GWD-R

Distributed Resource Management
Application API (DRMAA) Working Group

Hrabri Rajic*, Intel Coorporation (maintainer)
Roger Brobst, Cadence Design Systems
Waiman Chan, IBM
Fritz Ferstl, Sun Microsystems
Jeff Gardiner, John P. Robarts Res. Inst.
Andreas Haas*, Sun Microsystems
Bill Nitzberg, Altair Grid Technologies
Daniel Templeton, Sun Microsystems
John Tollefsrud[+], Sun Microsystems
Peter Tröger*, Hasso-Plattner-Inst.
*co-chairs
[+]founding co-chair
April, 2004
Updated June, 2007

Distributed Resource Management Application API Specification 1.0

Status of This Document

This document provides information to the Grid community regarding the specification of the Distributed Resource Management Application API. Distribution is unlimited.

Obsoletes

This document obsoletes GFD-22 OGF document [GDF.22].

Copyright Notice

Abstract

This document describes the Distributed Resource Management Application API (DRMAA), which provides a generalized API to distributed resource management systems (DRMSs) in order to facilitate integration of application programs.

The scope of DRMAA is limited to job submission, job monitoring and control, and retrieval of the finished job status. DRMAA provides application developers and distributed resource management builders with a programming model that enables the development of distributed applications tightly coupled to an underlying DRMS. For deployers of such distributed applications, DRMAA preserves flexibility and choice in system design.

Contents

1. Introduction

This document describes an API for the submission and control of jobs to one or more Distributed Resource Management Systems (DRMS). The specification encompasses the high-level functionality necessary for an application to consign a job to a DRMS, including common operations on jobs like termination or suspension. The objective is to facilitate the direct interfacing of applications to DRMS for application builders, portal builders, and independent software vendors. The specification abstracts the fundamental job interfaces of DRMS and provides an easy-to-use programming model, thereby encouraging adoption by both application builders and DRMS builders.

1.1 DRMAA Scope

The scope of DRMAA 1.0 is limited to job submission, job monitoring and control, and retrieval of the finished job status.

1.2 Language Issues

The document authors maintain that the API should be described such that it can be implemented in multiple languages. Therefore, in this document DRMAA interfaces are described using an Interface Definition Language (IDL) - like language.

1.3 Notational Conventions

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119 [BRADNER1].

The following abbreviations are used in this document:

API	Application Programming Interface
DRM	Distributed Resource Management
DRMS	Distributed Resource Management System
DRMAA	Distributed Resource Management Application API
ISV	Independent Software Vendor

2. Interface Design and Implementation Considerations

The DRMAA API has been developed to support what the authors believe will be desirable and common deployment scenarios of DRMAA implementations and applications. Specific attributes of library implementations of DRMAA should include, and the DRMAA specification anticipates:

2.1 Late Binding and Portability

DRMAA implementations SHOULD be provided as shared modules that could be interchangeably selected at the run time by the end user. DRMAA implementations MAY target one or more DRMSs. In the latter case a DRMAA-enabled application SHOULD be able to bind at run time to a specific DRMS via DRMAA library by setting an environment variable for the specific DRMS, or by providing the specific DRMS connection string in the initialization step.

2.2 Thread Safety

The authors expect that developers will link to a DRMAA library from serial and multithreaded codes; hence a DRMAA library SHOULD be thread-safe to allow a multithreaded application to use DRMAA interfaces without any explicit synchronization among the application threads. DRMAA implementers SHOULD label their implementations as thread safe if they meet the above criteria. Providers of non thread safe DRMAA implementations SHOULD document all the interfaces that are thread unsafe and provide a list of interfaces and their dependencies on external thread unsafe routines. Before a multithreaded application can use any of DRMAA interfaces, however, the DRMAA initialization routine SHOULD be called by only one thread, probably the main thread. The initialization routine is the only routine that MAY not be thread reentrant to still allow the implementers to call their implementation thread-safe. Similarly, the DRMAA library SHOULD be disengaged by only one thread. In case a standardized threads implementation such as POSIX threads exists it SHOULD be the preferred basis for thread-safe DRMAA implementation. Other threading implementations MAY be chosen, however, if documented accordingly.

2.3 Synchronization

DRMAA manages the asynchrony of job submission and job completion similarly to operating system process interfaces by blocking on the wait call for a specific job request.

2.4 Distributed Application Environment

DRMAA specifies mechanisms for submitting a job, monitoring and controlling it, and obtaining its final status. Ideally DRMAA implementations and distributed applications need not be concerned with a particular DRMS environment and DRMS site-specific policies. To facilitate deployments where this cannot be fully accomplished, Job Categories and Native Specification may be used to abstract or aggregate the site-specific policies into simple strings that are interpreted by DRMAA implementations.

2.4.1 Job Categories

DRMAA facilitates writing DRM-enabled applications even though the deployment properties, in particular the configuration of the DRMS, cannot be known in advance. This is a typical problem that has heretofore made writing DRM-enabled applications difficult for many Independent Software Vendors (ISVs), where the DRM system is selected by the end user.

Experience with integrations based on DRM command line interfaces show that even when the same ISV application is run as a job with the same DRMS, site-specific policies differ widely across users. These policies typically concern site-specific attributes such as what resources are to be used by the job, preferences where to run the job, and how the job should be scheduled relative to other jobs.

For supporting the variety of policies, job-specific requests expressed by DRMS submit options are common in the DRMS product space. Usually, however, these options do not affect the job from the perspective of the application or of the individual submitting the job request. This observation is the basis for “job categories,” which insulate the application and individual requester from site-specific policies.

DRMAA 1.0 provides interfaces for “job categories” which encapsulate site-specific details, hiding these details from applications using the DRMAA interface. Site administrators may create a job category suitable for an application to be dispatched by the DRMS; the associated category name could be specified as a job submission attribute. The DRMAA implementation MAY then use the category name to manage site-specific resource and functional requirements of jobs in the category. Such requirements need to be configurable by the site operating a DRMS and deploying an application on top of it.

An example can help to illustrate this idea:

> At site A, rendering application X is used in a heterogeneous clustered environment that is managed by a DRMS. Since application X is available only at a subset of these machines, the administrator sets up the DRMS so that the end users must put a -l X=true into their submit command line.
>
> At site B, the same application is used in a homogeneous clustered environment with rendering application X supported at all machines managed by the DRMS. However, since X jobs do compete with applications Y sharing the same resources and X applications are to be treated with higher priority than Y jobs, end users need to put a -p 1023 into their submit command line for raising the dispatch priority.
>
> An integration based on categories will allow submitting X jobs through the DRMAA interface in compliance with the policies of both sites A and B without the need to know about these policies. The ISV does this by specifying "X" as the category used for X rendering jobs submitted through the DRMAA interface and by mentioning this in the "DRMS integration" section of the X rendering software documentation.
>
> The administrators at site A and site B read the documentation or installation instructions about the "X" DRMAA category. The documentation of their DRMS contains directions about the category support of their DRMAA interface implementation. From this documentation they learn how to configure their DRMS in a way that "-l X=true" is used for "X" jobs at site A while "-p 1023" is used at site B for those jobs.

DRMAA describes a mechanism for specifying the category. Associating the policy-related portion of the submit command line to the job is implementation specific.

