GWD-R.xx
SAGA-RG
Version: 0.1

Andre Merzky (CCT/LSU)

January 15, 2008

SAGA API Extension: Remote Procedure Calls, Version 2

Status of This Document

This document provides information to the grid community, proposing a standard for a DRMAA extension to the Simple API for Grid Applications (SAGA). This document is intendeded to be used as input to the definition of language specific bindings for this API extension, and as reference for implementors of these language bindings. Distribution of this document is unlimited.

Copyright Notice

Abstract

This document provides information to the grid community, proposing a standard for a DRMAA extension to the Simple API for Grid Applications (SAGA). As such it depends upon the SAGA Core API Specification [1], und upon the DRMAA IDL specfication [?].

Contents

1 Introduction

The 'Distributed Resource Management Application API Working Group' (DRMAA-WG) in OGF has defined a high level API for application level job submission and management. That API is C-oriented, but was also mapped to other languages. Several implementations of the DRMAA API exist and are in use, and the specification has completed the OGF document process as a 'Recommended OGF Standard'. An IDL rendering of the DRMAA standard is currently being created by the DRMAA-WG.

The 'Simple Api for Grid Applications' Working Group (SAGA-WG) in OGF recently started an effort for a uniform OGF API, which is supposed to cover all high level application programming aspects of the OGF standardization landscape. The SAGA Core API specification has passed its first public comment period, and is at the moment advancing in the OGF document proces.

The SAGA API is modular, and consists of (i) a set of nonfunctional packages (the SAGA Look & Feel), and (ii) a set of functional packages, which provide the respective programming paradigms to the Grid application programmers. The SAGA Core API includes, amongst others, a package for job submission and management. That package builds upon the experiences from various OGF groups and from other Grid APIs (BES, DRMAA, JSDL, GAT, CoG, GridSphere, . . .).

This document describes another job submission and managment package for the SAGA API, which is supposed to provide an semantically *exact* and complete rendering of the DRMAA-API in the SAGA framework, i.e. with the SAGA Look & Feel applied. That document serves three purposes:

1. prove that a DRMAA rendering in SAGA is possible,
2. allow for an easy comparison between the original SAGA job API and the DRMAA API,

3. pave the path for a DRMAA API package in SAGA for the planned DRMAA version 2.0.

