Editor:
Andreas Savva

Authors:
Ali Anjomshoaa
Fred Brisard
An Ly
Stephen McGough
Darren Pulsipher

6 June 2004

Job Submission Description Language (JSDL) Specification

Version 0.4

Status of this Memo

This document is a draft of the Job Submission Description Language (JSDL) Specification from the Global Grid Forum (GGF). This is a public document being developed by the participants of the JSDL-Working Group (WG) of the Scheduling and Resource Management (SRM) Area of the GGF.

This document is being constantly updated. The latest version can be found at:
https://forge.gridforum.org/projects/jsdl-wg/document/draft-ggf-jsdl-spec/en/

Abstract

This specification document details the semantics and structure of the Job Submission Description Language (JSDL), which is used to describe the requirements of computational jobs for submission to resources, particularly in Grid environments, though not restricted to the latter. The normative XML schema for the JSDL is contained in the Appendix, along with examples of JSDL documents based on this schema.

GLOBAL GRID FORUM
office@gridforum.org
www.ggf.org

Full Copyright Notice

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contact information at GGF website).

Contents

1 Introduction

The Job Submission Description Language (JSDL) is a language for describing the requirements of computational jobs for submission to Grids and other systems. The language contains a vocabulary that facilitates the expression of those requirements as a set of JSDL job attributes. The structure of the language provides the grammar, and therefore, the syntax for the JSDL.

It is clear that there is a need for the JSDL as a language that will enable jobs to be described in a standard way so that their description may be ported to and understood by different systems. These systems, such as batch management systems, will typically have their own means and languages for describing computational jobs. For a user to be able to make use of multiple systems, therefore, it is currently necessary for them to have several job descriptions, one for each of the proprietary systems that they wish to use.

Moreover, in Grid environments, which involve the interaction of a number of proprietary or non-proprietary systems in a heterogeneous environment, it is essential that there is a standard intermediary language for the description of computational jobs, which can be understood by many systems. It will be necessary for implementers of those systems, to provide the necessary means by which to translate the JSDL into the proprietary language for that system, or to be able to parse and utilise the JSDL in its native form. This should then enable many existing systems in complex heterogeneous environments, such as computational Grids, to make use of a single job description for managing that job.

The JSDL language attributes, therefore, will allow the description of the requirements of computational jobs. These requirements broadly fall into several categories, including:

- Job Identification requirements;
- Resource requirements;
- Basic Scheduling requirements;
- Data requirements; and
- Security requirements.

Once the necessary job requirements have been captured in a JSDL document, the document can be submitted for job management and execution in potentially complex and heterogeneous environments. The JSDL document is a job template. A JSDL document, therefore, can be submitted for multiple job instances, leaving the identification of individual job instances to the underlying job management systems.

The job management systems that may consume a JSDL document may include:

- Job Submission Clients;
- Job and Resource Schedulers;
- Job and Resource Brokers;
- Accounting Systems;
- Security Systems;
- Archiving Systems; and
- Provenance Systems.

2 Notational Conventions

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119 [RFC 2119].

More conventions used in this document…

3 Setting the Context

[This section should include a brief description of computational Grids and the motivation behind them.]

[AA – Do we really need this?]

[AS: We should a section about the positioning and relation of JSDL relative to other efforts such as WS-Agreement, CDDLM, etc.]

3.1 Computational Grids

[AA – Do we really need this?]

3.2 Resource Management in Heterogeneous Grids

[The issues of resource management within a heterogeneous Grid should be outlined and briefly discussed in this section, with references to GGF, and other, documents.]

[AA – Do we really need this?]

3.3 Grid Languages as Common Interfaces in a Heterogeneous Environment

[JSDL motivation.]

4 Computational Jobs

A Computational Job is a uniquely identifiable task, or a number of tasks running under a workflow system, which may involve the execution of one or more processes or computer programs.

4.1 Computational Job Requirements

Computational Jobs have a number of requirements that will need to be satisfied in order for them to be executed accordingly. These requirements will include hardware and software requirements, as well as a user’s, or application’s, temporal and other requirements for the job that they are specifying and submitting for execution.

4.2 Satisfying Job Requirements

Satisfying job requirements will involve matching the requirements of a job to the status, capabilities and specifications of available computational resources, which can then be reserved and assigned for the execution of that job, in addition to the satisfaction of non-computing requirements such as scheduling requirements.

5 The JSDL Structure

The JSDL structure is about the schema for the language, that is, how parameters and their values are categorised and syntactically expressed etc. …(Steve to write more…?)