2.4.2 Native Specification

The categories concept provides a means for completely hiding site-specific policy details to be considered with a DRMAA job submission for a whole class of jobs. One job category MUST be maintained for each policy to be used. In order to allow the DRMAA interface to also be used for the submission of jobs where job-specific policy specification is required "native specification" is supported. Native specification MAY be used without the requirement to maintain job categories, and submit options MAY be specified directly.

An example can help to illustrate this idea:

> In order to implement the example from the previous section via native specifications, the native option string "-l X=true" has to be passed directly to the DRMAA interface while "-p 1023" has to be used at site B.

As far as the DRMAA interface specification is concerned, the native specification is an implementation-defined string and is interpreted by each DRMAA library. One MAY use job

categories and native specification with the same job submission for policy specification. In this case, the DRMAA library is assumed to be capable of joining the outcome of the two policy sources in a reasonable way. Care SHOULD be exercised to not change the job submission call semantics, pass options that conflict the already set attributes, or violate the DRMAA API in any way.

2.5 Interface Routines General Description

The interface routines are grouped in five categories: init and exit, job template handling, job submission, job monitoring and control, and auxiliary or informational routines that do not require initialization of a DRMAA session.

2.5.1 Init and Exit Routines

The calling sequence of the init routine allows all of the considered DRMS to be properly initialized, by interfacing either to the batch queue commands or to the DRMS API. Likewise, the exit routine requires parameters that will permit proper DRMS disengagement.

2.5.2 Job Template Routines

The remote jobs and their attributes SHALL be specified by the job template handle parameter. The job attributes SHALL be a string or a vector of string values.

The following job attributes are REQUIRED:

- Remote command to execute
- Remote command input parameters, a vector parameter
- Job state at submission
- Job environment, a vector parameter
- Job working directory
- Job category
- Native specification
- Standard input, output, and error streams
- Join output and error streams
- E-mail distribution list to report the job completion and status, a vector parameter
- E-mail suppression
- Job start time
- Job name to be used for the job submission

2.5.3 Job Submission Routines

Two job submission routines are described, one for submitting individual jobs and one for submitting bulk jobs.

2.5.4 Job Monitoring and Controlling Routines

The job monitoring and controlling API handles several functions:

- Job holding, releasing, suspending, resuming, and killing
- Checking the exit code of the finished remote job

- Checking the remote job status
- Waiting for the remote job till the end of its execution
- Waiting for all the jobs or a subset of the current session jobs to finish execution

The Unix like signals are replaced with the job control routines that have counterparts in DRMS. The only nontraditional feature is the passing of DRMAA_JOB_IDS_SESSION_ALL string as job_id parameter to indicate operations on all job Id's in the current session.

The remote job SHALL be in one of the following states:

- System hold
- User hold
- System and user hold simultaneously
- Queued active
- System suspended
- User suspended
- Running
- Finished (un)successfully

A rejected job is not assigned a job Id and consequently SHALL NOT have a state.

In a distributed system it may not be possible for the DRMAA implementation to determine the status of the remote job at all times.

2.5.5 Auxiliary Routines

The auxiliary routines are needed to obtain a textual representation of errors and other DRMAA implementation-specific information.

2.6 DRMAA Job State Transition Diagram

Figure 1 shows the DRMAA job state transition diagram in Harel notation:

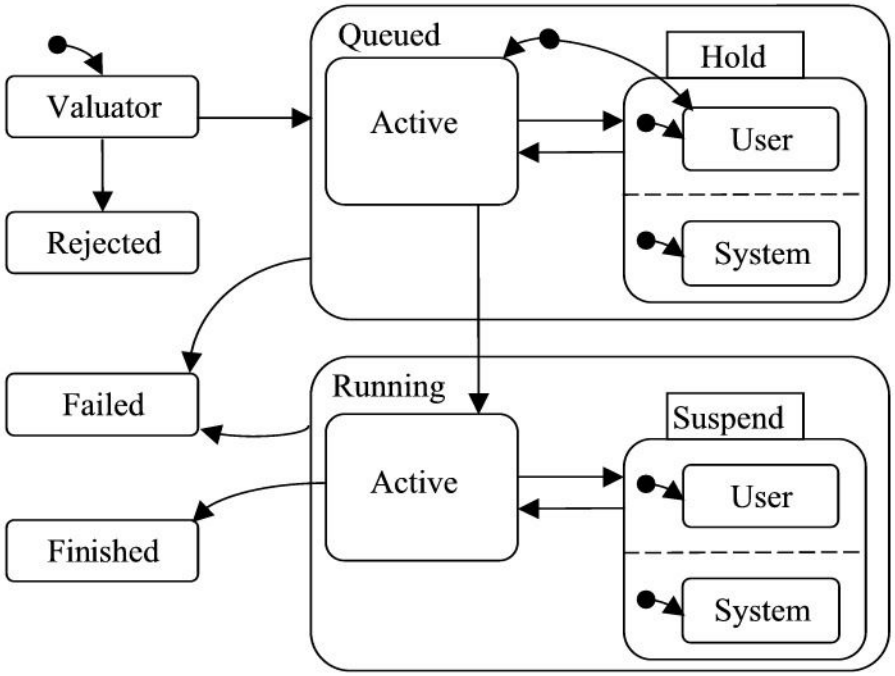

Figure 1: DRMAA job state transition diagram

3. API Specification

The API uses an IDL-like language to generalize discussion of allocation and deallocation, which have language specific implementations. The interface parameters could be IN, OUT, or INOUT parameters. Readers who are familiar with the C programming language should think of the parameters as being passed by value or by reference. Furthermore, the parameters could be scalar or vector values. The vector values are clearly documented.

3.1 Routines

In order to prevent interface name collisions all the routines have a prefix “drmaa.”

3.1.1 Error Codes

All of the interfaces return an error code on exit. Successful return SHALL be indicated by return value DRMAA_ERRNO_SUCCESS. All internal errors SHALL be indicated with DRMAA_ERRNO_INTERNAL_ERROR error. Out of memory errors SHALL be marked as DRMAA_ERRNO_NO_MEMORY. An invalid argument SHALL be flagged as DRMAA_ERRNO_INVALID_ARGUMENT. The return codes are specified and listed in Section 3.3.

The error code MAY be provided to the drmaa_strerror routine to retrieve a textual representation of the error. Routines MUST output a context-specific error string that MAY be used in addition to the textual representation obtained from the error code. Zero length string indicates that such information is not available. This string is undefined on normal returns. The parameter used to convey the context specific error SHALL be ignored by the routine whenever success is returned. The length of any output context-specific error string, including the ending null character SHALL NOT exceed DRMAA_ERROR_STRING_BUFFER.

3.1.2 DRMAA Sessions

An application process SHALL open only one DRMAA session at a time. Another session MAY be opened only after the current one is closed. Nesting of sessions SHOULD NOT be possible. It is RECOMMENDED that the DRMAA library SHALL free all the session resources, although this is not guaranteed, so it is RECOMMENDED that old session resources not be used later. Job Id’s SHALL remain valid from one session to another. Job control routines SHOULD work correctly if a job Id came from a previous DRMAA session, provided the current DRMAA session knows how to resolve this job Id. The burden is on the user to match previous job Id’s with appropriate DRMAA sessions (i.e., DRMAA implementations). It is RECOMMENDED that restartable applications make job Id’s persistent in order to access the already submitted jobs. Succesfull drmaa_wait() and drmaa_synchronize(), with dispose = true parameter, calls will make job id’s invalid by reaping the job run usage data.

