Altair Engineering Inc.

Altair Grid Engine Documentation

DRMAA2 Guide

Author:
Altair Engineering

Version:
2025.1.0 (8.10.0)

January 10, 2025

Contents

1 Distributed Resource Management Application API 2 (DRMAA2) Support

1.1 Introduction

DRMAA2 is an open standard developed by the Open Grid drmaa2_lasterror(3)Form(OGF) DRMAA working group. It standardizes an API(Application Programming Interface) which contains a set of functions for job submission, job monitoring, and cluster monitoring.

The standard OGF GFD-R-P.194 document defines the semantics of functions in an abstract interface definition language (IDL). Other standards, like OGF GFD-R-P.198 define so called language bindings, i.e. transformations of the abstract interface definition language to a programming language like ANSI C.

This document discusses the DRMAA2 API with a focus on the C API implementation, which is available in Altair Grid Engine 8.2.

> Note: The current version implements all mandatory DRMAA2 functions. Optional functionality like the reservation session are partially implemented. Please help to test the API functionality and report issues to the Univa support portal.

1.2 Differences to DRMAA1

Altair Grid Engine contains the first version of the DRMAA library (from now on called DRMAA1), which implements the first version of OGF DRMAA as specified in the documents OGF GFD-R-P.130. The C API implementation and the Java API of DRMAA1 are shipped as part as Altair Grid Engine. DRMAA1 will not be replaced by its successor DRMAA2 since DRMAA2 is not backward compatible to DRMAA1. Applications written using DRMAA1 can not be compiled against DRMAA2 without changing the source code. DRMAA1 implements about 40 functions while DRMAA2 has about 100 functions. Major differences are:

- The namespace of the functions is drmaa2, i.e. C functions have a drmaa2_ prefix.
- DRMAA2 implements a multi-session concept.
- DRMAA2 implements a monitoring-session concept, which allows monitoring any kind of jobs, hosts, and queues in the cluster.
- DRMAA2 allows for multiple job-sessions simultaneously, while DRMAA1 allows just one job session per application.
- DRMAA2 data structures are extensible, i.e. over time more Altair Grid Engine related functionality can be added without breaking compatiblity to existing applications or to the DRMAA2 standard.
- DRMAA2 applications are connected to the Altair Grid Engine master process on a subscription basis and not through polling when functions like status-queries are executed. This makes DRMAA2 an ideal candidate for long running monitoring and job submission applications.

1.3 Functional Overview

The DRMAA2 C API is implemented in a way that it subscribes objects from Altair Grid Engine qmaster. This means that function calls for retrieving status information (like job status queries) do not trigger any communication to the Altair Grid Engine master process. Whenever a state changes in the master for a subscribed object then this change is sent out to the application automatically and updated.

The available functions can be arranged in following groups:

- General functions
- Monitoring session functions
- Job session functions
- Job related functions

1.3.1 General Functions

General functions contain implementations of higher level data structures like lists and dictionaries, which are not part of ANSI C but mandatory for the DRMAA2 implementation.

Examples of such data structure related functions are:

```
drmaa2_list_create()
drmaa2_list_free()
drmaa2_list_get()
drmaa2_list_add()
drmaa2_list_del()
drmaa2_list_size()
```

and

```
drmaa2_dict_create()
drmaa2_dict_free()
drmaa2_dict_list()
drmaa2_dict_free()
drmaa2_dict_list()
drmaa2_dict_has()
drmaa2_dict_get()
drmaa2_dict_del()
drmaa2_dict_set()
```

Examples of other general DRMAA2 functions are drmaa2_get_drms_name(), drmaa2_get_drms_version(), drmaa2_get_drmaa_name(), dmraa2_get_drmaa_version(). Those functions are used to identify the Altair Grid Engine system, the versions, the DRMAA2 implementation name and the respective version.

Functionality which is defined by the DRMAA2 standard but does not necessarily need to be implemented by the DRMAA2 API is called a DRMAA2 *capability*. An example of how

capabilities can be discovered by the application during runtime is in the drmaa2_howto6.c in the $SGE_ROOT/examples/drmaa2/ directory.

Introspection related functions have to be used in order to get information about Altair Grid Engine DRMAA2 specific enhancements of the data structures. Following functions are available: drmaa2_get_instance_value(), drmaa2_describe_attribute(), and drmaa2_set_instance_value().

An example for a Altair Grid Engine specific data structure is defined in the drmaa2_jtemplate structure: uge_pe. This attribute represents an Altair Grid Engine parallel environment. An example of how to use it is in the drmaa2_howto6.c file.

1.3.2 Monitoring Session Functions

In a monitoring session **jobs**, **queues**, and **hosts** can be monitored. For an application at most one monitoring session can be open. It is recommended to use a monitoring session only for long term cluster wide monitoring applications since it requires the master to copy data about all jobs, queues, and hosts at the point in time when the session is opened to the client application. Hence an unnecessary open call can lead to large communication overhead even when the monitoring session is not used. During the lifetime of an open session only changes in the monitored objects are pushed from the qmaster automatically to the client application. Hence the actual calls for getting job lists and object statuses are cheap in that they don't result in any communication to the qmaster. Hence such calls can be done very frequently in the Altair Grid Engine DRMAA2 C implementation. Monitoring session related functions have the common prefix drmaa2_msession_.

Examples for monitoring session related functions are:

```
drmaa2_open_msession()
drmaa2_close_msession()
```

and

```
drmaa2_msession_get_all_jobs()
drmaa2_msession_get_all_queues()
drmaa2_msession_get_all_machines()
```

The *getter functions* expect filters as parameters. If a filter is set to NULL all related objects are returned as **deep copies** (which need to be freed), otherwise only a matching set of objects are returned. More details are explained in the man pages.

1.3.3 Job Session Functions

A job session is required in order to submit jobs and to monitor all jobs belonging to a particular job session. Multiple job sessions can be open simultaneously from one application. Job workflows can be isolated using job sessions. Job sessions need to be created before they can be opened. A job session is a persistent object in the Altair Grid Engine qmaster, which exists until the destroy function is called. When the application ends without destroying a job session the session can be re-used at a later point in time. Job sessions can

be managed by the Altair Grid Engine **admin user** alternatively with the qconf command. This allows the admin to perform cleanups of job sessions. Important commands are listening to all DRMAA2 job sessions with *qconf -sdjsl* and deleting specific DRMAA2 job sessions with qconf *-ddjs user@jobsession*.

Functions for managing job sessions are:

```
drmaa2_create_jsession()
drmaa2_open_jsession()
drmaa2_close_jsession()
drmaa2_destroy_jsession()
```

Functions for working with job sessions are:

```
drmaa2_jsession_get_contact()
drmaa2_jsession_get_session_name()
drmaa2_jsession_get_job_categories()
drmaa2_jsession_get_jobs()
drmaa2_jsession_get_job_array()
drmaa2_jsession_run_job()
drmaa2_jsession_run_bulk_jobs()
drmaa2_jsession_wait_any_started()
drmaa2_jsession_wait_any_terminated()
```

1.3.4 Job Related Functions

In order to submit jobs a job template needs to be allocated and filled in with values which represents the job. Jobs are treated as binary jobs in DRMAA2 like in DRMAA1, i.e the application needs to be present and accessible on the execution host where the job is executed. Then a job can either submitted as a single job or as a bulk job with the job session related run_job functions. A bulk job is a Altair Grid Engine array job. Job related functions have the common prefix *drmaa2_j_*.

Examples of job related functions are:

```
drmaa2_string drmaa2_j_get_id()
drmaa2_string drmaa2_j_get_session_name()
drmaa2_jtemplate drmaa2_j_get_jt()
drmaa2_error drmaa2_j_suspend()
drmaa2_error drmaa2_j_resume()
drmaa2_error drmaa2_j_hold()
drmaa2_error drmaa2_j_release()
drmaa2_error drmaa2_j_terminate()
drmaa2_error drmaa2_j_wait_started()
drmaa2_error drmaa2_j_wait_terminated()
drmaa2_jstate drmaa2_j_get_state()
drmaa2_jinfo drmaa2_j_get_info()
```

1.4 Compiling and Running Application

In order to compile a DRMAA2 C application the *drmaa2.h* file needs to be known by the compiler, as well as the DRMAA2 C library needs to be in the path for the linker. The C header file is located in the *$SGE_ROOT/drmaa/include* directory. The DRMAA2 library is in the architecture specific library directory of the Altair Grid Engine installation: *$SGE_ROOT/drmaa/lib/lx-amd64/libdrmaa2.so*

The following example shows how to compile the DRMAA2 howto examples with the gcc compiler.

```
gcc -L$SGE_ROOT/drmaa/lib/lx-amd64 -I$SGE_ROOT/drmaa/include -o howto1
    $SGE_ROOT/examples/drmaa2/drmaa2_howto1.c -ldrmaa2
```

In order to run the application the path to the shared library needs to be known. Typically the **$LD_LIBRARY_PATH** environment variable is set for doing so. Alternatively, but not recommended, is to put the DRMAA2 shared library in the library search path of the host on which the application is running.

The following example shows how to run a DRMAA2 application in a *bash* shell:

```
export LD_LIBRARY_PATH=$SGE_ROOT/drmaa/lib/lx-amd64
./howto1
```

2 API Reference Documentation

All DRMAA2 C functions are documented in separate man pages which are located in the *$SGE_ROOT/man/man3* directory. The man pages are the main resource for programmers in order to understand the functions. A higher level view describing the semantics of functions is available in the DRMAA2 specification. The original OGF DRMAA2 specifications are available from the Open Grid Forum at https://www.ogf.org/ogf/doku.php

Univa recorded a webinar about the DRMAA2 implementation, which is available online: http://www.univa.com/resources/webinar-drmaav2.php

The remaining parts of this documentation contains the description of the DRMAA2 C API functions from the man packages.

3 drmaa2_close_jsession

3.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_close_jsession(drmaa2_jsession jsession);
```

3.2 DESCRIPTION

Closes a job session. Closing means that jobs submitted within this session can not be used / controlled / queried anymore. Most drmaa2_jsession function calls will fail. No further reports of jobs belonging to the job session are sent from the Altair Grid Engine master process to the DRMAA2 application. If no other job session or monitoring session is open the DRMAA2 application is disconnected from the Altair Grid Engine master process. After closing the job session the job session object must be freed using the drmaa2_jsession_free(3) call.

3.3 RETURN VALUES

Returns a drmaa2_error value. In case of success DRMAA2_SUCCESS is returned otherwise the error value which indicates the error. In case of an error a more detailed error description is set for the calling thread. This description can be fetched with the drmaa2_lasterror_text(3) function.

3.4 EXAMPLE

```
/* "unique_jsession" must exist on AGE master process */

drmaa2_jsession js = drmaa2_open_jsession("unique_jsession");

if (js != NULL) {
   /* do something with the job session */
   drmaa2_j_list jobs = drmaa2_jsession_get_jobs(js, NULL);
   /* process jobs and free list ... */
   ...

   if (DRMAA2_SUCCESS != drmaa2_close_jsession(ms)) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during closing the job session: %s\n", error);
      drmaa2_string_free(&error);
   }
   drmaa2_jsession_free(&ms);
}
```

3.5 SEE ALSO

drmaa2_open_jsession(3), drmaa2_destroy_jsession(3), drmaa2_create_jsession(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

4 drmaa2_close_msession

4.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_close_msession(drmaa2_msession ms);
```

4.2 DESCRIPTION

Closes an active monitoring session. In case there is no other session (job sessions) open, the DRMAA2 library disengages from qmaster. In case of open job sessions the subscription scope is decreased so that less information is sent periodically from the Altair Grid Engine qmaster process. All jobs belonging to the monitoring session (and which are not part of any open job session) are reaped from internal caches. Especially finished jobs are removed and are not available anymore even after re-opening a monitoring job session.

After closing the monitoring session the monitoring session object is invalid and must be freed using the drmaa2_msession_free(3) function.

4.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case the session could be closed or an DRMAA2 error code indicating the error. In case of an error the drmaa2_lasterror_text(3) prints more detailed information.

4.4 EXAMPLE

```
drmaa2_msession ms = drmaa2_open_msession(NULL);

if (ms != NULL) {
   ...
   if (DRMAA2_SUCCESS != drmaa2_close_msession(ms)) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during closing the monitoring session: %s\n", error);
      drmaa2_string_free(&error);
   }
   drmaa2_msession_free(&ms);
}
```

4.5 SEE ALSO

drmaa2_msession_free(3), drmaa2_open_msession(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_queues(3), drmaa2_msession_get_all_machines(3)

5 drmaa2_close_rsession

5.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_close_rsession(drmaa2_rsession rsession);
```

5.2 DESCRIPTION

Reservation sessions are currently not supported in the Altair Grid Engine DRMAA2 implementation.

5.3 RETURN VALUES

Returns `DRMAA2_UNSUPPORTED_OPERATION`.

5.4 SEE ALSO

drmaa2_open_rsession(3), drmaa2_get_rsession_names(3), drmaa2_destroy_rsession(3), drmaa2_create_rsession(3), drmaa2_mession_get_all_reservations(3).

6 drmaa2_create_jsession

6.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jsession drmaa2_create_jsession(const char *session_name, const char *contact);
```

6.2 DESCRIPTION

Creates a new and persistent job session on the Altair Grid Engine master and opens it. For successful creation a session with the name may not already exist. The contact string must be NULL when using Altair Grid Engine.

Note: In Altair Grid Engine DRMAA2 job sessions can be listed, created, and deleted also by qconf(3) calls.

6.3 RETURN VALUES

On success a newly allocated drmaa2_jsession object is returned. In case case of failure NULL is returned and the error code and description is set for the calling thread.

6.4 EXAMPLE

```
drmaa2_jsession js = drmaa2_create_jsession("mysession", NULL);

if (js == NULL) {
   /* an error happend */
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during creation of job session with the name %s: %s\n",
              "mysession", error);
   drmaa2_string_free(&error);
} else {
   /* do something with the job session */
   ...
   /* close jsession */
   drmaa2_jsession_close(js);
   /* free jsession */
   drmaa2_jsession_free(&js);
}
```

6.5 SEE ALSO

drmaa2_close_jsession(3), drmaa2_open_jsession(3), drmaa2_destroy_jsession(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

7 drmaa2_create_rsession

7.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_rsession drmaa2_create_rsession(const char *session_name, const char *contact);
```

7.2 DESCRIPTION

Reservation sessions are currently not supported in the Altair Grid Engine DRMAA2 implementation.

7.3 RETURN VALUES

Returns NULL and sets the failure code `DRMAA2_UNSUPPORTED_OPERATION`.

7.4 SEE ALSO

drmaa2_open_rsession(3), drmaa2_close_rsession(3), drmaa2_destroy_rsession(3), drmaa2_get_rsession_names(3), drmaa2_msession_get_all_reservations(3)

8 drmaa2_describe_attribute

8.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_describe_attribute(const void *instance, const char *name)
```

8.2 DESCRIPTION

Returns a copy of a human readable description of a Altair Grid Engine specific attribute given by `name` for a certain DRMAA2 object (instance). Allowed types for the instance input values are:

- drmaa2_jtemplate
- drmaa2_jinfo
- drmaa2_rtemplate
- drmaa2_rinfo
- drmaa2_queueinfo
- drmaa2_machineinfo
- drmaa2_notification

8.3 RETURN VALUES

Returns a copy of the attribute value as `drmaa2_string`. In case there is no description available or an other error occurred NULL is returned.

