GWD-R
Distributed Resource Management
Application API (DRMAA) Working Group

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

Distributed Resource Management Application API Java™ Language Bindings 0.5

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.

Table of Contents

Index of Tables

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 in mind, however it should be implementable 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 as an add-on package for the Java Development Kit™ 1.2.

3 Design Decisions

In order to make the Java language binding as familiar to programmers as possible, whenever reasonable, 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.
Additionally, to make the DRMAA API itself as universal as possible, the Java language binding adheres to a more general object-oriented approach distilled from the Java language binding and the .Net binding. In this manner, DRMAA API understanding should be directly transferable by developers from the Java language to .Net and vice versa.

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.
A service provider wishing to implement a DRMAA binding for a specific DRM system must provide implementations of three classes. The SessionFactory and JobInfo classes are abstract and must be extended by a service provider implementation. The Session interface must also be implemented by a service provider implementation. The JobTemplate class may also be optionally extended by a service provider implementation, but is not required.
In order for the SessionFactory.getFactory() class to find the service provider implementation, the implementation must set the system variable *org.ggf.drmaa.SessionFactory* to the name of the class which extends the org.ggf.drmaa.SessionFactory class. This property is used by the org.ggf.drmaa.SessionFactory class to locate and instantiate the service provider's SessionFactory 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 property setter and getter for each DRMAA attribute. The advantage is that the property 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	args	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	email	java.lang.String[]
drmaa_block_email	BlockEmail	boolean
drmaa_start_time	startTime	org.ggf.drmaa.PartialTimestamp
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	org.ggf.drmaa.FileTransferMode
drmaa_deadline_time	deadlineTime	org.ggf.drmaa.PartialTimestamp
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 JavaBean™ 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 org.ggf.drmaa.FileTransferMode, the data is copied so that the data structure stored by the Java language binding is uncoupled from the data structure in the calling application.
Optional attributes are also represented by setters and getters. The Java binding implementation must provide implementations for setters and getters for all DRMAA attributes, both required and optional. The setter and getter implementations for optional attributes must throw org.ggf.drmaa.UnsupportedAttributeException. The service provider implementation should 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 Session Class

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

```
public abstract interface com.sun.grid.drmaa.Session {
    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.lang.String 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 USER_SYSTEM_SUSPENDED
    public static final int DONE
    public static final int FAILED
    protected org.ggf.drmaa.Session()
    public abstract void init(java.lang.String contactString)
       throws org.ggf.drmaa.DrmaaException
    public abstract void exit()
       throws org.ggf.drmaa.DrmaaException
    public abstract org.ggf.drmaa.JobTemplate
       createJobTemplate()
       throws org.ggf.drmaa.DrmaaException
    public abstract void
       deleteJobTemplate(org.ggf.drmaa.JobTemplate jobTemplate)
       throws org.ggf.drmaa.DrmaaException
    public abstract java.lang.String
       runJob(org.ggf.drmaa.JobTemplate jobTemplate)
       throws org.ggf.drmaa.DrmaaException
    public abstract java.util.List
       runBulkJobs(org.ggf .drmaa.JobTemplate jobTemplate,
                   int beginIndex, int endIndex, int step)
       throws org.ggf.drmaa.DrmaaException
    public abstract void control(java.lang.String jobName,
                                 int operation)
       throws org.ggf.drmaa.DrmaaException
```

```
    public abstract void synchronize(java.util.List jobList,
                                     long timeout, boolean dispose)
       throws org.ggf.drmaa.DrmaaException
    public abstract org.ggf.drmaa.JobInfo
       wait(java.lang.String jobName, long timeout)
       throws org.ggf.drmaa.DrmaaException
    public abstract int getJobProgramStatus(java.lang.String jobName)
       throws org.ggf.drmaa.DrmaaException
    public abstract java.lang.String getContact()
    public abstract org.ggf.drmaa.Version getVersion()
    public abstract java.lang.String getDRMSInfo()
    public abstract java.lang.String getDRMAAImplementation()
}
```

4.2 The SessionFactory 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 Session interface. The SessionFactory class serves this purpose and additionally allows the vendor the freedom to return different Session implementations depending on the need. The structure of the SessionFactory class is as follows:

```
public abstract class com.sun.grid.drmaa.SessionFactory {
    public static com.sun.grid.drmaa.SessionFactory getFactory()
    public abstract com.sun.grid.drmaa.Session getSession()
}
```

It is likely that with a future version of this specification, the SessionFactory class will include a method to explicitly request a specific service provider implementation.

