SAGA-NAREGI Adaptor System Specification

High Energy Accelerator Research Organization (KEK)
Computing Research Center

January 8, 2010

Index

1 Introduction 5
2 Overview of SNA 5
2.1 SNA operations 5
2.2 Scheme Definition to load SNA 5
2.3 JobID Format 6
2.4 Authorization Service 7
2.4.1 Sign on NAREGI Portal 7
2.4.2 Disable Certification Verification 7
3 How SNA works with NAREGI command tool 8
3.1 NAREGI commands used in SNA 8
3.2 How to use NAREGI commands by SAGA API 8
3.2.1 saga::job::service class 8
3.2.2 saga::job::job class 10
3.3 Command errors 12
4 NAREGI WFML creation 14
4.1 WFML structure 14
4.2 saga::job::description attributes vs WFML 16
4.2.1 Executable and Arguments 17
4.2.2 Environment Variables 18
4.2.3 Working Directory 19
4.2.4 Interactive mode 20
4.2.5 Standard output and error 21
4.2.6 File staging 21
4.2.7 Specify Max Wall time 24
4.2.8 Resource element 25
4.3 Options saga::job::description does not support 26
4.4 exportModel element creation 27
5 Job Status 29
6 Adaptor Configuration File 31
6.1 File name and location of Adaptor configuration file 31
6.2 Configuration 31
6.2.1 [saga.adaptors.naregi_job] section 31
6.2.2 [saga.adaptors.naregi_job.cli] section 31
6.2.3 [saga.adaptors.naregi_job.wfml.transfer] section 31
7 SAGA API specification by SNA 32

7.1 saga::job::service class 32
7.1.1 service(rm) 32
7.1.2 create_job(job_desc) 32
7.1.3 run_job(commandline, hostname, stdin_stream, stdout_stream, stderr_stream) 33
7.1.4 run_job(commandline, hostname) 33
7.1.5 list() 33
7.1.6 get_job(job_id) 34
7.1.7 get_self() 34
7.2 saga::job::job class 36
7.2.1 get_job_id() 36
7.2.2 run() 36
7.2.3 wait(timeout) 36
7.2.4 cancel(timeout) 37
7.2.5 get_state() 37
7.2.6 get_description() 38
7.2.7 get_stdin() 38
7.2.8 get_stdout() 38
7.2.9 get_stderr() 39
7.2.10 suspend() 39
7.2.11 resume() 39
7.2.12 checkpoint() 40
7.2.13 migrate(job_desc) 40
7.2.14 signal(signal) 40
8 Source Files 42
8.1 Source files related to Adaptor implementation 42
8.2 Source files related to NAREGI commands 42
8.3 Source files related to WFML creation 42
9 Class Reference 44
9.1 Namespace 44
9.2 Class 44
9.2.1 namespace naregi_job 44
9.2.2 namespace naregi_job::cli 45
9.2.3 namespace naregi_job::helper 45
9.2.4 namespace naregi_job::jsdl 45
9.2.5 namespace naregi_job::wfml 46

9.2.6 namespace naregi_job::wfml::jsdl 47
9.3 Functions 47
9.3.1 namespace naregi_job::cli 47
9.3.2 namespace naregi_job::helper 48

1 Introduction

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

2 Overview of SNA

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

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

2.1 SNA operations

NAREGI command line tool should be installed in the environment to use SNA for the reason that SNA required NAREGI command line tool to access NAREGI middleware. The following is the procedure that a SAGA application issues a NAREGI command via SAGA API.

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

2.2 Scheme Definition to load SNA

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

naregi://*localhost*/

naregi	Scheme name to load SNA
localhost	FQDN of the host executing SAGA application

This URL means that the SAGA application access NAREGI middleware by using NAREGI command line tool of the host, *localhost* , as backend commands. SNA can accept the scheme "any" because SAGA specification defines that SAGA application can select any adaptor by using "any" scheme. It is possible that SNA 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 NAREGI becomes like the following.

[naregi://*localhost*/] – [*NAREGI JOB_ID*]

naregi	Specify "naregi" to use NAREGI commands to access NAREGI jobs. Not specify "any" here.
localhost	FQDN of the host executing SAGA application
NAREGI JOB_ID	NAREGI JobID. This JobID should be specified as below. ➢ CID_*number* NAREGI JobID submitted by NAREGI command line tool. ➢ ID_*number* NAREGI JobID submitted by NAREGI GUI.

2.4 Authorization Service

2.4.1 Sign on NAREGI Portal

The current SAGA specification does not have an API to do authorization. Therefore, in order to use SNA in SAGA applications, the users should sign on the NAREGI Portal node by using the 'naregi-signon' command before to execute the SAGA applications.

