GWD-E (draft-gwde-jsdl-interop-v1-draft-008)
Category: Experimental

Editors:
A. Stephen McGough, Imperial College London
Andreas Savva, Fujitsu

4 September 2008

Implementation and Interoperability Experiences with the Job Submission Description Language(JSDL) 1.0 (Draft 008)

Status of This Document

This document provides information to the Grid community regarding the experiences of the Grid community in implementing JSDL 1.0. Distribution is unlimited.

Abstract

This document describes the implementation and interoperability experiences of independent implementations of the Job Submission Description Language (JSDL) 1.0.

Contents

1. Introduction

This document summarizes implementation and interoperability experiences with the JSDL specification. First, the results of a survey carried out by the JSDL-WG are presented. The survey collected the implementation experiences from eleven projects and was used to identify issues with the specification, common usage, as well as areas where extensions to the specification should be carried out in the future. The responses are summarized in Section 2. In addition the full text of each survey response is included in Appendix 3.

Second, this document provides a list of known implementations at the time of writing and a summary of interoperability efforts that covered parts of the JSDL specification, in particular the usage of JSDL 1.0[GFD.56] as profiled by the HPC Basic Profile.

Finally, a number of issues were identified since the publication of JSDL 1.0 through the experiences of the various implementers. These issues were recorded on the group's Gridforge tracker and a revision to the specification was carried out. The proposed changes to JSDL 1.0 are listed in Appendix 1. It is expected that the revised version will be subsequently published as the JSDL 1.0 Recommendation.

2. Survey Summary

Table 1 summarizes of the results of the implementation survey. The implementation status of each JSDL element is shown in aggregate. A 'Yes' indicates that an element was implemented; a 'No' indicates that the element was not implemented whilst a 'N/A' indicates that the implementers did not need the element for their work. Almost all elements, with the exception of MountSource[1], were implemented by at least one project, though various caveats exist as described in the individual responses. The survey response of each project is given in Appendix 3.

The following items were only implemented by one or less projects – we hereby justify whether they should remain within the specification or not. With the exception of MountSource (which was missed from the JSDL 1.0 Schema) all elements have been implemented at least once. Out of all the POSIX elements only ThreadCountLimit has not been implemented at least twice. As all other POSIX elements are being used it would seem incorrect to just delete one of them as this could cause more confusion in the future. Surprisingly GroupName has only been implemented once. This would seem counter-intuitive and may be due to the specific set of responses received. Many of the systems surveyed support rich security models, while this element was intended for use with simple mappings to batch systems. Further investigation is probably needed before making a decision on its utility.

We have argued here that the three elements that have not been implemented at least twice are elements which are needed: MountSource which was missed from the original specification, a POSIX element where omitting only one would cause confusion (ThreadCountLimit), or due to the subset of responses we have received (GroupName). Thus we feel that these items can justifiably remain within the specification. In any case removing one or two elements from a well-implemented schema seems counterproductive since there would be very little incentive for existing implementations to adopt such a revision. Instead such information on usage (or lack of) is best reserved for informing subsequent major revisions to the specification.

[1] MountSource was missing from the JSDL 1.0 schema

Table 1 Summary of Implemented JSDL Elements

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	8	0	0	
JobName	10	1	0	
JobAnnotation	3	6	1	
JobProject	3	7	0	
Description	2	0	0	
Application	6	1	0	
ApplicationName	6	5	0	
ApplicationVersion	4	6	0	
Resources	6	1	0	
CandidateHosts	6	5	0	
HostName	5	5	0	
FileSystem	3	7	0	
MountPoint	2	8	0	
MountSource	0	10	0	Note: Missing from JSDL 1.0 schema
DiskSpace	3	8	0	
FileSystemType	2	8	0	
ExclusiveExecution	4	6	0	
OperatingSystem	6	5	0	
OperatingSystemType	5	5	0	
OperatingSystemName	5	5	0	
OperatingSystemVersion	5	5	0	
CPUArchitecture	6	5	0	
CPUArchitectureName	5	5	0	
IndividualCPUSpeed	5	6	0	
IndividualCPUTime	6	5	0	
IndividualCPUCount	8	3	0	
IndividualNetworkBandwidth	2	8	0	
IndividualPhysicalMemory	8	3	0	
IndividualVirtualMemory	5	6	0	
IndividualDiskSpace	3	7	0	
TotalCPUTime	3	8	0	
TotalCPUCount	6	5	0	
TotalPhysicalMemory	4	7	0	
TotalVirtualMemory	3	8	0	
TotalDiskSpace	2	9	0	
TotalResourceCount	5	5	0	
DataStaging	6	1	0	
FileName	10	1	0	
FilesystemName	4	5	1	
CreationFlag	5	5	0	
DeleteOnTermination	4	5	0	
Source	9	1	0	
Target	9	1	0	
POSIXApplication	7	0	0	
Executable	10	0	0	
Argument	11	0	0	
Input	9	1	0	
Output	11	0	0	
Error	11	0	0	
WorkingDirectory	8	2	1	
Environment	9	1	0	
WallTimeLimit	5	5	0	
FileSizeLimit	4	7	0	
CoreDumpLimit	3	8	0	
DataSegmentLimit	3	8	0	
LockedMemoryLimit	3	8	0	

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
MemoryLimit	5	5	0	
OpenDescriptorsLimit	3	8	0	
PipeSizeLimit	2	8	0	
StackSizeLimit	3	8	0	
CPUTimeLimit	6	5	0	
ProcessCountLimit	2	8	0	
VirtualMemoryLimit	3	7	0	
ThreadCountLimit	1	8	0	
UserName	4	4	2	
GroupName	1	7	1	

Some projects did not provide a response for some elements hence the aggregate across the "Yes/No/N/A" columns is not always the same. Also one project responded separately for two different components of its architecture, hence the maximum possible aggregate is twelve even though there are only ten survey responses in the appendix.

3. HPC Basic Profile definition and Interoperability Tests

The work on defining the HPC Basic Profile and the subsequent interoperability testing[GFD.124] provided substantial feedback for JSDL 1.0. In particular clarifications made in the HPC Basic Profile[GFD.114] for the usage of JSDL 1.0 elements were included as errata issues for JSDL 1.0 (see Section 4). Also issues identified in the definition of the HPC Profile Application extension[GFD.111] were included as errata issues for the JSDL POSIXApplication extension (see Section 4).

4. Summary of Resolved Issues

A number of issues were received or collected from various sources after publication and were recorded in the group's Gridforge tracker[2]. Issues were divided into those that could be fixed in a schema-compatible update—minor editorial changes to fix typographical errors and other inaccuracies; clarifications on the definitions of various elements—and feature requests requiring new functionality.

Appendix 1 provides a detailed list of changes to address issues falling in the first category. In particular clarifications on the meaning of a number of Resources elements were necessary. Also an errata schema has to be added because an element defined in the original specification—MountSource—was missing from the normative JSDL 1.0 schema.

5. Author Information

Editors:
A. Stephen McGough
Imperial College London

Andreas Savva
Fujitsu

Contributors:
We gratefully acknowledge the survey responses from Dominic Battré, Mike Boniface, Christian Grimme, José Herrera, Eduardo Huedo, Nicholas Loulloudes, Mark Morgan, Vesso Novov, Alexander Papaspyrou, Morris Riedel, Gabor Roczei, Kazushige Saga, Bernd Schuller.

Acknowledgements:
We would like to thank the people who took the time to read and comment on earlier drafts. In particular, with apologies for anyone we may have missed, Michel Drescher, Donal Fellows, Steven Newhouse and Philipp Wieder. Their comments were valuable in helping us improve the readability and accuracy of this document.

[2] https://forge.gridforum.org/sf/projects/jsdl-wg

We would also like to thank the OGF HPC Profile WG for the interoperability work referenced in section 3.

6. Intellectual Property Statement

The OGF 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 OGF Secretariat.

The OGF 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 OGF Executive Director.

7. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

8. Full Copyright Notice

9. References

[GFD.56] Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., and Savva, A. Job Submission Description Language (JSDL), v. 1.0. GFD-R.056. Available at http://www.ogf.org/documents/GFD.56.pdf.

[GFD.111] Humphrey, M., Smith, C., Theimer, M., and Wasson, G. HPC Profile Application Extension v1.0. GFD-RP.111. Available at http://www.ogf.org/documents/GFD.111.pdf.

[GFD.114] Dillaway, B., Humphrey, M., Smith, C., Theimer, M. and Wasson, G. HPC Basic Profile, v. 1.0. GFD-R-P.114. Available at http://www.ogf.org/documents/GFD.114.pdf.

[GFD.124] Wasson, G. HPC Interoperability Experiences with the High Performance Computing Basic Profile (HPCBP), Version 1.0. GFD-E.124. Available at http://www.ogf.org/documents/GFD.124.pdf.

Appendix 1 Detailed Description of Proposed Changes to GFD.56

A number of issues were submitted after the publication of GFD.56. Following is the detailed description of changes agreed to in order to resolve these issues. Table and Section numbers refer to GFD.56.

