

GWD-R  
DRMAA-WG  
drmaa-wg@ogf.org

Peter Tröger, Hasso Plattner Institute<sup>1</sup>  
Roger Brobst, Cadence Design Systems  
Daniel Gruber, Univa  
Mariusz Mamoński, PSNC  
Andre Merzky, LSU  
September 2012

## **Distributed Resource Management Application API Version 2 (DRMAA) - C Language Binding**

### Status of This Document

Group Working Draft - Proposed Recommendation (GWD-R)

### Document Change History

<i>Date</i>	<i>Notes</i>
April 26th, 2012	Initial submission for public comment period
September 4th, 2012	Final version after public comment period

### Copyright Notice

Copyright © Open Grid Forum (2012-2012). Some Rights Reserved. Distribution is unlimited.

### Trademark

All company, product or service names referenced in this document are used for identification purposes only and may be trademarks of their respective owners.

### Abstract

This document describes the C language binding for the *Distributed Resource Management Application API Version 2 (DRMAA)*. The intended audience for this specification are DRMAA implementors.

---

<sup>1</sup>Corresponding author

## Notational Conventions

In this document, C language elements and definitions are represented in a **fixed-width** font.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are to be interpreted as described in RFC 2119 [1].

## Contents

1	Introduction . . . . .	4
2	General Design . . . . .	4
2.1	Error Handling . . . . .	5
2.2	Lists and Dictionaries . . . . .	6
3	Implementation-specific Extensions . . . . .	7
4	Complete Header File . . . . .	7
5	Security Considerations . . . . .	13
6	Contributors . . . . .	14
7	Intellectual Property Statement . . . . .	14
8	Disclaimer . . . . .	15
9	Full Copyright Notice . . . . .	15
10	References . . . . .	15

## 1 Introduction

The *Distributed Resource Management Application API Version 2 (DRMAA)* specification defines an interface for tightly coupled, but still portable access to the majority of DRM systems. The scope is limited to job submission, job control, reservation management, and retrieval of job and machine monitoring information.

The *DRMAA root specification* [2] describes the abstract API concepts and the behavioral rules of a compliant implementation, while this document standardizes the representation of API concepts in the C programming language.

## 2 General Design

The mapping of DRMAA IDL constructs to C follows a set of design principles. Implementation-specific extensions of the DRMAA C API SHOULD follow these conventions:

- Namespacing of the DRMAA API, as demanded by the root specification, is realized with the `drmaa2_` prefix for lower- and upper-case identifiers.
- In identifier naming, "job" is shortened as "j" and "reservation" is shortened as "r" for improved readability.
- The root specification demands a consistent parameter passing strategy for non-scalar values. All such values are passed as call-by-reference parameter in the C binding.
- Structs and enums are typedef'ed for better readability.
- Struct types have an `_s` suffix with their name. Structures with a non-standardized layout are defined as forward references for the DRMAA library implementation.
- Functions with IDL return type `void` have `drmaa2_error` as return type.
- The IDL `boolean` type maps to the `drmaa2_bool` type.
- The IDL `long` type maps to `long long` in C. One exception is the `exitStatus` variable, which is defined as `int` in order to provide a more natural mapping to existing operating system interfaces.
- The IDL `string` type is mapped in two different ways. Attributes and parameters with string values typically created by the implementation are mapped to the `drmaa2_string` type. The application frees such memory by calling the newly introduced function `drmaa2_string_free`. All other string parameters are mapped to the `const char *` type. Implementations MUST accept calls to `drmaa2_string_free` for all string pointers, regardless of their type.
- The language binding defines one `UNSET` macro per utilized C data type (`DRMAA2_UNSET_*`).
- All numerical types are signed, in order to support `-1` as numerical `UNSET` value.
- Application-created structs should be allocated by the additional support methods (such as `drmaa2_jinfo_create`) to realize the necessary initialization to `UNSET`.
- All structures have a specific support function for freeing them (`drmaa2_*_free`). Implementations SHOULD perform the freeing of struct members automatically if the struct itself is freed.
- Both `AbsoluteTime` and `TimeAmount` map directly to `time_t`. RFC 822 support as mandated by the root specification is given by the `%z` formatter for `sprintf`.