SNA returns an exception if a user executes SAGA API without signing on NAREGI Portal.

2.4.2 Disable Certification Verification

SNA is always executed with the option '-N' to disable the Certification Verification.

3 How SNA works with NAREGI command tool

This chapter describes how SNA works with NAREGI command tool.

3.1 NAREGI commands used in SNA

The following table shows the list of NAREGI commands to be used by SNA.

SAGA API	NAREGI commands
saga::job::service::run_job()	naregi-simplejob-submit
saga::job::job::run()	naregi-job-submit
saga::job::service::list()	naregi-job-list

saga::job::job::get_state()
saga::job::service::get_job()
	naregi-job-status

saga::job::job::cancel()	naregi-job-cancel

3.2 How to use NAREGI commands by SAGA API

This chapter describes each SAGA API methods to use NAREGI commands. The SAGA API implementation for SNA 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 'naregi-simplejob-submit' command. The following is the explanation about the arguments of this API.

commandline	This string will be split into tokens. The first token is used as the argument EXECUTABLE of the command 'naregi-simplejob-submit'. The rest of tokens are used as the VALUE of the option '--argument=VALUE' if they are exist.
hostname	This string is used as the NAME of the option '--host=NAME'. The

	option '--host' is not used if this hostname is not specified.

SNA converts the NAREGI JobID of the naregi-simplejob-submit standard output to SAGA JobID. Then, SNA stores the SAGA JobID in the saga::job::attributes::jobid in saga::job::job object.

Example:

The following example shows the case that NAREGI JobID is CID_1319.

```
Submitting is done...

JobID=CID_1319
```

list()

This API gets a list of NAREGI JobID by using 'naregi-job-list' command with the option '--noheader' to avoid to get job list headers. SNA takes information of NAREGI JobID in the job list of the output by the naregi-job-list standard output. SNA converts all of the NAREGI JobID information to SAGA JobID and returns them in form of std::vector.

Example:

For example, if qstat returns like the flowing output,

```
CID_28544  saga-app   Done       2009/03/26 13:48:07 JST  2009/03/26 13:50:40 JST

CID_28543  saga-app   Exception  2009/03/26 13:34:02 JST  2009/03/26 13:36:27 JST

...

ID_50995   uname      Exception  2008/10/09 12:48:59 JST  2008/10/09 12:49:15 JST

ID_50994   uname      Done       2008/10/08 15:14:50 JST  2008/10/08 15:15:41 JST

...
```

SNA returns the following SAGA JobID.

```
[naregi://example.com/]-[CID_28544]

[naregi://example.com/]-[CID_28543]

[naregi://example.com/]-[ID_50995]

[naregi://example.com/]-[ID_50994]
```

get_job(jobid)

This API executes 'naregi-job-state' to check the availability of the job corresponding to the SAGA JobID specified in the argument jobid.

The argument jobid can specify the only jobs created by this saga::job::service object. This API cannot get the job objects created in different SAGA applications, even if the 'naregi-job-list' command can show the jobs.

3.2.2 saga::job::job class

run()

This API submits a job by using the 'naregi-job-submit' command. The 'naregi-job-submit' command requires WFML file as its argument. This API creates a WFML file based on the saga::job::description specified in the argument of the saga::job::service::create_job(). The WFML file is created as a temporary file in the current directory and removed right after the job completion. The file name of the temporary file becomes in the form of "wfml_*XXXXX*" by using mkstep(3).

Also this API gets the NAREGI JobID from the naregi-job-submit standard output, converts NAREGI JobID to SAGA JobID, and then stores the NAREGI JobID to saga::job::attoributes::jobid of the saga::job::job object.

Example:

The following example shows the case that NAREGI JobID is CID_1319.

```
Submitting is done...

JobID=CID_1319
```

cancel (timeout)

This API executes 'naregi-job-cancel' in order to cancel jobs. The 'naregi-job-cancel' command needs NAREGI JobID as its argument. The NAREGI JobID is created based on the SAGA JobID in the saga::job::attributes::jobid attributes of this object saga::job::job.