6 The JSDL Core Attribute Set

The JSDL core attribute set contains the semantics for attributes that must be understood by JSDL compliant systems. It is not necessary that all of the core attributes be satisfied by those system. They must, however, be able to parse and handle all of the core attributes.

6.1 Job Identity Attributes

6.1.1 JobAnnotation

Deleted: Name

Definition: A string which MAY be specified by a 'user' to annotate the 'job'. It is not unique to a particular 'job', which means that a 'user' MAY specify the same 'JobAnnotation' for multiple 'jobs'.

Deleted: identify

Deleted: by 'name'

Deleted: and

Deleted: Name'

Multiplicity: {0..1}

Mutability:

Deleted: MUST not change for the 'lifetime' of the 'job' for which it is specified.

Usage: MAY be used for search and sort purposes.

6.1.2 SubmittingUserName

Definition: A string specifying the 'username' of the 'user' who is specifying the 'job', in the 'submitting environment'. A system must be able to map the 'SubmittingUserName' to a real 'user name'.

Multiplicity: {0..1}

Mutability:

Deleted: MUST not change for the 'lifetime' of the 'job' for which it is specified.

Usage: MAY be used for access control and accounting mechanisms.

6.1.3 ExecutingUserName

[AA – Do we need this?]

Definition: A string specifying the 'username' of the 'user' who is specifying the 'job', in the 'executing environment'.

Multiplicity: {0..n}

Mutability: MAY change for the 'job' for which it is specified (to support migration of the 'job' between different 'execution environments').

Deleted: the 'lifetime' of

Usage: MAY be used for access control and accounting mechanisms.

6.1.4 UserGroupName

Definition: A string specifying the 'group' of the 'user' who is specifying the 'job' (like the Unix group identifier (gid)).

The 'UserGroupName' attribute MAY represent the name of a 'Virtual Organisation' to which the 'user' belongs.

Multiplicity: {0..1}

Mutability: MAY change for the 'job' for which it is specified (to support migration of the 'job' between different 'execution environments').

Deleted: the 'lifetime' of

Usage: MAY be used for access control and accounting mechanisms.

[Is there a need for separate 'SubmittingUserGroupName' and 'ExecutingUserGroupName' to make the distinction between the 'user's' 'group' in the 'submitting environment' and the 'executing environment', respectively?]

[Alternatively, should we fold the 'GroupName' and 'UserName' attributes into the 'SubmittingUserName' and 'ExecutingUserName' attributes, which would then allow the specification of a user's username and group name in each environment respectively?]

[AA – Do we need specify user or group ID in the executing environment, or will this be mapped by the system from the submitting user and group ID?]

6.1.5 JobGroupName

Definition: A string specifying the 'group' to which the 'job' belongs. This attribute can be used when a 'user' 'submits' a 'job' on behalf of a 'group' to which the 'user' does not belong. In that case, the 'UserGroupName' is not the same as the 'JobGroupName'.

Deleted: , i.e. in such an instance

The 'JobGroupName' attribute MAY represent the name of a 'Virtual Organisation' to which the 'job' belongs.

Multiplicity: {0..1}

Mutability: MUST not change for the 'job' for which it is specified.

Deleted: the 'lifetime' of

Usage: The 'JobGroupName' attribute MAY be used for access control and accounting mechanisms.

6.1.6 JobProjectName

Definition: A string specifying the 'project' to which the 'job' belongs.

Multiplicity: {0..1}

Mutability: MUST not change for the 'job' for which it is specified.

Deleted: the 'lifetime' of

Usage: The 'JobProjectName' attribute MAY be used for access control and accounting mechanisms.

6.1.7 JobCategory

Definition: A string specifying the 'category' or 'type' of 'job' that is specific to a particular set of 'scheduling system' or 'execution system' 'policies'.

Multiple instances of 'JobCategory' are allowed for each job description. This should fulfill the need for the job to fall into a number of' categories'. The user MUST be able to specify the priority between the 'JobCategory' instances for the 'job' that they are specifying for 'submission'.

The 'JobCategory' attribute SHOULD be any string that is understood by a 'scheduling system' or an 'execution system', that MAY enable that system to associate a pre-defined set of 'policies' with the 'job'.

The 'JobCategory' attribute SHOULD be any string that is understood by a 'scheduling system' or an 'execution system', that MAY enable that system to associate a pre-defined set of 'resource requirements' with the 'job'.

The 'policies' and 'resource requirements' that a 'system' may associate with the 'job' according to the 'JobCategory' attribute, MAY be 'use domain' specific.

The 'policies' and 'resource requirements' that a system may associate with the 'job' according to the 'JobCategory' attribute, MAY be 'scheduling system' or 'execution system' specific.