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 Session implementation. A DRMAA application gets a JobTemplate from the active Session, specifies in the JobTemplate any required job parameters, and the passes the JobTemplate back to the Session when requesting that a job be executed. When finished, the DRMAA application should call the `Session.deleteJobTemplate()` 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 org.ggf.drmaa.JobTemplate {
    public static final int HOLD
    public static final int ACTIVE
    public static final int HOME_DIRECTORY
    public static final int WORKING_DIRECTORY
    public static final int PARAMETRIC_INDEX
    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
protected com.sun.grid.drmaa.JobTemplate()
public void setRemoteCommand(java.lang.String command)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getRemoteCommand()
public void setArgs(java.lang.String[] args)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String[] getArgs()
public void setJobSubmissionState(int state)
   throws org.ggf.drmaa.DrmaaException
public int getJobSubmissionState()
public void setJobEnvironment(java.util.Properties env)
   throws org.ggf.drmaa.DrmaaException
public java.util.Properties getJobEnvironment()
public void setWorkingDirectory(java.lang.String wd)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getWorkingDirectory()
public void setJobCategory(java.lang.String category)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getJobCategory()
public void setNativeSpecification(java.lang.String spec)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getNativeSpecification()
public void setEmail(java.lang.String[] email)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String[] getEmail()
public void setBlockEmail(boolean blockEmail)
   throws org.ggf.drmaa.DrmaaException
public boolean getBlockEmail()
public void setStartTime(org.ggf.drmaa.PartialTimestamp startTime)
   throws org.ggf.drmaa.DrmaaException
public org.ggf.drmaa.PartialTimestamp getStartTime()
public void setJobName(java.lang.String name)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getJobName()
public void setInputPath(java.lang.String inputPath)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getInputPath()
public void setOutputPath(java.lang.String outputPath)
   throws org.ggf.drmaa.DrmaaException
public java.lang.String getOutputPath()
public void setErrorPath(java.lang.String errorPath)
   throws org.ggf.drmaa.DrmaaException
```

```
    public java.lang.String getErrorPath()
    public void setJoinFiles(boolean joinFiles)
       throws org.ggf.drmaa.DrmaaException
    public boolean getJoinFiles()
    public void setTransferFiles(org.ggf.drmaa.FileTransferMode mode)
       throws org.ggf.drmaa.DrmaaException
    public org.ggf.drmaa.FileTransferMode getTransferFiles()
    public void setDeadlineTime(org.ggf.drmaa.PartialTimestamp
       deadline) throws org.ggf.drmaa.DrmaaException
    public org.ggf.drmaa.PartialTimestamp getDeadlineTime()
    public void setHardWallclockTimeLimit(long limit)
       throws org.ggf.drmaa.DrmaaException
    public long getHardWallclockTimeLimit()
    public void setSoftWallClockTimeLimit(long limit)
       throws org.ggf.drmaa.DrmaaException
    public long getSoftWallClockTimeLimit()
    public void setHardRunDurationLimit(long limit)
       throws org.ggf.drmaa.DrmaaException
    public long getHardRunDurationLimit()
    public void setSoftRunDurationLimit(long limit)
       throws org.ggf.drmaa.DrmaaException
    public long getSoftRunDurationLimit()
    public abstract java.util.List getAttributeNames()
    protected java.util.List getOptionalAttributeNames()
    public java.lang.String toString()
}
```

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 org.ggf.drmaa.JobInfo
                implements java.io.Serializable {
    protected java.lang.String jobId
    protected int status
    protected java.util.Map resourceUsage
    protected org.ggf.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()
    public java.lang.Object clone()
    public java.lang.String toString()
}
```

4.5 The Exception Hierarchy

All exceptions in the Java language binding inherit from the DrmaaException or DrmaaRuntimeException classes . The structure of DrmaaException as as 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 message)
}
```

The structure of DrmaaRuntimeException is as follows:

