[pbspro://kek-sna.soum.co.jp/] - [67.kek-sna131]
```

### **get\_job(jobid)**

This API executes ‘qstat *PBS\_JobID*’ by using the PBS 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 statuses.

## **3.2.2 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. SPA 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 PBS JobID from the qsub standard output, converts PBS JobID to SAGA JobID, and then stores the PBS JobID to `saga::job::attributes::jobid` of the `saga::job::job` object.

### **get\_state()**

This API executes ‘qstat -f’ in order to get a job status. PBS JobID will be specified in the argument of ‘qstat -f’. The PBS 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. SPA 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/spa-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/spa-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/pbspro/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/spa-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
```

## 4 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. SPA creates a PBS script based on the `saga::job::description` object.

In the latter way, SAGA application needs to specify the job information in the argument of the `saga::job::service::run_job()`. SPA 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 SPA.

### 4.1 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.

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

The following table shows the corresponding table the `saga::job::description` attributes and PBS directives. SPA 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.

<b>saga::job::attributes</b>	<b>PBS directives</b>	<b>Requirement</b>
<code>description_executable</code>	(executable)	Required
<code>description_arguments</code>	(argument)	Option
<code>description_environment</code>	<code>-v variable list[,...]</code>	Option
<code>description_working_directory</code>	<code>-d path</code>	Option
<code>description_interactive</code>	<code>-I</code> or <code>-W interactive=true</code>	Not Implemented
<code>description_input</code>	-	Not Implemented
<code>description_output</code>	<code>-o path</code>	Option
<code>description_error</code>	<code>-e path</code>	Option
<code>description_file_transfer</code>	<code>-W stagein=file_list</code> and <code>-W stageout=file_list</code>	Option
<code>description_cleanup</code>	-	Not Implemented
<code>description_job_start_time</code>	<code>-a date time</code>	Not Implemented
<code>Description_totall_cpu_time</code>	<code>-l cput=seconds</code>	Not Implemented
<code>description_wall_time_limit</code>	<code>-l walltime=seconds</code>	Option
<code>description_total_physical_memory</code>	<code>-l pmem=size</code>	Not Implemented
<code>description_cpu_architecture</code>	<code>-l arch=string</code>	Not Implemented
<code>description_operating_system_type</code>	<code>-l opsys=string</code>	Not Implemented
<code>description_candidate_hosts</code>	<code>-l host=string</code>	Not Implemented
<code>description_queue</code>	<code>-q destination</code>	Not Implemented
<code>description_job_contact</code>	<code>-M user list</code>	Option
<code>description_job_project</code>	-	Ignore
<code>description_spm�_variation</code>	-	Ignore
<code>description_total_cpu_count</code>	<code>-l nodes</code>	Not Implemented
<code>description_number_of_proceses</code>	<code>-l nodes</code>	Not Implemented
<code>description_processes_per_host</code>	<code>-l nodes</code>	Not Implemented
<code>description_threads_per_process</code>	-	Ignore

### 4.2.1 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 SPA 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
```

### 4.2.2 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*”. SPA 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

...
```

### 4.2.3 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, SPA 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_directory` 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, SPA handles the specified path as below.

- The working directory is specified as Relative path
  - ◆ SPA 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.

SPA 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

...
```

#### 4.2.4 Interactive mode

The current SPA does not support the interactive mode. SPA can accept the only “false” value of the `description_interactive`. If the specified value is “true”, SPA returns the exception, “Not Implemented”.

#### 4.2.5 Standard output and error

SPA supports the standard output/error. **TBD**.

#### 4.2.6 File staging

SPA 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
```

<i>local_file</i>	Only absolute or relative path can be specified. URL can NOT be specified.
<i>operator</i>	Only ‘>’ or ‘<’ can be specified. Existing files will be overwritten according to PBSPRO specification. If other characters are specified here, SPA returns exceptions.

<i>remote_file</i>	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 stagein=...’ option to transfer files to a job execution host.

