

GWD-R

Distributed Resource Management
Application API (DRMAA) Working Group

Daniel Templeton, Sun Microsystems (maintainer)
Roger Brobst, Cadence Design Systems
Andreas Haas, Sun Microsystems
Hrabri Rajic*, Intel Americas Inc.
John Tollefsrud*, Sun Microsystems
*co-chairs
March, 2004

Distributed Resource Management Application API Java™ Language Bindings 0.3

Status of This Memo

This memo is a Global Grid Forum Grid Working Draft - *Recommendations* (GWD-R) in process, in general accordance with the provisions of Global Grid Forum Document GFD-C.1, the Global Grid Forum Documents and Recommendations: Process and Requirements, revised April 2002.

Copyright Notice

Copyright © Global Grid Forum (2003). All Rights Reserved.

Table of Contents

1 Abstract.....	3
2 Overview.....	3
3 Design Decisions.....	3
3.1 Service Provider Interface.....	3
4 The Java Language Binding API.....	3
4.1 The DRMAASession Class.....	4
4.2 The DRMAASessionFactory Class.....	5
4.3 The JobTemplate Class.....	5
4.4 The JobInfo Class.....	6
4.5 The Exception Heirarchy.....	6
5 Java Language Binding Example.....	7
6 Security Considerations.....	10
7 Author Information.....	10
8 Intellectual Property Statement.....	11
9 Full Copyright Notice.....	11

1 Abstract

This document describes the Distributed Resource Management Application API (DRMAA) Java™ language bindings. The document is based on the implementations work of the DRMAA GWD-R document.

2 Overview

This document describes the Java™ language binding for the [DRMAA](#) interface. Issues regarding interface semantics, possible argument values, error conditions etc. are addressed in chapter "3.2 DRMAA API" of the interface specification. As a result the Java API defined in the following sections is complete only regarding the interfaces needed by a Java application to use the API.

This Java language binding was developed with the Java 2 Standard Edition 1.4.2, however it should be compatible with any Java 2 Software Development Kit version 1.2 or greater. This requirement stems from the use of the Collections API which was first introduced with JDK1.2.

3 Design Decisions

In order to make the Java language binding as familiar to programmers as possible, whenever possible, design elements were borrowed from common Java language APIs. The Java language binding makes use of an API/SPI factory pattern similar to the JAX Pack APIs. The Java language binding also abstracts exception handling to a single, declared, top-level exception as is done in the JDBC API. Properties in the Java language binding follow the standard JavaBean property pattern.

3.1 Service Provider Interface

The Java language binding allows vendors to implement the DRM-specific binding classes required to interface with a given DRM without changing the outward facing API. By extending the abstract classes in the Java language binding and providing implementations of the abstract methods, a vendor can tailor his implementation to his needs. The vendor implementation is completely transparent to the DRMAA application, however. The API hides the SPI and prevents the DRMAA application from needing to know anything about the underlying implementation.

4 The Java Language Binding API

The DRMAA Interface Specification was written originally with a slant towards a C/C++ binding. As such, several aspects of the DRMAA interface needed to be altered slightly to better fit with an object-oriented language like the Java language. Among the aspects that changed are variable and method naming and the error structure.

Additionally, some methods from the DRMAA specification fail to appear in the Java language binding specification. The `drmaa_get_vector_attribute()`, `drmaa_set_vector_attribute()`, and `drmaa_get_vector_attribute_names()` methods are not needed because the Java language binding specification handles all attributes as instances of `java.lang.List`, a vector representation.

In order to tell “String Vector” attributes from “String” attributes, a developer can check the length of the returned List instance. If the instance is of length 1, the attribute is a “String” attribute, and (String)List.get(0) can be used to retrieve the associated java.lang.String instance. If the attribute has any other length, the attribute is a “Vector String” attribute and can be operated upon as a normal List, e.g. List.iterator(). The advantage of this is that “String” attributes and “Vector String” attributes can be processed the same way. After the introduction of generics in the Java 2 Standard Edition 1.5, the instance of List returned may be bound to the type String.