- Multiple output parameters are realized by declaring all but one of them as pointer variable. For this reason, the `substate` parameter in `drmaa2_j_get_state` SHALL be interpreted as pointer to a string variable created by the DRMAA library.
- The `const` declarator is used to mark parameters declared as `readonly` in the root specification.
- The two string list types in DRMAA, ordered and unordered, are mapped to one ordered list with the `DRMAA2_STRING_LIST` type.
- The `any` member for job sub-state information is defined as `drmaa2_string` to achieve application portability.

The following structures are only used in result values. For this reason, the according allocation functions are not part of the API:

- `drmaa2_string`
- `drmaa2_slotinfo`
- `drmaa2_rinfo`
- `drmaa2_notification`
- `drmaa2_queueinfo`
- `drmaa2_version`
- `drmaa2_machineinfo`

The interface membership of a function is sometimes expressed by an additional prefix, as shown in Table 1.

DRMAA interface	C binding prefix
<code>DrmaaReflective</code>	<code>drmaa2_</code>
<code>SessionManager</code>	<code>drmaa2_</code>
<code>JobSession</code>	<code>drmaa2_jsession_</code>
<code>ReservationSession</code>	<code>drmaa2_rsession_</code>
<code>MonitoringSession</code>	<code>drmaa2_msession_</code>
<code>Reservation</code>	<code>drmaa2_r_</code>
<code>Job</code>	<code>drmaa2_j_</code>
<code>JobArray</code>	<code>drmaa2_jarray_</code>
<code>JobTemplate</code>	<code>drmaa2_jtemplate_</code>
<code>ReservationTemplate</code>	<code>drmaa2_rtemplate_</code>

Table 1: Mapping of DRMAA interface name to C method prefix

The C binding specifies the function pointer type `drmaa2_callback` for a notification callback function. This represents the `DrmaaCallback` interface from the root specification. The new constant value `DRMAA2_UNSET_CALLBACK` can be used by the application for the de-registration of callback functions.

## 2.1 Error Handling

The list of exceptions in the DRMAA root specification is mapped to the new enumeration `drmaa2_error`. The enumeration member `DRMAA2_LASTERROR` is intended to ensure application portability while allowing

additional implementation-specific error codes. It MUST always be the enumeration member with the highest value.

The language binding adds two new functions for fetching error number and error message of the last error that occurred: `drmaa2_lasterror` and `drmaa2_lasterror_text`. These functions MUST operate in a thread-safe manner, meaning that both error informations are managed per application thread by the DRMAA implementation.

## 2.2 Lists and Dictionaries

The C language binding adds generic support functions for the collection data types used by the root specification. The newly defined `drmaa2_lasterror` and `drmaa2_lasterror_text` functions MUST return according error information for these operations.

Both `drmaa2_list_create` and `drmaa2_dict_create` have an optional parameter `callback`. It allows the application to provide a callback pointer to a collection element cleanup function. It MUST be allowed for the application to provide `DRMAA2_UNSET_CALLBACK` instead of a valid callback pointer.

The following list operations are defined:

**drmaa2\_list\_create:** Creates a new list instance for the specified type of items. Returns a pointer to the list or NULL on error.

**drmaa2\_list\_free:** Frees the list and the contained members. If a callback function was provided on list creation, it SHALL be called once per list item.

**drmaa2\_list\_get:** Gets the list element at the indicated position. The element index starts at zero. If the index is invalid, the function returns NULL.

**drmaa2\_list\_add:** Adds a new item at the end of the list and returns a success indication. The list MUST contain only the provided pointer, not a deep copy of the provided data structure.

**drmaa2\_list\_del:** Removes the list element at the indicated position and returns a success indication. If a callback function was provided on list creation, it SHALL be called before this function returns.