Table 2 Detailed Description of Proposed Changes to GFD.56

Document Section	Description of Change
Table 2-1 Prefixes and namespaces used in this specification.	Added jsdl-errata namespace
Sections 5.2.4CreationFlagEnumeration Type, 6.5.4.1 Definition of CreationFlag	Clarification on CreationFlagdontOverwrite semantics
Section 5.2.5 RangeValue_Type Type	Minor clarifications on the definition and usage of the epsilon attribute
In a number of Example sections	Fixed miscapitalized range Exact
Section 6.1.1.6 Example of JobDefinition	Fixed minor error in example
Section 6.2.3 JobAnnotation	Added deprecation note to this element
Sections, 6.4.1 Resources, 6.4.16 - 6.4.27	Added note and cross-reference on relation between Individual and Total elements
Section 6.4.1.5 Pseudo Schema of Resources	Fixed misspelling of ExclusiveExecution
Section 6.4.4 FileSystem	▪ Added clarification that there is no expectation that a specific configuration action has to be carried out to satisfy this element ▪ Added Security Consideration section explaining ROOT may not be available
Section 6.4.4.5 Pseudo Schema of FileSystem	Updated order of elements in the pseudo schema order to match the normative schema
Section 6.4.4.6 Well-known FileSystem names	Changed text to make the provided definitions normative, rather than just examples
Sections 6.4.4 to 6.4.8, Examples using MountSource	Modified examples to use the jsdl-errata schema
Sections 6.4.4. to 6.4.8, Examples using FileSystem	Updated examples to use the correct order of elements (same as the corrected pseudo schema)
Sections 6.4.5 Mountpoint, 6.5.2 FileName, 8.1.3 Argument, 8.1.4 Input, 8.1.5 Output, 8.1.6 Error, 8.1.7 WorkingDirectory, 8.1.8 Environment	Added Security Consideration section with details of problematic characters in pathnames
Section 6.4.5 Mountpoint	Added Security Consideration section: stepping out of a mountpoint may be prohibited
Sections 6.4.4.6 MounSource, 6.4.4.8 FileSystemType	Fixed minor error in examples
Section 6.4.6 MountSource	Added note that the element was accidentally missing from the JSDL 1.0 schema.
Section 6.4.7 DiskSpace of FileSystem	Added clarification that it refers to “free” space on the filesystem
Sections 6.4.17 IndividualCPUTime, 6.4.23 TotalCPUTime	Added various clarifications on the expected meaning of these elements when values specify upper or lower bounds, etc.
Sections 6.4.18 IndividualCPUCount, 6.4.21 IndividualVirtualMemory, 6.4.22 IndividualDiskSpace, 6.4.27 TotalDiskSpace	Minor text change that these elements specify requirements rather than allocations in line with other Resources elements
Sections 6.4.18 IndividualCPUCount, 6.4.24 TotalCPUCount	Added non-normative interoperability note on usage of fractional values

Document Section	Description of Change
Section 6.4.19 IndividualNetworkBandwidth	Added clarification that this element refers to the maximum (nominal) bandwidth of a single network interface; and that multiple network interfaces are out of scope of the specification
Sections 6.4.20 IndividualPhysicalMemory, 6.4.21 IndividualVirtualMemory, 6.4.25 TotalPhysicalMemory, 6.4.26 TotalVirtualMemory	Changed text to specify that these requirements refer to the configured amount of memory
Section 6.4.22 IndividualDiskSpace	Added clarification that it refers to the raw disk capacity
Section 6.4.27 TotalDiskSpace	Added clarification that it refers to the total raw disk capacity across all resources
Section 6.5.1 DataStaging	Added text to specify the expected behavior when both Source and Target are not present
Sections 6.5.1.6, 6.5.3.6, 6.5.4.6, 6.5.5.6, 6.5.6.6 and 6.5.8.6. Examples using DataStaging	Updated examples to use the correct order of elements
Section 6.5.3 FilesystemName	Changed capitalization of the element name to match the normative schema
Section 8.1.3 Argument	Added clarifications: 1. An empty Argument element must map to an empty argument passed to the application 2. The order that arguments are passed to the application must match the order in the XML Minor terminology change: “collapsed” to “combined” Added note that the type of this element may change to a string in future versions.
Sections 8.1.5 Output, and 8.1.6 Error	Fixed minor error in examples
Section 8.1.8 Environment	Added clarifications ▪ Values are literal; and any other interpretation is implementation dependent and not interoperable. ▪ The order they are set in the execution environment SHOULD be the same as in the XML
References	Updated URI citation to RFC3986
Appendices 1. Normative JSDL, 2. PosixApplication schema	Added OGF copyright notice to the schemas
Appendix 3 JSDL Errata Schema	Added JSDL Errata schema to define missing MountSource element
Appendix 5 Detailed Document Change History	Added as part of errata release
Document at large	Minor text formatting changes, particularly removing smart quotes from XML text, for consistency

Appendix 2 Adoption of JSDL by projects and other groups

The following have implemented JSDL 1.0 and have participated in interoperability testing of the JSDL subset that is part of the HPC Basic profile. (Source “Interoperability Experiences with the High Performance Computing Basic Profile (HPCBP), Version 1.0”[GFD.124]).

- University of Virginia e-Science Group (.NET)
- University of Virginia e-Science Group (Linux/gSoap)
- Microsoft
- Platform Computing
- OMII-UK / GridSAM
- EGEE 2/OMII Europe CREAM-BES
- UNICORE
- NorduGrid/KnowARC A-REX
- Altair Engineering

The following have implemented JSDL 1.0 and have participated in interoperability testing of the JSDL subset that is part of the HPC Basic profile at SuperComputing 2006.

- University of Virginia .NET Implementation
- Microsoft HPC group
- Platform
- Globus
- Unicore
- Genesis II
- OMII-UK / GridSAM
- CROWN
- gLiteWMProxy (EGEE)
- gLite CREAM (EGEE)

The following projects have replied to the JSDL survey and are known to have implementations of JSDL 1.0.

- UNICORE 6
- Genesis II
- AssessGrid
- C3Grid
- g-Eclipse
- GlobusGridWay
- NAREGI
- OMII-UK / GridSAM
- NorduGrid / KnowARC
- GRIA

The following additional projects have presented at GGF or OGF JSDL sessions and are known to have implemented JSDL 1.0.

- HPC-Europa
- Business Grid Project (WS-Agreement & JSDL 1.0)
- Grid Programming Environment (GPE)
- UniGrids

The following groups also said they have implemented JSDL:

- Askalon (with WS-Agreement)
- DEISA

- GridBus Project (see OGSA Roadmap, section 8)
- OpenDSP

Appendix 3 Implementation Survey Responses

Appendix 3.1 UNICORE 6

Contact: Morris Riedel, Bernd Schuller
Date: 2008-08-01

3.1.1 Introduction

The WS-RF compliant UNICORE 6 Grid middleware uses JSDL as the job description language for submissions to the UNICORE proprietary job submission and management interface named UNICORE Atomic Services (UAS) that also provide file transfer and storage management functionalities. UNICORE 6 is based on the XFire SOAP Engine using the Jetty web server and XMLBeans data bindings. Please refer to the following URL for more information:
http://www.unicore.eu/documentation/manuals/unicore6/

3.1.2 Implemented JSDL Elements

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	X			
JobName	X			
JobAnnotation		X		
JobProject		X		
Application	X			
ApplicationName	X			
ApplicationVersion	X			
Resources	X			
CandidateHosts		X		
HostName		X		
FileSystem		X		
MountPoint		X		
MountSource		X		
DiskSpace		X		
FileSystemType		X		
ExclusiveExecution		X		
OperatingSystem		X		
OperatingSystemType		X		
OperatingSystemName		X		
OperatingSystemVersion		X		
CPUArchitecture		X		
CPUArchitectureName		X		
IndividualCPUSpeed		X		
IndividualCPUTime	X			
IndividualCPUCount	X			
IndividualNetworkBandwidth		X		
IndividualPhysicalMemory	X			
IndividualVirtualMemory		X		
IndividualDiskSpace		X		
TotalCPUTime		X		
TotalCPUCount		X		
TotalPhysicalMemory		X		
TotalVirtualMemory		X		
TotalDiskSpace		X		
TotalResourceCount	X			
DataStaging	X			
FileName	X			
FilesystemName	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
CreationFlag		X		
DeleteOnTermination		X		
Source	X			
Target	X			
POSIXApplication	X			
Executable	X			
Argument	X			array
Input	X			
Output	X			
Error	X			
WorkingDirectory		X		UNICORE always executes jobs in a temporary directory chosen by the system
Environment	X			array
WallTimeLimit		X		
FileSizeLimit		X		
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit		X		
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit		X		
ProcessCountLimit		X		
VirtualMemoryLimit		X		
ThreadCountLimit		X		
UserName	X			
GroupName		X		
HPCProfileApplication	X			Used like POSIXApplication
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory		X		UNICORE always executes jobs in a temporary directory chosen by the system
Environment	X			
UserName	X			

3.1.3 Other problems encountered

The JSDL specification as well as the HPCProfileApplication extension does not cover all necessary details for HPC-based job submissions (e.g. number of processes per host) and therefore we are already considering proprietary JSDL extensions and also had a look at the SPMD application extension. Since UNICORE 6 is a rather HPC driven Grid middleware this could be considered as a problem in the near future.