SNA verifies a successful job cancellation by the strings "Canceling is done…" of the standard output.

get_state()

This API executes 'naregi-job-status' in order to get a job status. NAREGI JobID will be specified in the argument of 'naregi-job-status'. The NAREGI JobID will be created based on the saga::job::attributes::jobid of the saga::job::job object. The command 'naregi-job-status' outputs the job information like the following.

```
saga-app(Done)
    mkdir#2(Done:www.naregi.org) ==> transfer#3(Done:www.naregi.org)
    transfer#3(Done:www.naregi.org) ==> program#1(Done:nrgclstr037.soum.co.jp)
    program#1(Done:nrgclstr037.soum.co.jp) ==> transfer#4(Done:www.naregi.org)
```

SNA gets the strings indicating the job state and converts the string to a saga::job::state value. In the above example, SNA gets "Done" in the string "saga-app(Done)" and then, converts to "saga::job::Done". Further information of the comparison between the job strings of NAREGI and saga::job::state is described in the chapter 5.

3.3 Command errors

SNA creates a saga::exception corresponding to the error messages the NAREGI command outputs when the executed NAREGI command by SNA returns an error. The SAGA application can see the error messages by the saga::exception::get_message().

The following shows error statuses assumed in using NAREGI commands, their error messages, and the SAGA exceptions corresponding to the errors.

Execute NAREGI commands without signing on the NAREGI Portal

Message	`[Error] Please sign-on again because of session timeout or invalidated session.`
	`(cause)`
	`java.io.FileNotFoundException: /tmp/.naregi=takando (No such file or directory)`

Exception	AuthenticationFailed

Session Time Over

Message	`[Error] Please sign-on again because of session timeout or invalidated session.`
	`(cause)`
	`Session timed out or unestablished yet.`

Exception	AuthenticationFailed

Execute a NAREGI command for the canceled or deleted job

Message	`Invalid job-id. [CID_41092] is not found.`

	`An exception occurs in canceling a job.`
	`Invalid job-id. [CID_41092] is not found.`

	`An exception occurs in deleting a job.`
	`Invalid job-id. [CID_41092] is not found.`

Exception	If the job object is created by the SAGA application, the job state moves to “Canceled”. If not, SNA returns the incorrect JobID error.

Specify a nonexistent WFML in executing the 'naregi-job-submit' command

Message	`An exception occurs in submitting a job.`
	`[Error] NAREGI-WFML file "xxxx" is not found.`

Exception	The current SNA does not define this exception.

Incorrect format of WFML

Message	`An exception occurs in submitting a job.`
	`result code [1009]`

Exception	The current SNA does not define this exception.

4 NAREGI WFML 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 needs to specify the job information in the argument of the saga::job::service::run_job(). SNA creates a saga::job::description object based on the API arguments, and then creates a NAREGI WFML by the object.

In the latter way, SAGA application should configure the job information in the saga::job::description object.

This chapter describes how to create a NAREGI WFML by SNA.

4.1 WFML structure

WFML is an XML like the following example.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<definitions xmlns="http://www.naregi.org/wfml/02" name="Workflow_Name">
  <activityModel>
    <activity name="Activity_Name">
      ...
    </activity>
    ...
  </activityModel>
  <compositionModel>
    <importModel/>
    <exportModel>
      <exportedActivity>
        ...
      </exportedActivity>
    </exportModel>
  </compositionModel>
</definitions>
```

SNA creates the activity elements based on the attributes specified in the saga::job::description. The activityModel element can include several activity elements. SNA creates three types of the activity elements as below.

- jsld activity

  This is the activity element that includes jsdl elements. The activity name is "program#*NUMBER*".
- mkdir activity

  This is the activity element that includes mkdir elements. The activity name is "mkdir#*NUMBER*".
- transfer activity

  This is the activity element that includes transfer elements. The activity name is "transfer#*NUMBER*".

The only one jsdl activity is created by the saga::job::description object. No or one mkdir activity is created due to its configuration. The number of transfer activities is the same number of the saga::job::attributes::description_file_transfer attributes. The number, "*#NUMBER*", of each activity name is specified in the order to add the activities to WFML. Typically, the number of the mandatory jsdl activity becomes 1, in other words, the activity name of the jsdl activity becomes "program#1".

The composisionModel element shows workflow. The composisionModel element includes importModel elements and exportModel elements. However, the importModel elements in the WFML created by SNA are always empty because the saga::job::description does not specify partial workflow. The further information about exportModel elements is described in the 4.4 section.

4.2 saga::job::description attributes vs WFML

The following table shows the corresponding table the saga::job::description attributes and WFML. The Activity column in the table shows the activity type corresponding to the attribute in saga::job::attributes column. The JSDL column shows the child elements to be configured when configuring a jsdl activity. SPA does not care about the attribute that is "Ignore" in the requirement column in the table.

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	Activity	JSDL	Requirement
description_executable	jsdl	Executable	Required
description_arguments	jsdl	Argument	Option
description_environment	jsdl	Environment	Option
description_working_directory	jsdl, mkdir	WorkingDirectory	Option
description_interactive			False
description_input			Ignore
description_output			Option
description_error			Option
description_file_transfer	transfer		Option
description_cleanup			Ignore
description_job_start_time			Ignore
Description_totall_cpu_time			Ignore
description_wall_time_limit	jsdl	WallTimeLimit	Option
description_total_physical_memory			Ignore
description_cpu_architecture			Ignore

description_operating_system_type	jsdl	OperatingSystemName	Option
description_candidate_hosts	jsdl	HostName	Option
description_queue			Ignore
description_job_contact			Ignore
description_job_project			Ignore
description_spmd_variation			Ignore
description_total_cpu_count			Ignore
description_number_of_proceses			Ignore
description_processes_per_host			Ignore
description_threads_per_process			Ignore

4.2.1 Executable and Arguments

The values of description_executable and description_arguments are specified in the Executable and Argument elements of JSDL respectively. The description_executable must be specified. If the description_executable is not specified, the exceptions are happen in the saga::job::service::create_job().

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: WFML sample