**drmaa2\_list\_size:** Returns the number of elements in the list. If the list is empty, then the function returns 0, which SHALL NOT be treated as an error case.

Similarly, a set of new functions for dictionary handling is introduced:

**drmaa2\_dict\_create:** Creates a new dictionary instance. Returns a pointer to the dictionary or NULL on error.

**drmaa2\_dict\_free:** Frees the dictionary and the contained members. If a callback function was provided on dictionary creation, it SHALL be called once per dictionary entry.

**drmaa2\_dict\_list:** Gets all dictionary keys as DRMAA `drmaa2_string_list`. If the dictionary is empty, a valid string list with zero elements SHALL be returned. The application is expected to use `drmaa2_list_free` for freeing the returned data structure.

**drmaa2\_dict\_has:** Returns a boolean indication if the given key exists in the dictionary. On error, the function SHALL return FALSE.

**drmaa2\_dict\_get:** Gets the dictionary value for the specified key. If the key is invalid, the function returns NULL.

**drmaa2\_dict\_del:** Removes the dictionary entry with the given key and returns a success indication. If a callback function was provided on dictionary creation, it SHALL be called before this function returns.

**drmaa2\_dict\_set:** Sets the specified dictionary key to the specified value. Key and value strings MUST be stored as the provided character pointers. If the dictionary already has an entry for this name, the value is replaced and the old value is removed. If a callback was provided on dictionary creation, it SHALL be called with a NULL pointer for the key and the pointer of the previous value.

### 3 Implementation-specific Extensions

The DRMAA root specification allows the product-specific extension of the DRMAA API in a standardized way.

New methods added to a DRMAA implementation SHOULD follow the conventions from Section 2. New **struct** attributes SHOULD use a product-specific prefix for a clear separation of non-portable and portable parts of the API. The extension MUST support the casting of product-specific **struct** pointers to their standard-compliant counterparts. Any compiler or linking options necessary for this feature MUST be documented accordingly by the DRMAA implementation.

### 4 Complete Header File

The following text shows the complete C header file for the DRMAAv2 application programming interface. DRMAA-compliant C libraries MUST declare all functions and data structures described here. Implementations MAY add custom parts in adherence to the extensibility principles of this specification and the root specification.