4.1 The DRMAASession Class

The main class in the Java language binding is the DRMAASession class. It represents the majority of the functionality defined by the DRMAA Interface Specification. It has the following structure:

```
public abstract class com.sun.grid.drmaa.DRMAASession {
    public static final int SUSPEND
    public static final int RESUME
    public static final int HOLD
    public static final int RELEASE
    public static final int TERMINATE
    public static final java.util.List JOB_IDS_SESSION_ALL
    public static final java.lang.String JOB_IDS_SESSION_ANY
    public static final long TIMEOUT_WAIT_FOREVER
    public static final long TIMEOUT_NO_WAIT
    public static final int UNDETERMINED
    public static final int QUEUED_ACTIVE
    public static final int SYSTEM_ON_HOLD
    public static final int USER_ON_HOLD
    public static final int USER_SYSTEM_ON_HOLD
    public static final int RUNNING
    public static final int SYSTEM_SUSPENDED
    public static final int USER_SUSPENDED
    public static final int DONE
    public static final int FAILED
    protected com.sun.grid.drmaa.DRMAASession()
    public abstract void init(java.lang.String contactString)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract void exit()
        throws com.sun.grid.drmaa.DRMAAException
    public abstract com.sun.grid.drmaa.JobTemplate
        createJobTemplate()
        throws com.sun.grid.drmaa.DRMAAException
    public abstract java.lang.String
        runJob(com.sun.grid.drmaa.JobTemplate jobTemplate)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract java.util.Set
        runBulkJobs(com.sun.grid.drmaa.JobTemplate jobTemplate,
            int beginIndex, int endIndex, int loopIndex)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract void control(java.lang.String jobName,
        int operation)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract void synchronize(java.util.List jobList,
        long timeout, boolean dispose)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract com.sun.grid.drmaa.JobInfo
        wait(java.lang.String jobName, long timeout)
```

```

        throws com.sun.grid.drmaa.DRMAAException
    public abstract int getJobProgramStatus(java.lang.String jobName)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract java.lang.String getContact()
    public abstract com.sun.grid.drmaa.DRMAASession$Version
        getVersion()
    public abstract java.lang.String getDRMSInfo()
    public java.lang.String getDRMAAImplementation()
}

```

4.2 The DRMAASessionFactory Class

In order to enable a Java language binding implementation to be supported by multiple different vendors, a factory class is needed to allow a DRMAA application to retrieve a vendor specific implementation of the DRMAASession class. The DRMAASessionFactory class serves this purpose and additionally allows the vendor the freedom to return different DRMAASession subclasses depending on the need. The structure of the DRMAASessionFactory class is as follows:

```

public abstract class com.sun.grid.drmaa.DRMAASessionFactory {
    public static com.sun.grid.drmaa.DRMAASessionFactory getFactory()
    public abstract com.sun.grid.drmaa.DRMAASession getSession()
}

```

4.3 The JobTemplate Class

In order to define the attributes associated with a job, a DRMAA application uses the JobTemplate class. JobTemplates are created via the active DRMAASession class. A DRMAA application gets a JobTemplate from the active DRMAASession, specifies in the JobTemplate any required job parameters, and then passes the JobTemplate back to the DRMAASession when requesting that a job be executed. When finished, the DRMAA application should call the `delete()` method to allow the underlying implementation to free any resources bound to the JobTemplate object. The structure of the JobTemplate class is as follows:

```

public abstract class com.sun.grid.drmaa.JobTemplate {
    public static final java.lang.String REMOTE_COMMAND
    public static final java.lang.String INPUT_PARAMETERS
    public static final java.lang.String JOB_SUBMISSION_STATE
    public static final java.lang.String JOB_ENVIRONMENT
    public static final java.lang.String WORKING_DIRECTORY
    public static final java.lang.String JOB_CATEGORY
    public static final java.lang.String NATIVE_SPECIFICATION
    public static final java.lang.String EMAIL_ADDRESS
    public static final java.lang.String EMAIL_SUPPRESSION
    public static final java.lang.String START_TIME
    public static final java.lang.String JOB_NAME
    public static final java.lang.String INPUT_PATH
    public static final java.lang.String OUTPUT_PATH
    public static final java.lang.String ERROR_PATH
    public static final java.lang.String JOIN_FILES
    public static final java.lang.String TRANSFER_FILES
    public static final java.lang.String DEADLINE_TIME
    public static final java.lang.String HARD_WALLCLOCK_TIME_LIMIT
    public static final java.lang.String SOFT_WALLCLOCK_TIME_LIMIT
    public static final java.lang.String HARD_RUN_DURATION_LIMIT
}

```

```

    public static final java.lang.String SOFT_RUN_DURATION_LIMIT
    public static final int HOLD
    public static final int ACTIVE
    protected com.sun.grid.drmaa.JobTemplate()
    public abstract void setAttribute(java.lang.String name,
                                     java.lang.String value)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract void setAttribute(java.lang.String name,
                                     java.util.List value)
        throws com.sun.grid.drmaa.DRMAAException
    public abstract java.util.List getAttribute(java.lang.String name)
    public abstract java.util.List getAttributeNames()
    public abstract void delete()
        throws com.sun.grid.drmaa.DRMAAException
}