2 SAGA DRMAA API

2.1 Introduction

2.2 Specification

```
package saga.drmaa
{
  //
  // Mapping of DRMAA exceptions to SAGA exceptions
  //
  // exception AlreadyActiveSession            = - not needed -
  // exception Authorization                   = AuthorizationFailed
  // exception ConflictingAttributeValues      = BadParameter
  // exception DefaultContactString            = NoSuccess
  // exception DeniedByDrm                     = PermissionDenied, NoSuccess
  // exception DrmCommunication                = NoSuccess
  // exception DrmsExit                        = - not needed -
  // exception DrmsInit                        = - not needed -
  // exception ExitTimeout                     = Timout
  // exception HoldInconsistentState           = IncorrectState
  // exception IllegalState                    = IncorrectState
  // exception Internal                        = NoSuccess
  // exception InvalidArgument                 = BadParameter
  // exception InvalidAttributeFormat          = BadParameter
  // exception InvalidAttributeValue           = BadParemeter
  // exception InvalidContactString            = IncorrectURL, BadParameter
  // exception InvalidJob                      = DoesNotExist
  // exception InvalidJobTemplate              = - not needed -
  // exception NoActiveSession                 = - not needed -
  // exception NoDefaultContactStringSelected  = - not needed -
  // exception OutOfMemory                     = - NoSuccess -
  // exception ReleaseInconsistentState        = IncorrectState
  // exception ResumeInconsistentState         = IncorrectState
  // exception SuspendInconsistentState        = IncorrectState
  // exception TryLater                        = TimeOut
  // exception UnsupportedAttribute            = BadParameter
  //

  enum job_control_action
  {
    Suspend,
    Resume,
    Hold,
    Release,
    Terminate
  }

  enum job_state
  {
    Undetermined,
    QueuedActive,
    SystemOnHold,
    UserOnHold,
    UserSystemOnHold,
    Running,
    SystemSuspended,
    UserSuspended,
    UserSystemSuspended,
    Done,
    Failed
  }

  enum job_submission_state
  {
    HoldState,
    ActiveState
  }

  struct file_transfer_mode
  {
    bool transfer_input_stream;
    bool transfer_output_stream;
    bool transfer_error_stream;
  }

  struct version
  {
    int major;
    int minor;
  }

  //
  // this should be defined on the SAGA Look & Feel level
  //
  // class PartialTimestamp
  // {
  // }
  //

  class job_info : extends saga::object,
                   extends saga::attributes
           // from object: saga::error_handler
  {
    CONSTRUCTOR (out job_info obj);
    DESTRUCTOR  (in  job_info obj);

    // Attributes:
    //
    //   name:  JobId
    //   desc:  The identifier for the completed job.
    //   type:  string
    //   mode:  ReadOnly
    //   value: ''
    //   notes:
    //
    //   name:  ResourceUsage
    //   desc:  the completed job's resource usage data.
    //   type:  Dictionary
    //   mode:  ReadOnly
    //   value: ''
    //   notes:
    //
    //   name:  HasExited
    //   desc:  true if the job terminated normally.
    //   type:  boolean
    //   mode:  ReadOnly
    //   value: True
    //   notes:
    //
    //   name:  ExitStatus
    //   desc:  the operating system exit code of the job.
    //   type:  long
    //   mode:  ReadOnly
    //   value: 0
    //   notes:
    //
    //   name:  HasSignaled
    //   desc:  true if the job terminated due to the receipt
    //          of a signal.
    //   type:  boolean
    //   mode:  ReadOnly
    //   value: False
    //   notes:
```

```
    //
    //   name:  TerminatingSignal
    //   desc:  a representation of the signal that caused
    //          termination.
    //   type:  string
    //   mode:  ReadOnly
    //   value: 0
    //   notes:
    //
    //   name:  HasCoreDump
    //   desc:  true if a core image of the terminated job was
    //          created
    //   type:  boolean
    //   mode:  ReadOnly
    //   value: False
    //   notes:
    //
    //   name:  WasAborted
    //   desc:  true if the job ended before entering the
    //          running state.
    //   type:  boolean
    //   mode:  ReadOnly
    //   value: False
    //   notes:
    //
  }

  job_template
  {
    // Attributes (extensible):
    //
    //   the following strings:
    //     HOME_DIRECTORY
    //     WORKING_DIRECTORY
    //     PARAMETRIC_INDEX
    //   are replaced in the job_template attribute values with the
    //   respective values of the attributes with the same name.
    //
    //   name:  HOME_DIRECTORY
    //   desc:
    //   type:  string
    //   mode:  ReadWrite
    //   value: -
    //   notes: -
    //
    //   name:  WORKING_DIRECTORY
    //   desc:
    //   type:  string
    //   mode:  ReadWrite
    //   value: -
    //   notes: -
    //
    //   name:  PARAMETRIC_INDEX
    //   desc:
    //   type:  string
    //   mode:  ReadWrite
    //   value: -
    //   notes: -
    //
    //   name:  remoteCommand
    //   desc:  command to be executed
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  args
    //   desc:  list of command-line arguments for the job
    //   type:  vString
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  jobSubmissionState
    //   desc:  state of the job at submission time
    //   type:  enum
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  jobEnvironment
    //   desc:  environment values that define the remote environment
    //   type:  vString
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  workingDirectory
    //   desc:  the directory where the job is executed
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  jobCategory
    //   desc:  how to resolve site-specific resources and/or policies
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  nativeSpecification
    //   desc:  site-specific resources and/or policies
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  email
    //   desc:  list of email addresses that is used to report the
    //          job completion and status
    //   type:  vString
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  blockEmail
    //   desc:  is sending of email is blocked or not
    //   type:  boolean
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  startTime
    //   desc:  the earliest time when the job is eligible to run
    //   type:  Time
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  jobName
    //   desc:  the client-provided job name
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  inputPath
    //   desc:  the job's standard input as a path to a file
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  outputPath
    //   desc:  the job's standard output as a path to a file
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  errorPath
    //   desc:  the job's standard error as a path to a file
    //   type:  string
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  joinFiles
    //   desc:  whether error stream should be intermixed with output stream
    //   type:  boolean
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  transferFiles (format)
    //   desc:  how to transfer files between hosts
    //   type:  String
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  deadlineTime
    //   desc:  deadline after which the DRMS will abort or terminate the job
    //   type:  Time
    //   mode:  ReadWrite
    //   value: ''
    //   notes: -
    //
    //   name:  hardWallclockTimeLimit
    //   desc:  specifies when the job's wall clock time limit has been exceeded
    //   type:  TimePeriod
    //   mode:  ReadWrite
    //   value: ''
```

```
  //   notes: -
  //
  //   name:  softWallClockTimeLimit
  //   desc:  an estimate of the wall clock time the job will need to complete
  //   type:  TimePeriod
  //   mode:  ReadWrite
  //   value: ''
  //   notes: -
  //
  //   name:  hardRunDurationLimit
  //   desc:  specifies how long the job MAY be in a running state
  //   type:  TimePeriod
  //   mode:  ReadWrite
  //   value: ''
  //   notes: -
  //
  //   name:  softRunDurationLimit
  //   desc:  an estimate as to how long the job will be running
  //   type:  TimePeriod
  //   mode:  ReadWrite
  //   value: ''
  //   notes: -
}


class job_service : implements saga::object
                    implements saga::async
                    implements saga::permissions
                // from object saga::error_handler
{
  CONSTRUCTOR   (in  session       s = default_session);
  DESTRUCTOR    (void);

  run_job       (in  job_template  tmpl,
                 out job           job);
  run_bulk_jobs (in  job_template  tmpl,
                 in  int           start,
                 in  int           end,
                 in  int           incr,
                 out array<job>    jobs);

  contact       (out array<string> contacts);
  version       (out string        version);
  drms_info     (out string        info);
  drmaa_impl    (out string        implementation);
}
```

```
  class job : implements saga::object
              implements saga::async
              implements saga::permissions
              extends    saga::task
          // from object saga::error_handler
  {
    // no constructor
    DESTRUCTOR    (void);

    suspend       (void);
    resume        (void);
    hold          (void);
    release       (void);
 // terminate     (void);   // == cancel    on saga::task
 // wait          (timeout) // == wait      on saga::task
 // job_status    (void);   // == get_state on saga::task
                            //    but with drmaa state model
  }


  class job_container : extends saga::task_container
         // from task_container saga::object
         // from object         saga::error_handler
  {
    ...

    synchronize   (in  float  timout = 0.0,
                   in  bool   cleanup);
    // == is wait on saga::task_container with additional
    // bool parameter
  }

              implements   saga::permissions
           // from object  saga::error_handler
  {
    CONSTRUCTOR (in    session          s,
                 in    saga::url        url = "",
                 out   rpc              obj          );
    DESTRUCTOR  (in    rpc              obj          );

    // rpc method invocation
    call        (inout array<remote_parameter>  parameters);

    // handle management
```

```
      close       (in    float            timeout = 0.0);
    }
  }
```