The source file is also available at <http://www.drmaa.org>.

```
#include <time.h>

#ifndef DRMAA2_H
#define 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;

typedef enum drmaa2_os {
    DRMAA2_OTHER_OS          = 0,
    DRMAA2_AIX               = 1,
    DRMAA2_BSD               = 2,
    DRMAA2_LINUX             = 3,
    DRMAA2_HPUX              = 4,
    DRMAA2_IRIX              = 5,
    DRMAA2_MACOS             = 6,
    DRMAA2_SUNOS             = 7,
    DRMAA2_TRU64             = 8,
    DRMAA2_UNIXWARE          = 9,
    DRMAA2_WIN               = 10,
    DRMAA2_WINNT             = 11
} drmaa2_os;
```

```

typedef enum drmaa2_cpu {
    DRMAA2_OTHER_CPU           = 0,
    DRMAA2_ALPHA               = 1,
    DRMAA2_ARM                 = 2,
    DRMAA2_ARM64               = 3,
    DRMAA2_CELL                = 4,
    DRMAA2_PARISC              = 5,
    DRMAA2_PARISC64            = 6,
    DRMAA2_X86                 = 7,
    DRMAA2_X64                 = 8,
    DRMAA2_IA64                = 9,
    DRMAA2_MIPS                = 10,
    DRMAA2_MIPS64              = 11,
    DRMAA2_PPC                 = 12,
    DRMAA2_PPC64               = 13,
    DRMAA2_SPARC               = 14,
    DRMAA2_SPARC64             = 15
} drmaa2_cpu;

typedef enum drmaa2_limit {
    DRMAA2_CORE_FILE_SIZE     = 0,
    DRMAA2_CPU_TIME           = 1,
    DRMAA2_DATA_SIZE          = 2,
    DRMAA2_FILE_SIZE          = 3,
    DRMAA2_OPEN_FILES         = 4,
    DRMAA2_STACK_SIZE         = 5,
    DRMAA2_VIRTUAL_MEMORY     = 6,
    DRMAA2_WALLCLOCK_TIME     = 7
} drmaa2_limit;

typedef enum drmaa2_jtemplate_placeholder {
    DRMAA2_HOME_DIRECTORY     = 0,
    DRMAA2_WORKING_DIRECTORY  = 1,
    DRMAA2_PARAMETRIC_INDEX   = 2
} drmaa2_jtemplate_placeholder;

typedef enum drmaa2_event {
    DRMAA2_NEW_STATE          = 0,
    DRMAA2_MIGRATED           = 1,
    DRMAA2_ATTRIBUTE_CHANGE   = 2
} drmaa2_event;

typedef enum {
    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;

typedef enum drmaa2_bool {
    DRMAA2_FALSE              = 0,
    DRMAA2_TRUE                = 1
} drmaa2_bool;

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;

typedef char * drmaa2_string;
void drmaa2_string_free(drmaa2_string *);

drmaa2_error drmaa2_lasterror(void);
drmaa2_string drmaa2_lasterror_text(void);

struct drmaa2_list_s;          /*forward*/
typedef struct drmaa2_list_s * drmaa2_list;
typedef struct drmaa2_list_s * drmaa2_string_list;
typedef struct drmaa2_list_s * drmaa2_j_list;
typedef struct drmaa2_list_s * drmaa2_queueinfo_list;
typedef struct drmaa2_list_s * drmaa2_machineinfo_list;
typedef struct drmaa2_list_s * drmaa2_slotinfo_list;
typedef struct drmaa2_list_s * drmaa2_r_list;

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 t, const drmaa2_list_entryfree callback);
void drmaa2_list_free (drmaa2_list * l);
const void * drmaa2_list_get (const drmaa2_list l, long pos);
drmaa2_error drmaa2_list_add (drmaa2_list l, const void * value);
drmaa2_error drmaa2_list_del (drmaa2_list l, long pos);
long drmaa2_list_size (const drmaa2_list l);

struct drmaa2_dict_s;          /*forward*/
typedef struct drmaa2_dict_s * drmaa2_dict;

typedef void (*drmaa2_dict_entryfree)(char **key, char **val);
drmaa2_dict drmaa2_dict_create (const drmaa2_dict_entryfree callback);
void drmaa2_dict_free (drmaa2_dict * d);
drmaa2_string_list drmaa2_dict_list (const drmaa2_dict d);
drmaa2_bool drmaa2_dict_has (const drmaa2_dict d, const char * key);
const char * drmaa2_dict_get (const drmaa2_dict d, const char * key);
drmaa2_error drmaa2_dict_del (drmaa2_dict d, const char * key);
drmaa2_error drmaa2_dict_set (drmaa2_dict d, const char * key, const char * val);

#define DRMAA2_ZERO_TIME ((time_t) 0)
#define DRMAA2_INFINITE_TIME ((time_t) -1)
#define DRMAA2_NOW ((time_t) -2)

#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

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);
void          drmaa2_jinfo_free  (drmaa2_jinfo * ji);

typedef struct {
    drmaa2_string       machineName;
    long long           slots;
} drmaa2_slotinfo_s;
typedef drmaa2_slotinfo_s * drmaa2_slotinfo;

void drmaa2_slotinfo_free  (drmaa2_slotinfo * si);

typedef struct {
    drmaa2_string       reservationId;
    drmaa2_string       reservationName;
    time_t              reservedStartTime;
    time_t              reservedEndTime;
    drmaa2_string_list  usersACL;
    long long           reservedSlots;
    drmaa2_slotinfo_list reservedMachines;
} drmaa2_rinfo_s;
typedef drmaa2_rinfo_s * drmaa2_rinfo;

void drmaa2_rinfo_free  (drmaa2_rinfo * ri);

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;
} drmaa2_jtemplate_s;
typedef drmaa2_jtemplate_s * drmaa2_jtemplate;

drmaa2_jtemplate drmaa2_jtemplate_create (void);
void drmaa2_jtemplate_free (drmaa2_jtemplate * jt);

typedef struct {
    drmaa2_string    reservationName;
    time_t          startTime;
    time_t          endTime;
    time_t          duration;
    long long        minSlots;
    long long        maxSlots;
    drmaa2_string    jobCategory;
    drmaa2_string_list usersACL;
    drmaa2_string_list candidateMachines;
    long long        minPhysMemory;
    drmaa2_os        machineOS;
    drmaa2_cpu        machineArch;
} drmaa2_rtemplate_s;
typedef drmaa2_rtemplate_s * drmaa2_rtemplate;

drmaa2_rtemplate drmaa2_rtemplate_create (void);
void drmaa2_rtemplate_free (drmaa2_rtemplate * rt);

typedef struct {
    drmaa2_event    event;
    drmaa2_string    jobId;
    drmaa2_string    sessionName;
    drmaa2_jstate    jobState;
} drmaa2_notification_s;
typedef drmaa2_notification_s * drmaa2_notification;

void drmaa2_notification_free (drmaa2_notification * n);

typedef struct {
    drmaa2_string    name;
} drmaa2_queueinfo_s;
typedef drmaa2_queueinfo_s * drmaa2_queueinfo;

void drmaa2_queueinfo_free (drmaa2_queueinfo * qi);

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

void drmaa2_version_free (drmaa2_version * v);

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_os        machineOS;
    drmaa2_version    machineOSVersion;
    drmaa2_cpu        machineArch;
} drmaa2_machineinfo_s;
typedef drmaa2_machineinfo_s * drmaa2_machineinfo;

void drmaa2_machineinfo_free (drmaa2_machineinfo * mi);

drmaa2_string_list drmaa2_jtemplate_impl_spec (void);
drmaa2_string_list drmaa2_jinfo_impl_spec (void);