8.4 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

9 drmaa2_destroy_jsession

9.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_destroy_jsession(const char * session_name);
```

9.2 DESCRIPTION

Removes a persistent DRMAA2 job session with the given session_name from the Altair Grid Engine qmaster process. The session must belong to the user of the application otherwise a failure is returned. Only Altair Grid Engine admins and operators are allowed to delete other user job sessions. If a job session is removed while an other application uses it, the behaviour is undefined. Existing DRMAA2 job session names can be fetched wth the drmaa2_get_jsession_names(3) function.

Note: In Altair Grid Engine DRMAA2 job sessions can be listed, created, and deleted also by qconf(3) calls.

9.3 RETURN VALUES

On success `DRMAA2_SUCCESS` is returned. In case case of an failure the error code indicating the failure reason is returned and a description is set for the calling thread. The failure description can be fetched with the drmaa2_lasterror_text(3) function call.

9.4 EXAMPLE

```
drmaa2_jsession js = drmaa2_create_jsession("mysession", NULL);

if (js == NULL) {
   /* an error happend */
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during creation of job session with the name %s: %s\n",
              "mysession", error);
   drmaa2_string_free(&error);
} else {
   /* do something with the job session */
   ...
   /* close jsession */
   drmaa2_jsession_close(js);
   /* free jsession */
   drmaa2_jsession_free(&js);

   /* remove the job session from the AGE master process */
   if (drmaa2_destroy_jsession("mysession") != DRMAA2_SUCCESS) {
```

```
      /* an error happend */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during destruction of job session with the name %s: %s\n",
                  "mysession", error);
      drmaa2_string_free(&error);
   }
}
```

9.5 SEE ALSO

drmaa2_close_jsession(3), drmaa2_open_jsession(3), drmaa2_create_jsession(3), drmaa2_get_jsession_names(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

10 drmaa2_destroy_rsession

10.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_destroy_rjsession(const char * session_name);
```

10.2 DESCRIPTION

Reservations sessions are currently only partially supported in the Altair Grid Engine DRMAA2 implementation.

10.3 RETURN VALUES

Returns `DRMAA2_UNSUPPORTED_OPERATION`.

10.4 SEE ALSO

drmaa2_create_rsession(3), drmaa2_get_rsession_names(3), drmaa2_msession_get_all_reservations(3)

11 drmaa2_dict_create

11.1 SYNOPSIS

```
#include "drmaa2.h"
```

```
typedef void (*drmaa2_dict_entryfree)(char **key, char **val);

drmaa2_dict drmaa2_dict_create(const drmaa2_dict_entryfree callback);
```

11.2 DESCRIPTION

Creates a new string dictionary where pairs of strings are stored as key and values. The given callback function is called for each entry when it is deleted either by drmaa2_dict_del(3) or by freeing the dictionary with drmaa2_dict_free(3). The callback function must be from type `drmaa2_dict_entryfree` or NULL of no callback should be used.

11.3 RETURN VALUES

Upon successful completion drmaa2_dict_create(3) returns a newly allocated `drmaa2_dict` dictionary. In case of an error NULL is returned and the specific error is set for the calling thread. The error can be read out by using drmaa2_lasterror(3) and/or drmaa2_lasterror_text(3).

11.4 EXAMPLE

```
drmaa2_j job;
drmaa2_jtemplate jt = drmaa2_jtemplate_create();

jt->jobName = strdup("EnvironmentTest");
jt->remoteCommand = strdup("env");

/* Create dictionary for job environment variables.
 * Because variables are non-allocated strings a callback
 * is not required.
 */
if ((environment = drmaa2_dict_create(NULL)) == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   error = 1;
}

if (drmaa2_dict_set(environment, "my_environment_variable", "has_a_value") != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new environment variable in the dictionary.\n");
   error = 1;
}

if (drmaa2_dict_set(environment, "my_empty_variable", "") != DRMAA2_SUCCESS) {
   printf("Error: Could not set an empty environment variable in the dictionary.\n");
   error = 1;
}
```

```
jt->jobEnvironment = environment;

/* submit job */
if ((job = drmaa2_jsession_run_job(js, jt)) == NULL) {
   printf("Error: Could not submit job.\n");
}

/* Calls drmaa2_dict_free(3) implicitly. */
drmaa2_jtemplate_free(&jt);
```

11.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

12 drmaa2_dict_del

12.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_dict_del(drmaa2_dict dict, const char *key)
```

12.2 DESCRIPTION

Deletes a key/value pair out of a given dictionary. If the dictionary was created using a callback function. the function is executed before the values are deleted. This is usually used for automatically freeing allocated memory.

12.3 RETURN VALUES

In case of success `DRMAA2_SUCCESS` is returned otherwise the error code indcating the error is returned.

12.4 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}

/* ... */
```

```
/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_del(dict, "key") != DRMAA2_SUCCESS) {
   printf("Error during deletion of the key value pair.");
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

12.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

13 drmaa2_dict_free

13.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_dict_free(drmaa2_dict* dict);
```

13.2 DESCRIPTION

Frees the DRMAA2 dictionary and sets it to NULL. If the dictionary was created with a callback function, this function is called for all elements of the dictionary.

13.3 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}
```

```
/* ... */

/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key2"), strdup("")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

13.4 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

14 drmaa2_dict_get

14.1 SYNOPSIS

```
#include "drmaa2.h"

const char* drmaa2_dict_get(const drmaa2_dict dict, const char* key)
```

14.2 DESCRIPTION

Searches the value of a given key in a given dictionary.

14.3 RETURN VALUES

Returns the value of the key or NULL in case the key was not found or an error occurred (like wrong arguments). The returned value is not a copy, it is a pointer to the value stored

in the dictionary.

14.4 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}

/* ... */

/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_get(dict, "key") != NULL) {
   printf("The key \"key\" has the value \"%s\".\n", drmaa2_dict_get(dict, "key"));
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

14.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

15 drmaa2_dict_has

15.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_bool drmaa2_dict_has(const drmaa2_dict dict, const char* key)
```

15.2 DESCRIPTION

Checks whether a specific key is part of the dictionary or not.

15.3 RETURN VALUES

Returns `DRMAA2_TRUE` if the given key is stored in the dictory and `DRMAA2_FALSE` if the key can not be found in the dictionary.

15.4 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}

/* ... */
drmaa2_string_list keys;

/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key2"), strdup("")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_has(dict, "key2") != DRMAA2_TRUE) {
   printf("Error! Expected that key2 is a key in the dictionary.\n");
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

15.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

16 drmaa2_dict_list

16.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_dict_list(const drmaa2_dict dict)
```

16.2 DESCRIPTION

Creates a newly allocated `drmaa2_string_list` out of the keys from the dictionary. When the dictionary is defined but has 0 keys, a list with 0 elements is returned. The list is required to get the keys in order to traverse a dictionary. After using the list it must be freed with drmaa2_dict_free(3) by the caller.

16.3 RETURN VALUES

Returns a `drmaa2_string_list` or NULL in case of an error.

16.4 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}

/* ... */
drmaa2_string_list keys;

/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
```

```
   return;
}

if (drmaa2_dict_set(dict, strdup("key2"), strdup("")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if ((keys = drmaa2_dict_list(dict)) != NULL) {
   long size, i;
   size = drmaa2_list_size(keys);
   for (i = 0; i < size; i++) {
      drmaa2_string key = drmaa2_list_get(keys, i);
      /* use key for getting the value in the dict */
      printf("Key: %s Value: %s\n", key, drmaa2_dict_get(dict, key));
   }
   drmaa2_list_free(&keys);
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

16.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

17 drmaa2_dict_set

17.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_dict_set(const drmaa2_dict dict, const char *key, const char *value)
```

17.2 DESCRIPTION

Stores a key / value pair in the given dictionary. Key and value strings are stored as pointers (no copies are made). If the dictionary already contains the key, the value pointer is removed. If the dictionary was created with a callback, the callback is called with NULL for the key. It is recommended that the user have to lookup if the key already exists before calling this function.

17.3 RETURN VALUES

In case of success DRMAA2_SUCCESS is returned otherwise the error code indcating the error is returned.

17.4 EXAMPLE

```
static void drmaa2_dict_string_free(char** key, char** value)
{
   drmaa2_string_free(key);
   drmaa2_string_free(value);
}

/* ... */

/* Create dictionary for job environment variables. */
drmaa2_dict dict = drmaa2_dict_create((drmaa2_dict_entryfree)drmaa2_dict_string_free);

if (dict == NULL) {
   printf("Error: Could not create a new dictionary.\n");
   return;
}

if (drmaa2_dict_set(dict, strdup("key"), strdup("value")) != DRMAA2_SUCCESS) {
   printf("Error: Could not set a new value in the dictionary.\n");
   return;
}

if (drmaa2_dict_del(dict, "key") != DRMAA2_SUCCESS) {
   printf("Error during deletion of the key value pair.");
}

/* Frees strings implicitly. */
drmaa2_dict_free(&dict);
```

17.5 SEE ALSO

drmaa2_dict_create(3), drmaa2_dict_free(3), drmaa2_dict_list(3), drmaa2_dict_has(3), drmaa2_dict_get(3), drmaa2_dict_del(3), drmaa2_dict_set(3)

18 drmaa2_get_drms_name

18.1 SYNOPSIS

```
#include "drmaa2.h"
```

```
drmaa2_string drmaa2_get_drms_name();
```

18.2 DESCRIPTION

Allocates a new `drmaa2_string` which contains the name of the distributed resource management system. Using Altair Grid Engine the string is typically Altair Grid Engine.

18.3 RETURN VALUES

This function returns a newly allocated `drmaa2_string` or NULL if insufficient memory was available.

18.4 EXAMPLE

```
drmaa2_string name = drmaa2_get_drms_name();

if (name != NULL) {
   fprintf("DRMS name: %s", (char *) v->major);
   drmaa2_string_free(&name);
}
```

18.5 SEE ALSO

drmaa2_string_free(3), drmaa2_get_drms_version(3)

19 drmaa2_get_drms_version

19.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_version drmaa2_get_drms_version();
```

19.2 DESCRIPTION

Allocates a new `drmaa2_version` object with the major and minor version of Altair Grid Engine.

19.3 RETURN VALUES

Upon successful completion drmaa2_get_drms_version(3) returns a newly allocated `drmaa2_version` object. In case of out-of-memory NULL is returned. The newly allocated object must be freed with drmaa2_version_free(3).

19.4 EXAMPLE

```
drmaa2_version v = drmaa2_get_drms_version();

if (v != NULL) {
   fprintf("Major version: %s", (char *) v->major);
   fprintf("Minor version: %s", (char *) v->minor);
   drmaa2_version_free(&v);
}
```

19.5 SEE ALSO

drmaa2_version(3), drmaa2_version_free(3), drmaa2_get_drms_name(3)

20 drmaa2_get_instance_value

20.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_get_instance_value(const void *instance, const char *name);
```

20.2 DESCRIPTION

Reads out a Altair Grid Engine specific attribute value with the given name from the object given as instance argument. A copy of the value is returned to the caller as `drmaa2_string`. The value must be free by the caller. A instance must be one of the following types: * jtemplate * jinfo * rtemplate * rinfo * queueinfo * machineinfo * notification

20.3 RETURN VALUES

Returns a copy of the attribute value as `drmaa2_string` or NULL in case the Altair Grid Engine specific value is not set or not available. In case of an error the error code and error text is set for the calling thread.

20.4 EXAMPLE

```
drmaa2_jtemplate jt = drmaa2_jtemplate_create();
drmaa2_string_list uge_attributes = drmaa2_jtemplate_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
```

```
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      if (strcmp(attr, "JTEMPLATE_SPECIAL_ATTRIBUTE") == 0) {
         drmaa2_string val;
         drmaa2_set_instance_value(jt, attr, strdup("My Value"));
         /* read "My Value" out ... */
         val = drmaa2_get_instance(jt, attr);
         /* val is "My Value" */
         drmaa2_string_free(&val);
      }
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

20.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

21 drmaa2_get_jsession_names

21.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_get_jsession_names();
```

21.2 DESCRIPTION

Fetches all available persistent DRMAA2 job session names from the Altair Grid Engine master process. There is no previous call needed to perform this action. In case of success a `drmaa2_string_list` with all DRMAA2 job session names is returned. Those job session names can be used for opening a DRMAA2 job session with the drmaa2_open_jsession(3) function.

Note: In Altair Grid Engine DRMAA2 job sessions can be listed, created, and deleted also by qconf(3) calls.

21.3 RETURN VALUES

On success a `drmaa2_string_list` is returned. It might contain 0 or more entries. The string list was initialized with a callback, which deletes all allocated strings when the list is freed with drmaa2_list_free(3). In case of an failure the error code indicating the failure reason

and a description is stored for the calling thread. The failure description can be fetched with the drmaa2_lasterror_text(3) function call. The failure code can be fetched with the drmaa2_lasterror(3) call.

21.4 EXAMPLE

```
drmaa2_string_list jsession_names = drmaa2_get_jsession_names();

if (jsession_names == NULL) {
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during fetching jsession names: %s\n",
              "mysession", error);
   drmaa2_string_free(&error);
} else {
   long size, i;

   size = drmaa2_list_size(jsession_names);
   for (i = 0; i < size; i++) {
      drmaa2_string jsession_name = (drmaa2_string) drmaa2_list_get(jsession_names, i);
      printf("job session: %s\n", (char *) jsession_name);
   }

   drmaa2_list_free(&jsession_names);
}
```

21.5 SEE ALSO

drmaa2_open_jsession(3), drmaa2_create_jsession(3), drmaa2_list_size(3), drmaa2_list_get(3), drmaa2_list_free(3)

22 drmaa2_get_rsession_names

22.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_get_rsession_names();
```

22.2 DESCRIPTION

Reservation sessions are currently unsupported in Altair Grid Engine.

22.3 RETURN VALUES

Returns `DRMAA2_UNSUPPORTED_OPERATION`.

22.4 SEE ALSO

drmaa2_open_rsession(3), drmaa2_create_rsession(3), drmaa2_list_size(3), drmaa2_list_get(3), drmaa2_list_free(3), drmaa2_get_jsession_names(3)

23 drmaa2_jarray_free

23.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_jarray_free(drmaa2_jarray *jarray);
```

23.2 DESCRIPTION

Frees a previously allocated job array object and sets it to NULL.

23.3 SEE ALSO

drmaa2_jsession_run_bulk_jobs(3)

24 drmaa2_jarray_get_id

24.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_jarray_get_id(const drmaa2_jarray jarray)
```

24.2 DESCRIPTION

Returns a newly allocated `drmaa2_string` copy of the id from the array job.

24.3 RETURN VALUES

Returns a newly allocated `drmaa2_string` (char *) or NULL in case of an error. The `drmaa2_string` must be freed from the calling function.

