GGF12: DRMAA Tutorial C and Java Language Bindings

Daniel Templeton
Senior Staff Engineer
Sun Microsystems GmbH

Agenda

- Architecture Overview
- API Overview
  - Session
  - Job Execution
  - Job Synchronization
  - Job Control & Monitoring
- Bindings
  - C Binding
  - Java™ Language Binding

Architectural Overview

- API for job submission and control
  - Keep it simple
    - Keep the number of functions down
    - Avoid tough issues
  - Address the areas of agreement
  - Leave room for areas of disagreement

Architectural Overview

- Implementable by any* DRM in any* language
  - N1™ Grid Engine – C, Perl, & Java language bindings
  - Condor – C binding, Perl binding?
  - Globus – C binding?

* OK. Not really, but we tried to make it as universal as possible

Design & Impl Considerations

- Implementations should be modular
  - Pluggable implementations
- Implementations should be multi-threaded
  - Handle synchronization like UNIX
- Single session per instance
  - Used for tracking jobs
  - Can only init and exit
- No user handling
  - Implies a single user per instance

Applications

- Bindings may be either client- or server-side
  - N1GE bindings use the facilities available to the client-side utilities, such as qsub
    - N1GE DRMAA bindings can only be used from submit hosts
    - DRMAA uses N1GE communications mechanisms
  - Web Service binding would be server side
    - Client would only need the WSDL for the service
  - Globus binding could be either
    - Client-side library
      - Uses web services to talk to Globus
    - Server-side web service hosted within Globus

API Overview

- All APIs based on an über spec
  - Language independent binding – version 1.0
  - C language binding – version 0.9.5
  - Java language binding – version 0.4.2
  - .Net language binding – version 0.2
  - Unspecified Perl binding
- Binding specs don't all agree 100%
  - Some things don't make sense in some languages
  - Original, language independent spec has a C slant
- Attempt to agree as much as possible
  - Reconciling .Net and Java language bindings

Session Handling

- Only two things to do with a session:
  - Init
    - Creates a session for the instance
    - Takes a string which specifies to server instance
      - More info on next slide
  - Exit
    - Destroys a session
  - Session exists from init to exit
    - Used by other calls to identify jobs
  - Exit should always be called
    - Allows implementation to do necessary cleanup

Instance Handling

- Get Contact
  - Before init, returns a list of available server instances
    - Items in the list can be passed to init
  - After init, returns the bound server instance
- Get DRM System
  - Before init, returns a list of available DRMs
  - After init, returns the DRM associated with the bound server instance

Instance Handling

- Get DRMAA Implementation
  - Before init, returns a list of available DRMAA implementations
  - After init, returns the bound implementation
- Get Version
  - Returns the major and minor version for the bound implementation
  - May only be called after init

Job Description

- Encapsulated in a job template
  - Properties sheet with predefined properties
  - Functions provided for accessing properties
    - Set Attribute, Get Attribute, Set Vector Attribute, Get Vector Attribute
      - List of defined Property names
    - Java language binding and uses JavaBean™ property accessors instead
      - e.g. setWorkingDirectory(), getWorkingDirectory()
    - .Net language binding offers both
    - Open debate

Job Templates

- Job Template is independent of jobs
  - 1:n relationship
  - Static – not updated during job execution
- Allocate Job Template
  - Creates a new job template
- Delete Job Template
  - Frees an allocated job template

Job Attributes: Required

- Must be implemented by all implementations
- Remote Command to Execute
  - The executable to be run as the job
- Input Parameteres
  - The args to be passed to the remote command
- Job State at Submission
  - Whether the job should be started in a suspended or runnable state
- Job Environment
  - The environment variable settings for the job

Job Attributes: Required

- Job Working Directory
  - The directory where the remote command should be executed
- Job Start Time
  - The earliest time that the job may run
  - Not a deadline
- Job Name
  - The name to be assigned to the job
- Input/Output/Error Stream
  - The input/output/error stream path

Job Attributes: Required

- Input/Output/Error Stream
  - The input/output/error stream path
- Join Files
  - Whether to attach (dup2) the error stream to the output stream

Job Attributes: Required

- Email Address
  - The address to which to send email
  - Email is only sent when the underlying DRM decides to
    - i.e. The DRM's settings control when email is sent
- Email Suppression
  - Whether to prevent the sending of email
  - Can only prevent email from being sent
    - Cannot force email to be sent if DRM settings say not to