3.1.4 Mappings to existing systems

The Web services layer forwards incoming JSDL execution requests to the internal backend named as enhanced Network Job Supervisor (XNJS). The XNJS in turns maps the JSDL to an internal proprietary NJS-Target System Interface (TSI) protocol and forwards execution requests in this format to different flavors of a TSI, for instance a Torque TSI or LoadLeveler TSI. These set of TSIs represent the interface to underlying resource management systems and thus do the actual job submission to them. So far, we have not experienced any problems in mapping JSDL content to TSI executions, except the above mentioned HPC-based problems.

3.1.5 Enhancements:

UNICORE 6 did not extend JSDL with our own features so far, maybe later if necessary to support HPC-based applications.

3.1.6 Participation in interoperability tests:

We plan to participate in the HPC Basic Profile interoperability demonstrations at Supercomputing 2007 organized from the Grid Interoperation Now (GIN) – Community Group (CG).

3.1.7 Security:

Transport Level Security (TLS) and message level security using WSS4J to sign the SOAP body of Web service message exchanges.

UNICORE 6 does not include security information in JSDL documents.

There were no problems combining JSDL with the UNICORE 6 security solution. JSDL is purely a part in the SOAP body, while UNICORE 6 security models are using the SOAP header or transport level security (TLS).

Appendix 3.2 Genesis II

Contact: Mark Morgan
Date: 20 August 2007

3.2.1 Introduction

Genesis II (http://vcgr.cs.virginia.edu/genesisII) is a complete, fully integrated, from-the-ground-up implementation of many of the OGF and OGSA grid service specifications and proto-specifications. We hope to provide a standards based compute and data grid for users here at the University of Virginia and around elsewhere while also vetting the standards in the OGF process. For the compute side of our work, we use JSDL to describe applications and jobs and in particular pass the JSDL through numerous reification stages from simple concept of a job to deployment and instantiation of that job using OGSA-BES. Please refer to http://vcgr.cs.virginia.edu/genesisII/documents/presentations/GenII.OGF19.ppt for a description of Genesis II.

3.2.2 Implemented JSDL Elements

JSDL Element	Yes	No	N/A	Comments
JobIdentification	X			
JobName	X			Only for human readable purposes
JobAnnotation		X		
JobProject		X		
Application	X			
ApplicationName	X			Only for human readable purposes
ApplicationVersion		X		
Resources				
CandidateHosts	X			Currently, we only use this for checking (not for scheduling).
HostName	X			See Candidate Hosts
FileSystem		X		We explicitly do not allow this
MountPoint		X		See above
MountSource		X		See above
DiskSpace		X		
FileSystemType		X		See FileSystem
ExclusiveExecution	X			If specified, we fault indicating that we cannot support this option
OperatingSystem	X			For checking only
OperatingSystemType	X			For checking only

JSDL Element	Yes	No	N/A	Comments
OperatingSystemName	X			For checking only
OperatingSystemVersion	X			For checking only
CPUArchitecture	X			For checking only
CPUArchitectureName	X			For checking only
IndividualCPUSpeed	X			For checking only
IndividualCPUTime	X			For checking only
IndividualCPUCount	X			For checking only
IndividualNetworkBandwidth		X		
IndividualPhysicalMemory	X			For checking only
IndividualVirtualMemory	X			For checking only
IndividualDiskSpace		X		
TotalCPUTime		X		
TotalCPUCount		X		
TotalPhysicalMemory		X		
TotalVirtualMemory		X		
TotalDiskSpace		X		
TotalResourceCount		X		
DataStaging	X			We make very heavy use of data staging.
FileName	X			
FilesystemName		X		We don't allow file systems.
CreationFlag	X			
DeleteOnTermination	X			
Source	X			The requirement of making this a URI is too restrictive.
Target	X			The requirement of making this a URI is too restrictive.
POSIXApplication	X			We suppose both POSIX App and HPC App.
Executable	X			It's unfortunate for us that this element is required. We wanted to be able to use POSIX App. For templating deployment-type runs as well and the deployment defines this value so we have to insert a dummy value that we later throw out.
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
WallTimeLimit		X		
FileSizeLimit		X		
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit		X		
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit		X		
ProcessCountLimit		X		
VirtualMemoryLimit		X		
ThreadCountLimit		X		
UserName		X		I don't think that this is a particularly useful parameter for grid apps. We need something MUCH more generic that fits into an overall grid-based security model (tied in with delegation and AuthN).
GroupName		X		Same as for UserName

JSDL Element	Yes	No	N/A	Comments
HPCProfileApplication	X			
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
UserName		X		See UserName under POSIXApp

3.2.3 Other problems encountered:

At some point in the past an element type was defined as being an simpleType extension of xsd:string which was causing my tooling (Axis 1.4) to barf. I don't know if that was fixed or not. Also, a lot of the JSDL stuff using floating point values for things that should obviously be integers (number of CPUs, etc.) I understand why the authors did that (for simplicity and type-reuse) but I would have preferred a more type-checked mechanism.

3.2.4 Mappings to existing systems:

We map JSDL onto a number of “systems” depending on your definition of the word “system”. We use it to describe the job from the point that it is submitted to Genesis II (by an end user) to the point that it gets executed in some environment. This path can include submitting the JSDL to a scheduler, an application deployer, a grid-based queing system, a BES container, or being fork-exec'd by our system.

3.2.5 Enhancements:

We extended JSDL to allow for jobs to be described using an application description (which can later be deployed in a target environment). Please refer to http://vcgr.cs.virginia.edu/genesisII/documents/presentations/AppDepDesc.ppt for a description of this.

3.2.6 Participation in interoperability tests:

We participated in the HPC Profile Interop at SC06.

3.2.7 Security:

We believe that JSDL is completely the wrong place to be putting security information. Security is a corss-cutting concern that the grid as a whole must deal with. JSDL should only be concerned with describing jobs and their restrictions, not how they get run. Further, job execution and management is a prime example of why delegation is necessary in the grid and should be considered an absolute necessity whenever grid security is discussed. In Genesis II, security information is passed along call chains using an implicit calling context (usually included in SOAP headers). This information can be used by out BES services to figure out which account to run the job under on the target system.

Appendix 3.3 AssessGrid

Contact: Dominic Battré
Date: Oct 24, 2007

3.3.1 Introduction

Project URL: http://www.assessgrid.eu

We use JSDL to describe jobs in a WS-Agreement context.

3.3.2 Implemented JSDL Elements:

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification				
JobName		X		
JobAnnotation		X		
JobProject		X		Might be added
Application				
ApplicationName		X		We use the POSIX extension
ApplicationVersion		X		
Resources				
CandidateHosts		X		
HostName		X		
FileSystem	X			Not fully supported, yet, static values are assumed
MountPoint		X		
MountSource		X		
DiskSpace	X			Not enforced, yet
FileSystemType	X			It was not quite clear to me, whether scratch is assumed to be cross-mounted for multi-node jobs
ExclusiveExecution	X			Always assumed to be true
OperatingSystem	X			
OperatingSystemType	X			
OperatingSystemName	X			
OperatingSystemVersion	X			
CPUArchitecture	X			
CPUArchitectureName	X			
IndividualCPUSpeed	X			
IndividualCPUTime	X			
IndividualCPUCount	X			
IndividualNetworkBandwidth		X		
IndividualPhysicalMemory	X			
IndividualVirtualMemory	X			
IndividualDiskSpace	X			
TotalCPUTime		X		
TotalCPUCount	X			
TotalPhysicalMemory		X		
TotalVirtualMemory		X		
TotalDiskSpace		X		
TotalResourceCount	X			
DataStaging				
FileName	X			
FilesystemName	X			
CreationFlag	X			
DeleteOnTermination	X			
Source	X			
Target	X			
POSIXApplication				
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
WallTimeLimit		X		
FileSizeLimit		X		
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit		X		
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit		X		
ProcessCountLimit		X		
VirtualMemoryLimit		X		
ThreadCountLimit		X		
UserName		X		
GroupName		X		
HPCProfileApplication				
Executable				
Argument				
Input				
Output				
Error				
WorkingDirectory				
Environment				
UserName				

3.3.3 Other problems encountered:

3.3.4 Mappings to existing systems:

We are mapping JSDL to OpenCCS, a planning based scheduler. The major problem was that the planning based scheduler expects information like earliest start time and a deadline for the job. It would be nice if this was included in JSDL or some extension.

3.3.4.1 Enhancements:

We have introduced a quality of service measure “Probability of Failure”:

```
<assessgrid:PoFassessgrid:unit="%">10</assessgrid:PoF>
```

Further more, we need scheduling attributes:

```
<assessgrid:EarliestStartTime>2007-03-
01T00:00:00+01:00</assessgrid:EarliestStartTime>
<assessgrid:LatestFinishTime>2007-03-
01T12:00:00+01:00</assessgrid:LatestFinishTime>
```

As well as tags, when the Stage-In files become available and when until when the results are kept.

3.3.5 Participation in interoperability tests:

We did not participate in any JSDL interop tests and do not plan to do that at the moment.

3.3.6 Security:

All communication is secured over WS-SecureConversation with the Globus Toolkit 4. This did not create any problems.