24.4 EXAMPLE

```
drmaa2_string id = drmaa2_jarray_get_id(ja);

if (id != NULL) {
   /* ... do something with the id ... */

   /* finally free the id again */
   drmaa2_string_free(&id);
}
```

24.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_jobs(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_session_names(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3)

25 drmaa2_jarray_get_jobs

25.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_j_list drmaa2_jarray_get_jobs(const drmaa2_jarray ja)
```

25.2 DESCRIPTION

A `drmaa2_jarray` is the equivalent of an Altair Grid Engine array job. It consists of several single `drmaa2_j` jobs. This function returns all jobs which belong to the `drmaa2_jarray` as a list of `drmaa2_j` jobs. If job tasks (i.e. DRMAA2 jobs) are finsihed during the DRMAA2 application was not connected to the Altair Grid Engine master process (because the job session was closed or the application was shut-down) those jobs are not available anymore. The job objects can be used for controling the job or getting more detailed job information. Most operations on the job objects require an open DRMAA2 job or monitoring session.

25.3 RETURN VALUES

Returns a list of `drmaa2_j` job objects in a `drmaa2_j_list` or NULL in case of an error. The newly allocated list was initialized with an appropriate callback function which frees the job object. After usage the `drmaa2_j_list` must be freed with `drmaa2_list_free`.

25.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (ja != NULL) {
  drmaa2_j_list jl = drmaa2_jarray_get_jobs(ja);

  if (jl != NULL) {
     /* ... do something with the job list ... */
     long i, size;
     size = drmaa2_list_size(jl);

     for (i = 0; i < size; i++) {
        drmaa2_j job = (drmaa2_j) drmaa2_list_get(jl, i);
        /* ... do something with the job - but don't free it ...
     }

     /* finally free the complete job list */
     drmaa2_list_free(&jl);
  }
}
```

25.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

26 drmaa2_jarray_get_job_template

26.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jtemplate drmaa2_jarray_get_job_template(const drmaa2_jarray jarray)
```

26.2 DESCRIPTION

Creates a newly allocated job template, which was used for submitting the job. After closing the job session which was used for submitting the job, the job template is not available anymore. The job template needs to be freed from the calling function.

26.3 RETURN VALUES

Return a newly allocated `drmaa2_jtemplate` or NULL in case of an error.

26.4 EXAMPLE

```
/* ... array job id stems from a previous drmaa2_jarray_get_id() call ... */
drmaa2_jarray ja = drmaa2_jsession_get_job_array(js, id);

if (ja != NULL) {
  drmaa2_jtemplate template = drmaa2_jarray_get_job_template(ja);

  if (template != NULL) {
     /* ... do something with the template ... */

     /* finally free the job template */
     drmaa2_jtemplate_free(&template);
  }

  drmaa2_jarray_free(&ja);
}
```

26.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_jobs(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_session_names(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3)

27 drmaa2_jarray_get_session_name

27.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_jarray_get_session_name(const drmaa2_jarray ja)
```

27.2 DESCRIPTION

Returns a newly allocated `drmaa2_string` copy of the name of the job session the array job belongs to.

27.3 RETURN VALUES

Returns a newly allocated `drmaa2_string` (char *) with the name or NULL in case of an error. The `drmaa2_string` must be freed from the calling function.

27.4 EXAMPLE

```
/* ... array job id stems from a previous drmaa2_jarray_get_id() call ... */
drmaa2_jarray ja = drmaa2_jsession_get_job_array(js, id);

if (ja != NULL) {
  drmaa2_string name = drmaa2_jarray_get_jsession_name(ja);

  if (name != NULL) {
     /* ... do something with the session name... */

     /* finally free the job template */
     drmaa2_string_free(&name);
  }

  drmaa2_jarray_free(&ja);
}
```

27.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_jobs(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_session_names(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3)

28 drmaa2_jarray_hold

28.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_jarray_suspend(drmaa2_jarray jarray);
```

28.2 DESCRIPTION

A `drmaa2_jarray` is the equivalent of an Altair Grid Engine array job. It consists of several single `drmaa2_j` jobs. This function sets all job in array job into hold state (h), i.e. all DRMAA2 jobs which belong the array job are not going to be rescheduled anymore, if they are in an

appropriate state (queued). Jobs which are already in running or a finished state are not directly affected.

This call is asynchronous in a way that after performing the action the result might be not immediately visible through job monitoring.

28.3 RETURN VALUES

Returns an error id in case something went wrong otherwise `DRMAA2_SUCCESS` is returned. In case of an error the error description is set in the context of the calling thread. The error description can be fetched with the drmaa2_lasterror_text(3) call.

28.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (DRMAA2_SUCCESS == drmaa2_jarray_hold(ja)) {
   printf("Putting jobs of job array in hold state...\n");
}
```

28.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

29 drmaa2_jarray_release

29.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_jarray_suspend(drmaa2_jarray jarray);
```

29.2 DESCRIPTION

A `drmaa2_jarray` is the equivalent of an Altair Grid Engine array job. It consists of several single `drmaa2_j` jobs. This function removes the hold state from all jobs in array job, i.e. all DRMAA2 jobs which were in hold state are allowed to be scheduled.

This call is asynchronous in a way that after performing the action the result might be not immediately visible through job monitoring.

29.3 RETURN VALUES

Returns an error id in case something went wrong otherwise `DRMAA2_SUCCESS` is returned. In case of an error the error description is set in the context of the calling thread. The error description can be fetched with the drmaa2_lasterror_text(3) call.

29.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (DRMAA2_SUCCESS == drmaa2_jarray_release(ja)) {
   printf("Releases the hold state of jobs of job array which are in hold state...\n");
}
```

29.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

30 drmaa2_jarray_resume

30.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_jarray_resume(drmaa2_jarray jarray);
```

30.2 DESCRIPTION

A `drmaa2_jarray` is the equivalent of an Altair Grid Engine array job. It consists of several single `drmaa2_j` jobs. This function resumes the array job, i.e. all DRMAA2 jobs which belong the array job are resumed if they are in an appropriate state (i.e. user suspended). Jobs which are not in suspended state can't be resumed, i.e. jobs which are queued, finished or running are not signalled. The action Altair Grid Engine performs on the job depends on the Altair Grid Engine configuration.

This call is asynchronous in a way that after performing the action the result might be not immediately visible.

30.3 RETURN VALUES

Returns an error id in case something went wrong otherwise `DRMAA2_SUCCESS` is returned. In case of an error the error description is set in the context of the calling thread. The error description can be fetched with the drmaa2_lasterror_text(3) call.

30.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (DRMAA2_SUCCESS == drmaa2_jarray_resume(ja)) {
   printf("All users suspended jobs are singalled in order to resume...\n");
}
```

30.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

31 drmaa2_jarray_suspend

31.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_jarray_suspend(drmaa2_jarray jarray);
```

31.2 DESCRIPTION

A `drmaa2_jarray` is the equivalent of an Altair Grid Engine array job. It consists of several single `drmaa2_j` jobs. This function suspends the array job, i.e. all DRMAA2 jobs which belong the array job are suspended if they are in an appropriate state. Jobs which are not in running state can't be suspended, i.e. jobs which are queued, finished or already suspended are not signalled. The action Altair Grid Engine performs on the job depends on the Altair Grid Engine configuration.

This call is asynchronous in a way that after performing the action the result might be not immediately visible through job monitoring.

31.3 RETURN VALUES

Returns an error id in case something went wrong otherwise DRMAA2_SUCCESS is returned. In case of an error the error description is set in the context of the calling thread. The error description can be fetched with the drmaa2_lasterror_text(3) call.

31.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (DRMAA2_SUCCESS == drmaa2_jarray_resume(ja)) {
   printf("All suspended jobs of jarray are singalled in order to resume...\n");
}
```

31.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

32 drmaa2_jarray_terminate

32.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_jarray_terminate(drmaa2_jarray jarray);
```

32.2 DESCRIPTION

A drmaa2_jarray is the equivalent of an Altair Grid Engine array job. It consists of several single drmaa2_j jobs. This function sends a termination signal to all jobs of the array job.

This call is asynchronous in a way that after performing the action the result might be not immediately visible through job monitoring.

32.3 RETURN VALUES

Returns an error id in case something went wrong otherwise DRMAA2_SUCCESS is returned. In case of an error the error description is set in the context of the calling thread. The error description can be fetched with the drmaa2_lasterror_text(3) call.

32.4 EXAMPLE

```
/* ... ja is the array job which was returned by drmaa2_jsession_run_bulk_jobs() ... */

if (DRMAA2_SUCCESS == drmaa2_jarray_terminate(ja)) {
   printf("Terminating all jobs of job array...\n");
}
```

32.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3)

33 drmaa2_j_free

33.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_j_free(drmaa2_j *job);
```

33.2 DESCRIPTION

Frees a previously allocated job object and sets it to NULL.

33.3 SEE ALSO

drmaa2_jsession_run_job(3)

34 drmaa2_j_get_id

34.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_j_get_id(drmaa2_j job);
```

34.2 DESCRIPTION

Creates the Altair Grid Engine job identifier for the job. It is used for faster access to the job id than creating a job info object.

34.3 RETURN VALUES

Returns a newly allocated job identifier as `drmaa2_string` which must be freed from the caller. In case of an error NULL is returned and the error id and error message is set in the context of the calling thread.

34.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

35 drmaa2_j_get_info

35.1 SYNOPSIS

```
#include "drmaa2.h"

typedef struct {
   drmaa2_string      jobId;
   int                exitStatus;
   drmaa2_string      terminatingSignal;
   drmaa2_string      annotation;
   drmaa2_jstate      jobState;
   drmaa2_string      jobSubState;
   drmaa2_string_list allocatedMachines;
   drmaa2_string      submissionMachine;
   drmaa2_string      jobOwner;
   long long          slots;
   drmaa2_string      queueName;
   time_t             wallclockTime;
   long long          cpuTime;
   time_t             submissionTime;
   time_t             dispatchTime;
   time_t             finishTime;
} drmaa2_jinfo_s;
typedef drmaa2_jinfo_s * drmaa2_jinfo;

drmaa2_jinfo drmaa2_j_get_info(const drmaa2_j job)
```

35.2 DESCRIPTION

Creates a `drmaa2_jinfo` object containing all currently available job information. Not all fields might be set depeding on the job state.

35.3 RETURN VALUES

Returns a newly allocated `drmaa2_jinfo` object which must be freed from the calling function or NULL in case of an error. In case of an error the error reason is set in the context of the calling thread. It can be read out by using the drmaa2_lasterror(3) and drmaa2_lasterror_text(3) functions.

35.4 SEE ALSO

drmaa2_jinfo_free(3), drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

36 drmaa2_j_get_jt

36.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jtemplate drmaa2_j_get_jt(const drmaa2_j job);
```

36.2 DESCRIPTION

A DRMAA2 job is submitted with a `drmaa2_jtemplate` object, which describes the characteristics of the job (job name, command, parameter, resource selection etc.). This function returns a deep copy of the job template used for job submission. It is not available for jobs not submitted with DRMAA2, or after closing all DRMAA2 sessions (even when a session is reopened later). If the job session is closed the job templated is reaped internally only when the monitoring session is closed as well otherwise it stays in memory until the monitoring session is closed, too.

36.3 RETURN VALUES

Returns a newly allocated job template which was used for submitting the job. It must be freed from the caller. In case of an error NULL is returned and the error id and error message is set in the context of the calling thread.

36.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

37 drmaa2_j_get_session_name

37.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_j_get_session_name(const drmaa2_j job);
```

37.2 DESCRIPTION

Allocates a `drmaa2_string` which contains the DRMA2 job session the job was submitted in. It might be empty for jobs not submitted with DRMAA2 but accessible with the monitoring interface.

37.3 RETURN VALUES

Returns a newly allocated job session name as `drmaa2_string` which must be freed from the caller. In case of an error NULL is returned and the error id and error message is set in the context of the calling thread. Can be NULL if the job was not submitted with DRMAA2.

37.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

38 drmaa2_j_get_state

38.1 SYNOPSIS

```
#include "drmaa2.h"

typedef enum drmaa2_jstate {
```

```
   DRMAA2_UNDETERMINED              =  0,
   DRMAA2_QUEUED                    =  1,
   DRMAA2_QUEUED_HELD               =  2,
   DRMAA2_RUNNING                   =  3,
   DRMAA2_SUSPENDED                 =  4,
   DRMAA2_REQUEUED                  =  5,
   DRMAA2_REQUEUED_HELD             =  6,
   DRMAA2_DONE                      =  7,
   DRMAA2_FAILED                    =  8
} drmaa2_jstate;

drmaa2_jstate drmaa2_j_get_state(const drmaa2_j job, drmaa2_string *substate)
```

38.2 DESCRIPTION

Determines the current state of a DRMAA2 job. Since the Altair Grid Engine DRMAA2 implemenation is event based, the function call does not result in any addionaly communication to the Altair Grid Engine master process. Hence this function can be savely called in a loop while waiting for a specific job state.

Currently the sub-state does not provide any information so the second argument must be NULL.

A job state can be fetched for any kind jobs which are currently visible through open job and monitoring sessions.

38.3 RETURN VALUES

Returns a DRMAA2 job state (`drmaa2_jstate`). In case the job stated can not be fetched DRMAA2_UNDETERMINED is returned and the failure reason is set in the context of the calling thread. It can be read out by using the drmaa2_lasterror(3) and drmaa2_lasterror_text(3) functions.

38.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

39 drmaa2_j_hold

39.1 SYNOPSIS

```
#include "drmaa2.h"
```

```
drmaa2_error drmaa2_j_hold(drmaa2_j job);
```

39.2 DESCRIPTION

Sets a user hold from to the given job. The job must be valid and the job session the job was submitted in must be open. If the job was not scheduled before the hold state was set, the job stays waiting until it is explicitly released by dmraa2_j_release(3). Setting a user hold and releasing the job from the hold state is equivalent the Altair Grid Engine command line commands `qhold -h u` and `qrls -h u`. Note that operator and system holds can not be set by this function.

Note that triggering resume is asynchronous. That means that the call does not block until the job state transistion is reported back from the Altair Grid Engine master process. Hence a following drmaa2_j_get_state(3) can still see the job as beeing suspended. The state change is visiable as soon as the Altair Grid Engine master process reports the new state to the application in the background of a job or monitoring session.

39.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case of success otherwise the error ID indicating the problem is returnd. In case of an error the text description of the error is stored in the context of the calling thread and can be read out with the drmaa2_lasterror_text(3) function.

39.4 SEE ALSO

drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3), qhold(3), qrls(3)

40 drmaa2_jinfo_create

40.1 SYNOPSIS

```
#include "drmaa2.h"

#define DRMAA2_UNSET_BOOL      DRMAA2_FALSE
#define DRMAA2_UNSET_STRING    NULL
#define DRMAA2_UNSET_NUM       -1
#define DRMAA2_UNSET_ENUM      -1
#define DRMAA2_UNSET_LIST      NULL
#define DRMAA2_UNSET_DICT      NULL
#define DRMAA2_UNSET_TIME      ((time_t) -3)
```

```
#define DRMAA2_UNSET_CALLBACK  NULL
#define DRMAA2_UNSET_JINFO     NULL
#define DRAMA2_UNSET_VERSION   NULL

typedef struct {
   drmaa2_string      jobId;
   int                exitStatus;
   drmaa2_string      terminatingSignal;
   drmaa2_string      annotation;
   drmaa2_jstate      jobState;
   drmaa2_string      jobSubState;
   drmaa2_string_list allocatedMachines;
   drmaa2_string      submissionMachine;
   drmaa2_string      jobOwner;
   long long          slots;
   drmaa2_string      queueName;
   time_t             wallclockTime;
   long long          cpuTime;
   time_t             submissionTime;
   time_t             dispatchTime;
   time_t             finishTime;
} drmaa2_jinfo_s;
typedef drmaa2_jinfo_s * drmaa2_jinfo;

drmaa2_jinfo drmaa2_jinfo_create(void);
```

40.2 DESCRIPTION

Allocates memory for a job info object, UNSETs all values (DRMAA2_UNSET_STRING, DRMAA2_UNSET_LIST etc.), and returns the object. The object must be freed by the caller. This function is required for creating a job filter based on a `drmaa2_jinfo` object.

40.3 RETURN VALUES

Returns a newly allocated `drmaa2_jinfo` object or NULL in case of an error (like out-of-memory).

40.4 EXAMPLE

```
drmaa2_jinfo filter = drmaa2_jinfo_create();
drmaa2_j_list running_jobs = NULL;

if (filter != NULL) {
    filter->jobState = DRMAA2_RUNNING;
    running_jobs = drmaa2_msession_get_all_jobs(ms, filter);

    /* do something with the list of running jobs ... */
```

```
    drmaa2_list_free(&running_jobs);
    drmaa2_jinfo_free(&filter);
}
```

40.5 SEE ALSO

drmaa2_jinfo_free(3), drmaa2_mession_get_all_jobs(3), drmaa2_jsession_get_jobs(3)

41 drmaa2_jinfo_free

41.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_jinfo_free(drmaa2_jinfo * jinfo);
```

41.2 DESCRIPTION

Frees the memory of a `drmaa2_jinfo` object and sets it to NULL.