Job Attributes: Required

- Two catch-all attributes
- Native Specification
  - Allows a user to specify settings not allowed by DRMAA
    - For example, in N1GE, -ckpt for checkpointing
- Job Category
  - Allows an administrator to assign default DRM settings to groups of jobs
  - Not true job category
    - Reference into DRM specific settings
    - “Dissolves” at submission time

Job Attributes: Optional

- Not required of every implementation
- Implementations may also implement other attributes
- Transfer Files
  - Whether to treat the input, output, and error paths as locations for file staging
- Absolute Job Termination Time
  - Deadline after which the DRM will terminate the job

Job Attributes: Optional

- Wall Clock Time Limit
  - The amount of wall clock time the job is allowed to execute before being terminated
- Soft Wall Clock Time Limit
  - The amount of wall clock time the job is expected to need to execute
- Job Run Duration Hlimit
  - The amount of CPU time the job is allowed to use before being terminated
- Job Run Duration Slimit
  - The amount of CPU time the job is expected to

Job Execution

- Two ways to run a job:
- Run Job
  - Runs a single job based on the parameters in the supplied job template
  - Returns an opaque job id
- Run Bulk Jobs
  - Runs a number of jobs all based on the same job template
    - AKA a parametric job
    - Number calculated from start, end, and increment
  - Returns a list of opaque job ids
    - More about this with the C language binding
- e.g, equivalent to “qsub [-t start-end:increment]”

Job Synchronization

- Wait
  - Waits for a single job to complete
  - Can wait for any, i.e. the next, job to complete
  - Returns information about the job's execution
    - Exit status
    - Resource usage
      - List of “name=value” strings
    - Other terminating info
  - Can only wait for a job once
    - Wait reaps the job's exit info
    - Additional waits return an error

Job Synchronization

- Synchronize
  - Waits for all jobs in a list to complete
  - Can wait for all jobs in the session to complete
    - All jobs submited before the synchronize
  - Does not return exit information
    - May reap the exited jobs' exit info – dispose parameter
    - Can be used in conjuction with Wait to get exit info
      - e.g synchronize on the id list, then wait for each id in the list

Finish Status

- Finish status is an opaque value interpreted by:
  - If Exited
    - Whether the job exited normally
  - Exit Status
    - The jobs exit code
  - If Signaled
    - If the job exited due to a signal
  - Terminating Signal
    - The terminating signal
  - If Core Dumped
    - If the job created a core dump file
  - If Aborted
    - If the job was terminated abnormally

Job Control

- Control
  - Works on a single job id or all jobs
  - Can:
    - Terminate
    - Suspend
    - Resume
    - Hold
    - Release
  - May return before action completes

Job Monitoring

- Job PS
  - Returns the status of a particular job:
    - Queued & Active
    - Running
    - System Hold
    - User Hold
    - User & System Hold
    - System Suspended
    - User Suspended
    - User & System Suspended
    - Done
    - Failed
    - Undetermined

Bindings

- C language Binding
  - Very close to platform-independent spec
  - Should use shared libraries
    - Modular C for plugability
- Java language Binding
  - Strays from platform-independent spec a little
    - OO implementation requires different prespective
      - JavaBeans properties
    - Java language provides some things for free
      - Strings
      - Exception handling
      - Collections

The rest of the presentation is intended for programmers

C Language Binding

- 0.9.5 is the current C language binding spec
  - Likely final
- Implemented by Sun™ (N1 Grid Engine 6.0) and Condor (Condor 6.7)

Error Handling

- Every function returns an error code
  - DRMAA_ERRNO_SUCCESS – good
  - Everything else – varying degrees of bad
- Most take an error buffer parameter
  - Contains error message on error
- Actually specified in platform independent spec
  - Belongs in C binding spec
- Data is returned from functions via pointers

Buffer Lengths

- DRMAA_ERROR_STRING_BUFFER
  - Maximum error string length
- DRMAA_JOBNAME_BUFFER
  - Maximum job name length
- DRMAA_SIGNAL_BUFFER
  - Maximum signal name length
- DRMAA_ATTR_BUFFER
  - Maximum attribute name length
- DRMAA_CONTACT_BUFFER
  - Maximum contact string length
- DRMAA_DRM_SYSTEM_BUFFER
  - Maximum DRM system name length

Session Handling

- int drmaa_init (
  ```
      const char *contact,
      char *error_diagnosis,
      size_t error_diag_len
  );
  ```
- int drmaa_exit (
  ```
      char *error_diagnosis,
      size_t error_diag_len
  );
  ```
- Idiom: signal handler thread that calls drmaa_exit() on SIGINT.