2.2.1 Specification Details

Enum `data_mode`

The `data_mode` enum specifies the storage properties of the `rpc::remote_parameter` instances:

`Volatile`

the paramater data are not stored on server sideafter computation

`Sticky`

the parameter data are stored on the rpc server after the rpc call finishes, and can be re-used for subsequent calls. That implies that `InOut` and `Out` parameter get their `src` url set and pointed to the intermediate results, but do not have a copy of the data stored.

`StickyReturn`

As `Sticky`, but a copy of the data are returned after the call, i.e., `InOut` and `Out` parameter have a copy of the data, and their `src` url is set and points to the intermediate results. The data remain at that single service location.

`Persistent`

As `Sticky`, but the data can be migrated to other service instances as needed.

`PersistentReturn`

As `Persistent`, but a copy of the data are returned after the call, i.e., `InOut` and `Out` parameter have a copy of the data, and their `src` url is set and points to the intermediate results.

Class `remote_parameter`

The `parameter` class inherits the `saga::parameter` class, adds an additional readonly state attribute: `data_mode`, and two additional URLs, `srs` and `tgt`. The `remote_parameter` uses these additional information to allow for data persistency between subsequent RPC calls, which can significantly improve performance.

Class `rpc`

This class replaces the `saga::rpc` class from the SAGA Core API specification [?], but now accepts both arrays of `saga::parameter` and `saga::remote_parameter` for rpc calls. The implementation is responsible to manage the data persistency for the remote parameters, according to their `data_mode` and `src` and `tgt` URLs.

As this class does not add any syntax to the original rpc class, no detailed specification is given here.

2.2.2 Examples

3 Conclusions

As stated in the Introduction 1, this document serves three different purposes. This conclusions provides a number of observations in respect to these points – more discussions are needed to finalize the conclusions.

Prove that a DRMAA rendering in SAGA is possible

In general, a DRMAA rendering within SAGA seems very well possible, and allows to maintain the DRMAA semantics. A number of differences, mostly in the API Look & Feel, must be noted however:

- DRMAA Exceptions do not map 1:1 to SAGA exceptions. The SAGA API does not, however, allow to introduce new exceptions in packages. Some care must thus be taken to prvide a reasonable mapping from DRMAA exceptions to SAGA exceptions.
- The DRMAA API is procedural, the SAGA API is object oriented. The mapping of procedural APIs to object oriented representations is inherently difficult and can be done in multuitude of ways. The presented mapping stays true to the mapping procedure used for the original SAGA job package, and has the implication that a number of job management methods are not performed on the job service handle (as in DRMAA), but on the job instance itself (as normal in SAGA).

allow for an easy comparison between the original SAGA job API and the DRMAA API

pave the path for a DRMAA API package in SAGA for the planned DRMAA version 2.0

4 Intellectual Property Issues

4.1 Contributors

This document is the result of the joint efforts of several contributors. The authors listed here and on the title page are those committed to taking permanent stewardship for this document. They can be contacted in the future for inquiries about this document.

Andre Merzky
andre@merzky.net
Center for Computation and Technology
Louisiana State University
216 Johnston Hall
70803 Baton Rouge
Louisiana, USA

In particular, the document build heavily on the specifications of the OGF GridRPC Working Group – we want to thank Eddy Caron, Craig Lee, Hidemoto Nakata and Yusuke Tanimura for their input and cooperation.

4.2 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

4.3 Disclaimer

4.4 Full Copyright Notice

References

[1] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. A Simple API for Grid Applications (SAGA). Grid Forum Document GFD.xx, 2007. Global Grid Forum.