**SAGA-NAREGI Adaptor**

**System Specification**

High Energy Accelerator Research Organization (KEK)

Computing Research Center

January 8, 2010

Index

[1 Introduction 5](#_Toc250714719)

[2 Overview of SNA 5](#_Toc250714720)

[2.1 SNA operations 5](#_Toc250714721)

[2.2 Scheme Definition to load SNA 5](#_Toc250714722)

[2.3 JobID Format 6](#_Toc250714723)

[2.4 Authorization Service 7](#_Toc250714724)

[2.4.1 Sign on NAREGI Portal 7](#_Toc250714725)

[2.4.2 Disable Certification Verification 7](#_Toc250714726)

[3 How SNA works with NAREGI command tool 8](#_Toc250714727)

[3.1 NAREGI commands used in SNA 8](#_Toc250714728)

[3.2 How to use NAREGI commands by SAGA API 8](#_Toc250714729)

[3.2.1 saga::job::service class 8](#_Toc250714730)

[3.2.2 saga::job::job class 10](#_Toc250714731)

[3.3 Command errors 12](#_Toc250714732)

[4 NAREGI WFML creation 14](#_Toc250714733)

[4.1 WFML structure 14](#_Toc250714734)

[4.2 saga::job::description attributes vs WFML 16](#_Toc250714735)

[4.2.1 Executable and Arguments 17](#_Toc250714736)

[4.2.2 Environment Variables 18](#_Toc250714737)

[4.2.3 Working Directory 19](#_Toc250714738)

[4.2.4 Interactive mode 20](#_Toc250714739)

[4.2.5 Standard output and error 21](#_Toc250714740)

[4.2.6 File staging 21](#_Toc250714741)

[4.2.7 Specify Max Wall time 24](#_Toc250714742)

[4.2.8 Resource element 25](#_Toc250714743)

[4.3 Options saga::job::description does not support 26](#_Toc250714744)

[4.4 exportModel element creation 27](#_Toc250714745)

[5 Job Status 29](#_Toc250714746)

[6 Adaptor Configuration File 31](#_Toc250714747)

[6.1 File name and location of Adaptor configuration file 31](#_Toc250714748)

[6.2 Configuration 31](#_Toc250714749)

[6.2.1 [saga.adaptors.naregi\_job] section 31](#_Toc250714750)

[6.2.2 [saga.adaptors.naregi\_job.cli] section 31](#_Toc250714751)

[6.2.3 [saga.adaptors.naregi\_job.wfml.transfer] section 31](#_Toc250714752)

[7 SAGA API specification by SNA 32](#_Toc250714753)

[7.1 saga::job::service class 32](#_Toc250714754)

[7.1.1 service(rm) 32](#_Toc250714755)

[7.1.2 create\_job(job\_desc) 32](#_Toc250714756)

[7.1.3 run\_job(commandline, hostname, stdin\_stream, stdout\_stream, stderr\_stream) 33](#_Toc250714757)

[7.1.4 run\_job(commandline, hostname) 33](#_Toc250714758)

[7.1.5 list() 33](#_Toc250714759)

[7.1.6 get\_job(job\_id) 34](#_Toc250714760)

[7.1.7 get\_self() 34](#_Toc250714761)

[7.2 saga::job::job class 36](#_Toc250714762)

[7.2.1 get\_job\_id() 36](#_Toc250714763)

[7.2.2 run() 36](#_Toc250714764)

[7.2.3 wait(timeout) 36](#_Toc250714765)

[7.2.4 cancel(timeout) 37](#_Toc250714766)

[7.2.5 get\_state() 37](#_Toc250714767)

[7.2.6 get\_description() 38](#_Toc250714768)

[7.2.7 get\_stdin() 38](#_Toc250714769)

[7.2.8 get\_stdout() 38](#_Toc250714770)

[7.2.9 get\_stderr() 39](#_Toc250714771)

[7.2.10 suspend() 39](#_Toc250714772)

[7.2.11 resume() 39](#_Toc250714773)