Appendix 3.4 Collaborative Climate Community Data and Processing Grid (C3Grid)

WP6: C3Grid Workflow Scheduling Service
Contact: Alexander Papaspyrou, Christian Grimme
Date:20.02.2008

3.4.1 Introduction

The Collaborative Climate Community Data and Processing Grid (C3Grid) is a cooperation of earth system science and computer science researchers that aims to provide an integrated Grid technology solution for Earth System Science.
Major challenges regarding workflow planning and management include

- Standardized access to heterogeneous and distributed data archives and intelligent data preselection and preprocessing to minimize wide-area transfers.
- Automatic co-allocation of compute and data resources to ensure just-in-time data availability for compute jobs using planned transfers.
- Unified usage of compute resources with next-generation scheduling features such as negotiation and agreement for advance reservation.

The C3Grid Workflow Scheduling Service (WSS) provides comprehensive support for submission, planning, execution, and control of workflows with respect to the aforementioned requirements.

3.4.2 Implemented JSDL Elements

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification				
JobName	X			
JobAnnotation		X		Not currently.
JobProject		X		Not currently.
Application				
ApplicationName		X		Planned (for predefined, user portal-selectable workflows).
ApplicationVersion		X		Planned (for predefined, Grid portal-selectable workflows).
Resources				
CandidateHosts	X			For user preselection, portal preselection (data staging jobs only), and scheduler (candidate set generator) preselection
HostName	X			Can be virtual (aka. key for Information System)
FileSystem	X			
MountPoint	X			If provider-defined in Information System
MountSource		X		
DiskSpace		X		
FileSystemType		X		
ExclusiveExecution		X		
OperatingSystem		X		Planned (pending implementation)
OperatingSystemType		X		Planned (pending implementation)
OperatingSystemName		X		Planned (pending implementation)
OperatingSystemVersion		X		Planned (pending implementation)
CPUArchitecture		X		Planned (pending implementation)
CPUArchitectureName		X		Planned (pending implementation)
IndividualCPUSpeed		X		
IndividualCPUTime		X		
IndividualCPUCount		X		
IndividualNetworkBandwidth		X		
IndividualPhysicalMemory		X		
IndividualVirtualMemory		X		
IndividualDiskSpace		X		
TotalCPUTime		X		Planned (pending implementation)

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
TotalCPUCount		X		Planned (pending implementation)
TotalPhysicalMemory		X		Planned (pending implementation)
TotalVirtualMemory		X		
TotalDiskSpace		X		Planned (pending implementation)
TotalResourceCount		X		
DataStaging[3]				
FileName	X			
FilesystemName	X			
CreationFlag		X		
DeleteOnTermination		X		Planned (pending implementation)
Source	X			
Target	X			
POSIXApplication				
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
WallTimeLimit	X			
FileSizeLimit		X		
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit	X			
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit	X			
ProcessCountLimit		X		
VirtualMemoryLimit		X		
ThreadCountLimit		X		
UserName		X		
GroupName		X		
HPCProfileApplication				
Executable		X		Planned (pending implementation)
Argument		X		Planned (pending implementation)
Input		X		Planned (pending implementation)
Output		X		Planned (pending implementation)
Error		X		Planned (pending implementation)
WorkingDirectory		X		Planned (pending implementation)
Environment		X		Planned (pending implementation)
UserName		X		

3.4.3 Other problems encountered:

A major concern of our project was the support of workflows. Within C3Grid, a simple, proprietary XML dialect for DAG-style job definitions has been designed and implemented. However, JSDL does not directly support the connection of output files from one job to input files of another job. To overcome this deficiency, the <jsdl:DataStaging> “name” attribute has been used for this: the workflow engine detects identical values of this attribute and connects the corresponding data staging elements from two JSDL definitions as input and output. Then, the scheduler dynamically inserts data transfer tasks into the workflow depending on job and network allocation decisions. To this end, the DataStaging contents are interpreted as follows:

[3] See Section “Other Problems encountered.”

- No <jsdl:Source> and no <jsdl:Target> leads to an automatic transfer injection by the scheduler, depending on its decisions.
- A <jsdl:Source> and no <jsdl:Target> leads to an import from a user-defined source to a scheduler-selected target, depending on its decisions. This applies to data within the workflow at the beginning of a branch within the graph.
- No <jsdl:Source> and a <jsdl:Target> leads to an export from a scheduler-selected source (based on previous decisions) to a user-defined target. This applies to data within the workflow at the end of a branch within the graph.

Regarding the <jsdl:URI> element, project-specific namespaces are used.

3.4.4 Mappings to existing systems:

JSDL w/ POSIXApplication	RSL for Globus Toolkit 4.x
Argument	Argument
WorkingDirectory	Directory
Environment	environment
Executable	Executable
CPUTimeLimit	maxCpuTime
MemoryLimit	maxMemory
WalltimeLimit	maxWallTime
Error	Stderr
Input	Stdin
Output	Stdout

Mappings for HPCApplication and SPMDApplication are pending implementation.

3.4.5 Enhancements:

An additional application profile has been created, which is C3Grid-proprietary and encapsulates the extraction of climate data sets from Web Service-accessible databases to file systems. Currently, however, the specification has not been published and is used internally only.

3.4.6 Participation in interoperability tests:

No.

3.4.7 Security:

Our security model is the usage of WS-Security (both message and conversation level) and TLS for the transport. It is planned to use Shibboleth SAML assertions for authorization in the future; currently, however, it is unclear whether this information (or a pointer to it) will be included in JSDL.

Appendix 3.5 g-Eclipse

Contact: contact@g-eclipse.eu
Date: ***October 15th, 2007 (Updated 26 June 2008)***

3.5.1 Introduction

The g-Eclipse project (www.eclipse.org/geclipse) aims to build an integrated workbench framework to access the power of existing Grid infrastructures. The framework is built on top of the reliable eco-system of the Eclipse community to enable a sustainable development. The framework will provide tools to customize Grid users' applications, to manage Grid resources and to support the development cycle of new Grid applications. The g-Eclipse framework will be middleware agnostic and its architecture is designed to extend it for many different Grid middlewares (such as gLite, UNICORE, Globus toolkit). Implementation started with the gLite middleware, and followed with the support of the GRIA middleware and also the Amazon Web Services (AWS).

The g-Eclipse framework aims for full support of the JSDL standard to be independent from the underlying Grid middleware. JSDL is the main description language used in g-Eclipse. Therefore the g-Eclipse team developed wizards and a multi-page editor for JSDL files.

Job Description Wizard

With the help of a JSDL wizard provided by the g-Eclipse framework, the user can easily create the functional skeleton of a JSDL file. Further details can be entered with the help of the JSDL editor, which will automatically be opened, when the wizard is finished. The Wizard introduces only those JSDL fields that are essential form user's point of view – those include application, POSIX application and data staging related elements.

JSDL Editor

The g-Eclipse project developed a user-friendly, fully functional JSDL editor for editing JSDL documents. The editor follows Eclipses' multi-page editor style andconsists currently of 5 pages (Overview, Job Definition, Application, DataStaging and Resources) plus a page showing the resulted XML file of the JSDL document (changes here will be reflected in the other pages).

NOTE: As of September28th 2007, **g-Eclipse 0.5.0** is publicly available. g-Eclipse is an official technology project at the Eclipse Foundation.

3.5.2 Implemented JSDL Elements:

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification				
JobName	X			
JobAnnotation	X			
JobProject	X			
Description	X			
Application				
ApplicationName	X			
ApplicationVersion	X			
Resources				
CandidateHosts	X			
HostName	X			
FileSystem	X			
MountPoint	X			
MountSource		X		
DiskSpace	X			
FileSystemType	X			
ExclusiveExecution	X			
OperatingSystem	X			
OperatingSystemType	X			
OperatingSystemName	X			
OperatingSystemVersion	X			
CPUArchitecture	X			
CPUArchitectureName	X			
IndividualCPUSpeed	X			
IndividualCPUTime	X			
IndividualCPUCount	X			
IndividualNetworkBandwidth	X			
IndividualPhysicalMemory	X			
IndividualVirtualMemory	X			
IndividualDiskSpace	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
TotalCPUTime	X			
TotalCPUCount	X			
TotalPhysicalMemory	X			
TotalVirtualMemory	X			
TotalDiskSpace	X			
TotalResourceCount	X			
DataStaging				
FileName	X			
FilesystemName	X			
CreationFlag	X			
DeleteOnTermination	X			
Source	X			
Target	X			
POSIXApplication				
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
WallTimeLimit	X			
FileSizeLimit	X			
CoreDumpLimit	X			
DataSegmentLimit	X			
LockedMemoryLimit	X			
MemoryLimit	X			
OpenDescriptorsLimit	X			
PipeSizeLimit	X			
StackSizeLimit	X			
CPUTimeLimit	X			
ProcessCountLimit	X			
VirtualMemoryLimit	X			
ThreadCountLimit	X			
UserName	X			
GroupName	X			
HPCProfileApplication				
Executable			X	Currently there is no plan for implementation of HPC Profile Applications.
Argument			X	
Input			X	
Output			X	
Error			X	
WorkingDirectory			X	
Environment			X	
UserName			X	

3.5.3 Other problems encountered:

3.5.4 Mappings to existing systems:

With the help of the g-Eclipse framework, the JSDL files can easily been submitted to any gLite based Infrastructure like EGEE. For the submission, a working webservice based Workload Management System (version 3.1) of gLite is required.