3.1.3 Run Usage Data

A DRMAA implementation SHALL collect remote run usage data (rusage variable) after the remote job run and job finish information (stat variable). The user MAY reap this data only once. The implementation is free to "garbage collect" the reaped data at a convenient time. Only the data from the current session's job Id MUST be available. Reaping data from other session job Id's MAY be supported in a DRMAA implementation.

3.1.4 Precedence Rules

The attributes set by using API routines SHALL be set at the compile time. The attributes set by job categories SHALL be set at installation time. The attributes set by the native specification SHALL be set at the run time. In principle these should determine the precedence rules, but these ideal precedence rules are not always achievable in practice because of complex interaction of attributes. Moreover, certain attributes in job categories may not be allowed to be overridden. The precedence rules are therefore implementation specific.

3.1.5 Site-Specific Requirements

Job categories and native specifications are two means for describing site-specific requirements. Setting of job categories is implementation specific. On the other hand, setting the native specification, while straightforward in the user code, could be a challenge if the user needs to provide a complex set of options. Quotation marks are especially problematic if only one variable is used for a set of native specification options.

The following are RECOMMENDED to developers to use this feature effectively:

- For each class of remote jobs, give end users a chance to specify site-specific environments, such as a queue where to send remote jobs or architecture(s) where the remote applications are available.
- Let users specify native specifications in a file if the distributed application has several classes of jobs to submit or several DRMAA sessions.
- Applications with a graphical user interface could have a dedicated dialog for this purpose.

3.1.6 Job Valuator

Before a submitted job enters a queue, it SHALL be passed through a valuator that determines whether the job attributes as specified are valid. If yes, a job Id SHALL be returned, and the job is successfully queued. If not, the job is rejected, job Id SHALL NOT be returned, and no job state is possible.

3.2 DRMAA API

For convenience, the API is divided in its five logical sections: init/exit, job template handling, job submission, job monitoring and control, and auxiliary routines.

```
/* ---------- Major Assumptions/Restrictions ---------- */

  No explicit file staging.
  Job Id Uniqueness -- "As unique as the underlying DRM makes them"


   /*Global constants */

      DRMAA_ERROR_STRING_BUFFER       = 1024
      DRMAA_JOBNAME_BUFFER            = 1024
      DRMAA_SIGNAL_BUFFER             =   32
      DRMAA_TIMEOUT_WAIT_FOREVER      =   -1
      DRMAA_TIMEOUT_NO_WAIT           =    0
      JOB_IDS_SESSION_ANY             = "DRMAA_JOB_IDS_SESSION_ANY"
```

JOB_IDS_SESSION_ALL = "DRMAA_JOB_IDS_SESSION_ALL"

3.2.1 Initialization and Exit Routines

drmaa_init(contact, drmaa_context_error_buf)
IN contact /* contact information for DRM system (string) */
OUT drmaa_context_error_buf /*Contains a context-sensitive error upon failed return*/

Initialize DRMAA API library and create a new DRMAA session. 'Contact' is an implementation-defined string that MAY be used to specify which DRM system to use. This routine MUST be called before any other DRMAA calls, except for drmaa_version(), drmaa_get_DRM_system(), drmaa_get_DRMAA_implementation(), drmaa__strerror(), or drmaa_get_contact(). If 'contact' is NULL, the default DRM system SHALL be used provided there is only one DRMAA implementation in the provided binary module. When there is more than one DRMAA implementation in the binary module, drmaa_init() SHALL return the DRMAA_ERRNO_NO_DEFAULT_CONTACT_STRING_SELECTED error. drmaa_init() SHOULD be called by only one of the threads. The main thread is RECOMMENDED. A call by another thread SHALL return DRMAA_ERRNO_ALREADY_ACTIVE_SESSION.

drmaa_init routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_INVALID_CONTACT_STRING,
DRMAA_ERRNO_ALREADY_ACTIVE_SESSION,
DRMAA_ERRNO_NO_DEFAULT_CONTACT_STRING_SELECTED,
DRMAA_ERRNO_AUTH_FAILURE, DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_DRMS_INIT_FAILED,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE, or
DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR.

drmaa_exit(drmaa_context_error_buf)
OUT drmaa_context_error_buf /*Contains a context-sensitive error upon failed return*/

Disengage from DRMAA library and allow the DRMAA library to perform any necessary internal cleanup.
This routine SHALL end the current DRMAA session but SHALL NOT affect any jobs (e.g., queued and running jobs SHALL remain queued and running). drmaa_exit() SHOULD be called by only one of the threads. The first call to call drmaa_exit() by a thread will operate normally. All other calls from the same and other threads SHALL fail, returning a DRMAA_ERRNO_NO_ACTIVE_SESSION error code.

drmaa_exit routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRMS_EXIT_ERROR, DRMAA_ERRNO_AUTH_FAILURE,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE, or
DRMAA_ERRNO_NO_ACTIVE_SESSION.

3.2.2 Job Template Routines

drmaa_allocate_job_template(jt, drmaa_context_error_buf)
OUT jt /* job template (implementation-defined handle) */
OUT drmaa_context_error_buf /*Contains a context-sensitive error upon failed return*/

Allocate a new job template.
drmaa_allocate_job_template() SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_AUTH_FAILURE,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE.

drmaa_delete_job_template(jt, drmaa_context_error_buf)
INOUT jt /* job template (implementation-defined handle) */
OUT drmaa_context_error_buf /*Contains a context-sensitive error upon failed return*/

Deallocate a job template. This routine has no effect on jobs.
drmaa_delete_job_template() SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_NO_ACTIVE_SESSION, DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_AUTH_FAILURE, or
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE.

drmaa_set_attribute(jt, name, value, drmaa_context_error_buf)
INOUT jt /* job template (implementation-defined handle) */
IN name /* attribute name (string) */
IN value /* attribute value (string) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

Adds ('name', 'value') pair to list of attributes in job template 'jt'.
Only non-vector attributes SHALL be passed.

drmaa_set_attribute routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT,
DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE,
DRMAA_ERRNO_NO_ACTIVE_SESSION,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_AUTH_FAILURE, or
DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES.

drmaa_get_attribute(jt, name, value, drmaa_context_error_buf)
IN jt /* job template (implementation-defined handle) */
IN name /* attribute name (string) */
OUT value /* attribute value (string) */

OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

If 'name' is an existing non-vector attribute name in the job template
'jt', then the value of 'name' SHALL be returned; otherwise, NULL MUST be returned.

drmaa_get_attribute routine SHALL return DRMAA_ERRNO_SUCCESS
on success, otherwise
DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_AUTH_FAILURE.

drmaa_set_vector_attribute(jt, name, values, drmaa_context_error_buf)

INOUT jt /* job template (implementation-defined handle) */
IN name /* attribute name (string) */
IN values /* vector of attribute value (string vector) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

Adds ('name', 'values') pair to list of vector attributes in job template 'jt'.
Only vector attributes SHALL be passed.