**Source file** In use of SPA, 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 stageout=...’ 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 SPA, 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("pbspro://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

...
```

### 4.2.7 Max Wall time

SPA 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

...
```

### 4.2.8 JobContact

SPA 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 PBS 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, PBS 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(), PBS 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

...
```

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

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

-N <i>name</i>	<i>name</i> is shown in the Name column of the ‘qstat’ command output. The fixed value is “saga-app” in SPA.
----------------	--

## 5 Job Status

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

PBS job_state		saga::job::state
-	(Right after a job object is created)	saga::job::New
-	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 SPA 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 a queue

list and the 'tracejob' command to get the "exit\_status".

- The job does not exist in the list that 'qstat' returns.
- 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 get a queue list and the 'tracejob' command to get the "exit\_status".

- The job does not exist in the list that 'qstat' returns.
- The "exit\_status" is NOT 0.

### **saga::job::Canceled**

The current SPA does not support the Canceled state.

## 6 Adaptor Configuration File

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

### 6.1 File name and location of Adaptor configuration file

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

### 6.2 Configuration

#### 6.2.1 [saga.adaptors.pbspro\_job] section

name	Specified as “pbspro_job”. No change in typical use.
path	Specified as “\$[saga.location]/lib”. No change in typical use.
enabled	Specified as “false” when SPA is disabled. No change in typical use

#### 6.2.2 [saga.adaptors.pbspro\_job.cli] section

Reserved.

#### 6.2.3 [saga.adaptors.pbspro\_job.cli.description] section

JobContact	<p>Specifies the email address to receive from PBS server. The format is <i>mailto:user@host</i>. This JobContact is used as the default value when the <code>description_job_contact</code> is not specified. Also, this JobContact is always used in the case of jobs submitted by using <code>saga::job::service::run_job()</code>.</p> <p>Specifying this JobContact is mandatory to load SPA. If JobContact is not specified, SPA returns errors in being loaded.</p>
------------	--



## 7 SAGA API specification by SPA

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

### 7.1 `saga::job::service` class

#### 7.1.1 `service(rm)`

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

#### 7.1.2 `create_job(job_desc)`

Purpose	
Creates a <code>saga::job::job</code> object. This API checks following attributes of the <code>saga::job::description</code> . <ul style="list-style-type: none"><li>➤ <code>description_executable</code></li><li>➤ <code>description_interactive</code></li></ul>	
Inputs	
<code>job_desc</code>	Specify the <code>saga::job::description</code> to be submitted.
Outputs	
Returns a <code>saga::job::job</code> . The job status becomes <code>saga::job::New</code> .	
Exceptions	
<code>BadParameter</code>	Occurs if the mandatory attribute, <code>descriptin_executable</code> , is not specified or null.
<code>Not Implemented</code>	Occurs if the <code>description_interactive</code> is 'True'.

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

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

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

<b>Purpose</b>	
Submits a job without a saga::job::description	
<b>Inputs</b>	
commandline	Specifies a command to be executed.
hostname	The current SPA does not support
<b>Outputs</b>	
Returns the saga::job::job of the submitted job. The job status becomes saga::job::Running, saga::job::Done, or saga::job::Failed.	
<b>Exceptions</b>	
NoSuccess	Occurs when executing the 'qsub' command has problems.

### 7.1.5 list()

<b>Purpose</b>	
Gets the job list that PBS server controls.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
Returns SAGA JobID in the std::vector<std::string> type	

Exceptions	
NoSuccess	Occurs when executing the 'qstat' command has problems.

### 7.1.6 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.

### 7.1.7 get\_self()

Purpose	
The current SPA does not support.	
Inputs	
n/a	
Outputs	
n/a	
Exceptions	
Not Implemented	Always occurs.

## 7.2 saga::job::job class

### 7.2.1 get\_job\_id()

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

### 7.2.2 run()

<b>Purpose</b>	
Submits the job whose status is saga::job::New	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