JSDL is used also in the job submission process of GRIA jobs.

3.5.5 Enhancements:

3.5.6 Participation in interoperability tests:

NO

3.5.7 Security:

g-Eclipse expects a Job Submission service (Resource Broker, ...) including a Grid security model. Currently, g-Eclipse provides the VOMS based security system based on X.509 certificates and delegated proxies. The g-Eclipse framework checks before job submission if a valid security token exists. If not, the user is asked to create the requested security token (i.e. a VOMS proxy).

Appendix 3.6 GlobusGridWay

Contact: Eduardo Huedo, José Herrera
Date: 10/05/07

3.6.1 Introduction

The GridWayMetascheduler enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRM (Local Resource Management) systems, such as PBS, SGE, LSF or Condor, within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid). GridWay is a Globus project, adhering to Globus philosophy and guidelines for collaborative development and so welcoming code and support contributions from individuals and corporations around the world.
GridWays allows users to specify their jobs using JSDL. In the future, GridWay will generate JSDL to interface LRM systems through Globus.
More information at http://www.gridway.org

3.6.2 Implemented JSDL Elements:

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	√			
JobName	√			
JobAnnotation		√		
JobProject		√		
Application	√			
ApplicationName	√			
ApplicationVersion	√			
Resources	√			
CandidateHosts	√			
HostName	√			
FileSystem		√		
MountPoint		√		
MountSource		√		
DiskSpace		√		
FileSystemType		√		
ExclusiveExecution		√		
OperatingSystem	√			
OperatingSystemType	√			
OperatingSystemName	√			
OperatingSystemVersion	√			
CPUArchitecture	√			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
CPUArchitectureName	√			
IndividualCPUSpeed	√			
IndividualCPUTime		√		Needs changes in GridWay
IndividualCPUCount	√			
IndividualNetworkBandwidth		√		Not usually reported by Info. Serv.
IndividualPhysicalMemory	√			
IndividualVirtualMemory		√		Will be supported in the future
IndividualDiskSpace	√			
TotalCPUTime		√		
TotalCPUCount		√		Will be supported in the future
TotalPhysicalMemory		√		Will be supported in the future
TotalVirtualMemory		√		Will be supported in the future
TotalDiskSpace		√		Will be supported in the future
TotalResourceCount		√		Will be supported in the future
DataStaging	√			
FileName	√			
FilesystemName		√		Out of scope
CreationFlag	√			
DeleteOnTermination	√			
Source	√			
Target	√			
POSIXApplication	√			
Executable	√			
Argument	√			
Input	√			
Output	√			
Error	√			
WorkingDirectory		√		Working dir is taken from OS
Environment	√			
WallTimeLimit		√		Needs changes in GridWay
FileSizeLimit		√		Out of scope
CoreDumpLimit		√		Out of scope
DataSegmentLimit		√		Out of scope
LockedMemoryLimit		√		Out of scope
MemoryLimit		√		Needs changes in GridWay
OpenDescriptorsLimit		√		Out of scope
PipeSizeLimit		√		Out of scope
StackSizeLimit		√		Out of scope
CPUTimeLimit		√		Needs changes in GridWay
ProcessCountLimit		√		Needs changes in GridWay
VirtualMemoryLimit		√		Needs changes in GridWay
ThreadCountLimit		√		Out of scope
UserName			√	User name is taken from OS
GroupName		√		Not needed
HPCProfileApplication	√			
Executable	√			
Argument	√			
Input	√			
Output	√			
Error	√			
WorkingDirectory			√	Working dir is taken from OS
Environment	√			
UserName			√	User name is taken from OS

3.6.3 Other problems encountered:

Due to GridWay Job Template specification, the POSIX Application schema must be always included in the JSDL document.

3.6.4 Mappings to existing systems:

We mapped JSDL to GridWay Job Template (GWJT), and we did not find any significant problem. However, there are a considerable number of unsupported parameters because they are out of the scope of GridWay.
More information at http://www.gridway.org/documentation/stable/userguide/c587.htm

3.6.5 Enhancements:

We didn't extend JSDL.

3.6.6 Participation in interoperability tests:

No, we didn't.

3.6.7 Security:

We map JSDL documents to GWJT files, so we use the GridWay security model, as well as GSI (Globus Security Infrastructures).
More information at http://gridway.org

Appendix 3.7 NAREGI

Contact: Kazushige Saga
Date: 04-June-08

3.7.1 Introduction

Project web site: http://www.naregi.org/index_e.html.

3.7.2 Implemented JSDL Elements:

(1) JSDL Elements in Resource Broker (NAREGI Super Scheduler)

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	X			Silently ignored in GridSS
JobName	X			Same as above
JobAnnotation	X			Same as above
JobProject	X			Same as above
Application	X			Silently ignored in GridSS
ApplicationName	X			Same as above
ApplicationVersion	X			Same as above
Resources	X			
CandidateHosts	X			
HostName	X			
FileSystem		X		Currently not supported

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
MountPoint		X		Same as above
MountSource		X		Same as above
DiskSpace		X		Same as above
FileSystemType		X		Same as above
ExclusiveExecution	X			Silently ignored in GridSS
OperatingSystem	X			
OperatingSystemType	X			
OperatingSystemName	X			
OperatingSystemVersion	X			
CPUArchitecture	X			
CPUArchitectureName	X			
IndividualCPUSpeed		X		Silently ignored in GridSS
IndividualCPUTime		X		Silently ignored in GridSS
IndividualCPUCount	X			
IndividualNetworkBandwidth	X			
IndividualPhysicalMemory	X			
IndividualVirtualMemory	X			
IndividualDiskSpace		X		Silently ignored in GridSS
TotalCPUTime		X		Silently ignored in GridSS
TotalCPUCount	X			
TotalPhysicalMemory	X			
TotalVirtualMemory	X			
TotalDiskSpace		X		Silently ignored in GridSS
TotalResourceCount	X			
DataStaging	X			third-party transfer only
FileName	X			
FilesystemName		X		
CreationFlag		X		
DeleteOnTermination		X		
Source	X			
Target	X			
POSIXApplication	X			
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
WallTimeLimit	X			
FileSizeLimit	X			Silently ignored in GridSS
CoreDumpLimit	X			Same as above
DataSegmentLimit	X			Same as above
LockedMemoryLimit	X			Same as above
MemoryLimit	X			Same as above
OpenDescriptorsLimit	X			Same as above
PipeSizeLimit	X			Same as above
StackSizeLimit	X			Same as above
CPUTimeLimit	X			Same as above
ProcessCountLimit	X			Same as above
VirtualMemoryLimit				Same as above
ThreadCountLimit				Same as above
UserName				Same as above
GroupName				Same as above
HPCProfileApplication		X		Currently not supported
Executable				Same as above

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
Argument				Same as above
Input				Same as above
Output				Same as above
Error				Same as above
WorkingDirectory				Same as above
Environment				Same as above
UserName				Same as above

(2) JSDL Elements in Computing Resources (NAREGI GridVM)

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	X			
JobName	X			
JobAnnotation		X		
JobProject		X		
Application		X		
ApplicationName		X		
ApplicationVersion		X		
Resources	X			
CandidateHosts		X		
HostName		X		
FileSystem		X		
MountPoint		X		
MountSource		X		
DiskSpace		X		
FileSystemType		X		
ExclusiveExecution		X		
OperatingSystem		X		
OperatingSystemType		X		
OperatingSystemName		X		
OperatingSystemVersion		X		
CPUArchitecture		X		
CPUArchitectureName		X		
IndividualCPUSpeed		X		
IndividualCPUTime		X		
IndividualCPUCount		X		
IndividualNetworkBandwidth		X		
IndividualPhysicalMemory		X		
IndividualVirtualMemory		X		
IndividualDiskSpace		X		
TotalCPUTime		X		
TotalCPUCount	X			Only for ‘exact’
TotalPhysicalMemory		X		
TotalVirtualMemory		X		
TotalDiskSpace		X		
TotalResourceCount	X			Only for ‘exact’
DataStaging		X		
FileName		X		
FilesystemName		X		
CreationFlag		X		
DeleteOnTermination		X		
Source		X		
Target		X		
POSIXApplication	X			
Executable	X			Required
Argument	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
Input		X		
Output	X			
Error	X			
WorkingDirectory	X			
Environment	X			
WallTimeLimit	X			
FileSizeLimit	X			
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit	X			
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit	X			
ProcessCountLimit		X		
VirtualMemoryLimit	X			
ThreadCountLimit		X		
UserName	X			
GroupName		X		
HPCProfileApplication		X		
Executable		X		
Argument		X		
Input		X		
Output		X		
Error		X		
WorkingDirectory		X		
Environment		X		
UserName		X		

3.7.3 Other problems encountered:

3.7.4 Mappings to existing systems:

PBSprofessional, LoadLeveler, SGE, ParallelNavi(Fujitsu SPARC), NQS-II(NEC SX)

3.7.5 Enhancements:

NAREGI extended some elements for our co-allocation functionality.