```
<jsdl>
  ...
    <Executable>/usr/bin/ci</Executable>
    <Argument>-m"add #include"</Argument>
    <Argument>sample.c</Argument>
  ...
</jsdl>
```

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*". SNA splits the sting in "*name*" and "*value*". The "*name*" and "*value*" is specified in the name and value attribute of the Environment element respectively.

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: WFML sample

```
<jsdl>
  ...
    <Environment name="FOO">HOGE</Environment>
    <Environment name="BAR">FUGA</Environment>
  ...
</jsdl>
```

4.2.3 Working Directory

The working directory defined in description_working_directory is specified in the WorkingDirectory element of JSDL. Also, a mkdir activity is added to WFML in order to create this working directory on the job execution host. If the description_working_directory is not specified, SNA will not create the WFML element to mkdir and the working directory becomes the home directory.

How to configure a mkdir activity

In the case that a mkdir activity is to be created by the description_working_directory, SNA converts the specified path to the WFML own URL.

default://*Management_Node*/*path*

The *Management_Node* is specified as "/FixMe(*Activity_Name*)eMxiF" based on the activity name of jsdl activity. SAGA application cannot change this. SNA handles the specified *path* as below.

- The working directory is specified as Relative path
  The working directory becomes the relative directory to the home directory, "~".
- The working directory is specified as Absolute path
  The specified path is used as the working directory directly.

Example: SAGA application example

```
namespace sja = saga::job::attributes;

saga::job::description jd;

jd.set_vector_attribute(sja::description_working_directory, "work");
```

Example: WFML sample

```
<activity name="program#1">
  <jsdl>
  ...
     <WorkingDirectory>work</WorkingDirectory>
  ...
  </jsdl>
</activity>
...
<activity name="mkdir#2">
  <mkdir WallTimeLimit="300">
    <target name="default:///FixMe(program%231)eMxiF/~/work">
  </mkdir>
</activity>
```

Note that the WallTimeLimit attribute of the mkdir activity is not used by SS(Super Scheduler) in NAREGI middleware. SAGA application cannot configure the attribute.

4.2.4 Interactive mode

The current SNA does not support the interactive mode. SNA can accept the only "false" value of the description_interactive. If the specified value is "true", SNA returns the exception, "BadParameter". SNA does not use the description_input for the same reason, neither. The description_input is ignored even if the description_input is specified..

4.2.5 Standard output and error

SNA supports the standard output/error. TBD.

4.2.6 File staging

SNA creates transfer activities by the specified values in the description_file_transfer.

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 NAREGI specification. If other characters are specified here, SNA returns exceptions.
remote_file	Only absolute or relative path can be specified. URL can NOT be specified.

How to configure transfer activities

The transfer activity requires URLs expressed in the own style of WFML. When SAGA application specifies a relative path in the description_file_transfer, SNA converts the specified path to the WFML URL style.

***local_file* path**

The *local_file* is used as a file on the local host that is executing SAGA application. The *local_file* path becomes a relative path to the current directory if the *local_file* path is a relative path. The local host name becomes the host name in the URL specified in the arguments of the saga::job::service constructor.

(Example 1) The local host is "example.com", the *local_file* is specified as "tiger.eps", and the current directory is /home/user/sample. SNA converts the URL to the following in this example.

```
default://example.com/home/user/sample/tiger.eps
```

(Example 2) The *local_file* path is used directly if the *local_file* path is an absolute path. If the *local_file* is specified as "/tmp/tiger.eps", SNA converts the URL to the following.