```
public class com.sun.grid.drmaa.DrmaaRuntimeException
    extends java.lang.RuntimeException{
    public com.sun.grid.drmaa.DrmaaRuntimeException()
    public com.sun.grid.drmaa.DrmaaRuntimeException(java.lang.String
                                                    message)
}
```

The DRMAA exception hierarchy is as follows:

- *java.lang.Object*
  - *java.lang.Throwable*
    - *java.lang.Exception*
      - *org.ggf.drmaa.DrmaaException**
        - *org.ggf.drmaa.AuthorizationException*
        - *org.ggf.drmaa.InvalidContactStringException*
          - *org.ggf.drmaa.DefaultContactStringException*
          - *org.ggf.drmaa.NoDefaultContactStringException*
        - *org.ggf.drmaa.DeniedByDrmException*
        - *org.ggf.drmaa.DrmCommunicationException*
        - *org.ggf.drmaa.DrmsExitException*
        - *org.ggf.drmaa.InconsistentStateException**
          - *org.ggf.drmaa.HoldInconsistentStateException*
          - *org.ggf.drmaa.ReleaseInconsistentStateException*
          - *org.ggf.drmaa.ResumeInconsistentStateException*
          - *org.ggf.drmaa.SuspendInconsistentStateException*
        - *org.ggf.drmaa.DrmsInitException*
        - *org.ggf.drmaa.InvalidArgumentException*
          - *org.ggf.drmaa.InvalidJobException*
        - *org.ggf.drmaa.InvalidAttributeException**
          - *org.ggf.drmaa.ConflictingAttributeValuesException*
          - *org.ggf.drmaa.InvalidAttributeFormatException*
          - *org.ggf.drmaa.InvalidAttributeValueException*
        - *org.ggf.drmaa.NoResourceUsageException*
        - *org.ggf.drmaa.ExitTimeoutException*
        - *org.ggf.drmaa.SessionException**
          - *org.ggf.drmaa.NoActiveSessionException*
          - *org.ggf.drmaa.AlreadyActiveSessionException*

        - *org.ggf.drmaa.TryLaterException*
    - *java.lang.RuntimeException*
        - *org.ggf.drmaa.DrmaaRuntimeException**
            - *org.ggf.drmaa.InternalException*
            - *org.ggf.drmaa.UnsupportedAttributeException*
            - *org.ggf.drmaa.InvalidJobTemplateException*

Exceptions marked by an asterisk exist only for behavior aggregation and have been declared as abstract.
All exceptions under the DrmaaException and DrmaaRuntimException classes 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 message)
}
```

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

4.6 Utility Classes

The DRMAA Java language binding is supported by three utility classes.

4.6.1 The PartialTimestamp Class

The PartialTimestamp class is used by the JobTemplate class to represent partially specified time stamps, as required by the DRMAA language independent specification, version 1.0. The PartialTimestamp class inherits its methods from the java.util.Calendar class, overriding the abstract methods to implement DRMAA-specific behavior. For more information, see the JavaDoc documentation for the java.util.Calendar class.
The PartialTimestamp class adds a new field to the list of fields defined by java.util.Calendar. The CENTURY field represents all but the last two digits of the year, and replaces the ERA field of java.util.Calendar. The YEAR field is changed to represent only the last two digits of the year.
The PartialTimestamp class adds two new methods to those defined by java.util.Calendar. The getModifier() method returns any modifiers pending for a field. Modifiers represent values to be added to unset fields when the PartialTimestamp is resolved to a specific time. The setModifier() is used to explicitly set the modifier for a specific field. Calling the add() method on an unset field is equivalent to calling setModifier() with the same parameters.
The PartialTimestamp class also adds a new constant, UNSET. UNSET is the value assigned to a field before the field is set. Any time that get() returns UNSET, isSet() will return false. The reverse is not necessarily true, however, as fields may be internally assigned values even though the fields haven't been explicitly set.
The structure of the PartialTimestamp class is as follows:

```
public class org.ggf.drmaa.PartialTimestamp
       extends java.util.Calendar {
    public static final int CENTURY;
    public static final int UNSET;
    protected org.ggf.drmaa.PartialTimestamp ()
    protected org.ggf.drmaa.PartialTimestamp (java.util.TimeZone tz)
```

```
    protected org.ggf.drmaa.PartialTimestamp (java.util.Locale locale)
    protected org.ggf.drmaa.PartialTimestamp (java.util.TimeZone tz,
                                              java.util.Locale locale)
    protected org.ggf.drmaa.PartialTimestamp (int year, int month,
                                              int date, int hour,
                                              int minute)
    protected org.ggf.drmaa.PartialTimestamp (int hour, int minute,
                                              int second)
    protected org.ggf.drmaa.PartialTimestamp (int year, int month,
                                              int date, int hour,
                                              int minute, int second)
    public int getModifier(int field)
    public void setModifier(int field, int value)
    protected void computeFields()
    protected void computeTime()
    public void add(int field, int value)
    public void roll(int field, boolean up)
    public int getMinimum(int field)
    public int getMaximum(int field)
    public int getGreatestMinimum(int field)
    public int getLeastMaximum(int field)
    public java.lang.Object clone()
    public java.lang.String toString()
    public boolean equals(java.lang.Object obj)
    public int hashCode()
}
```