3.7.6 Participation in interoperability tests:

NAREGI did a small interoperability test with GridSAM's JSDL.

3.7.7 Security:

GSI

Appendix 3.8 GridSAM—Grid Job Submission and Monitoring web service

Contact: Dr. A. Steven McGough, Vesso A. Novov
Date: 27 June 2008

3.8.1 Introduction

The aim of GridSAM is to provide a Web Service for submitting and monitoring the execution of jobs described in JSDL documents. GridSAM maps parsed JSDL documents to the proprietary job submission mechanisms of a variety of Distributed Resource Managers (DRM). Acting as a thin translation system between the emerging standard and existing DRM systems processing job submissions and file staging.
Project web site - http://gridsam.sourceforge.net
Project development – OMII-UK kit (Java, Axis, Tomcat), XmlBeans, HiveMind, Hibernate, Quartz.

3.8.2 Implemented JSDL Elements:

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification	X			
JobName	X			Stored as urn:gridsam:JobName job property.
JobAnnotation	X			Stored as urn:gridsam:JobAnnotation job property
JobProject	X			Stored as urn:gridsam:JobProject job property
Description	X			Stored as urn:gridsam:Description job property
Application				
ApplicationName		X		Not interpreted. Planned support in future version for preconfigured application identifiable by name
ApplicationVersion		X		Not interpreted. Planned support in future version for preconfigured application identifiable by name
Resources	X			Currently only a subset of these are supported for Condor. Others will follow for Condor and other supported DRMs.
CandidateHosts				
HostName				
FileSystem				
MountPoint				
MountSource				
DiskSpace				
FileSystemType				
ExclusiveExecution				
OperatingSystem	X			Condor
OperatingSystemType				
OperatingSystemName				
OperatingSystemVersion				
CPUArchitecture	X			Condor
CPUArchitectureName				
IndividualCPUSpeed				
IndividualCPUTime				
IndividualCPUCount				
IndividualNetworkBandwidth				
IndividualPhysicalMemory				
IndividualVirtualMemory				
IndividualDiskSpace				
TotalCPUTime				
TotalCPUCount				
TotalPhysicalMemory				
TotalVirtualMemory				
TotalDiskSpace				
TotalResourceCount				
DataStaging				
FileName	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
FilesystemName		X		Not supported. The element is ignored and mapped to a file system provided by the launching mechanism.
CreationFlag	X			Only "overwrite" is supported. This will be fixed in future version when the Virtual File System support is improved.
DeleteOnTermination				This will be fixed in future version when the Virtual File System support is imporoved.
Source	X			
Target	X			
URI	X			For URL schemes ftp://, http://, webdav:// and sftp://. Only simple username/password authentication are supported, however the username/password must be embedded as plain-text in the JSDL document. (e.g. http://myname:mypassword@www.myhost.com/myfile). For gsiftp:// URL scheme, the jsdl:JobDefinition/myproxy:MyProxy element is used to retrieve a Globus credential in order to perform the staging.
POSIXApplication				
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			Supported in Globus 2.4.3. Ignored in Fork, SSH, Condor. The working directory for these DRMConnectors are dynamically generated in the spool directory.
Environment	X			
WallTimeLimit		X		
FileSizeLimit		X		
CoreDumpLimit		X		
DataSegmentLimit		X		
LockedMemoryLimit		X		
MemoryLimit		X		
OpenDescriptorsLimit		X		
PipeSizeLimit		X		
StackSizeLimit		X		
CPUTimeLimit		X		
ProcessCountLimit		X		
VirtualMemoryLimit		X		
CPUTimeLimit		X		
UserName		X		
GroupName		X		
HPCProfileApplication				
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory	X			Supported in Globus 2.4.3. Ignored in Fork, SSH, Condor. The working directory for these DRMConnectors are dynamically generated in the spool directory.
Environment	X			
UserName		X		

3.8.3 Other problems encountered:

No.

3.8.4 Mappings to existing systems:

Currently, GridSAM maps JSDL to the job submission mechanisms of: Globus 2.4.3, Condor, Sun Grid Engine, PBS, UNICORE (by third party), LSF, Linux fork and Linux SSH.
There are plans to extend this functionality to include mappings to: EGEE.

3.8.5 Enhancements:

3.8.5.1 MPIApplication (Globus 2.4.3 DRM Connector only)

GridSAM-proposed extension to the jsdl-posix:POSIXApplication description. GridSAM user can use the mpi:MPIApplication element instead of the jsdl-posix:POSIXApplication to denote a MPI compiled application.
An mpi:MPIApplication element contains all the elements defined in jsdl-posix:POSIXApplication as well as the following additional elements. MPI application support is only currently implemented in the Globus 2.4.3 DRMConnector.

```
..
<jsdl:Application>
..
<mpi:MPIApplicationxmlns:mpi="urn:gridsam:mpi">
<jsdl-posix:*/>*
<mpi:ProcessorCount>xsd:positiveInteger</mpi:ProcessorCount>
</mpi:MPIApplication>
..
</jsdl:Application>
..
```

mpi:ProcessorCount: The number of processor to be used for the MPI application.

This functionality will be mapped to the SPDM extensions shortly.

3.8.5.2 MyProxy Authentication

GridSAM introduces a non-standard JSDL extension so that user can specify a MyProxy credential to be passed to a GridSAM instance in order for it to interact with Globus 2.4.3 based compute and file resources.

```
..
<jsdl:JobDefinition>
<jsdl:JobDescription>
        ...
</jsdl:JobDescription>
<myproxy:MyProxyxmlns:myproxy="urn:gridsam:myproxy">
<myproxy:MyProxyServer>xsd:string</myproxy:MyProxyServer>
<myproxy:ProxyServerDN>xsd:string</myproxy:ProxyServerDN>?
<myproxy:ProxyServerPort>xsd:positiveInteger</myproxy:ProxyServerPort>?
<myproxy:ProxyServerUserName>xsd:string</myproxy:ProxyServerUserName>
<myproxy:ProxyServerPassPhrase>xsd:string</myproxy:ProxyServerPassPhrase>
<myproxy:ProxyServerLifetime>xsd:int</myproxy:ProxyServerLifetime>?
</myproxy:MyProxy>
</jsdl:JobDefinition>
..
```

myproxy:MyProxyServer: MyProxy server hostname

myproxy:ProxyServerDN: Expected MyProxy server distinguished name so GridSAM can authenticate the server.
myproxy:ProxyServerPort: MyProxy server port. The default is 7512.
myproxy:ProxyServerUserName: The MyProxy username
myproxy:ProxyServerPassPhrase: The plain-text MyProxypassphrase. User should authenticate and authorise the GridSAM server before passing this information across the network.
myproxy:ProxyServerLifetime: The lifetime of the delegated proxy retrieved from the MyProxy server

3.8.6 Participation in interoperability tests:

GridSAM participated in HPC Profile Interoperability demonstrations at both SuperComputing'06 and SuperComputing'07.
Details and results of the interoperability tests:
SC07 - http://forge.ogf.org/sf/go/projects.ogsa-hpcp-wg/wiki
SC06 - http://forge.ogf.org/sf/wiki/do/viewPage/projects.ogsa-hpcp-wg/wiki/SC2006WikiPage

3.8.7 Security:

Authentication:
GridSAM relies on the OMII-UK Container to authenticate and obtain the credentials of the user submitting the JSDL document using transport-level or message-level security.
By default, GridSAM is preconfigured to perform WS-Security signature verification on all request messages and signing of all response messages. HTTPS can optionally be used to encrypt the data stream. WS-Security, together with HTTPS, provides strong encryption at the transport level and verification of message signature at the message level.
GridSAM also supports HTTPS mutual authentication without WS-Security.

Authorization:
The Authorisation sub-system in GridSAM provides fine-grain control of who (the distinguished name of the subject who submits the job) can submit what job (the structure of the JSDL description). The default allows any authenticated users to submit any job.
The configuration defines 'deny' and 'allow' rules that apply to the user identity (their authenticated distinguished name) and the structure of the job description (XPath pattern matching). Submission requests that match the deny directive and do match the allow directive will be permitted access. Requests that does not match the deny directive will allow access without evaluating the allow directive.

In the HPC Profile Interoperability test at SuperComputing'07 a new XML element 'Credential' was used as an extension to JSDL elements 'Application' and 'DataStaging'.
The purpose of the element was to allow for different user credential data to be used at different stages of a JSDL document processing: stage-in input files; job execution; stage-out output files.

Appendix 3.9 NorduGrid / KnowARC

Contact: Gabor Roczei
Date: 2008-07-03
Website: www.nordugrid.org

3.9.1 Introduction

Next generation ARC (ARC1) uses JSDL as the native job description language among execution management services. JSDL is used for communication among ARC components dealing with job instances (e.g. job submission client and A-REX). End-users are not expected specifying their jobs using JSDL directly. The users may use other job descriptions, such as xRSL, JDL.