41.3 EXAMPLE

```
drmaa2_jinfo filter = drmaa2_jinfo_create();
drmaa2_j_list running_jobs = NULL;

if (filter != NULL) {
    filter->jobState = DRMAA2_RUNNING;
    running_jobs = drmaa2_msession_get_all_jobs(ms, filter);

    /* do something with the list of running jobs ... */

    drmaa2_list_free(&running_jobs);
    drmaa2_jinfo_free(&filter);
}
```

41.4 SEE ALSO

drmaa2_jinfo_create(3), drmaa2_mession_get_all_jobs(3), drmaa2_jsession_get_jobs(3)

42 drmaa2_jinfo_impl_spec

42.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_jinfo_impl_spec(void);
```

42.2 DESCRIPTION

Returns all Altair Grid Engine specific job info (`drmaa2_jinfo`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

42.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

42.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_jtemplate_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

42.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

43 drmaa2_j_release

43.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_j_release(drmaa2_j job);
```

43.2 DESCRIPTION

Releases a user hold from the given job. The job must be valid and the job session the job was submitted in must be open. A job with a user hold, which was not running before the hold state was set, is not scheduled from the Altair Grid Engine scheduler until it is released by drmaa2_j_release(3). Setting a user hold and releasing the job from the hold state is equivalent the Altair Grid Engine command line commands `qhold -h u` and `qrls -h u`. Note that operator holds and system holds (triggered by Altair Grid Engine command line tools) can not be removed with this function.

Note that triggering a release from hold is asynchronous. That means that the call does not block until the job state transistion is reported back from the Altair Grid Engine master process. Hence a following drmaa2_j_get_state(3) can still see the job as beeing suspended. The state change is visiable as soon as the Altair Grid Engine master process reports the new state to the application in the background of a job or monitoring session.

43.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case of success otherwise the error ID indicating the problem is returnd. In case of an error the text description of the error is stored in the context of the calling thread and can be read out with the drmaa2_lasterror_text(3) function.

43.4 SEE ALSO

drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3), qhold(3), qrls(3)

44 drmaa2_j_resume

44.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_j_resume(drmaa2_j job);
```

44.2 DESCRIPTION

Resumes the job given as argument. The job must be valid and the job session the job was submitted in must be open.

Note that triggering resume is asynchronous. That means that the call does not block until the job state transistion is reported back from the Altair Grid Engine master process. Hence a following drmaa2_j_get_state(3) can still see the job as beeing suspended. The state change is visiable as soon as the Altair Grid Engine master process reports the new state to the application in the background of a job or monitoring session.

44.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case of success otherwise the error ID indicating the problem is returnd. In case of an error the text description of the error is stored in the context of the calling thread and can be read out with the drmaa2_lasterror_text(3) function.

44.4 SEE ALSO

drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

45 drmaa2_jsession_free

45.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_jsession_free(drmaa2_jsession *jobsession);
```

45.2 DESCRIPTION

Frees a previously allocated jsession object and sets it to NULL. A `drmaa2_jsession` is allocated by either by drmaa2_jsession_create(3) or by drmaa2_jsession_open(3).

45.3 EXAMPLE

```
/* "unique_jsession" must exist in AGE master process */

drmaa2_jsession js = drmaa2_open_jsession("unique_jsession");

if (js != NULL) {
```

```
   /* do something with the job session */
   drmaa2_j_list jobs = drmaa2_jsession_get_jobs(js, NULL);
   /* process jobs and free list ... */
   ...

   if (DRMAA2_SUCCESS != drmaa2_close_jsession(ms)) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during closing the job session: %s\n", error);
      drmaa2_string_free(&error);
   }
   drmaa2_jsession_free(&ms);
}
```

45.4 SEE ALSO

drmaa2_open_jsession(3), drmaa2_close_jsession(3), drmaa2_create_jsession(3), drmaa2_destroy_jsession(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

46 drmaa2_jsession_get_contact

46.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_jsession_get_contact(const drmaa2_jsession jsession);
```

46.2 DESCRIPTION

Returns the contact string of a DRMAA2 job session. In Altair Grid Engine a contact string is not used and therfore always NULL is returned.

46.3 RETURN VALUES

Returns always NULL. In the future or in other implementations a newly allocated `drmaa2_string` is expected.

46.4 SEE ALSO

drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_categories(3), drmaa2_jsession_get_jobs(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3),

drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3)

47 drmaa2_jsession_get_jarray

47.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jarray drmaa2_jsession_get_job_array(const drmaa2_jsession jsession,
  const drmaa2_string id);
```

47.2 DESCRIPTION

Searches in the given `drmaa2_jsession` for a given array job. The `drmaa2_jsession` must be open. If the array job is found a newly allocated `drmaa2_jarray` is returned. The `drmaa2_jarray` stores a list of DRMAA2 jobs, which can be queried with drmaa2_jarray_get_jobs(3).

47.3 RETURN VALUES

Returns NULL in case of an error or when the id was not found in the DRMAA2 job session. Otherwise a newly allocated `drmaa2_jarray` object is returned, which must be freed by the calling function (drmaa2_jarray_free(3)).

47.4 EXAMPLE

```
/* ... array job id stems from a previous drmaa2_jarray_get_id() call ... */
drmaa2_jarray ja = drmaa2_jsession_get_job_array(js, id);

if (ja != NULL) {
   drmaa2_j_list jl = drmaa2_jarray_get_jobs(aj);
   if (jl != NULL) {
      printf("Amount of single jobs: %lld\n", drmaa2_list_size(jl));
      /* do something other with the jobs ... */
      drmaa2_list_free(&jl);
   }

  drmaa2_jarray_free(&ja);
}
```

47.5 SEE ALSO

drmaa2_jarray_free(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_get_jobs(3), drmaa2_jarray_get_id(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3)

48 drmaa2_jsession_get_job_categories

48.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_jsession_get_job_categories(const drmaa2_jsession jsession);
```

48.2 DESCRIPTION

Returns a list of job categories available in the job session. For Altair Grid Engine a job category is represented by a job class. Hence requesting a job category in the `drmaa2_jtemplate` is equivalent to requesting a job class with qsub(3). The job category list is a list of job classes defined in Altair Grid Engine. If there is no job category defined in Altair Grid Engine the job category list is defined but empty (size of 0). After a sucessful creation of the list (with 0 or more elements) the list must be freed by the caller after usage with the drmaa2_list_free(3) function. The `drmaa2_string_list` is created with a callback function which frees all allocated strings.

48.3 RETURN VALUES

Returns a list of strings with job category names. The list has 0 or more entries. In case of an failure NULL is returned and the error id and error string is set for the calling thread.

48.4 EXAMPLE

```
drmaa2_jsession js = drmaa2_create_jsession("mysession", NULL);

if (js == NULL) {
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during creation of job session with the name %s: %s\n",
               "mysession", error);
   drmaa2_string_free(&error);
} else {
   drmaa2_string_list categories = drmaa2_jsession_get_job_categories(js);

   if (categories == NULL) {
```

```
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during getting job categories list: %s\n", error);
      drmaa2_string_free(&error);
   } else {
      long size, i;

      size = drmaa2_list_size(categories);
      for (i = 0; i < size; i++) {
         fprintf(stdout, "%s\n", (char *) drmaa2_list_get(categories, i));
         /* ... if job category has a specific name -> add it to jtemplate ... */
      }
      drmaa2_list_free(&categories);
   }

   /* close jsession */
   drmaa2_jsession_close(js);
   /* free jsession */
   drmaa2_jsession_free(&js);

   /* remove the job session from the AGE master process */
   if (drmaa2_destroy_jsession("mysession") != DRMAA2_SUCCESS) {
      /* an error happend */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during destruction of job session with the name %s: %s\n",
                  "mysession", error);
      drmaa2_string_free(&error);
   }
}
```

48.5 SEE ALSO

drmaa2_jtemplate_create(3), drmaa2_jsession_get_session_names(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_jobs(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_size(3), drmaa2_list_get(3), drmaa2_list_free(3)

49 drmaa2_jsession_get_jobs

49.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_j_list drmaa2_jsession_get_jobs(const drmaa2_jsession jsession,
  const drmaa2_jinfo filter);
```

49.2 DESCRIPTION

Returns a list of jobs in a `drmaa2_j_list` which belong to the given job session. The job list contains only jobs which are submitted in the same job session. If the application was restarted between job submission with drmaa2_jsession_run_job(3) and the drmaa2_jession_get_jobs(3) call the list contains only jobs which are still in Altair Grid Engine (meaning jobs finshed in between are not part of the job list). When a job finishes while the job session is open, the finished job is stored together with its usage in the `drmaa2_jsession` until the session is closed by drmaa2_close_jsession(3). Depending on the job state the `drmaa2_j` objects have varying information available. Freshly submitted jobs might have just a few of the fields set, while finsihed jobs have full usage information available. Array jobs are returned as single, separated jobs.

The second argument is a filter for the job list. If the filter is set to NULL all availble jobs are returned. If the filter has some UNSET entries only jobs matching the filter are returned. The filter works in the same way than for drmaa2_msession_get_all_jobs(3). For more details about job filtering please consider the drmaa2_msession_get_all_jobs(3) man page.

49.3 RETURN VALUES

Returns a newly allocated list of jobs which belong to the DRMAA2 job session. The list has 0 or more entries. In case of an failure NULL is returned and the error id and error string is set for the calling thread. An appropriate callback function was set so that drmaa2_list_free(3) removes all allocated memory.

49.4 EXAMPLE

```
/* expecting that the session was created before and some jobs are submitted */
drmaa2_jsession js = drmaa2_open_jsession("mysession");

if (js == NULL) {
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during opening of job session with the name %s: %s\n",
               "mysession", error);
   drmaa2_string_free(&error);
} else {
   drmaa2_j_list jobs = drmaa2_jsession_get_jobs(js, NULL);

   if (jobs == NULL) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during getting job session job list: %s\n", error);
      drmaa2_string_free(&error);
   } else {
      long size, i;

      size = drmaa2_list_size(jobs);
      for (i = 0; i < size; i++) {
         /* do something with the jobs...here we just terminate all jobs */
```

```
         drmaa2_j job = (drmaa2_j) drmaa2_list_get(jobs, i);
         drmaa2_j_terminate(j);
      }
      drmaa2_list_free(&categories);
   }

   /* close jsession */
   drmaa2_jsession_close(js);
   /* free jsession */
   drmaa2_jsession_free(&js);
}
```

49.5 SEE ALSO

drmaa2_jtemplate_create(3), drmaa2_jsession_get_session_names(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_list_get(3), drmaa2_list_size(3), drmaa2_list_free(3), drmaa2_j_terminate(3)

50 drmaa2_jsession_get_session_name

50.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_jsession_get_session_name(const drmaa2_jsession jsession);
```

50.2 DESCRIPTION

Returns the name of a DRMAA2 job session. This name can be used for opening a job session after it was closed and it can be used for destroying a job session (after the session was closed). The name is identical to the name which was used for creating the job session.

50.3 RETURN VALUES

Returns a newly allocated `drmaa2_string` with the name of the given job session, which must be freed after usage.

50.4 EXAMPLE

```
drmaa2_jsession js = drmaa2_create_jsession("mysession", NULL);
```

```
if (js == NULL) {
   /* an error happend */
   drmaa2_string error = drmaa2_lasterror_text();
   fprintf(stderr, "Error during creation of job session with the name %s: %s\n",
              "mysession", error);
   drmaa2_string_free(&error);
} else {
   drmaa2_string session_name = drmaa2_jsession_get_session_name(js);
   fprintf(stdout, "Session name is: %s\n", session_name?session_name:"NULL");
   drmaa2_string_free(&session_name);

   /* close jsession */
   drmaa2_jsession_close(js);
   /* free jsession */
   drmaa2_jsession_free(&js);

   /* remove the job session from the AGE master process */
   if (drmaa2_destroy_jsession("mysession") != DRMAA2_SUCCESS) {
      /* an error happend */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during destruction of job session with the name %s: %s\n",
                  "mysession", error);
      drmaa2_string_free(&error);
   }
}
```

50.5 SEE ALSO

drmaa2_jsession_get_contact(3), drmaa2_jsession_get_job_categories(3), drmaa2_jsession_get_jobs(3), drmaa2_jsession_get_job_array(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_open_jsession(3), drmaa2_create_jsession(3), drmaa2_destroy_jsession(3)

51 drmaa2_jsession_run_bulk_jobs

51.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jarray drmaa2_jsession_run_bulk_jobs(const drmaa2_jsession jsession,
                                            const drmaa2_jtemplate jtemplate,
                                            long long begin_index,
                                            long long end_index,
                                            long long step,
                                            long long max_parallel);
```

51.2 DESCRIPTION

This function submits one or more jobs (in Altair Grid Engine terminology an array job) which is based on the given job template in the given job session. For a successful array job submission the given job session needs to be open. The returned array job object can be used for monitoring and controlling the array job by either extracting the single jobs (job tasks) out of the job array or by using the job array functions directly.