Example: Session Handling

```
char error[DRMAA_ERROR_STRING_BUFFER];
int errnum = 0;

errnum = drmaa_init (argv[0], error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't init DRMAA library: %s\n", error);
   return 1;
}

/* Do Stuff */

errnum = drmaa_exit (error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't exit DRMAA library: %s\n", error);
   return 1;
}

return 0;
```

Instance Handling

- int drmaa_get_contact (
      char *contact, size_t contact_len,
      char * error_diagnosis, size_t error_diag_len
  );
- int drmaa_get_DRM_system (
      char *drm_system, size_t drm_system_len,
      char * error_diagnosis, size_t error_diag_len
  );
- int drmaa_get_DRMAA_implementation (
      char *drmaa_impl, size_t drmaa_impl_len,
      char * error_diagnosis, size_t error_diag_len
  );

Instance Handling

- int drmaa_version (
      unsigned int *major,
      unsigned int *minor,
      char * error_diagnosis,
      size_t error_diag_len
  );

Example: Instance Handling

```
char contact[DRMAA_CONTACT_BUFFER];
char drm_system[DRMAA_DRM_SYSTEM_BUFFER];
char drmaa_impl[DRMAA_DRM_SYSTEM_BUFFER];
unsigned int major = 0;
unsigned int minor = 0;

drmaa_get_contact (contact, DRMAA_CONTACT_BUFFER, NULL,
   0);
drmaa_get_DRM_system (drm_system, DRMAA_DRM_SYSTEM_BUFFER,
                      NULL, 0);
drmaa_get_DRMAA_implementation (drm_system,
                                DRMAA_DRM_SYSTEM_BUFFER,
                                NULL, 0);
drmaa_version (&major, &minor, NULL, 0);

printf (“Contact: %s\n”, contact);
printf (“DRM System: %s\n”, drm_system);
printf (“DRMAA Implementation: %s\n”, drmaa_impl);
printf (“Version: %d.%d\n”, major, minor);
```

Job Templates

- int drmaa_allocate_job_template (
      drmaa_job_template_t **jt,
      char *error_diagnosis,
      size_t error_diag_len
  );
- int drmaa_delete_job_template (
      drmaa_job_template_t *jt,
      char *error_diagnosis,
      size_t error_diag_len
  );

Example: Job Templates

```
char error[DRMAA_ERROR_STRING_BUFFER];
int errnum = 0;
drmaa_job_template_t *jt = NULL;

/* Init Session */

errnum = drmaa_allocate_job_template (&jt, error,
                                      DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't allocate job template: %s\n", error);
   return 1;
}

/* Run Job */

errnum = drmaa_delete_job_template (jt, error,
                                    DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't delete job template: %s\n", error);
   return 1;
}

/* Exit Session */
```

Scalar Job Attributes

- int drmaa_set_attribute (
        drmaa_job_template_t *jt,
        const char *name,
        const char *value,
        char * error_diagnosis,
        size_t error_diag_len
  );
- int drmaa_get_attribute (
        drmaa_job_template_t *jt,
        const char *name,
        char *value,
        char * error_diagnosis,
        size_t error_diag_len
  );

Vector Job Attributes

-
  ```
  int drmaa_set_vector_attribute (
      drmaa_job_template_t *jt,
      const char *name,
      const char *value[], /* NULL-terminated */
      char * error_diagnosis,
      size_t error_diag_len
  );
  ```

-
  ```
  int drmaa_get_vector_attribute (
      drmaa_job_template_t *jt,
      const char *name,
      drmaa_attr_values_t **values,
      char * error_diagnosis,
      size_t error_diag_len
  );
  ```

Attribute Values

- drmaa_attr_values_t type
  - Implementation independent vector
- int drmaa_get_next_attr_value (

  ```
      drmaa_attr_values_t *values,
      char *values,
      int value_len
  );
  ```
- int drmaa_release_attr_values (

  ```
      drmaa_attr_values_t *values
  );
  ```