[7.2.12 checkpoint() 40](#_Toc250714774)

[7.2.13 migrate(job\_desc) 40](#_Toc250714775)

[7.2.14 signal(signal) 40](#_Toc250714776)

[8 Source Files 42](#_Toc250714777)

[8.1 Source files related to Adaptor implementation 42](#_Toc250714778)

[8.2 Source files related to NAREGI commands 42](#_Toc250714779)

[8.3 Source files related to WFML creation 42](#_Toc250714780)

[9 Class Reference 44](#_Toc250714781)

[9.1 Namespace 44](#_Toc250714782)

[9.2 Class 44](#_Toc250714783)

[9.2.1 namespace naregi\_job 44](#_Toc250714784)

[9.2.2 namespace naregi\_job::cli 45](#_Toc250714785)

[9.2.3 namespace naregi\_job::helper 45](#_Toc250714786)

[9.2.4 namespace naregi\_job::jsdl 45](#_Toc250714787)

[9.2.5 namespace naregi\_job::wfml 46](#_Toc250714788)

[9.2.6 namespace naregi\_job::wfml::jsdl 47](#_Toc250714789)

[9.3 Functions 47](#_Toc250714790)

[9.3.1 namespace naregi\_job::cli 47](#_Toc250714791)

[9.3.2 namespace naregi\_job::helper 48](#_Toc250714792)

# Introduction

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

# 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.

## 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.

## 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”.

## 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. |

## Authorization Service

### 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.

### Disable Certification Verification

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

# How SNA works with NAREGI command tool

This chapter describes how SNA works with NAREGI command tool.

## 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 |

## 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.

### 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.

### 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.

## 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. |

# 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.

## 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.

## 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 |

### 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>

### 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>

### 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.

### 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..

### Standard output and error

SNA supports the standard output/error. TBD.

### 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>

### 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.

### 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>

## 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”. |

## 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>

# 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.

# Adaptor Configuration File

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

## 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.

## Configuration

### [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 |

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

Reserved.

### [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. |

# 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.

## 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 SNA 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 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. |

### 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. |

### 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. |

### get\_self()

|  |  |
| --- | --- |
| **Purpose** | |
| The current SNA 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** | |
| 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. |

### wait(timeout)

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

### 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. |

### 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. |

### 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 SNA 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 SNA does not support. | |
| **Inputs** | |
| n/a |  |
| **Outputs** | |
| n/a | |
| **Exceptions** | |
| Not Implemented | Always occurs. |

### resume()

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

### checkpoint()

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

### migrate(job\_desc)

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

### signal(signal)

|  |  |
| --- | --- |
| **Purpose** | |
| The current SNA 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.

* 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*

## 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

## 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

# Class Reference

## Namespace

SNA uses the following namespace.

|  |  |
| --- | --- |
| naregi\_job | Contains whole SNA |
| naregi\_job::cli | Contains the classes and functions related to NAREGI 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 element 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 attribute 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 activity names in WFML. |
| naregi\_job::wfml::attribute | Contains the const variables that are used as attribute names in WFML. |
| naregi\_job::wfml::element | Contains the const variables that are used as element names in WFML. |
| naregi\_job::wfml::jsdl | Contains the classes to build JSDL parts in WFML structures. |

## Class

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

### namespace naregi\_job

|  |  |
| --- | --- |
| adaptor (struct) | The adaptor implementation inherited from the saga::adaptor |
| 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 |

### 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. |

### 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. |

### 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 element. |
| resources\_formatter | The class that formats the Resource element. |

### 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 activity 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 element. |
| export\_model\_writer | The class that writes the WFML export\_model element. |
| exported\_activity\_writer | The class that writes the WFML exported\_activity element. |
| import\_model\_writer | The class that writes the WFML import\_model element. |
| 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. |

### 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. |

## Functions

This section describes the functions belonging to no class.

### 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\_cancel command. |

### namespace naregi\_job::helper

|  |  |
| --- | --- |
| convert\_saga\_job\_state(naregi\_status) | The function that converts a NAREGI 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(). |