Please refer to the following URL for more information (standard conformance roadmap second release):

http://www.knowarc.eu/documents/Knowarc_D3.3-1_08.pdf

3.9.2 Implemented JSDL Elements:

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
JobIdentification				
JobName	X			
JobAnnotation			X	It is deprecated (JSDL errata)
JobProject		X		We plan to use it in the future
Application	X			
ApplicationName		X		Not parsed and no plan to use it
ApplicationVersion		X		Not parsed and no plan to use it
Resources	X			
CandidateHosts		X		We plan to use it in the future
HostName		X		We plan to use it in the future
FileSystem		X		We plan to use it in the future
MountPoint		X		Not parsed and no plan to use it
MountSource		X		Not parsed and no plan to use it
DiskSpace		X		We plan to use it in the future
FileSystemType		X		Not parsed and no plan to use it
ExclusiveExecution		X		We plan to use it in the future
OperatingSystem		X		We plan to use it in the future
OperatingSystemType		X		We plan to use it in the future
OperatingSystemName		X		We plan to use it in the future
OperatingSystemVesion		X		We plan to use it in the future
CPUArchitecture		X		We plan to use it in the future
CPUArchitectureName		X		We plan to use it in the future
IndividualCPUSpeed		X		Not parsed and no plan to use it
IndividualCPUTime	X			
IndividualCPUCount	X			
IndividualNetworkBandwidth		X		Not parsed and no plan to use it
IndividualPhysicalMemory	X			
IndividualVirtualMemory		X		We plan to use it in the future
IndividualDiskSpace		X		We plan to use it in the future
TotalCPUTime	X			
TotalCPUCount	X			
TotalPhysicalMemory	X			
TotalVirtualMemory		X		We plan to use it in the future
TotalDiskSpace		X		We plan to use it in the future
TotalResourceCount		X		We plan to use it in the future
DataStaging	X			
FileName	X			
FilesystemName			X	HPC File Staging Profile does not support this element
CreationFlag		X		We plan to use it in the future
DeleteOnTermination		X		We plan to use it in the future
Source	X			
Target	X			

JSDL Element	*Yes*	*No*	*N/A*	*Comments*
POSIXApplication	X			
Executable	X			
Argument	X			
Input	X			
Output	X			
Error	X			
WorkingDirectory			X	The A-REX's job is running in the session directory
Environment		X		We plan to use it in the future
WallTimeLimit	X			
FileSizeLimit		X		Not parsed and no plan to use it
CoreDumpLimit		X		Not parsed and no plan to use it
DataSegmentLimit		X		Not parsed and no plan to use it
LockedMemoryLimit		X		Not parsed and no plan to use it
MemoryLimit	X			
OpenDescriptorsLimit		X		Not parsed and no plan to use it
PipeSizeLimit		X		Not parsed and no plan to use it
StackSizeLimit		X		Not parsed and no plan to use it
CPUTimeLimit	X			
ProcessCountLimit		X		Not parsed and no plan to use it
VirtualMemoryLimit		X		We plan to use it in the future
ProcessCountLimit		X		Not parsed and no plan to use it
ThreadCountLimit		X		Not parsed and no plan to use it
UserName			X	We cannot use it on grid level (security reason)
GroupName			X	We cannot use it on grid level (security reason)
HPCProfileApplication	X			
Executable	X			
Argument	X			
Input				
Output	X			
Error	X			
WorkingDirectory			X	The A-REX's job is running in the session directory
Environment		X		We plan to use it in the future
UserName			X	We cannot use it on grid level (security reason)

3.9.3 Other problems encountered:

Due to its deliberately limited scope there are numerous extensions emerged. “Raw JSDL” alone is hardly used in production deployments. The rather ambiguous semantics and the limited scope of JSDL elements necessitate the creation of profiles and extensions. In conformant to the NorduGrid development plans, it can be stated that JSDL was not designed and is not suitable to be exposed directly to end-users.

3.9.4 Mappings to existing systems:

ARC1 comes with a general-purpose client library which can perform translations among different job descriptions, for example: from xRSL to JSDL. The client library and the translator modules are modular therefore we can easily add a new job description handler to the system.

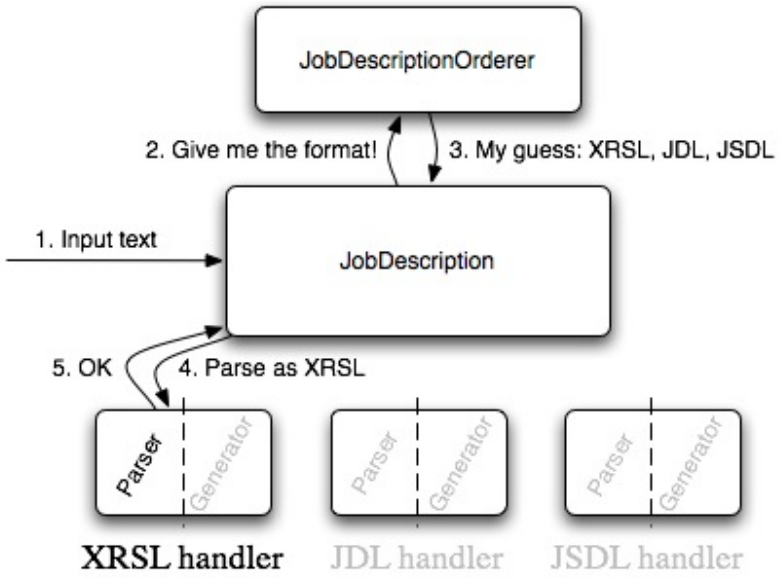

Figure 1 Source Parsing

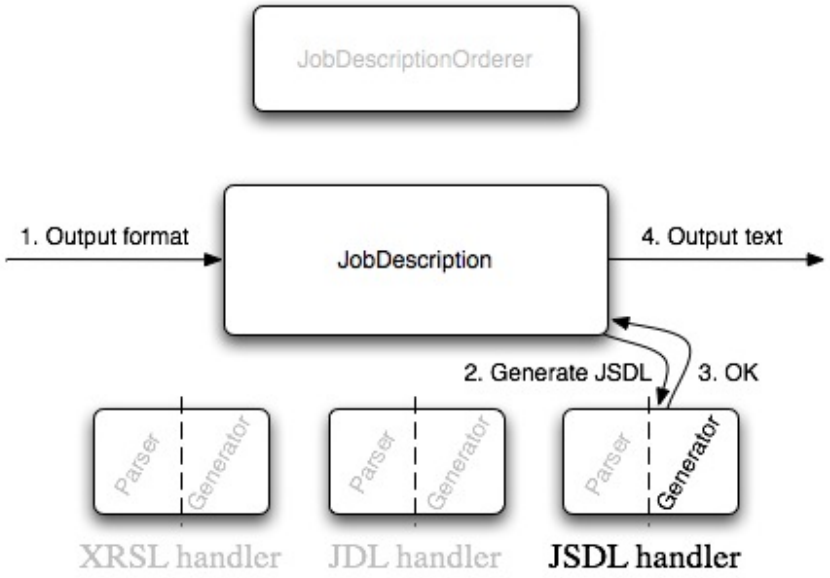

Figure 2 Product generation

Svn directory of this translator:
http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/libs/client

3.9.5 Enhancements:

To overcome the limited scope of JSDL version 1.0 NorduGrid found necessary the introduction of the following extensions: jsdl-arc:CredentialServer, jsdl-arc:RemoteLogging, jsdl-arc:Reruns, jsdl-arc:Notify, jsdl-arc:ProcessingStartTime, jsdl-arc:AccessControl, jsdl-arc:LocalLogging within the JobDescription element. The jsdl-arc:GridTimeLimit, jsdl-arc:RunTimeEnvironment, jsdl-arc:Middleware, jsdl-arc:CandidateTarget with jsdl-arc:HostName and jsdl-arc:QueueName, jsdl-arc:SessionFileTimePart of the Resources element. And the DataStaging element was extended by the jsdl-arc:IsExecutable subelement. Other elements: jsdl-arc:ProcessingStartTime and jsdl-arc:GridTimeLimit.

The NorduGrid extensions, the xRSL to JSDL mapping, are described in the NorduGrid manual (NORDUGRID-MANUAL-4):

http://www.nordugrid.org/documents/xrsl.pdf

3.9.6 Participation in interoperability tests:

SC07 interoperability test:
http://forge.ogf.org/sf/wiki/do/viewPage/projects.ogsa-hpcp-wg/wiki/SC07Demos

OGF23 exercise/interoperability test:
http://ogf.org/gf/event_schedule/index.php?id=1369

3.9.7 Security:

In general, both transport level security and message level security are developed and supported by ARC1. For transport level security, TLS/SSL is supported by implementing a pluggable component for communication processing in transport level. For message level security, two WS-Security specific profiles (UsernameToken and X509Token) are developed for SOAP message level authentication, integrity and confidentiality, the SAMLToken could be available later.

ARC1 does not include any security information in JSDL document so far. JSDL is supposed to be one part of SOAP body, so it is not necessary to consider security specifically for JSDL while we already have message level and transport level security.

Appendix 3.10 GRIA

Contact: Mike Boniface
Date: 6th July 2008

3.10.1 Introduction

The GRIA Job service architecture is shown in the figure below. The diagram shows where JSDL is used between components and how underlying resource management platforms are integrated.