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.

### 7.2.3 wait(timeout)

<b>Purpose</b>	
The current SPA does not support.	

<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

#### 7.2.4 cancel(timeout)

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

#### 7.2.5 get\_state()

<b>Purpose</b>	
Gets the state of this job.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
Returns the <code>saga::job::state</code>	
<b>Exceptions</b>	
NoSuccess	Occurs when executing the 'qstat' command has problems.

#### 7.2.6 get\_description()

<b>Purpose</b>
----------------

Returns the <code>saga::job::description</code> object of this job if the <code>saga::job::job</code> object corresponds to either of the following.	
<ul style="list-style-type: none"> <li>➤ The object is given by <code>saga::job::service::run_job()</code> .</li> <li>➤ The object is created by <code>saga::job::service::create_job()</code>.</li> </ul>	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
Returns a <code>saga::job::description</code>	
<b>Exceptions</b>	
DoesNotExist	Occurs if the <code>saga::job::job</code> object does not correspond to the above cases.

### 7.2.7 get\_stdn()

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

### 7.2.8 get\_stdout()

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

### 7.2.9 get\_stderr()

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

### 7.2.10 suspend()

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

### 7.2.11 resume()

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

### 7.2.12      **checkpoint()**

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

### 7.2.13      **migrate(job\_desc)**

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	
Not Implemented	Always occurs.

### 7.2.14      **signal(signal)**

<b>Purpose</b>	
The current SPA does not support.	
<b>Inputs</b>	
n/a	
<b>Outputs</b>	
n/a	
<b>Exceptions</b>	



Not Implemented	Always occurs.
-----------------	----------------

## 8 Source Files

### 8.1 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.

- `pbspro_job_adaptor.cpp`
- `pbspro_job_adaptor.hpp`
- `pbspro_job_service.cpp`
- `pbspro_job_service.hpp`
- `pbspro_job.cpp`
- `pbspro_job.hpp`
- `pbspro_job_adaptor.ini`
- *`pbspro_job_async.cpp`*
- *`pbspro_job_service_async.cpp`*
- *`pbspro_job_istream.hpp`*
- *`pbspro_job_ostream.hpp`*
- *`pbspro_job_stream.hpp`*

### 8.2 Source files related to PBSPro commands

The following files are newly created to implement SPA.

- `debug.hpp`
- `directives.hpp`
- `directives_impl.cpp`
- `directives_impl.hpp`
- `script.cpp`
- `script.hpp`
- `staging.hpp`
- `pbspro_cli.cpp`
- `pbspro_cli.hpp`
- `pbspro_cli_staging.cpp`
- `pbspro_cli_staging.hpp`
- `pbspro_helper.cpp`
- `pbspro_helper.hpp`

## 9 Class Reference

### 9.1 Namespace

SPA uses the following namespace.

<code>pbspro_job</code>	Contains whole SPA
<code>pbspro_job::cli</code>	Contains the classes and functions related to PBS Pro command executions.
<code>pbspro_job::helper</code>	Contains helper functions.

### 9.2 Class

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

#### 9.2.1 namespace `pbspro_job`

<code>adaptor (struct)</code>	The adaptor implementation inherited from the <code>saga::adaptor</code>
<code>job_cpi_impl</code>	The SPA implementation for the <code>saga::job::job</code> .
<code>job_srvice_cpi_impl</code>	The SPA implementation for the <code>saga::job::service</code>

#### 9.2.2 namespace `pbspro_job::cli`

<code>directives</code>	The interface that configures the PBS directives.
<code>directives_checker</code>	The interface that checks the PBS directives.
<code>directives_builder</code>	The interface that builds the PBS directives.
<code>directives_impl</code>	The class that configures the PBS directives.
<code>directives_checker_impl</code>	The class that checks the PBS directives.
<code>directives_builder_impl</code>	The class that builds the PBS directives.
<code>job_script</code>	The PBS script file class.
<code>job_script_builder</code>	The class that builds the PBS script files.
<code>_directives_checker_impl</code>	This class that checks the PBS directives is used by the <code>job_script_builder</code> class.
<code>file_transfer</code>	The class that defines the file transfer.
<code>file_transfer_parser</code>	The interface that parses the file transfer directives.

output_parser	The class that parses the PBSPro 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.

### 9.2.3 namespace pbspro\_job::helper

jobid_converter	The class that converts JobID formats between PBSPro to JobID and SAGA JobID.
-----------------	---

## 9.3 Functions

This section describes the functions belonging to no class.

### 9.3.1 namespace pbspro\_job::helper

convert_saga_job_state(pbspro_status)	The function that converts a PBSPro 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().