drmaa_set_vector_attribute routine SHALL return DRMAA_ERRNO_SUCCESS
on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT,
DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE,
DRMAA_ERRNO_NO_ACTIVE_SESSION,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_AUTH_FAILURE, or
DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES.

drmaa_get_vector_attribute(jt, name, values, drmaa_context_error_buf)

IN jt /* job template (implementation-defined handle) */
IN name /* attribute name (string) */
OUT values /* vector of attribute value (string vector) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

If 'name' is an existing vector attribute name in the job template 'jt',
then the values of 'name' SHALL be returned; otherwise, NULL MUST be returned.

drmaa_get_vector_attribute routine SHALL return DRMAA_ERRNO_SUCCESS
on success, otherwise
DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_NO_MEMORY,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_AUTH_FAILURE.

drmaa_get_attribute_names(names, drmaa_context_error_buf)

OUT names /* vector of attribute name (string vector) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

SHALL return the set of supported attribute names whose associated value type is String. This set SHALL include supported DRMAA reserved attribute names and native attribute names.

drmaa_get_attribute_names routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_NO_MEMORY,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_AUTH_FAILURE.

drmaa_get_vector_attribute_names(names, drmaa_context_error_buf)

```
OUT names                    /* vector of attribute name (string vector) */
OUT drmaa_context_error_buf  /*Contains a context sensitive error upon failed return*/
```

SHALL return the set of supported attribute names whose associated value type is String Vector.This set SHALL include supported DRMAA reserved attribute names and native attribute names.

drmaa_get_vector_attribute_names routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_NO_MEMORY,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_AUTH_FAILURE.

3.2.3 Job attributes

Mandatory attributes

The following reserved attributes SHALL be available in all implementations of DRMAA. Vector attributes are marked with a 'V':

remote command to execute (string)
It is relative to the execution host.
It is evaluated on the execution host.
No binary file management is done.
The attribute name is drmaa_remote_command.

V input parameters (vector of strings)
These parameters SHALL be passed as arguments to the job.
The attribute name is drmaa_v_argv.

job state at submission (string value)
This might be useful for a rather rudimentary, but very general job-dependent execution.
The states SHALL be drmaa_hold and drmaa_active:
drmaa_active means job has been queued, and is eligible to run
drmaa_hold means job has been queued, but it is NOT eligible to run
The attribute name is drmaa_js_state.

V job environment (vector of strings)
The environment values that define the remote environment.
Each string SHALL comply with the format <name>=<value>.
The values override the remote environment values if there is a collision.
If above is not possible, it is implementation dependent.
The attribute name is drmaa_v_env.

job working directory (string)
This attribute specifies the directory where the job is executed.
If not set, it is implementation dependent.
Evaluated relative to the execution host.

A $drmaa_hd_ph$ placeholder at the begin denotes the remaining portion of the directory_name as a relative directory name resolved relative to the job users home directory at the execution host.
The $drmaa_incr_ph$ placeholder MAY be used at any position within the directory_name of parametric job templates and SHALL be substituted by the underlying DRM system with the parametric jobs' index.
The directory_name MUST be specified in a syntax that is common at the host where the job is executed.
If set and no placeholder is used, an absolute directory specification is expected.
If set and the job was successfully submitted and the directory does not exist, the job enters the state DRMAA_PS_FAILED.
The attribute name is drmaa_wd.

job category (string)
An implementation-defined string specifying how to resolve site-specific resources and/or policies. Detailed description is provided in section 2.4.1.
The attribute name is drmaa_job_category.

native specification (string)
An implementation-defined string that is passed by the end user to DRMAA to specify site-specific resources and/or policies. Detailed description is provided in section 2.4.2.
The attribute name is drmaa_native_specification.

V e-mail address (vector of strings)
It is used to report the job completion and status. The new values replace old values,
The attribute name is drmaa_v_email.

e-mail suppression (string)
It is used to block sending e-mail by default, regardless of the DRMS setting.
1 block
0 do not block.
The attribute name is drmaa_block_email

job start time (string)

This attribute specifies the earliest time when the job may be eligible to be run.
The attribute name is drmaa_start_time
The value of the attribute SHALL be of the form
[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]
where
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two-digit day of the month [01,31]
hh is the two-digit hour of the day [00,23]
mm is the two-digit minute of the day [00,59]
ss is the two-digit second of the minute [00,61]
UU is the two-digit hours since (before) UTC
uu is the two-digit minutes since (before) UTC
If the optional UTC-offset is not specified, the offset associated with the local timezone SHALL be used.
If the day (DD) is not specified, the current day SHALL be used unless the specified hour:mm:ss has already elapsed, in which case the next day SHALL be used.
Similarly for month (MM), year (YY), and century-1 (CC).
If seconds (ss) value is omitted it is set to 0.

Example:
The time: Sep 3 4:47:27 PM PDT 2002,
could be represented as: 2002/09/03 16:47:27 -07:00

job name

A job name SHALL comprise alphanumeric and _ characters.
The drmaa-implementation SHALL NOT provide the client with a job name longer than DRMAA_JOBNAME_BUFFER -1 (1023) characters.
The drmaa-implementation MAY truncate any client-provided job name to an implementation-defined length that is at least 31 characters.
The attribute name is drmaa_job_name

input stream (string)

Specifies the jobs' standard input.
Unless set elsewhere, if not explicitly set in the job template, the job is started with an empty input stream.
If set, specifies the network path of the jobs input stream file of the form [hostname]:file_path
When the drmaa_transfer_files job template attribute is supported and contains the character 'i', the input file SHALL be fetched by the underlying DRM system from the specified host or from the submit host if no hostname is specified.
When the drmaa_transfer_files job template attribute is not supported or does not contain the character 'i', the input file is always expected at the host where the job is executed, irrespective of a possibly hostname specified.
The $drmaa_incr_ph$ placeholder can be used at any position within the file_path of parametric job templates and SHALL be substituted by the underlying DRM system with the parametric jobs' index.
A $drmaa_hd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved

relative to the job users home directory at the host where the file is located.
A $drmaa_wd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved relative to the jobs working directory at the host where the file is located.
The file_path MUST be specified in a syntax that is common at the host where the file is located.
If set and the job was successfully submitted and the file can't be read, the job enters the state DRMAA_PS_FAILED.
The attribute name is drmaa_input_path.

output stream (string)

Specifies how to direct the jobs' standard output.
If not explicitly set in the job template, the whereabouts of the jobs output stream is not defined.
If set, specifies the network path of the jobs output stream file of the form [hostname]:file_path
When the drmaa_transfer_files job template attribute is supported and contains the character 'o', the output file SHALL be transferred by the underlying DRM system to the specified host or to the submit host if no hostname is specified.
When the drmaa_transfer_files job template attribute is not supported or does not contain the character 'o', the output file is always kept at the host where the job is executed irrespectively of a possibly hostname specified.
The $drmaa_incr_ph$ placeholder can be used at any position within the file_path of parametric job templates and SHALL be substituted by the underlying DRM system with the parametric jobs' index.
A $drmaa_hd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved relative to the job users home directory at the host where the file is located.
A $drmaa_wd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved relative to the jobs working directory at the host where the file is located.
The file_path MUST be specified in a syntax that is common at the host where the file is located.
If set and the job was successfully submitted and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.
The attribute name is drmaa_output_path.

error stream (string)