```
default://example.com/tmp/tiger.eps
```

remote_file **path**

The *remote_file* is used as a file on the remote host that is executing the job. The *remote_file* path becomes a relative path to the working directory if the *remote_file* path is a relative path. As described in the 4.2.3 section, the working directory becomes home directory or the directory specified in the description_working_directory. The remote host name is specified as "/Fixme(*Activity_Name*)eMxiF" based on the activity name of the jsdl activity. SAGA application cannot change this name.

(Example 1) The *remote_file* is specified as "tiger.eps", and the description_working_dir ectory is "work". SNA converts the URL to the following in this example.

```
default:///FixMe(program¥%231)eMxiF/~/work/tiger.eps
```

(Example 2) If the description_working_directory is an absolute path like a "/tmp/work" in the above example1, SNA converts the URL to the following.

```
default:///FixMe(program¥%231)eMxiF/tmp/work/tiger.pdf
```

(Example 3) The *remote_file* path is used directly if the *remote_file* path is an absolute path. If the *remote_file* is specified as "/tmp/tiger.pdf", SNA converts the URL to the following.

```
default:///FixMe(program¥%231)eMxiF/tmp/tiger.pdf
```

operator

The source and target of the transfer activities are configured based on the operator types as below.

- operator is specified as '>'
    - *local_file* becomes source
    - *remote_file* becomes target
- operator is specified as '>'
    - *remote_file* becomes source
    - *local_file* becomes target

Max wall time

The max wall time cannot be specified in the description_file_transfer even if transfer activities under NAREGI middleware can specify the max wall time in the WallTimeLimit attribute. Therefore, SAGA application cannot configure the max wall time. SNA uses the same value in all transfer activities and the value is specified in the adaptor configuration file.

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("tiger.eps > tiger.eps");
ft.push_back("tiger.pdf < tiger.pdf");


jd.set_attribute(sja::description_working_directory, "work");
jd.set_vector_attribute(sja::description_file_transfer, ft);
```

Example: WFML sample

```
<activity name="program#1">
  <jsdl>
  ...
      <WorkingDirectory>work</WorkingDirectory>
  ...
  </jsdl>
</activity>
...
<activity name="mkdir#2">
  <mkdir WallTimeLimit="300">
    <target name="default:///FixMe(program%231)eMxiF/~/work">
  </mkdir>
</activity>
<activity name="transfer#3">
  <transfer WallTimeLimit="60">
    <source name="default://example.com/home/user/sample/tiger.eps">
    <target name="default:///FixMe(program%231)eMxiF/~/work/tiger.eps">
  </transfer>
</activity>
<activity name="transfer#4">
  <transfer WallTimeLimit="60">
    <source name="default:///FixMe(program%231)eMxiF/~/work/tiger.pdf">
    <target name="default://example.com/home/user/sample/tiger.pdf">
  </transfer>
</activity>
```

4.2.7 Specify Max Wall time

WFML has several max wall time configurations as below.

- WallTimeLimit element of POSIXApplication element
- WallTimeLimit attribute of transfer element

➢ WallTimeLimit attribute of mkdir element

SNA uses the value of the description_wall_time for the WallTimeLimit element of the POSIXApplication element.

The WallTimeLimit of the transfer element uses the value in the adaptor configuration file. The WallTimeLimit of the mkdir element is specified as a certain value by SNA because SS does not use the WallTimeLimit. Both WallTimeLimit cannot be specified by SAGA application.

4.2.8 Resource element

The std::vector values in the description_operating_system_type are used as the values of the OperatingSystemName element directly. The std::vector values in the description_candidate_hosts are used as the values of the HostName element directly. Both values are output to WFML without being checked by SNA. If the specified values are not correct, NAREGI middleware returns errors.

Example: SAGA application example

```
namespace sja = saga::job::attributes;
namespace sjad = saga::job::attributes::detail;

saga::job::description jd;

std::vector <std::string> ostype;
ostype.push_back (sjad::description_operating_system_linux);

jd.set_vector_attribute(sja::description_operating_system_type, ostype);

std::vector<std::string> hosts;
hosts.push_back("kek-sna131.soum.co.jp");
hosts.push_back("kek-sna132.soum.co.jp");

jd.set_vector_attribute(sja::description_candidate_hosts, hosts);
```

Example: WFML sample

```
<activity name="program#1">
  <jsdl>
      ...
      <JobDescription>
      ...
        <Resources>
          <CandidateHosts>
            <HostName>kek-sna131.soum.co.jp</HostName>
            <HostName>kek-sna132.soum.co.jp</HostName>
          </CandidateHosts>
          <OperatingSystem>
            <OperatingSystemType>
              <OperatingSystemName>LINUX</OperatingSystemName>
            </OperatingSystemType>
          </OperatingSystem>
      ...
        </Resources>
      </JobDescription>
      ...
  </jsdl>
</activity>
```