```

4.4 The JobInfo Class

The information regarding a job's execution is encapsulated in the JobInfo class. Via the JobInfo class a DRMAA application can discover information about the resource usage and exit status of a job. The structure of the JobInfo class is as follows:

```

public abstract class com.sun.grid.drmaa.JobInfo
    implements java.io.Serializable {
    protected java.lang.String jobId
    protected int status
    protected java.util.Map resourceUsage
    protected com.sun.grid.drmaa.JobInfo (java.lang.String jobName,
                                         int statusCode,
                                         java.util.Map resourceUsage)

    public java.lang.String getJobId()
    public java.util.Map getResourceUsage()
    public abstract boolean hasExited()
    public abstract int getExitStatus()
    public abstract boolean hasSignaled()
    public abstract java.lang.String getTerminatingSignal()
    public abstract boolean hasCoreDump()
    public abstract boolean wasAborted()
}

```

4.5 The Exception Heirarchy

All exception in the Java language binding inherit from the DRMAAException class. The structure of DRMAAException follows:

```

public class com.sun.grid.drmaa.DRMAAException
    extends java.lang.Exception{
    public com.sun.grid.drmaa.DRMAAException();
    public com.sun.grid.drmaa.DRMAAException(java.lang.String);
}

```

The exception heirarchy is:

- *java.lang.Object*
 - *java.lang.Throwable*

- *java.lang.Exception*
 - *com.sun.grid.drmaa.DRMAAException*
 - *com.sun.grid.drmaa.AuthorizationException*
 - *com.sun.grid.drmaa.DefaultContactStringException*
 - *com.sun.grid.drmaa.DeniedByDRMException*
 - *com.sun.grid.drmaa.DRMCommunicationException*
 - *com.sun.grid.drmaa.ExitTimeoutException*
 - *com.sun.grid.drmaa.InconsistentStateException*
 - *com.sun.grid.drmaa.InternalException*
 - *com.sun.grid.drmaa.InvalidArgumentException*
 - *com.sun.grid.drmaa.InvalidAttributeException*
 - *com.sun.grid.drmaa.ConflictingAttributeValuesException*
 - *com.sun.grid.drmaa.InvalidAttributeFormatException*
 - *com.sun.grid.drmaa.InvalidAttributeValueException*
 - *com.sun.grid.drmaa.InvalidContactStringException*
 - *com.sun.grid.drmaa.InvalidJobException*
 - *com.sun.grid.drmaa.NoResourceUsageDataException*
 - *com.sun.grid.drmaa.SessionException*
 - *com.sun.grid.drmaa.DRMSExitException*
 - *com.sun.grid.drmaa.NoActiveSessionException*
 - *com.sun.grid.drmaa.SessionAlreadyActiveException*
 - *com.sun.grid.drmaa.TryLaterException*

All exceptions under the DRMAAException have the following structure:

```
public class com.sun.grid.drmaa.<NAME>Exception
    extends <PARENT>Exception{
    public com.sun.grid.drmaa.<NAME>Exception();
    public com.sun.grid.drmaa.<NAME>Exception(java.lang.String);
}
```

Where <NAME> is the name of the error and <PARENT> is the name of the parent error.

5 Java Language Binding Example

The Java application below is an example of an application that uses the DRMAA Java language binding interface. It illustrates submission of both single and bulk jobs. After submission DRMAASession.synchronize() is used to synchronize with all jobs to finish. Finally DRMAASession.wait() is used to retrieve and print out information about the exit status of each job.

The path, which must be passed as argument to the program, is directly used for the job template JobTemplate.REMOTE_COMMAND attribute. The Java language binding example passes "5" as first argument to the job template JobTemplate.INPUT_PARAMETERS attribute. Assuming the example is run under "/bin/sleep" unix command and that a command "/bin/sleep" exists at the remote machine which behaves like the UNIX sleep(1) command, running this application with the parameter "/bin/sleep" will result in 32 jobs being run that sleep for 5 seconds each before finishing.

