



Using JSDL for DRMAA Job Templates - A Short Analysis

Peter Tröger,
Hasso-Plattner-Institute (HPI) @ University of Potsdam

GGF 14 JSDL session
Chicago, June 30, 2005

DRMAA in Practice

- Multiple implementations since GFD-P-R 0.22 in 2004
 - Product implementation in Sun Grid Engine 6
 - C- and Java-binding implementation
 - Prototype in Condor 6.7 series
 - C-binding implementation
 - CPAN Perl DRMAA module (Tim Harsch)
 - On-top-of DRMAA C-library
 - GridWay DRMAA implementation
 - Allows DRMAA on-top-of Globus
 - Prototype for Globus 3 DRMAA job manager (HPI)
 - Based on DRMAA Perl implementation
- Tutorials, programming examples, test suites
 - <http://gridengine.sunsource.net>
 - <http://www.dcl.hpi.uni-potsdam.de/research/grid/drmaa>
 - GGF12 tutorial, JavaOne 05 tutorial materials

DRMAA “1.(>0)” IDL Spec

- Started work in early 2005
- Specification through standardized OMG Interface Definition Language (IDL)
 - Avoid C-centric interface description of DRMAA 1.0
 - Easier development of OO language bindings
 - Usage of IDL avoids wording issues (i.e. ,attribute‘ vs. ,property‘)
 - Allows for true language-independent description of namespaces, enumerations, constants, and time values
 - More detailed and consistent description text
 - More placeholders
 - More error codes
 - More possible job states

Next Steps

- Putting the DRMAA-IDL spec in the GGF document chain (GGF14 version is nearly final)
- Prerequisite for some announced activities
 - .NET-binding implementation (HPI)
 - On-top-of DRMAA C-library
 - Improved Condor C library
 - Join the efforts at <http://sf.net/projects/condor-ext>
 - Python binding specification
 - Maybe more implementations
- **JSDL for DRMAA job template attributes ?!?**
 - Might be useful in case of a DRMAA-SOAP binding
 - Breaks backward compatibility to DRMAA 1.0

Things That Match

DRMAA JobTemplate

JSDL

remoteCommand (string)

<POSIXApplication> <Executable>...

args (string array)

<POSIXApplication> <Argument>...

jobName (string)

<JobIdentification> <JobName>

inputPath ([hostname]:file_path)

<POSIXApplication> <Input>...

outputPath ([hostname]:file_path)

<POSIXApplication> <Output>...

errorPath ([hostname]:file_path)

<POSIXApplication> <Error>...

*transferFiles (transferInput,
transferOutput, transferError)*

Existence of <DataStaging> element

Things We Have To Live With

DRMAA JobTemplate	JSDL
startTime (PartialTimestamp)	intentionally not (SDL)
<i>deadlineTime (PartialTimestamp)</i>	intentionally not (SDL)
	<POSIXApplication>
	<WallTimeLimit>???
<i>hardWallclockTimeLimit (long)</i>	<CPUTimeLimit>???

Unclear / Missing Stuff

DRMAA JobTemplate	JSDL
<i>softWallclockTimeLimit (long)</i>	???
<i>hardRunDurationLimit (long)</i>	???
<i>softRunDurationLimit (long)</i>	???
email (string array)	???
blockEmail (boolean)	???
joinFiles (boolean)	???
Placeholders (HOME_DIR,	???

Conclusion

- Some things simply match
- Some things are unclear
- Some things are explicitly out-of-scope for JSDL
- Some ideas
 - Extension of <POSIXApplication> with respect to DRMAA IDL job template attributes
 - New JSDL application type <DRMAAJob> - use case ?
 - Allow DRMAA interface to consume JSDL, by defining default values for the missing mandatory DRMAA attributes
 - Still remaining question of placeholders