Example: Job Attributes

```
char error[DRMAA_ERROR_STRING_BUFFER];
int errnum = 0;
drmaa_job_template_t *jt = NULL;

/* Init Session & Allocate Job Template */

errnum = drmaa_set_attribute (jt, DRMAA_REMOTE_COMMAND, "sleeper.sh",
                              error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't set remote command: %s\n", error);
   return 1;
}

errnum = drmaa_set_vector_attribute (jt, DRMAA_V_ARGV, argv, error,
                                     DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't set remote command args: %s\n", error);
   return 1;
}

/* Run Job, Delete Job Template, & Exit Session */
```

Job Execution

-
  ```
  int drmaa_run_job (
      char *job_id,
      size_t job_id_len,
      drmaa_job_template_t *jt,
      char *error_diagnosis,
      size_t error_diag_len
  );
  ```

-
  ```
  int drmaa_run_bulk_jobs (
      drmaa_job_ids_t **jobids,
      drmaa_job_template_t *jt,
      int start,
      int end,
      int increment,
      char *error_diagnosis,
      size_t error_diag_len
  );
  ```

Job IDs

- drmaa_job_ids_t type
  - Implementation independent vector
- int drmaa_get_next_job_id (
      drmaa_job_ids_t *values,
      char *value,
      int value_len
  );
- int drmaa_release_job_ids (
      drmaa_job_ids_t *values
  );

Example: Job Execution

```
char error[DRMAA_ERROR_STRING_BUFFER];
char jobid[DRMAA_JOBNAME_BUFFER];
int errnum = 0;
drmaa_job_template_t *jt = NULL;

/* Init Session, Allocate Job Template, & Set Attributes */

errnum = drmaa_run_job (jobid, DRMAA_JOBNAME_BUFFER, jt,
                        error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't run job: %s\n", error);
   return 1;
}
else {
   printf ("Your job has been submitted with id %s\n", jobid);
}

/* Delete Job Template, & Exit Session */
```

Example: Bulk Job Execution

```
char error[DRMAA_ERROR_STRING_BUFFER];
char jobid[DRMAA_JOBNAME_BUFFER];
int errnum = 0;
drmaa_job_template_t *jt = NULL;
drmaa_job_ids_t *ids = NULL;

/* Init Session, Allocate Job Template, & Set Attributes */

errnum = drmaa_run_bulk_jobs (&ids, jt, 1, 30, 2, error,
                              DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't run bulk jobs: %s\n", error);
   return 1;
}
else {
   while (drmaa_get_next_job_id (ids, jobid, DRMAA_JOBNAME_BUFFER) ==
          DRMAA_ERRNO_SUCCESS) {
      printf ("A job task has been submitted with id %s\n", jobid);
   }

   if (drmaa_release_job_ids (ids) != DRMAA_ERRNO_SUCCESS) ...
}

/* Delete Job Template, & Exit Session */
```

Job Synchronization

- int drmaa_wait (

```
        const char *job_id,
        char *job_id_out,
        size_t job_id_len,
        int *stat,
        signed long timeout,
        drmaa_attr_values_t **rusage,
        char *error_diagnosis,
        size_t error_diag_len
);
```

Bulk Job Synchronization

- int drmaa_synchronize (

```
        const char *job_ids[],
        signed long timeout,
        int dispose,
        char *error_diagnosis,
        size_t error_diag_len
);
```

- job_ids must be NULL-terminated

Example: Job Synchronization

```
char error[DRMAA_ERROR_STRING_BUFFER];
char jobid[DRMAA_JOBNAME_BUFFER];
char jobid_out[DRMAA_JOBNAME_BUFFER];
int errnum = 0;
int status = 0;
drmaa_attr_values_t *rusage = NULL;

/* Init Session, Allocate Job Template, Set Attributes, & Run Job */

errnum = drmaa_wait (jobid, jobid_out, DRMAA_JOBNAME_BUFFER, &status,
                     DRMAA_TIMEOUT_WAIT_FOREVER, &rusage, error,
                     DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't wait for job: %s\n", error);
   return 1;
}
else {
   /* Print Job Finish Status */
}

/* Delete Job Template, & Exit Session */
```

Example: Bulk Job Synchronization

```
char error[DRMAA_ERROR_STRING_BUFFER];
char *jobids[2] = {DRMAA_JOB_IDS_SESSION_ALL, NULL}; /* Idiom */
int errnum = 0;

/* Init Session, Allocate Job Template,
   Set Attributes, & Run Jobs */

errnum = drmaa_synchronize (jobids, DRMAA_TIMEOUT_WAIT_FOREVER,
                             1, error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't wait for jobs: %s\n", error);
   return 1;
}

/* Delete Job Template, & Exit Session */
```