```

```

drmaa2_string_list drmaa2_rtemplate_impl_spec      (void);
drmaa2_string_list drmaa2_rinfo_impl_spec          (void);
drmaa2_string_list drmaa2_queueinfo_impl_spec      (void);
drmaa2_string_list drmaa2_machineinfo_impl_spec    (void);
drmaa2_string_list drmaa2_notification_impl_spec   (void);

drmaa2_string drmaa2_get_instance_value (const void * instance, const char * name);
drmaa2_string drmaa2_describe_attribute (const void * instance, const char * name);
drmaa2_error  drmaa2_set_instance_value (void * instance, const char * name, const char * value);

typedef void (*drmaa2_callback)(drmaa2_notification * notification);

struct drmaa2_jsession_s; /*forward*/
struct drmaa2_rsession_s; /*forward*/
struct drmaa2_msession_s; /*forward*/
struct drmaa2_j_s; /*forward*/
struct drmaa2_jarray_s; /*forward*/
struct drmaa2_r_s; /*forward*/

typedef struct drmaa2_jsession_s * drmaa2_jsession;
typedef struct drmaa2_rsession_s * drmaa2_rsession;
typedef struct drmaa2_msession_s * drmaa2_msession;
typedef struct drmaa2_j_s * drmaa2_j;
typedef struct drmaa2_jarray_s * drmaa2_jarray;
typedef struct drmaa2_r_s * drmaa2_r;

void drmaa2_jsession_free(drmaa2_jsession * js);
void drmaa2_rsession_free(drmaa2_rsession * rs);
void drmaa2_msession_free(drmaa2_msession * ms);
void drmaa2_j_free (drmaa2_j * j);
void drmaa2_jarray_free (drmaa2_jarray * ja);
void drmaa2_r_free (drmaa2_r * r);

drmaa2_string drmaa2_rsession_get_contact (const drmaa2_rsession rs);
drmaa2_string drmaa2_rsession_get_session_name (const drmaa2_rsession rs);
drmaa2_r drmaa2_rsession_get_reservation (const drmaa2_rsession rs, const drmaa2_string reservationId);
drmaa2_r drmaa2_rsession_request_reservation (const drmaa2_rsession rs, const drmaa2_rtemplate rt);
drmaa2_r_list drmaa2_rsession_get_reservations (const drmaa2_rsession rs);

drmaa2_string drmaa2_r_get_id (const drmaa2_r r);
drmaa2_string drmaa2_r_get_session_name (const drmaa2_r r);
drmaa2_rtemplate drmaa2_r_get_reservation_template (const drmaa2_r r);
drmaa2_rinfo drmaa2_r_get_info (const drmaa2_r r);
drmaa2_error drmaa2_r_terminate (drmaa2_r r);

drmaa2_string drmaa2_jarray_get_id (const drmaa2_jarray ja);
drmaa2_j_list drmaa2_jarray_get_jobs (const drmaa2_jarray ja);
drmaa2_string drmaa2_jarray_get_session_name (const drmaa2_jarray ja);
drmaa2_jtemplate drmaa2_jarray_get_job_template (const drmaa2_jarray ja);
drmaa2_error drmaa2_jarray_suspend (drmaa2_jarray ja);
drmaa2_error drmaa2_jarray_resume (drmaa2_jarray ja);
drmaa2_error drmaa2_jarray_hold (drmaa2_jarray ja);
drmaa2_error drmaa2_jarray_release (drmaa2_jarray ja);
drmaa2_error drmaa2_jarray_terminate (drmaa2_jarray ja);

drmaa2_string drmaa2_jsession_get_contact (const drmaa2_jsession js);
drmaa2_string drmaa2_jsession_get_session_name (const drmaa2_jsession js);
drmaa2_string_list drmaa2_jsession_get_job_categories (const drmaa2_jsession js);
drmaa2_j_list drmaa2_jsession_get_jobs (const drmaa2_jsession js,
const drmaa2_jinfo filter);
drmaa2_jarray drmaa2_jsession_get_job_array (const drmaa2_jsession js,
const drmaa2_string jobarrayId);
drmaa2_j drmaa2_jsession_run_job (const drmaa2_jsession js,
const drmaa2_jtemplate jt);
drmaa2_jarray drmaa2_jsession_run_bulk_jobs (const drmaa2_jsession js,
const drmaa2_jtemplate jt,
unsigned long begin_index,
unsigned long end_index,
unsigned long step,
unsigned long max_parallel);
drmaa2_j drmaa2_jsession_wait_any_started (const drmaa2_jsession js,