The source code follows:

```

import java.util.*;
import com.sun.grid.drmaa.*;

public class DRMAAExample {
    private static int NBULKS = 3;
    private static int JOB_CHUNK = 8;
    private DRMAASession session = null;

    public void main (String[] args) throws Exception {
        String jobPath = args[0];
        session = DRMAASessionFactory.getFactory ().getSession ();
        session.init (null);

        JobTemplate jt = createJobTemplate (jobPath, 5, true);

        List allJobIds = new LinkedList ();
        Set jobIds = null;
        boolean retry = true;

        for (int count = 0; count < NBULKS; count++) {
            do {
                try {
                    jobIds = session.runBulkJobs (jt, 1, JOB_CHUNK, 1);
                    retry = false;
                }
                catch (DRMCommunicationException e) {
                    System.err.println ("runBulkJobs() failed - retry: " +
                                         e.getMessage ());

                    Thread.sleep (1000);
                }
            }
            while (retry);

            allJobIds.add (jobIds);

            System.out.println ("submitted bulk job with jobids:");

            Iterator i = jobIds.iterator ();

            while (i.hasNext ()) {
                System.out.println ("\t \"" + i.next () + "\"");
            }
        }

        jt.delete ();

        /* submit some sequential jobs */
        jt = createJobTemplate (jobPath, 5, false);

        String jobId = null;
        retry = true;

        for (int count = 0; count < JOB_CHUNK; count++) {
            do {
                try {
                    jobId = session.runJob (jt);
                    retry = false;
                }
                catch (DRMCommunicationException e) {

```



```

        System.err.println ("runBulkJobs() failed - retry: " +
                            e.getMessage ());

        Thread.sleep (1000);
    }
}
while (retry);

System.out.println ("\t \"" + jobId + "\"");
jobIds.add (jobId);
}

jt.delete ();

/* synchronize with all jobs */
session.synchronize (allJobIds,
                    DRMAASession.TIMEOUT_WAIT_FOREVER,
                    false);
System.out.println ("synchronized with all jobs");

/* wait all those jobs */
Iterator i = allJobIds.iterator ();

while (i.hasNext ()) {
    JobInfo status = null;

    status = session.wait ((String)i.next (),
                          DRMAASession.TIMEOUT_WAIT_FOREVER);

    /* report how job finished */
    if (status.wasAborted ()) {
        System.out.println ("job \"" + i.next () + "\" never ran");
    }
    else if (status.hasExited ()) {
        System.out.println ("job \"" + i.next () +
                            "\" finished regularly with exit status " +
                            status.getExitStatus ());
    }
    else if (status.hasSignaled ()) {
        System.out.println ("job \"" + i.next () +
                            "\" finished due to signal " +
                            status.getTerminatingSignal ());
    }
    else {
        System.out.println ("job \"" + i.next () +
                            "\" finished with unclear conditions");
    }
}
}

private JobTemplate createJobTemplate (String jobPath,
                                       int seconds,
                                       boolean isBulkJob)
    throws DRMAAException {
    JobTemplate jt = session.createJobTemplate ();

    jt.setAttribute (JobTemplate.WORKING_DIRECTORY, "$drmaa_hd_pd$");
    jt.setAttribute (JobTemplate.REMOTE_COMMAND, jobPath);
    jt.setAttribute (JobTemplate.INPUT_PARAMETERS,
                    Arrays.asList (new String[] {
                                    Integer.toString (seconds)
                                }
    ));
}

```

```

        }));
jt.setAttribute (JobTemplate.JOIN_FILES, "y");
if (!isBulkJob) {
    jt.setAttribute (JobTemplate.OUTPUT_PATH,
        "$drmaa_hd_pd$/DRMĀA_JOB");
}
else {
    jt.setAttribute (JobTemplate.OUTPUT_PATH,
        "$drmaa_hd_pd$/DRMĀA_JOB$drmaa_incr_ph$");
}
return jt;
}
}

```

6 Security Considerations

Security issues are not discussed in this document. The scheduling scenario described here assumes that security is handled at the point of job authorization/execution on a particular resource. Also, the Java 2 Standard Edition Runtime Environment applies a fine-grained security model that can be assumed to provide some measure or protection at the point of execution.

7 Author Information

Roger Brobst
 rbrobst@cadence.com
 Cadence Design Systems, Inc
 555 River Oaks Parkway
 San Jose, CA 95134

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

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

Daniel Templeton
 dan.templeton@sun.com
 Sun Microsystems GmbH
 Dr.-Leo-Ritter-Str. 7
 D-93049 Regensburg
 Germany

John Tollefsrud

j.t@sun.com
Sun Microsystems
200 Jefferson Drive UMPK29-302
Menlo Park, CA 94025

8 Intellectual Property Statement

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

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

9 Full Copyright Notice

Copyright (C) Global Grid Forum (date). All 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 GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

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

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM 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."