Example: Hybrid Job Synchronization

```
/* Init Session, Allocate Job Template, Set Attributes, & Run Jobs
   */

errnum = drmaa_synchronize (jobids, DRMAA_TIMEOUT_WAIT_FOREVER,
                            0, error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't wait for jobs: %s\n", error);
   return 1;
}else {
   for (count = start; count <= end; count += incr) {
      drmaa_wait (DRMAA_JOB_IDS_SESSION_ANY, jobid,
   DRMAA_JOBNAME_BUFFER,
                 &status, DRMAA_TIMEOUT_WAIT_FOREVER, &rusage,
   error,
                 DRMAA_ERROR_STRING_BUFFER);

      /* Print Job Finish Status */
   }
}

/* Delete Job Template, & Exit Session */
```

Finish Status

- int drmaa_wifexited (
  int *exited,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );
- int drmaa_wexitstatus (
  int *exit_status,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );
- int drmaa_wifaborted (
  int *aborted,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );
- int drmaa_wifsignaled (
  int *signaled,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );
- int drmaa_wtermsig (
  char *signal,
  size_t signal_len,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );
- int drmaa_wcoredump (
  int *core_dumped,
  int status,
  char *error_diagnosis,
  size_t error_diag_len
  );

Example: Finish Status

```
int aborted = 0;

drmaa_wifaborted (&aborted, status, NULL, 0);

if (aborted == 1)
   printf (“Job never ran\n”);
else {
   int exited = 0;

   drmaa_wifexited (&exited, status, NULL, 0);

   if (exited == 1) {
      int exit_status = 0;

      drmaa_wexitstatus (&exit_status, status, NULL, 0);
      printf (“Job exited with status %d\n”, exit_status);
   } else {
      int signaled = 0;

      drmaa_wifsignaled (&signaled, status, NULL, 0);

      if (signaled == 1) {
         char termsig[DRMAA_SIGNAL_BUFFER+1];

         drmaa_wtermsig (termsig, DRMAA_SIGNAL_BUFFER, NULL, 0);
         printf (“Job exited doe to signal: %s\n”, termsig);
      }
   }
}
```

Job Control

- int drmaa_control (
      const char *jobid,
      int action,
      char *error_diagnosis,
      size_t error_diag_len
  );
- Actions
  - DRMAA_CONTROL_SUSPEND
  - DRMAA_CONTROL_RESUME
  - DRMAA_CONTROL_HOLD
  - DRMAA_CONTROL_RELEASE
  - DRMAA_CONTROL_TERMINATE

Example: Job Control

```
char error[DRMAA_ERROR_STRING_BUFFER];
char jobid[DRMAA_JOBNAME_BUFFER];
int errnum = 0;

/* Init Session, Allocate Job Template,
   Set Attributes & Run Job */

errnum = drmaa_control (jobid, DRMAA_CONTROL_TERMINATE,
                        error, DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, "Couldn't delete job: %s\n", error);
   return 1;
}

/* Delete Job Template, & Exit Session */
```

Job Monitoring

- int drmaa_job_ps (
        const char *jobid,
        int *remote_ps,
        char *error_diagnosis,
        size_t error_diag_len
  );
- States
  - DRMAA_PS_QUEUED_ACTIVE
  - DRMAA_PS_RUNNING
  - DRMAA_PS_SYSTEM_ON_HOLD
  - DRMAA_PS_USER_ON_HOLD
  - DRMAA_PS_USER_SYSTEM_ON_HOLD
  - DRMAA_PS_SYSTEM_SUSPENDED
  - DRMAA_PS_USER_SUSPENDED
  - DRMAA_PS_DONE
  - DRMAA_PS_FAILED
  - DRMAA_PS_UNDETERMINED

Example: Job Monitoring

```
char error[DRMAA_ERROR_STRING_BUFFER];
char jobid[DRMAA_JOBNAME_BUFFER];
int errnum = 0;
int status = 0;

/* Init Session, Allocate Job Template,
   Set Attributes & Run Job */

errnum = drmaa_job_ps (jobid, &status, error,
                       DRMAA_ERROR_STRING_BUFFER);

if (errnum != DRMAA_ERRNO_SUCCESS) {
   fprintf (stderr, “Couldn't get job status: %s\n”, error);
   return 1;
}
else {
   /* Print Job Status */
}

/* Delete Job Template, & Exit Session */
```

