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 0.4.1

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

Index of Tables

1 Abstract

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

2 Overview

This document describes the Java^{TM} 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^{TM} 1.4.2 in mind, however it should be implementable with any Java 2 Software Development Kit^{TM} version 1.2 or greater. This requirement stems from the use of the Collections API which was first introduced with Java Development Kit^{TM} 1.2.

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 handing 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_attribute(), drmaa_set_attribute(), drmaa_get_vector_attribute(), drmaa_set_vector_attribute(), and drmaa_get_vector_attribute_names() methods are not needed because the Java language binding specification specifies a specific setter and getter for each DRMAA attribute. The advantage is that the specific setters and getters allow for compile-time type checking of DRMAA attributes, and allow special treatment of attributes which are better represented as something other than a String. Below is a table of the DRMAA attributes, their corresponding Java property names, and their types.

Table 1: DRMAA Attributes

DRMAA Attribute	***Java Property***	***Type***
drmaa_remote_command	remoteCommand	java.lang.String
drmaa_v_argv	inputParameters	java.lang.String[]
drmaa_js_state	jobSubmissionState	int
drmaa_v_env	jobEnvironment	java.util.Properties
drmaa_wd	workingDirectory	java.lang.String
drmaa_job_category	jobCategory	java.lang.String
drmaa_native_specification	nativeSpecification	java.lang.String
drmaa_v_email	emailAddresses	java.lang.String[]
drmaa_block_email	BlockEmail	boolean
drmaa_start_time	startTime	java.util.Date
drmaa_job_name	jobName	java.lang.String
drmaa_input_path	inputPath	java.lang.String
drmaa_output_path	outputPath	java.lang.String
drmaa_error_path	errorPath	java.lang.String
drmaa_join_files	joinFiles	boolean
drmaa_transfer_files	transferFiles	byte
drmaa_deadline_time	deadlineTime	java.util.Date
drmaa_wct_hlimit	hardWallclockTimeLimit	long
drmaa_wct_slimit	softWallclockTimeLimit	long
drmaa_run_duration_hlimit	hardRunDurationLimit	long
drmaa_run_duration_slimit	softRunDurationLimit	long

The setters and getters follow the JavaBeans pattern for properties. For an attribute named *attribute* of type *Type*, the signature of the setter and getter would be:

```
public void setAttribute (Type value) throws DRMAAException;
public Type getAttribute ();
```

All attribute setters and getters operate in a pass-by-value mode. For data types which are not natively pass-by-value, such as java.util.Date, the data is copied so that the data structure stored by the Java binding is uncoupled from the data structure in the calling application. The only

exception to this rule is the jobEnvironment property. Because the java.util.Properties class supports the separation of original values from values set by the client, it is possible to allow the client to access the java.util.Properties object directly.
Optional attributes are also represented by setters and getters. The Java binding implementation should provide implementations for setters and getters for all DRMAA attributes, both required and optional. The setter and getter implementations for optional attributes should throw org.ggf.drmaa.UnsupportedAttributeException. The service provider implementation must then override the setters and getters for supported optional attributes with methods that operate normally.
The JobTemplate.getAttributeNames() method returns the names of all properties supported by the service provider implementation, including required, optional, and implementation specific attributes. In order to get the values for supported attributes, such as in a property sheet, one should use introspection to call the appropriate setter and getter for each attribute.