```

```

                                const drmaa2_j_list l,
                                const time_t timeout);
drmaa2_j      drmaa2_jsession_wait_any_terminated (const drmaa2_jsession js,
                                const drmaa2_j_list l,
                                const time_t timeout);

drmaa2_string drmaa2_j_get_id      (const drmaa2_j j);
drmaa2_string drmaa2_j_get_session_name (const drmaa2_j j);
drmaa2_jtemplate drmaa2_j_get_jt   (const drmaa2_j j);
drmaa2_error  drmaa2_j_suspend     (drmaa2_j j);
drmaa2_error  drmaa2_j_resume      (drmaa2_j j);
drmaa2_error  drmaa2_j_hold        (drmaa2_j j);
drmaa2_error  drmaa2_j_release     (drmaa2_j j);
drmaa2_error  drmaa2_j_terminate   (drmaa2_j j);
drmaa2_jstate drmaa2_j_get_state   (const drmaa2_j j, drmaa2_string * substate);
drmaa2_jinfo  drmaa2_j_get_info    (const drmaa2_j j);
drmaa2_error  drmaa2_j_wait_started (const drmaa2_j j, const time_t timeout);
drmaa2_error  drmaa2_j_wait_terminated (const drmaa2_j j, const time_t timeout);

drmaa2_r_list drmaa2_msession_get_all_reservations (const drmaa2_msession ms);
drmaa2_j_list drmaa2_msession_get_all_jobs        (const drmaa2_msession ms,
                                const drmaa2_jinfo filter);
drmaa2_queueinfo_list drmaa2_msession_get_all_queues (const drmaa2_msession ms,
                                const drmaa2_string_list names);
drmaa2_machineinfo_list drmaa2_msession_get_all_machines (const drmaa2_msession ms,
                                const drmaa2_string_list names);

drmaa2_string drmaa2_get_drms_name      (void);
drmaa2_version drmaa2_get_drms_version  (void);
drmaa2_string drmaa2_get_drmaa_name     (void);
drmaa2_version drmaa2_get_drmaa_version (void);
drmaa2_bool    drmaa2_supports          (const drmaa2_capability c);
drmaa2_jsession drmaa2_create_jsession  (const char * session_name, const char * contact);
drmaa2_rsession drmaa2_create_rsession  (const char * session_name, const char * contact);
drmaa2_jsession drmaa2_open_jsession    (const char * session_name);
drmaa2_rsession drmaa2_open_rsession    (const char * session_name);
drmaa2_msession drmaa2_open_msession    (const char * session_name);
drmaa2_error  drmaa2_close_jsession     (drmaa2_jsession js);
drmaa2_error  drmaa2_close_rsession     (drmaa2_rsession rs);
drmaa2_error  drmaa2_close_msession     (drmaa2_msession ms);
drmaa2_error  drmaa2_destroy_jsession   (const char * session_name);
drmaa2_error  drmaa2_destroy_rsession   (const char * session_name);
drmaa2_string_list drmaa2_get_jsession_names (void);
drmaa2_string_list drmaa2_get_rsession_names (void);
drmaa2_error  drmaa2_register_event_notification (const drmaa2_callback callback);

#endif