Example: Compiling, Linking, & Running

- N1 Grid Engine 6.0
- Compiling
  - Include DRMAA header

    ```
    cc -o app.o -I$SGE_ROOT/include app.c
    ```
- Linking
  - Include DRMAA library in library path
    - LD_LIBRARY_PATH, SHLIB_PATH, et al
  - Link in DRMAA library

    ```
    cc -o app -ldrmaa app.o
    ```
- Running
  - Include DRMAA library in library path
    - LD_LIBRARY_PATH, SHLIB_PATH, et al

Java Language Binding

- 0.4.2 is the current spec
  - Close to final
  - May adapt to OO binding spec
- Implemented by Sun (N1 Grid Engine 6.0s1)
  - Built as layer on top of C binding
- Spec'ed using API/SPI model
  - Application Programming interface for using DRMAA
  - Service Provider Interface for building implementations
  - All implementations should be interchangable

Object Model

- DRMAASessionFactory
  - Creates a DRMAASession
  - Enables SPI
- DRMAASession
  - Job submission, control, and monitoring
- JobTemplate
  - Describes a job for submission
- JobInfo
  - Finish information from DRMAASession.wait()
- DRMAAException
  - Parent of Exception tree

Exception Handling

- Exceptions instead of error codes
  - Exceptions map roughly to error codes
    - InvalidAttributeException
    - InconsistentStateException
- Signatures declare DRMAAException
  - JDBC model
  - Can catch more specific Exceptions if needed

Session Handling

- DRMAASessionFactory.getFactory ()
  - Static method – returns a DRMAASessionFactory
- DRMAASessionFactory.getSession ()
  - Returns a DRMAASession
  - Still only single-session
- DRMAASession.init (String contact)
  - Initializes a session
- DRMAASession.exit ()
  - Destroys a session
  - Must be called if init() is called
- Idiom: Use a shutdown hook to call exit()

Example: Session Handling

```
DRMAASessionFactory factory = DRMAASessionFactory.getFactory ();
DRMAASession = factory.getSession ();

try {
   session.init (args[0]);
}
catch (DRMAAException e) {
   System.err.println ("Error: " + e.getMessage ());
}

System.addShutdownHook (new Runnable () {
   public void run () {
      try {
         session.exit ();
      }
      catch (DRMAAException e) {
         System.err.println ("Error exiting: " + e.getMessage ());
      }
   }
});

// Do Stuff
```

Instance Handling

- DRMAASession.getContact ()
  - Returns contact as a String
- DRMAASession.getDRMSystem ()
  - Returns DRM system as String
- DRMAASession.getDRMAAImplementation ()
  - Returns DRMAA implementation as a String
- DRMAASession.getVersion ()
  - Returns a DRMAASession.Version object
    - Contains major and minor version numbers as ints

Example: Instance Handling

```
DRMAASession session = null;
DRMAASession.Version version = null;

// Init Session

try {
   System.out.println ("Contact: " + session.getContact ());
   System.out.println ("DRM System: " + session.getDRMSystem ());
   System.out.println ("DRMAA Implementation: " +
                        session.getDRMAAImplementation ());

   version = session.getVersion ();

   System.out.println ("Version: " + version.major + "." +
                        version.minor);
}
catch (DRMAAException e) {
   System.err.println ("Error: " + e.getMessage ());
}

// Exit Session
```

Job Templates

- DRMAASession.createJobTemplate ()
  - Creates a new JobTemplate
- JobTemplate.delete ()
  - Deletes the JobTemplate
- JobTemplate.set<Property> (<type> value)
  - Sets the property
  - JobTemplate.setRemoteCommand (“sleeper.sh”)
- JobTemplate.get<Property> ()
  - Returns the property's current value
  - String cmd = JobTemplate.getRemoteCommand ();

Job Attributes

- boolean blockEmail
- String[] emailAddresses
- String errorPath
- String[] inputParameters
- String inputPath
- String jobCategory
- Properties jobEnvironment
- String jobName
- int jobSubmissionState
- boolean joinFiles
- String nativeSpecification
- String outputPath
- String remoteCommand
- Date startTime
- String workingDirectory

Optional

- Date deadlineTime
- long hardRunDurationLimit
- long hardWallclockTimeLimit
- long softRunDurationLimit
- long softWallclockTimeLimit
- byte transferFiles