A DRMAA2 job array is a set of jobs which are based on the same job template. It allows an efficient submission and handling of such jobs not only in the DRMAA2 application but also in Altair Grid Engine itself. The DRMAA2 job array corresponds to the Altair Grid Engine array job. The difference between the jobs is that specific environement variables (`$SGE_TASK_ID`) are set in order to identify the task number. The task number is for each job (which is an array job instance) different. The task ID is usually used to identify the data chunk the job has to work on or to identify which code should be executed.

The task numbers and the amount of jobs which should be part of the array job are determined by the parameters `begin_index`, `end_index`, `step`, and `max_parallel`.

- `begin_index`: The job task id if the first job array job. Allowed values are 1 and higher.
- `end_index`: The job task id of the last job of the array job. Allowed values are 1 and higher. It must be equal or higher than `begin_index`.
- `step`: The incrementor / step size of the job task ids. A common value is 1 that mean that each job task id ranging from `begin_index` to `end_index` is used. It must be equal or higher than 1. If a higher value is used the second job task id is `begin_index` + step.
- `max_parallel`: Influences how many job id tasks are allowed to run concurrently in parallel. Depending on the cluster load also less tasks can run. If DRMAA2_UNSET_NUM is used as limit, no limit is applied. Otherwise the number corresponds to the Altair Grid Engine **-tc** parameter. It must be equal or greater than 1.

Resource limits and general resource requests which are not part of the job template or the Altair Grid Engine job template enhancements (but which available in Altair Grid Engine installation with the **-l** requests - like administrator defined resources), can be requested by using the `resourceLimits` dictionary which is part of the DRMAA2 job template.

51.3 RETURN VALUES

Returns a newly allocated DRMAA2 job array object or NULL in case any error happend. In case of an error the error ID and error text is stored in the context of the thread which submitted the job. The error can be read out by drmaa2_lasterror(3) and drmaa2_lasterror_text(3). The returned job object must be freed by drmaa2_jarray_free(3) after usage. The job array contains a list of DRMAA2 jobs (type `drmaa2_j`) which can be used for a fainer grained job control and job status fetching.

51.4 EXAMPLE

```
/* Create and open a new job session. */
drmaa2_jsession js = drmaa2_create_jsession("test_session", NULL);
drmaa2_jarray ja = NULL;

if (js != NULL) {
   /* create a new job template. */
   drmaa2_jtemplate jt = drmaa2_jtemplate_create();

   /* add the job characteristics */
   jt->jobName = strdup("test_job");
   jt->remoteCommand = strdup("sleep");
   /* since no allocated strings are used we don't need to specify a callback */
   args = drmaa2_list_create(DRMAA2_STRINGLIST, NULL);
   drmaa2_list_add(args, "60");
   jt->args = args;

   /* submit 100 sleeper jobs, limiting to run 10 sleepers at a time in parallel */
   jarray = drmaa2_jsession_run_bulk_jobs(js, jt, 1, 100, 10);
}

drmaa2_jarray_free(&jarray);
drmaa2_jtemplate_free(&jt);
...
```

51.5 SEE ALSO

drmaa2_jtemplate_create(3), drmaa2_jtemplate_free(3), drmaa2_jsession_open(3), drmaa2_jsession_create(3), drmaa2_jsession_close(3), drmaa2_jsession_destroy(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_get_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_j_free(3), drmaa2_jarray_get_id(3), drmaa2_jarray_get_jobs(3), drmaa2_jarray_get_session_name(3), drmaa2_jarray_get_job_template(3), drmaa2_jarray_suspend(3), drmaa2_jarray_resume(3), drmaa2_jarray_hold(3), drmaa2_jarray_release(3), drmaa2_jarray_terminate(3)

52 drmaa2_jsession_run_job

52.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_j drmaa2_jsession_run_job(const drmaa2_jsession jsession,
  const drmaa2_jtemplate jtemplate);
```

52.2 DESCRIPTION

This function submits a job in the given job session based on the job description provided by the job template. For a successful job submission the given job session needs to be open. The returned job object can be used for monitoring and controlling the job.

Resource limits and general resource requests which are not part of the job template or the Altair Grid Engine job template enhancements (but which available in Altair Grid Engine installation with the `-l` requests - like user defined complexes), can be requested by using the `resourceLimits` dictionary which is part of the DRMAA2 job template.

52.3 RETURN VALUES

Returns a newly allocated DRMAA2 job object or NULL in case any error happend. In case of an error the error ID and error text is stored in the context of the thread which submitted the job. The error can be read out by drmaa2_lasterror(3) and drmaa2_lasterror_text(3). The returned job object must be freed by drmaa2_j_free(3) after usage.

52.4 EXAMPLE

```
/* Create and open a new job session. */
drmaa2_jsession js = drmaa2_create_jsession("test_session", NULL);
drmaa2_j job = NULL;

if (js != NULL) {
   /* create a new job template. */
   drmaa2_jtemplate jt = drmaa2_jtemplate_create();

   /* add the job characteristics */
   jt->jobName = strdup("test_job");
   jt->remoteCommand = strdup("sleep");
   /* since no allocated strings are used we don't need to specify a callback */
   args = drmaa2_list_create(DRMAA2_STRINGLIST, NULL);
   drmaa2_list_add(args, "60");
   jt->args = args;

   /* submit the jobs */
   job = drmaa2_jsession_run_job(js, jt);
}

drmaa2_j_free(&job);
drmaa2_jtemplate_free(&jt);

...
```

52.5 SEE ALSO

drmaa2_jtemplate_create(3), drmaa2_jtemplate_free(3), drmaa2_jsession_open(3), drmaa2_jsession_create(3), drmaa2_jsession_close(3), drmaa2_jsession_destroy(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_get_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_j_free(3)

53 drmaa2_jsession_wait_any_started

53.1 SYNOPSIS

```
#include "drmaa2.h"

#define DRMAA2_ZERO_TIME       ((time_t)  0)
#define DRMAA2_INFINITE_TIME   ((time_t) -1)

drmaa2_j drmaa2_jsession_wait_any_started(const drmaa2_jsession jsession,
  const drmaa2_j_list joblist, const time_t timeout);
```

53.2 DESCRIPTION

This function blocks until one of the given jobs is in or passed any started state. It also immediately returns when any job is in unknown state (`DRMAA2_UNDETERMINED`). The timeout argument determines a maximum time - in seconds - the function should block. Special constant values (defined in the `drmaa2.h` file) are `DRMAA2_ZERO_TIME` and `DRMAA2_INFINITE_TIME`. If `DRMAA2_ZERO_TIME` is used as argument, the function returns immediately after checking the job state. If the job is not in the expected state the return value is != `DRMAA2_SUCCESS` (when no other error happend it is `DRMAA2_INVALID_STATE`). When using `DRMAA2_INFINITE_TIME` the function blocks possibly endlessly. All given jobs must be part of the job session given as argument. The job session must be open.

53.3 RETURN VALUES

Returns one of the given job objects if it is (or passed) the required state. The returned object is **not** a copy. In case of a timeout or an error NULL is returned. The error reason is saved in the contex of the calling thread and can be fetched with the drmaa2_lasterror_text(3) function.

53.4 SEE ALSO

drmaa2_jsession_wait_any_terminated(3), drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3),

drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3), drmaa2_close_jsession(3), drmaa2_open_jsession(3), drmaa2_destroy_jsession(3), drmaa2_create_jsession(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

54 drmaa2_jsession_wait_any_terminated

54.1 SYNOPSIS

```
#include "drmaa2.h"

#define DRMAA2_ZERO_TIME       ((time_t)  0)
#define DRMAA2_INFINITE_TIME   ((time_t) -1)

drmaa2_j drmaa2_jsession_wait_any_terminated(const drmaa2_jsession jsession,
  const drmaa2_j_list joblist, const time_t timeout);
```

54.2 DESCRIPTION

This function blocks until one of the given jobs is in any terminated state. It also immediately returns when any job is in unknown state (`DRMAA2_UNDETERMINED`). The timeout argument determines a maximum time - in seconds - the function should block. Special constant values (defined in the `drmaa2.h` file) are `DRMAA2_ZERO_TIME` and `DRMAA2_INFINITE_TIME`. If `DRMAA2_ZERO_TIME` is used as argument, the function returns immediately after checking the job state. If the job is not in the expected state the return value is != `DRMAA2_SUCCESS` (when no other error happend it is `DRMAA2_INVALID_STATE`). When using `DRMAA2_INFINITE_TIME` the function blocks possibly endlessly. All given jobs must be part of the job session given as argument. The job session must be open.

54.3 RETURN VALUES

Returns one of the given job objects if it is in the required state. The returned object is **not** a copy. In case of a timeout or an error NULL is returned. The error reason is saved in the contex of the calling thread and can be fetched with the drmaa2_lasterror_text(3) function.

54.4 SEE ALSO

drmaa2_jsession_wait_any_terminated(3), drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3),

drmaa2_close_jsession(3), drmaa2_open_jsession(3), drmaa2_destroy_jsession(3), drmaa2_create_jsession(3), drmaa2_jsession_free(3), drmaa2_jsession_get_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

55 drmaa2_j_suspend

55.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_j_suspend(drmaa2_j job);
```

55.2 DESCRIPTION

Suspends a job given as argument. The job must be valid and the job session the job was submitted in must be open.

Note that triggering suspension is asynchronous. That means that the call does not block until the job state transistion is reported back from the Altair Grid Engine master process. Hence a following drmaa2_j_get_state(3) can still see the job as running. The state change is visiable as soon as the Altair Grid Engine master process reports the new state to the application in the background of a job or monitoring session.

55.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case of success otherwise the error ID indicating the problem is returnd. In case of an error the text description of the error is stored in the context of the calling thread and can be read out with the drmaa2_lasterror_text(3) function.

55.4 SEE ALSO

drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

56 drmaa2_jtemplate_create

56.1 SYNOPSIS

```
#include "drmaa2.h"

/* DRMAA2 defined keys for resource limits dictionary
   \GEShortName{} complex names are also allowed */
extern const char *const DRMAA2_CORE_FILE_SIZE;
extern const char *const DRMAA2_CPU_TIME;
extern const char *const DRMAA2_DATA_SIZE;
extern const char *const DRMAA2_FILE_SIZE;
extern const char *const DRMAA2_OPEN_FILES;
extern const char *const DRMAA2_STACK_SIZE;
extern const char *const DRMAA2_VIRTUAL_MEMORY;
extern const char *const DRMAA2_WALLCLOCK_TIME;

#define DRMAA2_UNSET_BOOL      DRMAA2_FALSE
#define DRMAA2_UNSET_STRING    NULL
#define DRMAA2_UNSET_NUM       -1
#define DRMAA2_UNSET_ENUM      -1
#define DRMAA2_UNSET_LIST      NULL
#define DRMAA2_UNSET_DICT      NULL
#define DRMAA2_UNSET_TIME      ((time_t) -3)
#define DRMAA2_UNSET_CALLBACK  NULL
#define DRMAA2_UNSET_JINFO     NULL
#define DRAMA2_UNSET_VERSION   NULL

typedef struct {
   drmaa2_string      remoteCommand;
   drmaa2_string_list args;
   drmaa2_bool        submitAsHold;
   drmaa2_bool        rerunnable;
   drmaa2_dict        jobEnvironment;
   drmaa2_string      workingDirectory;
   drmaa2_string      jobCategory;
   drmaa2_string_list email;
   drmaa2_bool        emailOnStarted;
   drmaa2_bool        emailOnTerminated;
   drmaa2_string      jobName;
   drmaa2_string      inputPath;
   drmaa2_string      outputPath;
   drmaa2_string      errorPath;
   drmaa2_bool        joinFiles;
   drmaa2_string      reservationId;
   drmaa2_string      queueName;
   long long          minSlots;
   long long          maxSlots;
```

```
   long long          priority;
   drmaa2_string_list candidateMachines;
   long long          minPhysMemory;
   drmaa2_os          machineOS;
   drmaa2_cpu         machineArch;
   time_t             startTime;
   time_t             deadlineTime;
   drmaa2_dict        stageInFiles;
   drmaa2_dict        stageOutFiles;
   drmaa2_dict        resourceLimits;
   drmaa2_string      accountingId;
   void *             implementationSpecific;
} drmaa2_jtemplate_s;
typedef drmaa2_jtemplate_s * drmaa2_jtemplate;

drmaa2_jtemplate drmaa2_jtemplate_create(void);
```

56.2 DESCRIPTION

Allocates memory for a job template object, UNSETs all values (DRMAA2_UNSET_STRING, DRMAA2_UNSET_LIST etc.), and returns the object. The object must be freed by the caller with drmaa2_jtemplate_free(3). This function is required for running a job with drmaa2_jsession_run_job(3) or drmaa2_jsession_run_bulk_jobs(3).

56.3 RETURN VALUES

Returns a newly allocated `drmaa2_jtemplate` object or NULL in case of an error (like out-of-memory).

56.4 SEE ALSO

drmaa2_jtemplate_free(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3)

57 drmaa2_jtemplate_free

57.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_jtemplate_free(drmaa2_jtemplate *jtemplate);
```

57.2 DESCRIPTION

Frees the memory of a `drmaa2_jtemplate` object and sets it to NULL.

57.3 SEE ALSO

drmaa2_jtemplate_create(3)

58 drmaa2_jtemplate_impl_spec

58.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_jtemplate_impl_spec(void);
```

58.2 DESCRIPTION

Returns all Altair Grid Engine specific job template attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

Following implementation specific job template attributes are available in Altair Grid Engine:

- uge_jt_pe : Specifies the name of a parallel environment the job requests. Setting this parameter is required in order to make use of the `jtemplate` attributes `minSlots` and `maxSlots`. If not parallel enviornment is requested both attributes are ignored.