```

## 5 Security Considerations

The DRMAA root specification [2] describes the behavioral aspects of a standard-compliant implementation. This includes also security aspects.

Software written in C language has well-known security attack vectors, especially with memory handling. Implementors MUST clarify in their documentation which kind of memory management is expected by the application. Implementations MUST also consider the possibility for multi-threaded applications performing re-entrant calls to the library. The root specification clarifies some of these scenarios.

## 6 Contributors

**Roger Brobst**

Cadence Design Systems, Inc.  
555 River Oaks Parkway  
San Jose, CA 95134, United States  
Email: rbrobst@cadence.com

**Daniel Gruber**

Univa GmbH  
c/o Rüter und Partner  
Prielmayerstr. 3  
80335 München, Germany  
Email: dgruber@univa.com

**Mariusz Mamoński**

Poznań Supercomputing and Networking Center  
ul. Noskowskiego 10  
61-704 Poznań, Poland  
Email: mamonski@man.poznan.pl

**Andre Merzky**

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

**Peter Tröger (Corresponding Author)**

Hasso Plattner Institute at University of Potsdam  
Prof.-Dr.-Helmert-Str. 2-3  
14482 Potsdam, Germany  
Email: peter@troeger.eu

Special thanks go to *Stefan Klauck (Hasso Plattner Institute)* for the DRMAA C binding reference implementation and the debugging of the implementation-related language binding issues.

## 7 Intellectual Property Statement

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

license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

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

## 8 Disclaimer

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

## 9 Full Copyright Notice

Copyright © Open Grid Forum (2012-2012). Some Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

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

## 10 References

- [1] Scott Bradner. Key words for use in RFCs to Indicate Requirement Levels. RFC 2119 (Best Current Practice), March 1997. URL <http://tools.ietf.org/html/rfc2119>.
- [2] Peter Tröger, Roger Brobst, Daniel Gruber, Mariusz Mamonski, and Daniel Templeton. Distributed Resource Management Application API Version 2 (DRMAA). <http://www.ogf.org/documents/GFD.194.pdf>, January 2012.