Job Execution

- DRMAASession.runJob (JobTemplate jt)
  - Submits a job based on jt
  - Returns the job id as an opaque String
- DRMAASession.RunBulkJobs (JobTemplate jt, int start, int end, int increment)
  - Submits (end – start)/increment jobs based on jt
  - Returns the job ids as a List of opaque Strings

Example: Job Execution

```
DRMAASession session = DRMAASessionFactory.getFactory ().getSession ();
JobTemplate jt = null;

session.init ("/sge:default");
jt = session.createJobTemplate ();
jt.setRemoteCommand ("/sge/examples/jobs/sleeper.sh");
jt.setInputParameters (new String[] {"300"});
jt.setOutputPath ("/dev/null");
jt.setJoinFiles (true);

String jobId = session.runJob (jt);
System.out.println ("Single job id is " + jobId);

List jobIds = session.runBulkJob (jt, 1, 10, 1);
Iterator i = jobIds.iterator ();

while (i.hasNext ()) {
   jobId = (String)i.next ();
   System.out.println ("Bulk job id is " + jobId);
}

jt.delete ();
session.exit ();
```

Job Synchronization

- DRMAASession.wait
  (String jobId, long timeout)
  – Waits for the job with the given id to succeed or fail
  – Returns a JobInfo object for the job
- DRMAASession.synchronize
  (List jobIds, long timeout, boolean dispose)
  – Wait for all jobs with ids in the list to succeed or fail
  – Must be used in conjunction with wait() to get the JobInfo objects

Example: Job Synchronization

```
DRMAASessionFactory factory = DRMAASessionFactory.getFactory ();
DRMAASession session = factory.getSession ();
JobTemplate jt = null;

session.init (“/sge:default”);
jt = session.createJobTemplate ();
jt.setRemoteCommand (“/sge/examples/jobs/sleeper.sh”);
jt.setInputParameters (new String[] {“300”});
jt.setOutputPath (“/dev/null”);
jt.setJoinFiles (true);

String jobId = session.runJob (jt);
System.out.println (“Job “ + jobId + “ has been submitted”);

JobInfo info = session.wait (jobId,
                             DRMAASession.TIMEOUT_WAIT_FOREVER);
System.out.println (“Job “ + jobId + “ has finished”);

// Print JobInfo

// Delete Job Template & Exit Session
```

Example: Bulk Job Synchronization

```
DRMAASessionFactory factory = DRMAASessionFactory.getFactory ();
DRMAASession session = factory.getSession ();
JobTemplate jt = null;

session.init ("/sge:default");
jt = session.createJobTemplate ();
jt.setRemoteCommand ("/sge/examples/jobs/sleeper.sh");
jt.setInputParameters (new String[] {"300"});
jt.setOutputPath ("/dev/null");
jt.setJoinFiles (true);

List jobIds = session.runBulkJob (jt, 1, 10, 1);

System.out.println ("Jobs 1-10 submitted.");

session.synchronize (jobIds, DRMAASession.TIMEOUT_WAIT_FOREVER,
                     true);

System.out.println ("All jobs have completed.");

// Delete Job Template & Exit Session
```

Example: Hybrid Job Synchronization

```
DRMAASession session = null;
JobTemplate jt = null;

// Init Session, Create JobTemplate, & Set Attributes

List jobIds = session.runBulkJob (jt, 1, 10, 1);

System.out.println ("Jobs 1-10 submitted.");

session.synchronize (jobIds, 1000, false);

Iterator i = jobIds.iterator ();

while (i.hasNext ()) {
    JobInfo info = session.wait ((String)i.next (),
                                    DRMAASession.TIMEOUT_NO_WAIT);

    // Print JobInfo
}

// Delete Job Template & Exit Session
```

Finish Status

- JobInfo.hasExited ()
  - Returns whether the job has exited – boolean
- JobInfo.getExitStatus ()
  - Returns the job's exit status – int
- JobInfo.wasAborted ()
  - Returns whether the job terminated abnormally – boolean
- JobInfo.hasSignaled ()
  - Returns whether the job terminated due to a signal – boolean
- JobInfo.getTerminatingSignal ()
  - Returns the name of the terminating signal – String
- JobInfo.hasCoreDump ()
  - Returns whether the job created a core dump – boolean
- JobInfo.getResourceUsage ()
  - Returns the job's resource usage – Map