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

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

definitions	The name attribute of the definitions element is specified as the fixed value “saga-app”.
JobName	The JobName value in JobIdetification element of the jsdl activity is specified as the fixed value “Program”. This name is used to be shown in the local scheduler.
LowerBoundedRange	The LowerBoundedRange value in IndividualCPUCount element of the jsdl activity is specified as the fixed value “1”.
exportedActivityInfo	The name attribute of the exportedActivityInfo element is

	specified as the fixed value “saga-app”. This name is used in the Name column of naregi-job-list and used in the output of naregi-job-status.
controlModel	The contrlIn attribute of the controlModel element is specified as the same value of the name attribute of exportedActivityInfo element. If both values are different, the submitted job status becomes “Missing”.

4.4 exportModel element creation

The exportModel element has the exportedActivity element as child elements. The exportedActivity element has the exportedActivityInfo and controlModel elements as child elements. The controlModel element specifies the order of activity executions by using the controlLink elements. SNA defines the order of the activity executions as below.

1. If the only jsdl activity exists, SNA creates a controlLink that has only jsdl activity.
2. If the mkdir activity also exists, SNA puts this activity as the first activity.
3. If the transfer activities exist, SNA puts the activities before/after the jsdl activity depending on each operator.
   - If the operator is ‘>’, the transfer activity will be put after the mkdir activity and before the jsdl activity.
   - If the operator is ‘<’, the transfer activity will be put after jsdl activity.

Example: WFML sample

```
  <activityModel>
  ...
  </activityModel>
  <compositionModel>
    <importModel/>
    <exportModel>
      <exportedActivity>
        <exportedActivityInfo name="saga-app" />
        <controlModel controlIn="saga-app">
          <controlLink label="WFTGEN" source="mkdir#2" target="transfer#3" />
          <controlLink label="WFTGEN" source="transfer#3" target="program#1" />
          <controlLink label="WFTGEN" source="program#1" target="transfer#4" />
        </controlModel>
      </exportedActivity>
    </exportModel>
  </compositionModel>
</definitions>
```

In the case of activityModel including only jsdl activity, the controlModel element becomes the following.

```
<controlModel controlIn="saga-app">
  <controlLink label="WFTGEN" source="program#1" />
</controlModel>
```

5 Job Status

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

saga::job::state	**NAREGI Job Status**
saga::job::New	(status right after a job object creation.)
saga::job::Running	Running
saga::job::Suspend	n/a
saga::job::Done,	Done
saga::job::Failed	Exception, Missing
saga::job::Canceled	(cance() is successfully completed.)

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. NAREGI does not have this job state because this state is the state before submitting the job.

saga::job::Running

This state, Running, is set in 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. If the process to submit a job fails, the job state does not enter the saga::job::Running and is still in the saga::job::New instead.

saga::job::Suspended

The current SNA does not support the Suspended state.

saga::job::Done

This state, Done, is set in the saga::job::job object when the saga::job::job::get_state() returns "Done".

saga::job::Failed

This state, Failed, is set in the saga::job::job object when the saga::job::job::get_state() returns "Exception" or "Missing".

saga::job::Canceled

This state, Canceled, is set in the saga::job::job object when the job is canceled by the saga::job::job::cancel(). Also, the saga::job::Canceled is set if SNA cannot get the job status when executing 'naregi-job-status' for the job in the saga::job::Runing state. In this case, it is assumed that SAGA application cannot refer to the job information because the 'naregi-job-cancel' command is issued out of the SAGA application.

6 Adaptor Configuration File

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

6.1 File name and location of Adaptor configuration file

The file name of the SNA adaptor configuration file is “saga_adaptor_naregi_job.ini”. The ini file is typically installed in the directory, $SAGA_LOCATION/share/saga.

6.2 Configuration

6.2.1 [saga.adaptors.naregi_job] section

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

6.2.2 [saga.adaptors.naregi_job.cli] section

Reserved.

6.2.3 [saga.adaptors.naregi_job.wfml.transfer] section

This section defines the default values of the transfer activity.

WallTimeLimit	This defines the WallTimeLimit attribute in the transfer elements. The default value is “300” if this attribute is not specified.

7 SAGA API specification by SNA

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

7.1 saga::job::service class

7.1.1 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.

7.1.2 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'.