58.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

58.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_jtemplate_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

58.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

59 drmaa2_j_terminate

59.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_j_terminate(drmaa2_j job);
```

59.2 DESCRIPTION

Terminates the job which is given as argument. The job must be valid and the job session the job was submitted in must be open. The Altair Grid Engine equivalent on the command line is `qdel -j`.

Note that triggering termination of a job is asynchronous. That means that the call does not block until the job state transistion is reported back from the Altair Grid Engine master process. Hence a following drmaa2_j_get_state(3) can still see the job as beeing running or in a different state. The state change is visiable as soon as the Altair Grid Engine master process reports the new state to the application in the background of a job or monitoring session.

59.3 RETURN VALUES

Returns DRMAA2_SUCCESS in case of success otherwise the error ID indicating the problem is returnd. In case of an error the text description of the error is stored in the context of the calling thread and can be read out with the drmaa2_lasterror_text(3) function.

59.4 SEE ALSO

drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

60 drmaa2_j_wait_started

60.1 SYNOPSIS

```
#include "drmaa2.h"

#define DRMAA2_ZERO_TIME       ((time_t)  0)
#define DRMAA2_INFINITE_TIME   ((time_t) -1)

drmaa2_error drmaa2_j_wait_started(const drmaa2_j job, const time_t timeout);
```

60.2 DESCRIPTION

This function blocks until the given job is in started state. The function should be called for jobs in DRMAA2_QUEUED and DRMAA2_QUEUED_HELD states. It immediately returns when the job is in any finished or unknown state (DRMAA2_FAILED, DRMAA2_DONE, DRMAA2_UNDETERMINED). The timeout argument determines a maximum time - in seconds - the function should block. Special constant values (defined in the drmaa2.h file) are DRMAA2_ZERO_TIME and DRMAA2_INFINITE_TIME. If DRMAA2_ZERO_TIME is used as argument, the function returns immediately after checking the job state. If the job is not in the expected state the return value is != DRMAA2_SUCCESS (when no other error happend it is DRMAA2_INVALID_STATE). When using DRMAA2_INFINITE_TIME the function blocks possibly endlessly.

60.3 RETURN VALUES

Returns DRMAA2_SUCCESS when the job is in any or passed a started state. Otherwise an error code is returned. The error reason is saved in the contex of the calling thread and can be fetched with the drmaa2_lasterror_text(3) function.

60.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

61 drmaa2_j_wait_terminated

61.1 SYNOPSIS

```
#include "drmaa2.h"

#define DRMAA2_ZERO_TIME       ((time_t)  0)
```

```
#define DRMAA2_INFINITE_TIME   ((time_t) -1)

drmaa2_error drmaa2_j_wait_terminated(const drmaa2_j job, const time_t timeout);
```

61.2 DESCRIPTION

This function blocks until the given job is in any terminated state. It also immediately returns when the job is in unknown state (DRMAA2_UNDETERMINED). The timeout argument determines a maximum time - in seconds - the function should block. Special constant values (defined in the drmaa2.h file) are DRMAA2_ZERO_TIME and DRMAA2_INFINITE_TIME. If DRMAA2_ZERO_TIME is used as argument, the function returns immediately after checking the job state. If the job is not in the expected state the return value is != DRMAA2_SUCCESS (when no other error happend it is DRMAA2_INVALID_STATE). When using DRMAA2_INFINITE_TIME the function blocks possibly endlessly.

61.3 RETURN VALUES

Returns DRMAA2_SUCCESS when the job is in any or passed a started state. Otherwise an error code is returned. The error reason is saved in the contex of the calling thread and can be fetched with the drmaa2_lasterror_text(3) function.

61.4 SEE ALSO

drmaa2_lasterror(3), drmaa2_lasterror_text(3), drmaa2_j_suspend(3), drmaa2_j_resume(3), drmaa2_j_hold(3), drmaa2_j_release(3), drmaa2_j_terminate(3), drmaa2_j_get_id(3), drmaa2_j_get_session_name(3), drmaa2_j_get_jt(3), drmaa2_j_get_state(3), drmaa2_j_get_info(3), drmaa2_j_wait_started(3), drmaa2_j_wait_terminated(3), drmaa2_jsession_run_job(3), drmaa2_j_free(3)

62 drmaa2_lasterror

62.1 SYNOPSIS

```
#include "drmaa2.h"

typedef enum drmaa2_error {
   DRMAA2_SUCCESS                =  0,
   DRMAA2_DENIED_BY_DRMS         =  1,
   DRMAA2_DRM_COMMUNICATION      =  2,
   DRMAA2_TRY_LATER              =  3,
   DRMAA2_SESSION_MANAGEMENT     =  4,
   DRMAA2_TIMEOUT                =  5,
   DRMAA2_INTERNAL               =  6,
   DRMAA2_INVALID_ARGUMENT       =  7,
   DRMAA2_INVALID_SESSION        =  8,
```

```
   DRMAA2_INVALID_STATE           =  9,
   DRMAA2_OUT_OF_RESOURCE         = 10,
   DRMAA2_UNSUPPORTED_ATTRIBUTE   = 11,
   DRMAA2_UNSUPPORTED_OPERATION   = 12,
   DRMAA2_IMPLEMENTATION_SPECIFIC = 13,
   DRMAA2_LASTERROR               = 14
} drmaa2_error;

drmaa2_error drmaa2_lasterror(void)
```

62.2 DESCRIPTION

Returns the last DRMAA2 related error happend within the thread the function has called. Internally thread local storage is used by DRMAA2 functions for storing errors. This function is usually called when a previous DRMAA2 function call failed resulting in returning a NULL value. A detailed description of the error can be get by using the drmaa2_lasterror_text(3) call.

62.3 RETURN VALUES

Returns a `drmaa2_error` value of the last error occurred. If no error happend yet `DRMAA2_SUCCESS` is returned.

62.4 EXAMPLE

```
drmaa2_rsession rs = drmaa2_open_rsession(NULL);

if (rs == NULL) {
   if (drmaa2_listerror() == DRMAA2_UNSUPPORTED_OPERATION) {
      printf("Don't use reservations sessions again, they are not supported.\n");
   } else {
      /* handle real error ... */
   }
} else {
   /* Do something with the reservation session ... */
}
```

62.5 SEE ALSO

drmaa2_lasterror_text(3)

63 drmaa2_lasterror_text

63.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string drmaa2_lasterror_text(void)
```

63.2 DESCRIPTION

Returns the last DRMAA2 related error happend within the thread the function has called. Internally thread local storage is used by DRMAA2 functions for storing errors. This function is usually called when a previous DRMAA2 function call failed resulting in returning a NULL value. A numerical description of the error can be get by using the drmaa2_lasterror(3) call.

63.3 RETURN VALUES

Returns a copy of an unspecified string value describing the last error occurred. If no error happend yet NULL is returned. The `drmaa2_string` must be freed from the calling function.

63.4 EXAMPLE

```
drmaa2_rsession rs = drmaa2_open_rsession(NULL);

if (rs == NULL) {
   if (drmaa2_listerror() == DRMAA2_UNSUPPORTED_OPERATION) {
      printf("Don't use reservations sessions again, they are not supported.\n");
   } else {
      /* handle real error ... */
      drmaa2_error error = drmaa2_lasterror_text();
      printf("Following error happend while opening a reservation session: %s\n", error);
      drmaa2_string_free(&error);
   }
} else {
   /* Do something with the reservation session ... */
}
```

63.5 SEE ALSO

drmaa2_lasterror(3)

64 drmaa2_list_add

64.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_list_add(const drmaa2_list list, const void* element)
```

64.2 DESCRIPTION

Appends a new element in the given list. The element itself must be from the same type as the list. When the list is freed with drmaa2_list_free(3) then the callback function specified during list creation time is called for the list elements.

64.3 RETURN VALUES

Returns `DRMAA2_SUCCESS` in case of success or an `drmaa2_error` value in case of an error. The error condition can be read out by drmaa2_lasterror_text(3).

64.4 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST,
                                    (drmaa2_list_entryfree) drmaa2_string_free);

if (DRMAA2_SUCCESS != drmaa2_list_add(strings, strdup("string one"))) {
   drmaa2_error error = drmaa2_lasterror_text();
   printf("Unexpected error happend: %s\n", (char *) error);
   drmaa2_string_free(&error);
}

drmaa2_list_add(strings, strdup("string two"));

const drmaa2_string string = (const drmaa2_string) drmaa2_list_get(strings, 0);

printf("First element of the list %s\n", (const char *) string);

drmaa2_list_free(&strings);
```

64.5 SEE ALSO

drmaa2_list_create(3), drmaa2_list_free(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_has(3)

65 drmaa2_list_create

65.1 SYNOPSIS

```
#include "drmaa2.h"

typedef enum drmaa2_listtype {
   DRMAA2_STRINGLIST,
   DRMAA2_JOBLIST,
   DRMAA2_QUEUEINFOLIST,
   DRMAA2_MACHINEINFOLIST,
   DRMAA2_SLOTINFOLIST,
   DRMAA2_RESERVATIONLIST
} drmaa2_listtype;
typedef void (*drmaa2_list_entryfree)(void **value);

drmaa2_list drmaa2_list_create(const drmaa2_listtype type,
                               const drmaa2_list_entryfree callback);
```

65.2 DESCRIPTION

Allocates a new drmaa2_list(3) of a specific type initialized with a specific callback function (or NULL). The callback acts as destroy function and is called when an element is deleted with drmaa2_list_del(3). When the list is freed with drmaa2_list_free(3) and a callback function was given then for each element in the list this function is called. When lists are returned as copies by DRMAA2 functions then appropriate destroy functions are set.

The drmaa2_listtype can be any of:

- `DRMAA2_STRINGLIST`: A list containing drmaa2_string(3) values (which are in C defined as char *). The appropiate destroy function is drmaa2_string_free(3).
- `DRMAA2_JOBLIST`: A list containing drmaa2_j(3) (job) objects. The appropriate destroy function is drmaa2_j_free(3).
- `DRMAA2_QUEUEINFOLIST`: A list containing drmaa2_queueinfo(3) objects. The appropriate destroy function is drmaa2_queueinfo_free(3).
- `DRMAA2_MACHINEINFOLIST`: A list containing drmaa2_machineinfo(3) objects. The appropriate destroy function is drmaa2_machineinfo_free(3).
- `DRMAA2_SLOTINFOLIST`: A list containing drmaa2_slotinfo(3) objects. The appropriate destroy function is drmaa2_slotinfo_free(3). The list is used only in a reservation session and currently not supported by Altair Grid Engine.
- `DRMAA2_RESERVATIONLIST`: A list containing drmaa2_reservation(3) objects. The appropriate destroy function is drmaa2_reservation_free(3). Currently not supported by Altair Grid Engine.

65.3 RETURN VALUES

Upon successful completion drmaa2_list_create(3) returns a newly allocated `drmaa2_list`. The depending on the given type the returned list can be either a `drmaa2_string_list`, `drmaa2_j_list`, `drmaa2_queueinfo_list`, `drmaa2_machineinfo_list`, `drmaa2_slotinfo_list`, or a `drmaa2_r_list`.

65.4 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST, (drmaa2_list_entryfree) drmaa2

drmaa2_list_add(strings, strdup("string one"));
drmaa2_list_add(strings, strdup("string two"));
drmaa2_list_del(strings, "string one"));

printf("Size of list %ld", drmaa2_list_size(strings));

drmaa2_list_free(&strings);
```

65.5 SEE ALSO

drmaa2_list_free(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_get(3), drmaa2_list_has(3)

66 drmaa2_list_del

66.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_list_del(const drmaa2_list list, const long pos)
```

66.2 DESCRIPTION

Deletes a new element in the given list. When the list was created with a callback function, the callback is called for the element before it is removed from the list.

66.3 RETURN VALUES

Returns `DRMAA2_SUCCESS` in case of success or an `drmaa2_error` value in case of an error. The error condition can be read out by drmaa2_lasterror_text(3).

66.4 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST,
                                    (drmaa2_list_entryfree) drmaa2_string_free);

drmaa2_list_add(strings, strdup("string one"));
drmaa2_list_add(strings, strdup("string two"));
drmaa2_list_del(strings, 0);

const drmaa2_string string = (const drmaa2_string) drmaa2_list_get(strings, 0);

printf("First element of the list %s\n", (const char *) string);

drmaa2_list_free(&strings);
```

66.5 SEE ALSO

drmaa2_list_create(3), drmaa2_list_free(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_has(3), drmaa2_lasterror(3), drmaa2_lasterror_text(3)

67 drmaa2_list_free

67.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_list_free(drmaa2_list * l);
```

67.2 DESCRIPTION

Frees the memory of the given `drmaa2_list` (independent of its type) and sets it to NULL. If the list was created by using a callback function (see drmaa2_list_create(3)) then the callback function is called for each element in the list. Usually the callback function frees an element. Lists returned by function of the Altair Grid Engine DRMAA2 C implementation are initialized with appropriate callback functions so that the list and its entries are completely freed.

67.3 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST,
                                                (drmaa2_list_entryfree) drmaa2_string_free);

drmaa2_list_add(strings, strdup("string one"));
drmaa2_list_add(strings, strdup("string two"));
drmaa2_list_del(strings, "string one"));
```

```
printf("Size of list %ld", drmaa2_list_size(strings));

drmaa2_list_free(&strings);
```

67.4 SEE ALSO

drmaa2_list_create(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_get(3), drmaa2_list_has(3)

68 drmaa2_list_get

68.1 SYNOPSIS

```
#include "drmaa2.h"

const void * drmaa2_list_get(const drmaa2_list list, const long pos);
```

68.2 DESCRIPTION

Returns a pointer (not a copy) to an element of the given list at the given position. The function is independent of the list type. The returned element must be casted into the specific element type.

68.3 RETURN VALUES

Returns NULL in case of an error or when there is no element on the given position. The error condition can be read out with drmaa2_lasterror(3) and drmaa2_lasterror_text(3). In case of success a pointer to the element is returned.

68.4 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST,
                                  (drmaa2_list_entryfree) drmaa2_string_free);

drmaa2_list_add(strings, strdup("string one"));
drmaa2_list_add(strings, strdup("string two"));

const drmaa2_string string = (const drmaa2_string) drmaa2_list_get(strings, 0);

printf("First element of the list %s\n", (const char *) string);

drmaa2_list_free(&strings);
```

68.5 SEE ALSO

drmaa2_list_create(3), drmaa2_list_free(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_has(3)

69 drmaa2_list_size

69.1 SYNOPSIS

```
#include "drmaa2.h"

long drmaa2_list_size(const drmaa2_list list)
```

69.2 DESCRIPTION

Determines the amount of entries in the given list and returns the value.

69.3 RETURN VALUES

Returns the amount of entries in the given list. The value is 0 or greater. In case of any errors (e.g. list is NULL), 0 is returned.

69.4 EXAMPLE

```
drmaa2_string_list strings = drmaa2_create_list(DRMAA2_STRINGLIST,
                                    (drmaa2_list_entryfree) drmaa2_string_free);

drmaa2_list_add(strings, strdup("string one"));
drmaa2_list_add(strings, strdup("string two"));

printf("List size: %ld\n", drmaa2_list_size(strings));

drmaa2_list_free(&strings);
```

69.5 SEE ALSO

drmaa2_list_create(3), drmaa2_list_free(3), drmaa2_list_add(3), drmaa2_list_del(3), drmaa2_list_size(3), drmaa2_list_has(3), drmaa2_lasterror(3), drmaa2_lasterror_text(3)

70 drmaa2_machineinfo_free

70.1 SYNOPSIS

```
#include "drmaa2.h"

typedef struct {
   drmaa2_string   name;
   drmaa2_bool     available;
   long long       sockets;
   long long       coresPerSocket;
   long long       threadsPerCore;
   float           load;
   long long       physMemory;
   long long       virtMemory;
   drmaa2_cpu      machineArch;
   drmaa2_version  machineOSVersion;
   drmaa2_os       machineOS;
} drmaa2_machineinfo_s;
typedef drmaa2_machineinfo_s *drmaa2_machineinfo;

void drmaa2_machineinfo_free(drmaa2_machineinfo *machineinfo);
```

70.2 DESCRIPTION

Releases the memory of an allocated `drmaa2_machineinfo` object and sets it to NULL.