Example: Finish Status

```
JobInfo info = null;

// Init Session, Create Template, Set Attributes, Submit Job, &
// Wait For Job

if (info.wasAborted ()) {
   if (info.hasCoreDump ()) {
      System.out.println ("Job dumped core");
   }
   else {
      System.out.println ("Job crashed");
   }
}
else if (info.hasExited ()) {
   System.out.println ("Job exited with " + info.getExitStatus ());
}
else if (info.hasSignaled ()) {
   System.out.println ("Job got signal:" + info.getTerminatingSignal ());
}
else {
   System.out.println ("Job exit with unclear conditions");
}

// Delete Job Template & Exit Session
```

Example: Resource Usage

```
JobInfo info = null;

// Init Session, Create Template, Set Attributes,
// Submit Job, & Wait For Job

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

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

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

// Delete Job Template & Exit Session
```

Job Control

- DRMAASession.control (String jobId, int action)
  - Action may be:
    - DRMAASession.HOLD
    - DRMAASession.RELEASE
    - DRMAASession.SUSPEND
    - DRMAASession.RESUME
    - DRMAASession.TERMINATE

Example: Job Control

```
DRMAASessionFactory factory = DRMAASessionFactory.getFactory ();
DRMAASession session = factory.getSession ();
JobTemplate jt = null;

session.init ("/sge:default");
jt = session.createJobTemplate ();
jt.setRemoteCommand ("/sge/examples/jobs/sleeper.sh");
jt.setInputParameters (new String[] {"300"});
jt.setOutputPath ("/dev/null");
jt.setJoinFiles (true);

String jobId = session.runJob (jt);
System.out.println ("Job " + jobId + " has been submitted");

Thread.sleep (5000);
session.control (jobId, DRMAASession.TERMINATE);
System.out.println ("Job " + jobId + " has been deleted");

// Delete Job Template & Exit Session
```

Job Monitoring

- DRMAASession.getJobProgramStatus (String jobId)
  - Returns a status code as an int
    - DRMAASession.QUEUED_ACTIVE
    - DRMAASession.RUNNING
    - DRMAASession.SYSTEM_ON_HOLD
    - DRMAASession.USER_ON_HOLD
    - DRMAASession.USER_SYSTEM_ON_HOLD
    - DRMAASession.SYSTEM_SUSPENDED
    - DRMAASession.USER_SUSPENDED
    - DRMAASession.DONE
    - DRMAASession.FAILED
    - DRMAASession.UNDETERMINED

Example: Job Monitoring

```
// Init Session, Create Template, Set Attributes, & Submit Job

switch (session.getJobProgramStatus (jobId)) {
   case DRMAASession.UNDERTERMINED:
      System.out.println ("undetermined"); break;
   case DRMAASession.QUEUED_ACTIVE:
      System.out.println ("queued and active"); break;
   case DRMAASession.SYSTEM_ON_HOLD:
   case DRMAASession.USER_ON_HOLD:
   case DRMAASession.USER_SYSTEM_ON_HOLD:
      System.out.println ("queued and on hold"); break;
   case DRMAASession.RUNNING:
      System.out.println ("running"); break;
   case DRMAASession.SYSTEM_SUSPENDED:
   case DRMAASession.USER_SUSPENDED:
   case DRMAASession.USER_SYSTEM_SUSPENDED:
      System.out.println ("suspended"); break;
   case DRMAASession.DONE:
      System.out.println ("done"); break;
   case DRMAASession.FAILED:
      System.out.println ("failed"); break;
}


// Delete Job Template & Exit Session
```

Example: Compiling & Running

- N1 Grid Engine 6.0s1
- Compiling
  - Include drmaa.jar in the classpath

    ```
    javac -classpath \
    $SGE_ROOT/lib/drmaa.jar *.java
    ```
- Running
  - Include libdrmaa in the library path
    - LD_LIBRARY_PATH, SHLIB_PATH, et al
  - Include drmaa.jar in the classpath

    ```
    java -cp $SGE_ROOT/lib/drmaa.jar:. \
    MyApp
    ```

Additional Information

- DRMAA Working Group
  http://www.drmaa.org
  https://forge.gridforum.org/projects/drmaa-wg
- N1 Grid Engine
  http://www.sun.com/software/gridware
  http://gridengine.sunsource.net
  - Tutorials
    /project/gridengine/howto/drmaa.html
    /project/gridengine/howto/drmaa_java.html
- Condor
  http://www.cs.wisc.edu/condor
- GGF
  http://www.ggf.org

GGF12: DRMAA Tutorial C and Java Language Bindings

dan.templeton@sun.com