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.List
       runBulkJobs(com.sun.grid.drmaa.JobTemplate jobTemplate,
                   int beginIndex, int endIndex, int step)
       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 the 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 int HOLD
    public static final int ACTIVE
    public static final byte TRANSFER_NONE;
    public static final byte TRANSFER_ERROR_FILES;
    public static final byte TRANSFER_INPUT_FILES;
    public static final byte TRANSFER_OUTPUT_FILES;
    protected java.lang.String remoteCommand;
    protected java.lang.String[] args;
    protected int state;
    protected java.util.Properties env;
    protected java.lang.String wd;
    protected java.lang.String category;
    protected java.lang.String spec;
    protected java.lang.String[] email;
    protected boolean blockEmail;
    protected java.util.Date startTime;
    protected java.lang.String name;
    protected java.lang.String inputPath;
    protected java.lang.String outputPath;
    protected java.lang.String errorPath;
    protected boolean join;
    public void setRemoteCommand(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getRemoteCommand();
    public void setInputParameters(java.lang.String[]);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String[] getInputParameters();
    public void setJobSubmissionState(int);
       throws org.ggf.drmaa.DRMAAException
    public int getJobSubmissionState();
    public void setJobEnvironment(java.util.Properties);
       throws org.ggf.drmaa.DRMAAException
    public java.util.Properties getJobEnvironment();
    public void setWorkingDirectory(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getWorkingDirectory();
    public void setJobCategory(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getJobCategory();
    public void setNativeSpecification(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getNativeSpecification();
    public void setEmailAddresses(java.lang.String[]);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String[] getEmailAddresses();
    public void setBlockEmail(boolean);
       throws org.ggf.drmaa.DRMAAException
    public boolean getBlockEmail();
    public void setStartTime(java.util.Date);
       throws org.ggf.drmaa.DRMAAException
    public java.util.Date getStartTime();
    public void setJobName(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getJobName();
    public void setInputPath(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getInputPath();
    public void setOutputPath(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getOutputPath();
    public void setErrorPath(java.lang.String);
       throws org.ggf.drmaa.DRMAAException
    public java.lang.String getErrorPath();
    public void setJoinFiles(boolean);
       throws org.ggf.drmaa.DRMAAException
    public boolean getJoinFiles();
    public void setTransferFiles(byte);
       throws org.ggf.drmaa.DRMAAException
    public byte getTransferFiles();
    public void setDeadlineTime(java.util.Date);
       throws org.ggf.drmaa.DRMAAException
    public java.util.Date getDeadlineTime();
```

```
    public void setHardWallclockTimeLimit(long);
       throws org.ggf.drmaa.DRMAAException
    public long getHardWallclockTimeLimit();
    public void setSoftWallClockTimeLimit(long);
       throws org.ggf.drmaa.DRMAAException
    public long getSoftWallClockTimeLimit();
    public void setHardRunDurationLimit(long);
       throws org.ggf.drmaa.DRMAAException
    public long getHardRunDurationLimit();
    public void setSoftRunDurationLimit(long);
       throws org.ggf.drmaa.DRMAAException
    public long getSoftRunDurationLimit();
    protected com.sun.grid.drmaa.JobTemplate()
    public abstract java.util.List getAttributeNames()
    public abstract void delete()
       throws com.sun.grid.drmaa.DRMAAException
    protected java.util.List getOptionalAttributeNames();
}
```

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 Hierarchy

All exception in the Java language binding inherits 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 hierarchy is:

- *java.lang.Object*
  - *java.lang.Throwable*
    - *java.lang.Exception*
      - *org.ggf.drmaa.DRMAAException*
        - *org.ggf.drmaa.AuthorizationException*
        - *org.ggf.drmaa.DefaultContactStringException*
        - *org.ggf.drmaa.DeniedByDRMException*
        - *org.ggf.drmaa.DRMCommunicationException*
        - *org.ggf.drmaa.ExitTimeoutException*
        - *org.ggf.drmaa.InconsistentStateException*
        - *org.ggf.drmaa.InternalException*
        - *org.ggf.drmaa.InvalidArgumentException*
        - *org.ggf.drmaa.InvalidAttributeException*
          - *org.ggf.drmaa.ConflictingAttributeValuesException*
          - *org.ggf.drmaa.InvalidAttributeFormatException*
          - *org.ggf.drmaa.InvalidAttributeValueException*
        - *org.ggf.drmaa.InvalidContactStringException*
        - *org.ggf.drmaa.InvalidJobException*
        - *org.ggf.drmaa.NoResourceUsageDataException*
        - *org.ggf.drmaa.SessionException*
          - *org.ggf.drmaa.DRMSExitException*
          - *org.ggf.drmaa.NoActiveSessionException*
          - *org.ggf.drmaa.SessionAlreadyActiveException*
        - *org.ggf.drmaa.TryLaterException*
      - *java.lang.RuntimeException*
        - *org.ggf.drmaa.UnsupportedAttributeException*

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 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.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.setWorkingDirectory ("$drmaa_hd_pd$");
    jt.setRemoteCommand (jobPath);
    jt.setInputParameters (new String[] {
                                          Integer.toString (seconds)
                                        });
    jt.setJoinFiles (true);

    if (!isBulkJob) {
      jt.setOutputPath ("$drmaa_hd_pd$/DRMAA_JOB");
    }
    else {
      jt.setOutputPath ("$drmaa_hd_pd$/DRMAA_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

9 Full Copyright Notice