70.3 SEE ALSO

drmaa2_mession_get_all_machines(3), drmaa2_machineinfo_impl_spec(3)

71 drmaa2_machineinfo_impl_spec

71.1 SYNOPSIS

```
#include "drmaa2.h"
```

drmaa2_string_list drmaa2_machineinfo_impl_spec(void);

71.2 DESCRIPTION

Returns all Altair Grid Engine specific machine info (`drmaa2_machineinfo`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

71.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

71.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_machineinfo_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

71.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

72 drmaa2_msession_free

72.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_msession_free(drmaa2_msession *monitoring_session);
```

72.2 DESCRIPTION

Frees the memory of a allocated drmaa2_mession object. A drmaa2_mession object is returned by drmaa2_open_mession(3). After freeing the monitoring session is set to NULL.

72.3 EXAMPLE

```
drmaa2_msession ms = drmaa2_open_msession(NULL);

if (ms != NULL) {
```

```
   ...
   if (DRMAA2_SUCCESS != drmaa2_close_msession(ms)) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during closing the monitoring session: %s\n", error);
      drmaa2_string_free(&error);
   }
   drmaa2_msession_free(&ms);
}
```

72.4 SEE ALSO

drmaa2_open_msession(3)

73 drmaa2_msession_get_all_jobs

73.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_j_list drmaa2_msession_get_all_jobs(const drmaa2_msession monitoring_session,
                const drmaa2_jinfo filter);
```

73.2 DESCRIPTION

Returns a list of all jobs of the user currently stored in Altair Grid Engine. The job list contains also jobs not submitted in Altair Grid Engine. Calling this function does not necessarly trigger communication with Altair Grid Engine master since all job related information is sent from Altair Grid Engine master process to the DRMAA2 client as soon as there are job changes in the system. Hence it is safe to call drmaa2_msession_get_all_jobs(3) in short intervalls. The monitoring session given as argument must be valid, i.e. it must be opened with drmaa2_open_mession(3) before.

The second argument defines a filter for the jobs to be returned. If NULL is used as filter, all available jobs are returned otherwise only jobs which match the given filter. The filter is based on a drmaa2_jinfo object. After allocation with drmaa2_jinfo_create(3) all fields are UNSET, i.e. no filtering is applied. When setting fields with different values than UNSET the semantic for filtering is defined as follows:

- `jobId`: The job id to filter for or DRMAA2_UNSET_STRING when all job ids should match.
- `exitStatus` Filter allows only finished jobs with the given exit status.
- `terminatingSignal` Filter allows only finsihed jobs with the given termination signal.
- `annotation` Ignored for filtering.

- `jobState` Filter allows only jobs with the given job state.
- `jobSubState` Currently unsupported by Altair Grid Engine.
- `allocatedMachines` Filter allows all jobs which have at least one slot allocated on one of the given machines.
- `submissionMachine` Filter allows all jobs which are submitted on the given machine.
- `jobOwner` Filter allows all jobs which are owned (i.e. submitted) by the given user.
- `slots` Filter allows all jobs which have this amount of slots granted.
- `queueName` Filter allows only jobs which have at least one part running in the given queue instance.
- `wallclockTime` Filter allows only jobs which are running the given amount of time (in seconds) or longer.
- `cpuTime` Filter allows only jobs which had consumed this amount of cpu time ore more.
- `submissionTime` Filter allows only jobs which are submitted at or after the given time.
- `dispatchTime` Filter allows only jobs which are started at or after the given time on the host.
- `finishTime` Filter allows only jobs which are finshed at or after the given time.

73.3 RETURN VALUES

Returns a newly allocated list of jobs in a `drmaa2_j_list` structure or NULL in case of an error. The job list was initialized with an appropriate callback function so that drmaa2_list_free(3) frees the complete list with all job objects inside. In case of an error the error number and error message can be fetched with drmaa2_lasterror(3) and drmaa2_lasterror_text(3).

73.4 EXAMPLE

```
drmaa2_jinfo filter = drmaa2_jinfo_create();
drmaa2_msession monitoring_session = drmaa2_open_msession(NULL);

/* filter only for jobs which consume one slot */
filter->slots = 1;

if (ms != NULL) {
   drmaa2_j_list job_list = drmaa2_msession_get_all_jobs(monitoring_session, filter);

   if (job_list == NULL) {
      /* handle error */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error fetching the job list from the monitoring session: %s\n",
```

```
                        error);
      drmaa2_string_free(&error);
   } else {
      int i;
      size_t size = drmaa2_list_size(job_list);

      fprintf(stdout, "There are %lld jobs in the system with 1 slot:\n", size);
      for (i = 0; i < size; i++) {
          drmaa2_j job = (drmaa2_j) drmaa2_list_get(job_list, i);
          if (drmaa2_j_get_id(job) != NULL) {
             fprintf(stdout, "Job with id %s.\n", (char *) drmaa2_j_get_id(job));
          } else {
             fprintf(stdout, "Job has unknown id.\n");
          }
      }
      drmaa2_list_free(&job_list);
   }

   ...
}

drmaa2_jinfo_free(&filter);
```

73.5 SEE ALSO

drmaa2_open_msession(3), drmaa2_close_msession(3), drmaa2_msession_free(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_queues(3), drmaa2_msession_get_all_machines(3), drmaa2_list_free(3), drmaa2_jinfo_create(3), drmaa2_jinfo_free(3), drmaa2_j_get_id(3)

74 drmaa2_msession_get_all_machines

74.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_machineinfo_list drmaa2_msession_get_all_machines(const drmaa2_msession
  monitoring_session, const drmaa2_string_list filter);
```

74.2 DESCRIPTION

Returns a list of all execution hosts currently managed by Altair Grid Engine in a `drmaa2_machineinfo_list`.

The second argument defines a filter for the hosts to be returned. Only hosts with names given by the filter are returned if filter is != NULL. If filter is NULL all available hosts are returned.

The drmaa2_machineinfo_list consists of drmaa2_machineinfo elements which are pointers to the drmaa2_machineinfo_s struct. The struct offers at least following elements:

```
typedef struct {
   drmaa2_string   name;
   drmaa2_bool     available;
   long long       sockets;
   long long       coresPerSocket;
   long long       threadsPerCore;
   float           load;
   long long       physMemory;
   long long       virtMemory;
   drmaa2_cpu      machineArch;
   drmaa2_version  machineOSVersion;
   drmaa2_os       machineOS;
} drmaa2_machineinfo_s;

typedef drmaa2_machineinfo_s *drmaa2_machineinfo;
```

All values, but especially allocated values like drmaa2_string name and drmaa2_version, which can be NULL needs to be tested if they are not UNSET or != NULL before they are used. The UNSET defines can be found in the drmaa2.h file.

The availability of additional values can be queried with the drmaa2_machineinfo_impl_spec(3) function.

74.3 RETURN VALUES

Returns a newly allocated list of hosts in a drmaa2_machineinfo_list structure or NULL in case of an error. The machine list was initialized with an appropriate callback function so that drmaa2_list_free(&result) frees the complete list with all drmaa2_machineinfo objects inside. In case of an error the error number and error message can be fetched with drmaa2_lasterror(3) and drmaa2_lasterror_text().

74.4 EXAMPLE

```
drmaa2_msession monitoring_session = drmaa2_open_msession(NULL);

if (ms != NULL) {
   drmaa2_machineinfo_list mi_list = drmaa2_msession_get_all_machines(monitoring_session,
                                                                      NULL);

   if (mi_list == NULL) {
      /* handle error */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during fetching the machineinfo list from the monitoring
                      session: %s\n", error);
```

```
      drmaa2_string_free(&error);
   } else {
      int i;
      size_t size = drmaa2_list_size(mi_list);

      fprintf(stdout, "There are %lld machines in the system:\n", size);
      for (i = 0; i < size; i++) {
          drmaa2_machineinfo mi = (drmaa2_machineinfo) drmaa2_list_get(mi_list, i);
          fprintf(stdout, "machine name: %s\n", (mi->name==DRMAA2_UNSET_STRING)?
                  "UNSET":mi->name);
          fprintf(stdout, "amount of sockets: %lld\n", mi->sockets);
          fprintf(stdout, "amount of cores per socket: %lld\n", mi->coresPerSocket);
          fprintf(stdout, "amount of threads per core: %lld\n", mi->threadsPerCore);
          fprintf(stdout, "1 min. avg. load on machine: %f\n", mi->load);
          fprintf(stdout, "physical memory in kilobyte: %lld\n", mi->physMemory);
          fprintf(stdout, "virtual memory in kilobyte: %lld\n", mi->virtMemory);
          fprintf(stdout, "OS: %s\n", drmaa2_os_to_string(mi->machineOS));
          fprintf(stdout, "CPU architecture as enum: %d\n", mi->machineArch);
      }
      drmaa2_list_free(&mi_list);
   }

   ...
   drmaa2_close_msession(monitoring_session);
   drmaa2_msession_free(&monitoring_session);
}
...
```

Example for the convertion from a drmaa2_os object into a string:

```
static char* drmaa2_os_to_string(const drmaa2_os os)
{
     switch (os) {
        case DRMAA2_OTHER_OS:
           return "DRMAA2_OTHER_OS";
        case DRMAA2_AIX:
           return "DRMAA2_AIX";
        case DRMAA2_BSD:
           return "DRMAA2_BSD";
        case DRMAA2_LINUX:
           return "DRMAA2_LINUX";
        case DRMAA2_HPUX:
           return "DRMAA2_HPUX";
        case DRMAA2_IRIX:
           return "DRMAA2_IRIX";
        case DRMAA2_MACOS:
           return "DRMAA2_MACOS";
        case DRMAA2_SUNOS:
           return "DRMAA2_SUNOS";
```

```
         case DRMAA2_TRU64:
            return "DRMAA2_TRU64";
         case DRMAA2_UNIXWARE:
            return "DRMAA2_UNIXWARE";
         case DRMAA2_WIN:
            return "DRMAA2_WIN";
         case DRMAA2_WINNT:
            return "DRMAA2_WINNT";
         default:
            return "UNKNOWN";
      }
      return "UNKNOWN";
}
```

74.5 SEE ALSO

drmaa2_open_msession(3), drmaa2_close_msession(3), drmaa2_msession_free(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_queues(3), drmaa2_list_free(3), drmaa2_machineinfo_impl_spec(3)

75 drmaa2_msession_get_all_queues

75.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_queueinfo_list drmaa2_msession_get_all_queues(const drmaa2_msession
  monitoring_session, const drmaa2_string_list filter);
```

75.2 DESCRIPTION

Returns a list of all queues currently stored in Altair Grid Engine. The queue list might also contain queues which are not available for the user.

The second argument defines a filter for the queues to be returned. Only queues with names given by the filter are returned if filter is != NULL. If filter is NULL all queues are returned.

75.3 RETURN VALUES

Returns a newly allocated list of queues in a `drmaa2_queueinfo_list structure` or NULL in case of an error. The queue list was initialized with an appropriate callback function so that `drmaa2_list_free(&result)` frees the complete list with all `drmaa2_queueinfo` objects inside. In case of an error the error number and error message can be fetched with `drmaa2\_lasterror(3)` and `drmaa2_lasterror_text()`.

75.4 EXAMPLE

```
drmaa2_msession monitoring_session = drmaa2_open_msession(NULL);

if (ms != NULL) {
   drmaa2_queueinfo_list qi_list = drmaa2_msession_get_all_queues(monitoring_session, NULL);

   if (qi_list == NULL) {
      /* handle error */
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during fetching the queueinfo list from the
                       monitoring session: %s\n", error);
      drmaa2_string_free(&error);
   } else {
      int i;
      size_t size = drmaa2_list_size(qi_list);

      fprintf(stdout, "There are %lld queues in the system:\n", size);
      for (i = 0; i < size; i++) {
          drmaa2_queueinfo qi = (drmaa2_queueinfo) drmaa2_list_get(qi_list, i);
          fprintf(stdout, "Queue name is %s.\n", (char *) qi->name);
      }
      drmaa2_list_free(&qi_list);
   }

   ...
   drmaa2_close_msession(monitoring_session);
   drmaa2_msession_free(&monitoring_session);
}
```

75.5 SEE ALSO

drmaa2_open_msession(3), drmaa2_close_msession(3), drmaa2_msession_free(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_machines(3), drmaa2_list_free(3)

76 drmaa2_msession_get_all_reservations

76.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_r_list drmaa2_msession_get_all_reservations(const drmaa2_msession monitoring_session);
```

76.2 DESCRIPTION

Optional by the DRMAA2 standard. Currently not implemented.

76.3 SEE ALSO

drmaa2_open_msession(3), drmaa2_close_msession(3), drmaa2_msession_free(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_queues(3), drmaa2_msession_get_all_machines(3)

77 drmaa2_notification_free

77.1 SYNOPSIS

```
#include "drmaa2.h"

typedef struct {
   drmaa2_event   event;
   drmaa2_string  jobId;
   drmaa2_string  sessionName;
   drmaa2_jstate  jobState;
} drmaa2_notification_s;
typedef drmaa2_notification_s * drmaa2_notification;
typedef void (*drmaa2_callback)(drmaa2_notification *notification);

void drmaa2_notification_free(drmaa2_notification *notification)
```

77.2 DESCRIPTION

Frees the memory of a `drmaa2_notification` object and sets it to NULL.

77.3 SEE ALSO

drmaa2_notification_impl_spec(3)

78 drmaa2_notification_impl_spec

78.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_notification_impl_spec(void);
```

78.2 DESCRIPTION

Returns all Altair Grid Engine specific notification (`drmaa2_notification`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

78.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

78.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_notification_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

78.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

79 drmaa2_open_jsession

79.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_jsession drmaa2_open_jsession(const char* session_name);
```

79.2 DESCRIPTION

Opens a DRMAA2 job session with a specific name. The job session must exist on the Altair Grid Engine master process. A job session can be created by the drmaa2_create_jsession(3) function or by qconf(3) commands on command line. The session name argument must not be NULL. In case of success a connection is established to the Altair Grid Engine master process. If there is already a connection (because the process has already other job sessions or a monitoring session open) the existing connection to the master process is shared, but additionally events relating to jobs in this job session are subscribed from the master process. The session needs to be closed and freed when it is not going to be used anymore in order

to reduce the network traffic or close the connection to the master process completely (if the sesssion is the last open session).

A DRMAA2 job session is used for submitting jobs, monitoring and controling jobs. Operations on jobs can only be performed when the job session they belong to is open.

79.3 RETURN VALUES

Returns a newly allocated `drmaa2_jsession` object in case of success. This object is going to be used in further job session related function calls. The session object needs to be freed with drmaa2_jsession_free(3) after closing a job session with drmaa2_close_jsession(3). Finally the persistent job session can be removed from the master process by drmaa2_destroy_jsession(3).

In case of an error NULL is returned. The error number and error text is set in the current thread context.

79.4 EXAMPLE

```
/* "unique_jsession" must exist on AGE master process */

drmaa2_jsession js = drmaa2_open_jsession("unique_jsession");

if (js != NULL) {
   /* do something with the job session */
   drmaa2_j_list jobs = drmaa2_jsession_get_jobs(js, NULL);
   /* process jobs and free list ... */
   ...

   if (DRMAA2_SUCCESS != drmaa2_close_jsession(ms)) {
      drmaa2_string error = drmaa2_lasterror_text();
      fprintf(stderr, "Error during closing the job session: %s\n", error);
      drmaa2_string_free(&error);
   }
   drmaa2_jsession_free(&ms);
}
```

79.5 SEE ALSO

drmaa2_close_jsession(3), drmaa2_destroy_jsession(3), drmaa2_jsession_free(3), drmaa2_get_jsession_names(3), drmaa2_jsession_all_jobs(3), drmaa_jsession_get_job_categories(3), drmaa2_jsession_run_job(3), drmaa2_jsession_run_bulk_jobs(3), drmaa2_jsession_wait_any_started(3), drmaa2_jsession_wait_any_terminated(3), drmaa2_jsession_get_contact(3), drmaa2_jsession_get_session_name(3), drmaa2_jsession_get_job_array(3)

80 drmaa2_open_msession

80.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_msession drmaa2_open_msession(const char* monitoring_session_name);
```

80.2 DESCRIPTION

If there is no active job session, the function establishes a new connection to the Altair Grid Engine master process and subscribes objects which can be requested by DRMAA2 monitoring session calls. If a job session is already open, the connection is shared by the job and the monitoring session. It is only allowed to have one monitoring session open at a time. The given monitoring session name has no effect, it can be any string of NULL and is not further considered by the DRMAA2 Altair Grid Engine implementation.