4.6.2 The FileTransferMode Class

The FileTransferMode class is used by the JobTemplate class to indicate the value for the transferFiles property. The class has three properties which determine which streams will be staged in or out. See the drmaa_transfer_files attribute in the DRMAA language independent specification, version 1.0 for more information. The structure of the FileTransferMode class is as follows:

```
public class org.ggf.drmaa.FileTransferMode
       implements java.io.Serializable {
    public org.ggf.drmaa.FileTransferMode()
    public org.ggf.drmaa.FileTransferMode(boolean, boolean, boolean)
    public void setInputStream(boolean)
    public boolean getInputStream()
    public void setOutputStream(boolean)
    public boolean getOutputStream()
    public void setErrorStream(boolean)
    public boolean getErrorStream()
    public java.lang.Object clone()
    public java.lang.String toString()
    public boolean equals(java.lang.Object)
    public int hashCode()
}
```

4.6.3 The Version Class

The Version class is a holding class for the major and minor version numbers of the DRMAA implementation as returned by the `Session.getVersion()` method. The class structure follows:

```
public class org.ggf.drmaa.Version
       implements java.io.Serializable {
    public org.ggf.drmaa.Version(int, int)
    public int getMajor()
    public int getMinor()
    public java.lang.Object clone()
    public java.lang.String toString()
    public boolean equals(java.lang.Object)
    public int hashCode()
}
```

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 Session.synchronize() is used to synchronize with all jobs to finish. Finally Session.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 remoteCommand property. The Java language binding example passes “5” as first argument to the job template args property. 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 16 jobs being run that sleep for 5 seconds each before finishing.

The source code follows:

```
import java.util.*;

import org.ggf.drmaa.*;

public class DrmaaExample {
   public static void main (String[] args) {
      SessionFactory factory = SessionFactory.getFactory ();
      Session session = factory.getSession ();

      try {
         session.init (null);
         JobTemplate jt = session.createJobTemplate ();
         jt.setRemoteCommand (args[0]);
         jt.setArgs (new String[] {"5"});

         int start = 1;
         int end  = 30;
         int step = 2;
```

```
List ids = session.runBulkJobs (jt, start, end, step);

ids.add (session.runJob (jt));

Iterator i = ids.iterator ();

while (i.hasNext ()) {
   System.out.println ("Your job has been submitted with id " +
                       i.next ());
}

session.deleteJobTemplate (jt);
session.synchronize (Collections.singletonList
                        (Session.JOB_IDS_SESSION_ALL),
                     Session.TIMEOUT_WAIT_FOREVER, false);

for (int count = start; count < end; count += step) {
   JobInfo info = session.wait (Session.JOB_IDS_SESSION_ANY,
                                Session.TIMEOUT_WAIT_FOREVER);

   if (info.wasAborted ()) {
      System.out.println ("Job " + info.getJobId () +
                          " never ran");
   }
   else if (info.hasExited ()) {
      System.out.println ("Job " + info.getJobId () +
      " finished regularly with exit status " +
      info.getExitStatus ());
   }
   else if (info.hasSignaled ()) {
      System.out.println ("Job " + info.getJobId () +
      " finished due to signal " +
      info.getTerminatingSignal ());
   }
   else {
      System.out.println ("Job " + info.getJobId () +
      " finished with unclear conditions");
   }

   System.out.println ("Job Usage:");

   Map rmap = info.getResourceUsage ();
   Iterator r = rmap.keySet ().iterator ();

   while (r.hasNext ()) {
      String name = (String)r.next ();
      String value = (String)rmap.get (name);

      System.out.println ("  " + name + "=" + value);
   }
}
```

```
            session.exit ();
         }
         catch (DrmaaException e) {
            System.out.println ("Error: " + e.getMessage ());
         }
      }
}
```

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 of 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

Peter Tröger
peter.troeger@hpi.uni-potsdam.de

Universität Potsdam
Am Neuen Palais 10
14469 Potsdam
Germany

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