GWD-R

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

Distributed Resource Management
Application API (DRMAA) Working Group

September, 2003

Distributed Resource Management Application API Java™ Language Bindings 0.1

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

Table of Contents

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 J2SDK version 1.2 or greater. This requirement stems from the use of the Collections API which was first introduced with JDK1.2.

3 The Java Language Binding API

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

3.1 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;
    public com.sun.grid.drmaa.DRMAASession();
    public abstract void init(java.lang.String contact);
       throws com.sun.grid.drmaa.DRMAAException
```

```
    public abstract void exit();
       throws com.sun.grid.drmaa.DRMAAException
    public abstract com.sun.grid.drmaa.JobTemplate
       allocateJobTemplate();
       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.List
       runBulkJobs(com.sun.grid.drmaa.JobTemplate jobTemplate,int
           start,int end,int incr);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract void control(java.lang.String jobId,int action);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract void synchronize(java.util.List jobIds,long
          timeout,Boolean dispose);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract com.sun.grid.drmaa.JobInfo
       wait(java.lang.String jobId,long timeout);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract int getJobProgramStatus(java.lang.String jobId);
       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();
}
```

3.2 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 structure of the DRMAASessionFactory class is as follows:

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

3.3 The JobTemplate Class

In order to define the attributes associated with a job, a DRMAA application uses the JobTemplate class. JobTemplates are managed by the active DRMAASession class. A DRMAA application gets a JobTemplate from the active DRMAASession, specifies in the JobTemplate any required job parameters, and the passes the JobTemplate back to the DRMAASession when requesting that the job be executed. The structure of the JobTemplate class is as follows:

```
public abstract class com.sun.grid.drmaa.JobTemplate extends {
    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 BLOCK_EMAIL;
    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;
    public static final int JOBNAME_BUFFER;
    public com.sun.grid.drmaa.JobTemplate();
    public abstract void delete();
       throws com.sun.grid.drmaa.DRMAAException
    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 values);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract java.lang.String
       getAttributeValue(java.lang.String name);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract java.util.List getAttributeValues(java.lang.String
          name);
       throws com.sun.grid.drmaa.DRMAAException
    public abstract java.util.List getAttributeNames();
    public abstract java.util.List getVectorAttributeNames();
}
```

3.4 The JobInfo Class

The information regarding a job's status 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;
    public com.sun.grid.drmaa.JobInfo
       (java.lang.String jobId,int status,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();
}
```

3.5 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 msg);
}
```

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.HoldInconsistentStateException*
          - *com.sun.grid.drmaa.ReleaseInconsistentStateException*
          - *com.sun.grid.drmaa.ResumeInconsistentStateException*
          - *com.sun.grid.drmaa.SuspendInconsistentStateException*
        - *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.

4 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 that 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 which 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.getSession ();
    session.init (null); // null string asks for default connection

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

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

    // submit NBULK sets of JOB_CHUNK jobs as bulk jobs
    // retrying if the link with the DRM system is temporarily
    // unavailable
    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;

// submit JOB_CHUNK single jobs
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.allocateJobTemplate ();

    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,
                       "$drmaa_hd_pd$/DRMAA_JOB");
    }
    else {
      jt.setAttribute (JobTemplate.OUTPUT_PATH,
                       "$drmaa_hd_pd$/DRMAA_JOB$drmaa_incr_ph$");
    }
```

```
    return jt;
  }
}
```

5 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.

6 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

7 Intellectual Property Statement

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.

8 Full Copyright Notice