Specifies how to direct the jobs' standard error.
If not explicitly set in the job template, the whereabouts of the jobs error stream is not defined.
If set, specifies the network path of the jobs error stream file of the form [hostname]:file_path
When the drmaa_transfer_files job template attribute is supported and contains the character 'e', the output file SHALL be transferred by the underlying DRM system to the specified host or to the submit host if no hostname is specified.
When the drmaa_transfer_files job template attribute is not supported or does not contain the character 'e', the error file is always kept at the host where the job is executed irrespectively of a possibly hostname specified.
The $drmaa_incr_ph$ placeholder can be used at any position within the

file_path of parametric job templates and SHALL be substituted by the underlying DRM system with the parametric jobs' index.
A $drmaa_hd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved relative to the job users home directory at the host where the file is located.
A $drmaa_wd_ph$ placeholder at the begin of the file_path denotes the remaining portion of the file_path as a relative file specification resolved relative to the jobs working directory at the host where the file is located.
The file_path MUST be specified in a syntax that is common at the host where the file is located.
If set and the job was successfully submitted and the file can't be written before execution the job enters the state DRMAA_PS_FAILED.
The attribute name is drmaa_error_path.

join files (string)
Specifies if the error stream should be intermixed with the output stream.
If not explicitly set in the job template the attribute defaults to 'n'.
Either 'y' or 'n' can be specified.
If 'y' is specified the underlying DRM system SHALL ignore the value of the drmaa_error_path attribute and intermix the standard error stream with the standard output stream as specified with drmaa_output_path.
The attribute name is drmaa_join_files.

Optional Attributes

The following reserved attribute names are OPTIONAL in a conforming DRMAA implementation. For attributes that are implemented, the meanings are REQUIRED to be as follows:

Note that the list of attributes that are implemented may be programmatically obtained by using the drmaa_get_attribute_names and drmaa_get_vector_attribute_names routines.

transfer files (string)
Specifies how to transfer files between hosts.
If not explicitly set in the job template the attribute defaults to ''.
Any combination of 'e', 'i' and 'o' MAY be specified.
Whether the character 'e' is specified impacts the behavior of the drmaa_error_path attribute.
Whether the character 'i' is specified impacts the behavior of the drmaa_input_path attribute.
Whether the character 'o' is specified impacts the behavior of the drmaa_output_path attribute.
The attribute name is drmaa_transfer_files.

absolute job termination time (string)
Specifies a deadline after which the DRMS will terminate a job.
This is a reserved attribute named drmaa_deadline_time
The value of the attribute SHALL be of the form
[[[[CC]YY/]MM/]DD] hh:mm[:ss] [{-|+}UU:uu]

where
CC is the first two digits of the year (century-1)
YY is the last two digits of the year
MM is the two digits of the month [01,12]
DD is the two digit day of the month [01,31]
hh is the two digit hour of the day [00,23]
mm is the two digit minute of the day [00,59]
ss is the two digit second of the minute [00,61]
UU is the two digit hours since (before) UTC
uu is the two digit minutes since (before) UTC
If an optional portion of the time specification is omitted,
then the termination time SHALL be determined based upon
the job's earliest start time.
If the day (DD) is not specified, the earliest start day
for the job SHALL be used unless the specified hour:mm:ss
precedes the corresponding portion of the job start time,
in which case the next day SHALL be used.

Similarly for month (MM), year (YY), and century-1 (CC).
If seconds (ss) value is omitted it is set to 0.

Example:
The time: Sep 3 4:47:27 PM PDT 2002,
could be represented as: 2002/09/03 16:47:27 -07:00

wall clock time limit (string)
This attribute specifies when the job's wall clock time limit has been
exceeded. The DRMS SHALL terminate a job that has exceeded its wall
clock time limit. Suspended time SHALL also be accumulated here.
This is a reserved attribute named drmaa_wct_hlimit
The value of the attribute SHALL be of the form
[[h:]m:]s
where
h is one or more digits representing hours
m is one or more digits representing minutes
s is one or more digits representing seconds

Example:
To terminate a job after 2 hours and 30 minutes,
any of the following MAY be passed:
2:30:0, 1:90:0, 150:0

soft wall clock time limit (string)
This attribute specifies an estimate as to how long the
job will need wall clock time to complete. Note that
the suspended time is also accumulated here.
This attribute is intended to assist the scheduler.
If the time specified in insufficient, the
drmaa-implementation MAY impose a scheduling penalty.
This is a reserved attribute named drmaa_wct_slimit
The value of the attribute SHALL be of the form
[[h:]m:]s where

h is one or more digits representing hours
m is one or more digits representing minutes
s is one or more digits representing seconds

job run duration hlimit (string)
This attribute specifies how long the job MAY be in a running state
before its limit has been exceeded, and therefore is terminated by the DRMS.
This is a reserved attribute named drmaa_run_duration_hlimit
The value of the attribute SHALL be of the form
[[h:]m:]s where
h is one or more digits representing hours
m is one or more digits representing minutes
s is one or more digits representing seconds

job run duration slimit (string)
This attribute specifies an estimate as to how long the job will need to
remain in a running state to complete.
This attribute is intended to assist the scheduler. If the time specified
in insufficient, the drmaa-implementation MAY impose a scheduling penalty.
This is a reserved attribute named drmaa_run_duration_slimit
The value of the attribute SHALL be of the form
[[h:]m:]s where
h is one or more digits representing hours
m is one or more digits representing minutes
s is one or more digits representing seconds

3.2.4 Job Submission Routines

drmaa_run_job(job_id, jt, drmaa_context_error_buf)

```
OUT job_id                  /* job identifier (string) */
IN  jt                      /* job template (implementation-defined handle) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/
```

Submit a job with attributes defined in the job template 'jt'.
The job identifier 'job_id' is a printable, NULL terminated string,
identical to that returned by the underlying DRM system.
The call SHOULD return after the DRMAA implementation submits the job request to
the underlying DRM system.

drmaa_run_job routine SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_TRY_LATER,
DRMAA_ERRNO_DENIED_BY_DRM,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_INTERNAL_ERROR, DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_NO_ACTIVE_SESSION, or
DRMAA_ERRNO_AUTH_FAILURE.

drmaa_run_bulk_jobs(job_ids, jt, start, end, incr, drmaa_context_error_buf)

```
OUT job_ids                 /* job identifiers (array of strings) */
IN  jt                      /* job template (implementation-defined handle) */
```

IN start /* beginning index (unsigned integer)*/
IN end /* ending index (unsigned integer) */
IN incr /* loop increment (integer)*/
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

Submit a set of parametric jobs, dependent on the implied loop index, each
with attributes defined in the job template 'jt'.
The job identifiers 'job_ids' SHALL be all printable,
NULL terminated strings, identical to those returned by the underlying
DRM system. Nonnegative loop bounds SHALL NOT use file names
that start with minus sign like command line options. The call SHOULD return after
the DRMAA implementation submits the job request to the underlying DRM system.

The special index placeholder is a DRMAA defined string
drmaa_incr_ph /* == $incr_pl$ */
that is used to construct parametric job templates.
Due to DRM system differing implementations job working directory,
input stream, output stream, and error stream are the only attributes
that are allowed to have the index placeholder.