In case of success the function returns a valid drmaa2_msession object, which can be used for fetching queue, machine or job statuses. When the monitoring session is not used anymore it must be close by drmaa2_close_msession(3). Closing the monitoring session disenganges the connection from the Altair Grid Engine master process. If there is still one or more job session open, the connection to the master process remains active, but closing the monitoring session has the effect that less information is transferred from the Altair Grid Engine master process to the DRMAA2 application and the memory footprint of the application is reduced.

The connection to the qmaster is event driven hence with an open monitoring session the DRMAA2 application gets an continues data stream of changed lists and objects even no information (e.g. by calling drmaa2_mession_get_all_queues(3)) is requested by the application. Hence it is recommended to open a monitoring session only if the information is needed and close the montioring session as soon as possible in the application. Subsequent calls of drmaa2_msession_get_*() do not result in additional communication to the qmaster since all information is already locally available. An open monitoring session can increase the memory footprint of the application depending on the amount of information is available on qmaster (amount of jobs, amount of queues, amount of machines).

80.3 RETURN VALUES

This function returns a newly allocated drmaa2_msession object or NULL in case of an error. The error number and text can be fetched by subsequent calls of drmaa2_lasterror(3) and drmaa2_lasterror_text(3) within the same thread where drmaa2_open_msession(3) was called.

80.4 EXAMPLE

```
drmaa2_msession ms = drmaa2_open_msession(NULL);
```

```
if (ms != NULL) {
   drmaa2_j_list job_list = drmaa2_msession_get_all_jobs(ms, NULL);
   ...
   drmaa2_close_msession(ms);
   drmaa2_msession_free(&ms);
}
```

80.5 SEE ALSO

drmaa2_msession_free(3), drmaa2_close_msession(3), drmaa2_msession_get_all_jobs(3), drmaa2_msession_get_all_queues(3), drmaa2_msession_get_all_machines(3)

81 drmaa2_open_rsession

81.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_rsession drmaa2_open_rsession(const char* session_name);
```

81.2 DESCRIPTION

Reservation sessions are currently not supported in the Altair Grid Engine DRMAA2 implementation.

81.3 RETURN VALUES

Returns NULL and sets the failure code `DRMAA2_UNSUPPORTED_OPERATION`.

81.4 SEE ALSO

drmaa2_close_rsession(3), drmaa2_create_rsession(3), drmaa2_destroy_rsession(3), drmaa2_get_rsession_names(3), drmaa2_msession_get_all_reservations(3)

82 drmaa2_queueinfo_free

82.1 SYNOPSIS

```
#include "drmaa2.h"

typedef struct {
   drmaa2_string               name;
```

```
} drmaa2_queueinfo_s;
typedef drmaa2_queueinfo_s * drmaa2_queueinfo;

void drmaa2_queueinfo_free   (drmaa2_queueinfo *qi);
```

82.2 DESCRIPTION

Frees the memory of a `drmaa2_queueinfo` object and sets it to NULL.

82.3 SEE ALSO

drmaa2_queueinfo_impl_spec(3)

83 drmaa2_queueinfo_impl_spec

83.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_queueinfo_impl_spec(void);
```

83.2 DESCRIPTION

Returns all Altair Grid Engine specific queue info (`drmaa2_queueinfo`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

83.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

83.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_queueinfo_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
```

```
   drmaa2_list_free(&uge_attributes);
}
```

83.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

84 drmaa2_register_event_notification

84.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_register_event_notification(const drmaa2_callback callback)
```

84.2 DESCRIPTION

Events notifications are currently unsuppported in Altair Grid Engine.

84.3 RETURN VALUES

The function returns `DRMAA2_UNSUPPORTED_OPERATION`.

84.4 SEE ALSO

drmaa2_supports(3)

85 drmaa2_r_free

85.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_r_free(drmaa2_r *reservation);
```

85.2 DESCRIPTION

Frees a previously allocated reservation object and sets it to NULL.

85.3 SEE ALSO

drmaa2_rsession_get_reservation(3), drmaa2_rsession_request_reservation(3)

86 drmaa2_rinfo_free

86.1 SYNOPSIS

```
#include "drmaa2.h"
```

void drmaa2_rinfo_free(drmaa2_rinfo * rinfo);

86.2 DESCRIPTION

Frees the memory of a `drmaa2_rinfo` object and sets it to NULL. A `drmaa2_rinfo` object is part of the optional reservation session and not supported in Altair Grid Engine.

86.3 SEE ALSO

drmaa2_slotinfo_free(3)

87 drmaa2_rinfo_impl_spec

87.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_rinfo_impl_spec(void);
```

87.2 DESCRIPTION

Returns all Altair Grid Engine specific reservation info (`drmaa2_rinfo`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

87.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

87.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_rinfo_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

87.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

88 drmaa2_rsession_free

88.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_rsession_free(drmaa2_rsession *reservationsession);
```

88.2 DESCRIPTION

Frees a previously allocated rsession object and sets it to NULL.

88.3 SEE ALSO

drmaa2_create_rsession(3)

89 drmaa2_rtemplate_impl_spec

89.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_string_list drmaa2_rtemplate_impl_spec(void);
```

89.2 DESCRIPTION

Returns all Altair Grid Engine specific reservation template (`drmaa2_rtemplate`) attributes which are not defined by the DRMAA2 standard but available in the Altair Grid Engine DRMAA2 implemenation. The list must be freed by the caller.

89.3 RETURN VALUES

Returns a newly allocated `drmaa2_string_list` or NULL in case no Altair Grid Engine specific attributes are available.

89.4 EXAMPLE

```
drmaa2_string_list uge_attributes = drmaa2_rtemplate_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

89.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

90 drmaa2_set_instance_value

90.1 SYNOPSIS

```
#include "drmaa2.h"

drmaa2_error drmaa2_set_instance_value(void *instance, const char *name, const char *value)
```

90.2 DESCRIPTION

Sets a new value for the attribute with the given name in a DRMAA2 object (given by instance). A valid instance has one of the following types:

- jtemplate
- jinfo
- rtemplate
- rinfo
- queueinfo
- machineinfo
- notification

90.3 RETURN VALUES

Returns `DRMAA2_SUCCESS` if the attribute name could be found and the value was set successfully in the DRMAA2 object.

90.4 EXAMPLE

```
drmaa2_jtemplate jt = drmaa2_jtemplate_create();
drmaa2_string_list uge_attributes = drmaa2_jtemplate_impl_spec();

if (uge_attributes != NULL) {
   long size, i;
   size = drmaa2_list_size(uge_attributes);
   for (i = 0; i < size; i++) {
      drmaa2_string attr = drmaa2_list_get(uge_attributes, i);
      if (strcmp(attr, "JTEMPLATE_SPECIAL_ATTRIBUTE") == 0) {
         drmaa2_string val;
         if (DRMAA2_SUCCESS == (drmaa2_set_instance_value(jt, attr, strdup("My Value"))) {
            /* read "My Value" out ... */
            val = drmaa2_get_instance(jt, attr);
            /* val is "My Value" */
            drmaa2_string_free(&val);
         }
      }
      printf("Additionally supported attribute: %s\n", attr);
   }
   drmaa2_list_free(&uge_attributes);
}
```

90.5 SEE ALSO

drmaa2_jtemplate_impl_spec(3), drmaa2_jinfo_impl_spec(3), drmaa2_rtemplate_impl_spec(3), drmaa2_rinfo_impl_spec(3), drmaa2_queueinfo_impl_spec(3), drmaa2_machineinfo_impl_spec(3), drmaa2_notification_impl_spec(3), drmaa2_get_instance_value(3), drmaa2_describe_attribute(3), drmaa2_set_instance_value(3)

91 drmaa2_slotinfo_free

91.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_slotinfo_free(drmaa2_slotinfo * slotinfo);
```

91.2 DESCRIPTION

Frees the memory of a `drmaa2_slotinfo` object and sets it to NULL. A `drmaa2_slotinfo` object is part of the optional reservation session and not supported in Altair Grid Engine.

91.3 SEE ALSO

drmaa2_rinfo_free(3)

92 drmaa2_string_free

92.1 SYNOPSIS

```
#include "drmaa2.h"

void drmaa2_string_free(drmaa2_string *string)
```

92.2 DESCRIPTION

Releases the memory of an allocated `drmaa2_string`, which in the Altair Grid Engine implementation is a `char *`, and sets the variable to NULL.

92.3 SEE ALSO

drmaa2_get_drms_name(3), drmaa2_lasterror_text(3)

93 drmaa2_supports

93.1 SYNOPSIS

```
#include "drmaa2.h"

typedef enum drmaa2_capability {
```

```
   DRMAA2_ADVANCE_RESERVATION         =  0,
   DRMAA2_RESERVE_SLOTS               =  1,
   DRMAA2_CALLBACK                    =  2,
   DRMAA2_BULK_JOBS_MAXPARALLEL       =  3,
   DRMAA2_JT_EMAIL                    =  4,
   DRMAA2_JT_STAGING                  =  5,
   DRMAA2_JT_DEADLINE                 =  6,
   DRMAA2_JT_MAXSLOTS                 =  7,
   DRMAA2_JT_ACCOUNTINGID             =  8,
   DRMAA2_RT_STARTNOW                 =  9,
   DRMAA2_RT_DURATION                 = 10,
   DRMAA2_RT_MACHINEOS                = 11,
   DRMAA2_RT_MACHINEARCH              = 12
} drmaa2_capability;

drmaa2_bool drmaa2_supports(const drmaa2_capability capability);
```

93.2 DESCRIPTION

Allows a DRMAA2 application to check during runtime whether a specific (non-mandatory) DRMAA2 functionality is supported by the Altair Grid Engine DRMAA2 implementation or not. Optionally implementable functionality is denoted `capability`. Following capabilities are defined by the DRMAA2 standard:

- `DRMAA2_ADVANCE_RESERVATION`: Describes whether the reservation session functionality is supported or not. In Altair Grid Engine reservation sessions are currently not supported.
- `DRMAA2_RESERVE_SLOTS`: The granularity level for reservation session (slots versus whole machines). In Altair Grid Engine reservation sessions are currently not supported.
- `DRMAA2_CALLBACK`: Describes if event notification via the `drmaa2_callback` is supported (drmaa2_register_event_notification(3)) or not.
- `DMRAA2_BULK_JOBS_MAXPARALLEL`: Describes if the max_parallel parameter for drmaa2_jsession_run_job(3) is supported or not. In Altair Grid Engine it is supported.
- `DRMAA2_JT_EMAIL`: Describes if the job template attributes emailOnStarted and emailOnTerminated are supported or not. In Altair Grid Engine sending emails on job start and job end is supported.
- `DRMAA2_JT_STAGING`: Describes if file staging with the job template dictionaries stageInFiles and stageOutFile are supported or not. In Altair Grid Engine file staging as part of the job submission process is not supported. File staging is usually performed by the job itself or can be optionally configured in the prolog and epilog scripts.
- `DRMAA2_JT_DEADLINE`: Describes if the deadlineTime attribute of the job template is supported. Altair Grid Engine has a different semantic as the DRMAA2 specified deadline time hence it is not supported.

- DRMAA2_JT_MAXSLOTS: Describes if the maxSlots job template attribute is supported.
- DRMAA2_JT_ACCOUNTINGID: Describes if the accountingId job template attribute is supported or not. In Altair Grid Engine it is equivalent to the -A qsub(3) parameter.
- DRMAA2_RT_STARTNOW: Reservation template attributes are not supported because DRMAA2_ADVANCE_RESERVATION is not supported.
- DRMAA2_RT_DURATION: Reservation template attributes are not supported because DRMAA2_ADVANCE_RESERVATION is not supported.
- DRMAA2_RT_MACHINEOS: Reservation template attributes are not supported because DRMAA2_ADVANCE_RESERVATION is not supported.
- DRMAA2_RT_MACHINEARCH: Reservation template attributes are not supported because DRMAA2_ADVANCE_RESERVATION is not supported.

93.3 RETURN VALUES

Returns DRMAA2_TRUE or DRMAA2_FALSE depending if the capability is supported or not. An application should use this information to create DRMAA2 compatible code which can run on systems with or without the specific functionality.

93.4 EXAMPLE

```
/* Create and open a new job session. */
drmaa2_jsession js = drmaa2_create_jsession("test_session", NULL);
drmaa2_j job = NULL;

if (js != NULL) {
   /* create a new job template. */
   drmaa2_jtemplate jt = drmaa2_jtemplate_create();

   /* add the job characteristics */
   jt->jobName = strdup("test_job");
   jt->remoteCommand = strdup("sleep");
   /* since no allocated strings are used we don't need to specify a callback */
   args = drmaa2_list_create(DRMAA2_STRINGLIST, NULL);
   drmaa2_list_add(args, "60");
   jt->args = args;

   /* only add the accounting string if the DRM supports it */
   if (drmaa2_supports(DRMAA2_JT_ACCOUNTINGID) == DRMAA2_TRUE) {
     jt->accountingId = strdup("ProjectX");
   }

   /* submit the jobs */
   job = drmaa2_jsession_run_job(js, jt);
}
```

```
drmaa2_j_free(&job);
drmaa2_jtemplate_free(&jt);

...
```

93.5 SEE ALSO

drmaa2_jtemplate_create(3), drmaa2_rtemplate_create(3), drmaa2_jtemplate_impl_spec(3)

94 drmaa2_version_free

94.1 SYNOPSIS

```
#include "drmaa2.h"

typedef struct {
   drmaa2_string major;
   drmaa2_string minor;
} drmaa2_version_s;
typedef drmaa2_version_s * drmaa2_version;

void drmaa2_version_free(drmaa2_version *version);
```

94.2 DESCRIPTION

Releases the memory of an allocated `drmaa2_version` object and sets it to NULL.

94.3 EXAMPLE

```
drmaa2_version v = drmaa2_get_drms_version();

if (v != NULL) {
   fprintf("Major version: %s", (char *) v->major);
   fprintf("Minor version: %s", (char *) v->minor);
   drmaa2_version_free(&v);
}
```

94.4 SEE ALSO

drmaa2_get_drms_version(3), drmaa2_mession_get_all_machines(3)