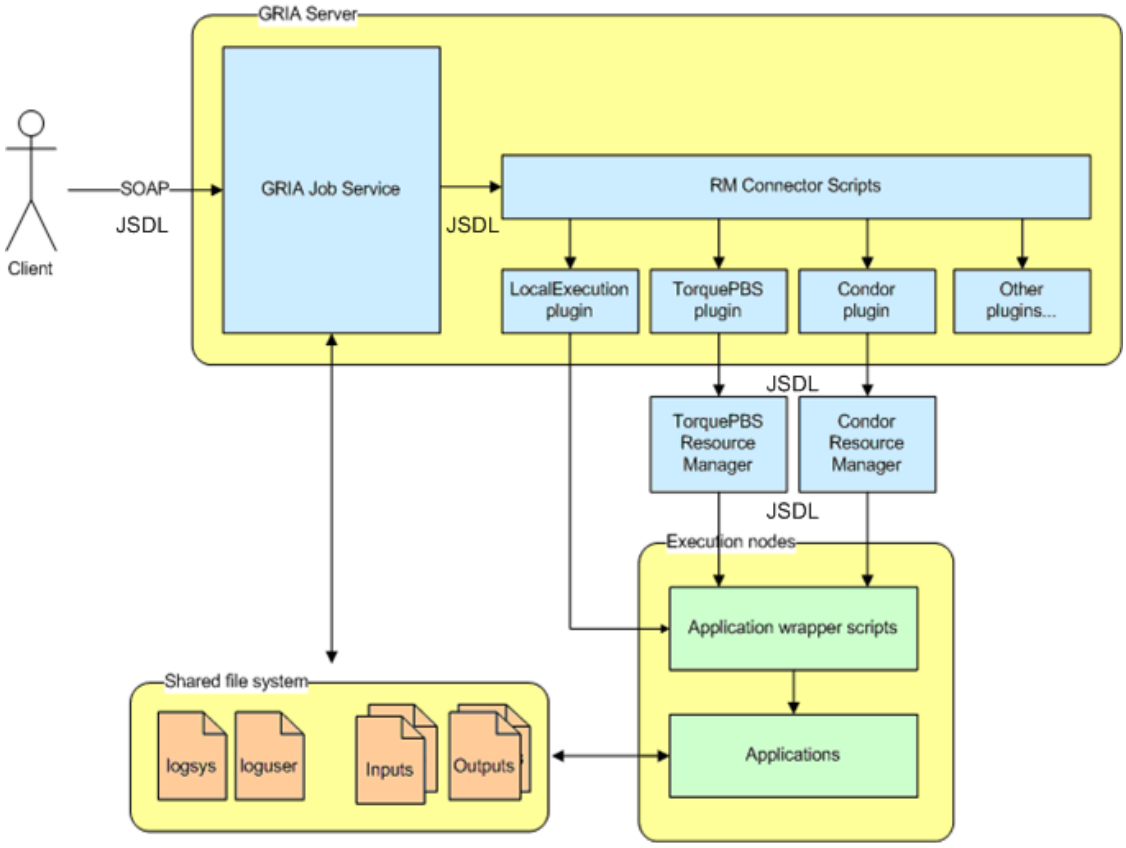

Figure 3 JSDL usage

The JSDL specification has been developed from existing Grid middleware and resource management technologies. It therefore provides a good coverage of possible computational requirements but there are some implicit assumptions about the lifecycle of persistent entities related to jobs such as data and projects that had to be considered in GRIA.

JSDL assumes that users of a JSDL job service have access to a physical file systems mounted with the computational nodes and that data can persist on these file systems beyond the lifetime of the job. This implies that JSDL jobs can have persistent side affects and that there is a workflow that goes beyond JSDL's staging in and staging out process.
JSDL assumes that access to these "other" files systems can be achieved through different communicate layer (not specified) and access control is based on O/S account permissions associated with the executing job. The lifecycle, security and management of these side-effects are not covered by the specification.

The JSDL specification works if users are given O/S accounts on computational nodes and storage devices however we do not do this with GRIA's Software-as-a-Service model. Therefore, we had to define a profile on JSDL that is acceptable for industry where the service provider runs jobs on the behalf of users and restrict the commands that can be executed based on requirements in the JSDL document (Application, SLA Metrics) rather than using the operating system commands. In addition, GRIA does not allow direct access to file systems but provides a logical mapping between EPRs and filenames. Therefore, data staging references needed to be extended to include references to EPRs.

GRIA's job service needed to check the user's usage requirements defined within the JSDL document against higher-level agreements (SLAs) to ensure that the user has sufficient resource to execute the job. Jobs could only be executed within a temporary "sandboxed" file space by the service provider. No O/S accounts are assigned to specific users and therefore persistent side-affects in file systems outside of the temporary space could not be supported. Input data should be staged in and output data staged out to managed storage locations that are either collocated with computational node or at a remote location. When a Job is destroyed the temporary space is deleted removing all associated data.

Considering these requirements, only the JSDL "TMP" file system types is supported rather than "SPOOL" or "NORMAL", which can both be used to store data after job termination.

JSDL assumes that jobs can belong to a project; however, there are no details on how a service provider should support a project's lifecycle or how to add a job to a project, therefore this is not supported

Apart from the restrictions defined above, JSDL provides a good specification for describing job requirements for GRIA applications. However, we note that focusing on just job submission rather than the definition of a "data processing workflow" means that data lifetime and relationship with other specifications initiatives such as OGSA-DAI needs to be resolved. We clearly need a specification that covers data processing, transfer and storage actions and is explicit about the lifecycle of all resources used.

3.10.2 Implemented JSDL Elements:

JSDL Element	Yes	No	N/A	Comments
JobIdentification	yes			
JobName	Yes			Used to set the label of the job resource that is created
JobAnnotation				
JobProject				
Description				
Application	yes			
ApplicationName	yes			Used to set the GRIA application URI
ApplicationVersion				
Resources	yes			
CandidateHosts	yes			Torque/PBS, Condor plugins
HostName				
FileSystem				
MountPoint				
MountSource				
DiskSpace	yes			
FileSystemType				
ExclusiveExecution				
OperatingSystem	yes			
OperatingSystemType				
OperatingSystemName				
OperatingSystemVersion				
CPUArchitecture	yes			
CPUArchitectureName				
IndividualCPUSpeed	yes			
IndividualCPUTime	yes			
IndividualCPUCount	yes			
IndividualNetworkBandwidth				
IndividualPhysicalMemory	yes			
IndividualVirtualMemory	yes			
IndividualDiskSpace				
TotalCPUTime	yes			

JSDL Element	Yes	No	N/A	Comments
TotalCPUCount	yes			
TotalPhysicalMemory	yes			
TotalVirtualMemory	yes			
TotalDiskSpace	yes			
TotalResourceCount				
DataStaging	yes			One DataStaging element should be provided for each input or output your job requires, "name" attribute Should match one of the input/output names in the application metadata. If the metadata describes an array, the name should have a numerical suffix, indicating which element of the array the stager represents (eg. inputarray-0, inputarray-1, etc.)
FileName	yes			Used as above if the "name" attribute is not specified
FilesystemName				
CreationFlag				
DeleteOnTermination				
Source				
Target				
URI				
POSIXApplication	yes			
Executable				
Argument	yes			Specifies a single commandline argument to pass to the application wrapper scripts
Input				
Output	yes			LSF plugin
Error	Yes			LSF plugin
WorkingDirectory	Yes			LSF plugin
Environment				
WallTimeLimit				
FileSizeLimit	yes			LocalExecution plugin on POSIX
CoreDumpLimit	Yes			LocalExecution plugin on POSIX
DataSegmentLimit	yes			LocalExecution plugin on POSIX
LockedMemoryLimit	yes			LocalExecution plugin on POSIX
MemoryLimit				
OpenDescriptorsLimit	yes			LocalExecution plugin on POSIX
PipeSizeLimit				
StackSizeLimit	yes			LocalExecution plugin on POSIX
CPUTimeLimit	yes			LocalExecution plugin on POSIX
ProcessCountLimit				
VirtualMemoryLimit	yes			
CPUTimeLimit	yes			
ProcessCountLimit				
VirtualMemoryLimit				
ThreadCountLimit				
UserName	yes			LSF plugin
GroupName				
HPCProfileApplication				
Executable				
Argument				
Input				
Output				
Error				
WorkingDirectory				

JSDL Element	Yes	No	N/A	Comments
Environment				
UserName				

3.10.3 Other problems encountered:

3.10.4 Mappings to existing systems:

We have mapped GRIA's implementation to Platform LSF and CNGrid GOS (which is actually based on GridSAM). We did encounter problems between GRIA and Platform LSF but this was resolved with a patch from Platform.

3.10.5 Enhancements:

Yes, we did not extended data staging to allow references to EPRs rather than just URI's

3.10.6 Participation in interoperability tests:

No, we did not participate in JSDL interoperability tests. Interoperability has been driven by customer needs for integration between different technologies supporting the JSDL specification.

3.10.7 Security:

The security model for JSDL is based on GRIA's security model and implementation decisions about the implementation of the job submission. Key differences to other JSDL implementations is that GRIA only allows consumers to execute a managed set of applications published by the service provider and does not allow consumers to submit JSDL to execute arbitrary applications. In addition it is the consumer's responsibility to stage the data in and out and therefore we avoid having to deal with staging workflows within the job service and the security consequences of this.