7.1.3 run_job(commandline, hostname, stdin_stream, stdout_stream, stderr_stream)

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

7.1.4 run_job(commandline, hostname)

Purpose	
Submits a job without a saga::job::description	
Inputs	
commandline	Specifies a command to be executed.
hostname	The hostname to be submitted. This hostname is used in the description_candidate_hosts attributes in the saga::job::description but the only one hostname can be specified here. If not specified, SS defines the hostname.
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	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
NoSuccess	Occurs when executing the NAREGI command has problems.

7.1.5 list()

Purpose
Gets the job list that NAREGI middleware controls.

Inputs	
n/a	
Outputs	
Returns SAGA JobID in the std::vector<std::string> type	
Exceptions	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
NoSuccess	Occurs when executing the NAREGI 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	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
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 NAREGI command has problems.

7.1.7 get_self()

Purpose	
The current SNA 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()

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

7.2.2 run()

Purpose	
Submits the job whose status is saga::job::New	
Inputs	
n/a	
Outputs	
n/a	
Exceptions	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
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 NAREGI command has problems.

7.2.3 wait(timeout)

Purpose

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

7.2.4 cancel(timeout)

Purpose	
Cancel the job execution. The job state becomes saga::job::Canceled when this cancellation is successfully completed or when the job is already canceled by other reasons.	
Inputs	
timeout	SNA does not support timeout argument.
Outputs	
n/a	
Exceptions	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
NoSuccess	Occurs when executing the NAREGI command has problems.

7.2.5 get_state()

Purpose	
Gets the state of this job.	
Inputs	
n/a	
Outputs	
Returns the saga::job::state	
Exceptions	
AuthenticationFailed	Occurs when signing on the NAREGI Portal is not completed.
NoSuccess	Occurs when executing the ‘qstat’ command has problems.

7.2.6 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.

7.2.7 get_stdin()

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

7.2.8 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.

7.2.9 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.

7.2.10 suspend()

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

7.2.11 resume()

Purpose	
The current SNA does not support.	
Inputs	

n/a	
Outputs	
n/a	
Exceptions	
Not Implemented	Always occurs.

7.2.12 checkpoint()

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

7.2.13 migrate(job_desc)

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

7.2.14 signal(signal)

Purpose
The current SNA does not support.

Inputs	
n/a	
Outputs	
n/a	
Exceptions	
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.

- naregi_job_adaptor.cpp
- naregi_job_adaptor.hpp
- naregi_job_service.cpp
- naregi_job_service.hpp
- naregi_job.cpp
- naregi_job.hpp
- naregi_job_adaptor.ini
- *naregi_job_async.cpp*
- *naregi_job_service_async.cpp*
- *naregi_job_istream.hpp*
- *naregi_job_ostream.hpp*
- *naregi_job_stream.hpp*

8.2 Source files related to NAREGI commands

The following files are newly created to implement SNA.

- naregi_cli.cpp
- naregi_cli.hpp
- naregi_helper.cpp
- naregi_helper.hpp
- desc_builder.cpp
- desc_builder.hpp

8.3 Source files related to WFML creation

The following files are newly created to implement SNA, specifically for WFML creation.

- wfml/debug.hpp
- wfml/jsdl.hpp

- wfml/jsdl_formatter.hpp
- wfml/jsdl_staging.hpp
- wfml/m.txt
- wfml/msg.txt
- wfml/utf_commit.sh
- wfml/wfml.hpp
- wfml/wfml_jsdl_writer.cpp
- wfml/wfml_jsdl_writer.hpp
- wfml/wfml_writer.cpp
- wfml/wfml_writer.hpp
- wfml/workflow.cpp
- wfml/workflow.hpp
- wfml/writer.cpp
- wfml/writer.hpp
- wfml/xml.cpp
- wfml/xml.hpp

9 Class Reference

9.1 Namespace

SNA uses the following namespace.

naregi_job	Contains whole SNA
naregi_job::cli	Contains the classes and functions related to NAR EGI command executions.
naregi_job::helper	Contains helper functions.
naregi_job::jsdl	Contains the classes to build JSDL structures.
naregi_job::jsdl::elements	Contains the const variables that are used as ele ment names in JSDL.
naregi_job::jsdl::jsdl_posix	Contains the const variables that are used as the names of the POSIX Application element in JSDL.
naregi_job::jsdl::attribute	Contains the const variables that are used as attri bute names in JSDL.
naregi_job::wfml	Contains the classes to build WFML structures.
naregi_job::wfml::activity_type	Contains the const variables that are used as acti vity names in WFML.
naregi_job::wfml::attribute	Contains the const variables that are used as attri bute names in WFML.
naregi_job::wfml::element	Contains the const variables that are used as ele ment names in WFML.
naregi_job::wfml::jsdl	Contains the classes to build JSDL parts in WFM L structures.