For example:
drmaa_set_attribute(pjt, "stderr", drmaa_incr_ph + ".err");
/*C++/java string syntax used */

drmaa_run_bulk_jobs routine SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_TRY_LATER, DRMAA_ERRNO_DENIED_BY_DRM,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_INTERNAL_ERROR, DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_NO_ACTIVE_SESSION, or
DRMAA_ERRNO_AUTH_FAILURE.

3.2.5 Job Control Routines

drmaa_control(job_id, action, drmaa_context_error_buf)
IN job_id /* job identifier (string) */
IN action /* control action (const) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

Start, stop, restart, or kill the job identified by 'job_id'.
If 'job_id' is DRMAA_JOB_IDS_SESSION_ALL, then this routine
SHALL act on all jobs submitted during this DRMAA session, at the moment it is called.
If 'job_id' is DRMAA_JOB_IDS_SESSION_ALL, then a call on an empty session
SHALL result to DRMAA_ERRNO_SUCCESS for all control operations.
To avoid thread races in multithreaded application the user of DRMAA
implementation should explicitly synchronize this call with any other
job submission call or control call that changes the number of remote jobs.
The legal values for 'action' and their meanings SHALL be:
DRMAA_CONTROL_SUSPEND: stop the job,
DRMAA_CONTROL_RESUME: (re)start the job,
DRMAA_CONTROL_HOLD: put the job on-hold,

DRMAA_CONTROL_RELEASE: release the hold on the job, and
DRMAA_CONTROL_TERMINATE: kill the job.

This routine SHALL return once the action has been acknowledged by the DRM system, but does not necessarily wait until the action has been completed. If there are multiple kinds of errors the routine SHALL return INTERNAL_ERROR error. The error description string SHOULD contain a good explanation of the problems.

drmaa_control routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_AUTH_FAILURE,
DRMAA_ERRNO_RESUME_INCONSISTENT_STATE,
DRMAA_ERRNO_SUSPEND_INCONSISTENT_STATE,
DRMAA_ERRNO_HOLD_INCONSISTENT_STATE,
DRMAA_ERRNO_RELEASE_INCONSISTENT_STATE,
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_ACTIVE_SESSION, or
DRMAA_ERRNO_INVALID_JOB.

drmaa_synchronize(job_ids, timeout, dispose, drmaa_context_error_buf)

IN job_ids	/* job identifiers (array of strings) */
IN timeout	/* how long we block in this call (signed long) */
IN dispose	/* dispose reaping information (boolean)*/
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

Wait until all jobs specified by 'job_ids' have finished execution. If an invalid or already waited job id is specified, the routine SHALL fail with a DRMAA_ERRNO_INVALID_JOB error. If 'job_ids' is DRMAA_JOB_IDS_SESSION_ALL, then this routine SHALL wait for all jobs submitted during this DRMAA session, at the moment it is called. If the session contains no jobs, the routine MUST immediately return with DRMAA_ERRNO_SUCCESS, regardless of the dispose parameter value.
Because a DRMAA implementation is not required to retain information about jobs which have been reaped, the routine is not required to, but MAY distinguish between non-existent and reaped jobs. The routine MUST return DRMAA_ERRNO_INVALID_JOB if a provided ‘job_ids’ contains a job that is unrecognized. If the routine successfully validates a ‘job_id’ for an already reaped job, it MAY return DRMAA_ERRNO_SUCCESS.
To avoid thread races in multithreaded application the user of DRMAA implementation should explicitly synchronize this call with any other job submission call or control call that changes the number of remote jobs.
To prevent blocking indefinitely in this call, the caller MAY use timeout specifying after how many seconds to time out in this call. The value DRMAA_TIMEOUT_WAIT_FOREVER (-1) MAY be specified to wait indefinitely for a result. The value DRMAA_TIMEOUT_NO_WAIT (0) MAY be specified to return immediately if no result is available. If the call exits before timeout, all the jobs have been waited on or there was an interrupt.

If the invocation exits on timeout, the return code is DRMAA_ERRNO_EXIT_TIMEOUT. The caller SHOULD check system time before and after this call in order to check how much time has passed.

The dispose parameter specifies how to treat reaping of the remote job's system resources consumption and other statistics. If dispose is set to false, the job's information remains available and can be retrieved through drmaa_wait(). If dispose is set to true, the job's information is not retained.

drmaa_ synchronize routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR, DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE, DRMAA_ERRNO_AUTH_FAILURE, DRMAA_ERRNO_EXIT_TIMEOUT, DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_INVALID_JOB.

drmaa_wait(job_id, job_id_out, stat, timeout, rusage, drmaa_context_error_buf)

IN job_id	/* job identifier (string) or DRMAA_JOB_IDS_SESSION_ANY (string) */
OUT job_id_out	/* job identifier of ended job (string) or NULL */
OUT stat	/* status code of job (integer) */
IN timeout	/ * how long we block in this call (signed long) */
OUT rusage	/* resource usage (string array) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

This routine SHALL wait for a job with job_id to fail or finish execution. If the special string DRMAA_JOB_IDS_SESSION_ANY is provided as the job_id, this routine SHALL wait for any job from the session. In a multithreaded environment, only the active thread gets the status of the finished or failed job in that case, while the rest of the threads continue waiting. drama_wait would wait on the submitted and not yet done/failed jobs at the moment it has been issued, so possible indeterminism is possible in multithreaded environments.
If there are no more running or completed jobs the routine SHOULD return DRMAA_ERRNO_INVALID_JOB error. If an invalid or already waited job id is specified, the routine SHOULD fail with a DRMAA_ERRNO_INVALID_JOB error.
The timeout value is used to specify the desired behavior when a result is not immediately available.
The value DRMAA_TIMEOUT_WAIT_FOREVER (-1) MAY be specified to wait indefinitely for a result. The value DRMAA_TIMEOUT_NO_WAIT (0) MAY be specified to return immediately if no result is available. Alternatively, a number of seconds MAY be specified to indicate how long to wait for a result to become available.

If the call exits before timeout, either the job has been waited on successfully or there was an interrupt.

If the invocation exits on timeout, the return code is DRMAA_ERRNO_EXIT_TIMEOUT. The caller SHOULD check system time before and after this call in order to check how much time has passed.

The routine reaps jobs on a successful call, so any subsequent calls to drmaa_wait SHOULD fail returning an error DRMAA_ERRNO_INVALID_JOB meaning that the job has been already reaped. This error is the same as if the job was unknown. Failing due to an elapsed timeout has an effect that it is possible to issue drmaa_wait multiple times for the same job_id. When successful, the rusage information SHALL be provided as an array of strings, where each string complies with the

format <name>=<value>.
The string portion <value> contains the amount of resources consumed by the job and is implementation-defined.

drmaa_ wait routine SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR, DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE, DRMAA_ERRNO_AUTH_FAILURE, DRMAA_ERRNO_NO_RUSAGE, DRMAA_ERRNO_EXIT_TIMEOUT, DRMAA_ERRNO_NO_RUSAGE, DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_INVALID_JOB.

The 'stat' drmaa_wait parameter is used in a series of functions, defined below, for providing more detailed information about job termination if it is available. If those functions: `drmaa_wcoredump(), drmaa_wexitstatus(), drmaa_wifaborted(), drmaa_wifexited(), drmaa_wifsignaled(), and drmaa_wtermsig()` are called when they should not have been, DRMAA_ERRNO_INVALID_ARGUMENT SHALL be returned and a descriptive message SHOULD be provided in drmaa_context_error_buf.

drmaa_wifexited(OUT exited, IN stat, OUT drmaa_context_error_buf)

OUT exited	/* more diagnosis indication value (integer) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

Evaluates into 'exited' a non-zero value if stat was returned for a job that either failed after running or finished after running (see section 2.6). More detailed diagnosis can be provided by means of drmaa_wexitstatus().
A zero result for the 'exited' parameter either indicates that
1) although it is known that the job was running, more information is not available
2) it is not known whether the job was running
In both cases drmaa_wexitstatus() SHALL NOT provide exit status information.

The function SHALL return DRMAA_ERRNO_SUCCESS on success, otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_wexitstatus(OUT exit_status, IN stat, OUT drmaa_context_error_buf)

OUT exit_status	/* exit code of the job (integer) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

If the OUT parameter 'exited' of drmaa_wifexited() is non-zero, this function evaluates into 'exit_code' the exit code that the job passed to _exit() (see exit(2)) or exit(3C), or the value that the child process returned from main.

The function SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or
DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_wifsignaled(OUT signaled, IN stat, OUT drmaa_context_error_buf)

OUT signaled	/* receipt of signal indication (integer) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

Evaluates into 'signaled' a non-zero value if stat was returned
for a job that terminated due to the receipt of a signal. A zero value
can also indicate that although the job has terminated due to the receipt
of a signal the signal is not available or that it is not known whether
the job terminated due to the receipt of a signal. In both cases
drmaa_wtermsig() SHALL NOT provide signal information. A non-zero value returned
in ‘signaled’ parameter indicates signal information can be retrieved by means
of drmaa_wtermsig.

The function SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or
DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_wtermsig(OUT signal, IN stat, OUT drmaa_context_error_buf)

OUT signal	/* returned signal (string) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

If the parameter 'signaled' of drmaa_wifsignaled is
non-zero, this function evaluates into signal which is a string representation of the signal
that caused the termination of the job. For signals declared by POSIX, the symbolic
names SHALL be returned (e.g., SIGABRT, SIGALRM).
For signals not declared by POSIX, any other string MAY be returned.

The function SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or
DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_wcoredump(OUT core_dumped, IN stat, OUT drmaa_context_error_buf)

OUT core_dumped	/* core image indication value (integer) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

If the parameter 'signaled' of drmaa_wifsignaled is
non-zero, this function evaluates into 'core_dumped', a non-zero value
if a core image of the terminated job was created.

The function SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or
DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_wifaborted(OUT aborted, IN stat, OUT drmaa_context_error_buf)

OUT aborted	/* job aborted indication value (integer) */
IN stat	/* status code of job (integer) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

Evaluates into 'aborted', a non-zero value if 'stat'
was returned for a job that ended before entering the running state.

The function SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_INVALID_ARGUMENT, DRMAA_ERRNO_NO_MEMORY, or
DRMAA_ERRNO_INTERNAL_ERROR.

drmaa_job_ps(IN job_id, OUT remote_ps, OUT drmaa_context_error_buf);

IN job_id	/* job identifier (string) */
OUT remote_ps	/* program status (constant) */
OUT drmaa_context_error_buf	/*Contains a context sensitive error upon failed return*/

Get the program status of the job identified by 'job_id'.
The possible values returned in 'remote_ps' and their meanings SHALL be:

DRMAA_PS_UNDETERMINED	= 00H : process status cannot be determined
DRMAA_PS_QUEUED_ACTIVE	= 10H : job is queued and active
DRMAA_PS_SYSTEM_ON_HOLD	= 11H : job is queued and in system hold
DRMAA_PS_USER_ON_HOLD	= 12H : job is queued and in user hold
DRMAA_PS_USER_SYSTEM_ON_HOLD	= 13H : job is queued and in user and system hold
DRMAA_PS_RUNNING	= 20H : job is running
DRMAA_PS_SYSTEM_SUSPENDED	= 21H : job is system suspended
DRMAA_PS_USER_SUSPENDED	= 22H : job is user suspended
DRMAA_PS_DONE	= 30H : job finished normally
DRMAA_PS_FAILED	= 40H : job finished, but failed.

DRMAA SHOULD always get the status of job_id from DRM system,
unless the previous status has been DRMAA_PS_FAILED or DRMAA_PS_DONE and the status has been successfully cached. Terminated jobs get DRMAA_PS_FAILED status.

drmaa_ synchronize routine SHALL return DRMAA_ERRNO_SUCCESS on success,
otherwise
DRMAA_ERRNO_NO_MEMORY, DRMAA_ERRNO_INTERNAL_ERROR,
DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE,
DRMAA_ERRNO_AUTH_FAILURE, DRMAA_ERRNO_INVALID_ARGUMENT,
DRMAA_ERRNO_NO_ACTIVE_SESSION, or DRMAA_ERRNO_INVALID_JOB.

3.2.6 Auxiliary Routines

error_string drmaa_strerror (IN errno, OUT error_string**);**
IN errno /* Errno number (integer) */
OUT error_string /* Readable text version of errno (constant string) */

SHALL return the error message text associated with the errno number. The routine SHALL return null string if called with invalid ERRNO number.

drmaa_get_contact(OUT contacts, OUT drmaa_context_error_buf);
OUT contacts /* Default contacts information for DRM systems (string) */
OUT drmaa_context_error_buf /* Contains a context sensitive error upon failed return */

If called before drmaa_init(), it SHALL return a comma delimited default DRMAA implementation contacts string, one per each DRM system provided implementation. If called after drmaa_init(), it SHALL return the selected contact string. The output (string) is Implementation dependent.

drmaa_get_contact routine SHALL return DRMAA_ERRNO_SUCCESS on success.

drmaa_version(OUT major, Out minor, OUT drmaa_context_error_buf)
OUT major /* major version number (non-negative integer) */
OUT minor /* minor version number (non-negative integer) */
OUT drmaa_context_error_buf /*Contains a context sensitive error upon failed return*/

SHALL return the major and minor version numbers of the DRMAA library; for DRMAA 1.0, 'major' is 1 and 'minor' is 0.

drmaa_version routine SHALL return DRMAA_ERRNO_SUCCESS on success.

drmaa_get_DRM_system(OUT drm_systems, OUT drmaa_context_error_buf)
OUT drm_systems /* DRM systems information (string) *
OUT drmaa_context_error_buf /* Contains a context sensitive error upon failed return*/

If called before drmaa_init(), it SHALL return a comma delimited DRM systems string, one per each DRM system provided implementation. If called after drmaa_init(), it SHALL return the selected DRM system. The output (string) is implementation dependent.

drmaa_get_DRM_system routine SHALL return DRMAA_ERRNO_SUCCESS on success.

drmaa_get_DRMAA_implementation(OUT drmaa_implementations, OUT drmaa_context_error_buf)
OUT drmaa_implementations /* DRMAA implementations information (string) *
OUT drmaa_context_error_buf /* Contains a context sensitive error upon failed return*/

If called before drmaa_init(), it SHALL return a comma delimited DRMAA implementations string, one per each DRM system provided implementation. If called after drmaa_init(), it SHALL return the selected DRMAA implementation. The output (string) is implementation dependent and COULD contain the DRM system as its part.

drmaa_get_DRM_implementation routine SHALL return DRMAA_ERRNO_SUCCESS on success.