9.2 Class

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

9.2.1 namespace naregi_job

adaptor (struct)	The adaptor implementation inherited from the saga::a daptor
job_cpi_impl	The SNA implementation for the saga::job::job.
job_srvice_cpi_impl	The SNA implementation for the saga::job::service
description_builder_impl	The implementation of the naregi_job::jsdl::description_

	builder
file_transfer_parser_impl	The implementation of the naregi_job::jsdl::file_transfer_parser

9.2.2 namespace naregi_job::cli

output_parser	The class that parses the NAREGI command outputs.
error_msg	The class that checks the error messages of NAREGI command outputs.

9.2.3 namespace naregi_job::helper

jobid_converter	The class that converts JobID formats between NAREGI JobID and SAGA JobID.
tempfile	The temporary file class for WFML creations.

9.2.4 namespace naregi_job::jsdl

application_impl	The Application interface of the JSDL Application element.
description	The class that contains the classes to built JSDL structures. The classes are jsdl_job_identity, jsdl_application, jsdl_resources, and jsdl_data_staging.
description_builder	The interface to build JSDL.
file_transfer	The class that defines file transfer.
file_transfer_parser	The parser interface of file transfer directives.
jsdl_job_identity	The JSDL JobIdentity element class.
jsdl_application	The JSDL Application element class.
posix_application	The POSIX Application element class. The implementation of the application_impl class.
jsdl_resources	The JSDL Resource element class.
jsdl_data_staging	The JSDL DataStaging element class.
job_identity_formatter	The class that formats the JobIdentity element.

posix_application_formatter	The class that formats the POSIXApplication elemen t.
resources_formatter	The class that formats the Resource element.

9.2.5 namespace naregi_job::wfml

activity_writer	The class that writes the WFML activity element.
activity_child_writer	The base class of the classes that writes the WFML a ctivity child element.
model_writer	The base class of the classes that writes the WFML model element.
activity_model_writer	The WFML activity_model element class
composition_model_writer	The WFML composition_model element class
element_writer	The base class of the classes that writes the WFML el ement.
export_model_writer	The class that writes the WFML export_model elemen t.
exported_activity_writer	The class that writes the WFML exported_activity ele ment.
import_model_writer	The class that writes the WFML import_model elemen t.
jsdl_element_writer	The class that writes the WFML jsdl element.
mkdir_element_writer	The class that writes the WFML mkdir element.
transfer_element_writer	The class that writes the WFML transfer element.
workflow_writer	The class that writes the NAREGI workflow (WFML).
writer	The base class of writer classes.
xml_formatter	The class that formats the XML element.
xml_tag	The XML tag class.
xml_writer	The base class that writes XML.
workflow	The NAREGI workflow class.
workflow_builder	The class that builds the NAREGI workflow.
activity_factory	The class that creates the workflow activity.
activity	The workflow activity base class.
jsdl_activity	The workflow jsdl activity class.
mkdir_activity	The workflow mkdir activity class.

transfer_activity	The workflow transfer activity class.
control_link	The workflow contrl_link definition class
staging_path_builder	The class that builds path names for the workflow file staging.

9.2.6 namespace naregi_job::wfml::jsdl

job_identity_writer	The class that writes the JobIdentity element.
extention_writer	The class that writes the Applictaion extention element.
application_writer	The class that writes the Application element.
resources_writer	The class that writes the Resource element.
posix_application_writer	The class that writes the POSIXApplication element.
posix_application_writer_extention	The class that writes the POSIXApplication extention element.
jobdefinition_formatter	The class that formats the JobDefinition element.
staging_path_builder	The class that writes the JobDefinition element.

9.3 Functions

This section describes the functions belonging to no class.

9.3.1 namespace naregi_job::cli

execute(command, options, arg)	The function that executes commands.
naregi_simplejob_submit(jd, id, os)	The function that executes the naregi-simplejob-submit command.
naregi_job_list(ids, os)	The function that executes the naregi_job_list command.
naregi_job_submit(wf, id, os)	The function that executes the naregi_job_submit command.
naregi_job_status(id, status, os)	The function that executes the naregi_job_status command.
naregi_job_cancel(id, os)	The function that executes the naregi_job_canc

	el command.

9.3.2 namespace naregi_job::helper

convert_saga_job_state(naregi_status)	The function that converts a NAREGI Job s tring to a SAGA state string.
create_saga_job_description(jd, cmd, host)	The function that builds a saga::job::descript ion for the saga::job::service::run_job().
split_command_line(cmd, executable, options)	The function that splits command line strin gs for the saga::job::service::run_job().