3.3 List of DRMAA Errors

---------- these are relevant to all sections ----------

DRMAA_ERRNO_SUCCESS
Routine returned normally with success.

DRMAA_ERRNO_INTERNAL_ERROR
Unexpected or internal DRMAA error like system call failure, etc.

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE
Could not contact DRM system for this request.

DRMAA_ERRNO_AUTH_FAILURE
The specified request is not processed successfully due to authorization failure.

DRMAA_ERRNO_INVALID_ARGUMENT
The input value for an argument is invalid.

DRMAA_ERRNO_NO_MEMORY
The system is unable to allocate resources.

---------- init and exit specific ----------

DRMAA_ERRNO_INVALID_CONTACT_STRING
Initialization failed due to invalid contact string.

DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR
DRMAA could not use the default contact string to connect to DRM system.

DRMAA_ERRNO_NO_DEFAULT_CONTACT_STRING_SELECTED
No defaults contact string was provided or selected.
DRMAA requires that the default contact string is selected when there is more than one default contact string due to multiple available DRMAA implementations.

DRMAA_ERRNO_DRMS_INIT_FAILED

Initialization failed due to failure to init DRM system.

DRMAA_ERRNO_ALREADY_ACTIVE_SESSION
Initialization failed due to existing DRMAA session.

DRMAA_ERRNO_NO_ACTIVE_SESSION
Exit routine failed because there is no active session.

DRMAA_ERRNO_DRMS_EXIT_ERROR
DRM system disengagement failed.

---------- job attributes specific ----------
DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT
The format for the job attribute value is invalid.

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE
The value for the job attribute is invalid.

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES
The value of this attribute is conflicting with a previously set attributes.

---------- job submission specific ----------

DRMAA_ERRNO_TRY_LATER
Could not pass job now to DRM system. A retry MAY succeed however (saturation).

DRMAA_ERRNO_DENIED_BY_DRM
The DRM system rejected the job. The job will never be accepted due to DRM configuration or job template settings.

---------- job control specific ----------

DRMAA_ERRNO_INVALID_JOB
The job specified by the 'jobid' does not exist.

DRMAA_ERRNO_RESUME_INCONSISTENT_STATE
The job has not been suspended. The RESUME request SHALL NOT be processed.

DRMAA_ERRNO_SUSPEND_INCONSISTENT_STATE

The job has not been running, and it cannot be suspended.

DRMAA_ERRNO_HOLD_INCONSISTENT_STATE
The job cannot be moved to a HOLD state.

DRMAA_ERRNO_RELEASE_INCONSISTENT_STATE
The job is not in a HOLD state.

DRMAA_ERRNO_EXIT_TIMEOUT
We have encountered a time-out condition for drmaa_synchronize or drmaa_wait.

DRMAA_ERRNO_NO_RUSAGE
This error code is returned by drmaa_wait() when a job has finished but no rusage and stat data could be provided.

4. Security Considerations

The DRMAA API does not specifically assume the existence of GRID Security infrastructure. The scheduling scenario described herein presumes that security is handled at the point of job authorization/execution on a particular resource. It is assumed that credentials owned by the process using the API are used by the DRMAA implementation to prevent abuse of the interface. In order to not unnecessarily restrict the spectrum of usable credentials, no explicit interface is defined for passing credentials.

It is conceivable an authorized but malicious user could use a DRMAA implementation or a DRMAA enabled application to saturate a DRM system with a flood of requests. Unfortunately for the DRM system this case is not distinguishable from the case of an authorized good-natured user that has many jobs to be processed. For this case DRMAA defines the DRMAA_ERRNO_TRY_LATER return code to allow a DRM system to reject requests and properly indicate DRM saturation.

DRMAA implementers should guard against buffer overflows that could be exploited through DRMAA enabled interactive applications or web portals. Implementations of the DRMAA API will most likely require a network to coordinate subordinate DRMS, however the API makes no assumptions about the security posture provided the networking environment. Therefore, application developers should further consider the security implications of "on-the-wire" communications.

For environments that allow remote or protocol based DRMAA clients DRMAA should consider implementing support for secure transport layers to prevent man in the middle attacks. DRMAA does not impose any security requirements on its clients.

5. Author Information

Roger Brobst
rbrobst@cadence.com
Cadence Design Systems, Inc
555 River Oaks Parkway

San Jose, CA 95134

Waiman Chan
waimanc@us.ibm.com
International Business Machines Corporation
2455 South Road
Poughkeepsie, NY 12601

Fritz Ferstl
fritz.ferstl@sun.com
Sun Microsystems GmbH
Dr.-Leo-Ritter-Str. 7
D-93049 Regensburg
Germany

Jeffrey T. Gardiner
gardiner@imaging.robarts.ca
Robarts Research Institute
PO Box 5015, 100 Perth Drive
London ON, N6A 5K8
Canada

Andreas Haas
andreas.haas@sun.com
Sun Microsystems GmbH
Dr.-Leo-Ritter-Str. 7
D-93049 Regensburg
Germany

Bill Nitzberg
nitzberg@pbspro.com
Altair Grid Technologies
2685 Marine Way, Suite 1209
Mountain View, CA 94043

Hrabri L. Rajic
hrabri.rajic@intel.com
Intel Americas Inc.
1906 Fox Drive
Champaign, IL 61820

Daniel Templeton
dan.templeton@sun.com
Sun Microsystems
18 Network Circle, UMPK18-117
Menlo Park, CA 94025

John Tollefsrud
j.t@sun.com
Sun Microsystems
18 Network Circle, UMPK18-211
Menlo Park, CA 94025

Peter Tröger
peter.troeger@hpi.uni-potsdam.de
Hasso-Plattner-Institute at University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
D-14482 Potsdam
Germany

Contributors

We gratefully acknowledge the contributions made to this specification by Nicholas Geib, Tim Harsch, Krzysztof Kurowski, Ignacio M. Llorente, and Ruben S. Montero.

Acknowledgements

We are grateful to numerous colleagues for discussions on the topics covered in this document, in particular (in alphabetical order, with apologies to anybody we've missed) Guillaume Alleon, Ali Anjomshoaa, Matthieu Cargnelli, Karl Czajkowski, Paul Foley, Becky Gietzel, Ancor Gonzalez Sosa, Greg Hewgill, Rayson Ho, Eduardo Huedo, Peter G. Lane, Miron Livny, Andre Merzky, Steven Newhouse, Michaell Primeaux, Greg Quinn, Alexander Saar, Martin Sarachu, Jennifer Schopf, Enrico Sirola, Chris Smith, and Douglas Thain.

6. Intellectual Property Statement

The GGF 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 GGF Secretariat.

The GGF 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 GGF Executive Director.

7. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the GGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

8. Full Copyright Notice

developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assignees.

9. References

[BRADNER1] Bradner, S. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.

[GDF.22] Hrabri Rajic, Roger Brobst, Waiman Chan, Fritz Ferstl, Jeff Gardiner, Andreas Haas, Bill Nitzberg, and John Tollefsrud. Distributed Resource Management Application API Specification 1.0. http://forge.ggf.org/projects/drmaa-wg/, 2004