Multiplicity: {0..n}

Mutability: The 'JobCategory' instances MUST not change for the 'job' for which they are specified.

Deleted: the 'lifetime' of

Usage: The 'JobCategory' attribute MAY be used for access control and accounting mechanisms.

6.2 Resource Attributes

The resource attributes are described in groups for clarity only. Actual structure will be defined in the schema. It is anticipated that most attributes will be at the base level.

6.2.1 Hardware Requirements

This section describes the attributes needed for the target environment.

6.2.1.1 Host

Definition: A string specifying the 'name' of the 'host' required by the 'job' in the 'execution environment'. This attribute specifies a physical machine name or a logical group of physical machines.

[AA – can the Host attribute be an IP address?]

Multiplicity: {0..n}

Usage: The 'Host' attribute contains the following properties:

Description: A string specifying a 'description' of the 'host' required by the 'job' in the 'execution environment'. The 'user' or 'execution environment' may use this property to provide more descriptive information about the 'host'. Also, this property may be used to further qualify the 'host' required by the 'job'.

Affinity: An indicator that identifies the preference for a given host. Possible values of high and low.

[AA – can we use an integer index to specify preference? e.g. 1=most significant, 1,000,000=least significant]

Example:

```
<host>hostname affinity=high</host>
```

6.2.1.2 CPU

Definition: A string specifying the 'type' of the 'processor' required by the 'job' in the 'execution environment'.

Multiplicity: {0..n}

Usage: The 'CPU' attribute SHOULD be any string that is understood by the 'execution environment', that MAY enable that environment to identify the 'type' of 'processor' required by the 'job'.

Usage: The 'CPU' attribute contains the following properties:

Description – A string specifying a 'description' of the 'processor' required by the 'job' in the 'execution environment'. This property may be used to further qualify the 'processor' required by the 'job'.

Count – A string specifying the 'number' of 'processors' required by the 'job' in the 'execution environment'. This property may be used to specify an exact number, a minimum number, or a maximum number.

Speed – A string specifying the 'speed' of the 'processor' required by the 'job' in the 'execution environment'. This 'speed' is specified in 'MIPS' (millions of instructions per second. This property may be used to specify an exact speed, a minimum speed, or a maximum speed.

Example:

Notes:

Q. How is a request for 2 cpus if it's a fast box and 4 cpus if it's a slower box?

A. The description of the processor should identify a fast or slower processor. Code 2 CPU attributes where one attribute specifies the fast cpu and a count of two. Code the other attribute with the slower cpu and a count of 4.

[AA – can we apply a preference to which of "2 fast processors" OR "4 slow processors" to use for the job? May be using an integer index as suggested for the Host attribute.]

6.2.1.3 PhysicalMemory

Definition: A string specifying the 'amount' of 'physical memory' required by the 'job' in the 'execution environment'.

[AA – Should we say "RAM memory" or even "physical RAM memory"?]

Multiplicity: {0..1}.

Usage: The 'Physical Memory' attribute contains the following properties:

Description – A string specifying the amount of 'PhysicalMemory'. This amount is specified as a value combined with a units specification (K, M, G, T)

[AA – K, M, G, T Bytes or bits?]

Example:

```
<PhysicalMemory>4000 Units=M Constraint=Min</PhysicalMemory>
```

6.2.1.4 Network

Definition: A string specifying the 'type' of the 'network connection' required by the 'job' in the 'execution environment'. This represents the type of network connections required for the job to run on the target machine. It is distinct from the Data Attributes since they refer to connections required to transfer data before and after execution.

Multiplicity: {0..n}

Usage: The 'Network' attribute SHOULD be any string that is understood by the 'execution environment', that MAY enable that environment to identify the 'type' of 'network connection' required by the 'job'.

[AS: Is it possible to use this attribute to describe a pool of resources, e.g., a cluster.]

The 'Network' attribute contains the following properties:

Description – A string specifying a 'description' of the 'network connection' required by the 'job' in the 'execution environment'. This property may be used to further qualify the 'network connection' required by the 'job'.

Count – A string specifying the 'number' of 'network connections' required by the 'job' in the 'execution environment'. This property may be used to specify an exact number, a minimum number, or a maximum number.

Bandwidth – A string specifying the 'bandwidth' of the 'network connection' required by the 'job' in the 'execution environment'. This 'bandwidth' is specified in 'MBps' (megabytes per second) or 'KBps' (kilobytes per second). This property may be used to specify an exact number, a minimum number, or a maximum number.

[AA – why not use the same unit specification as for physical memory, i.e. K, M, G, T?]

Latency – A string specifying the 'latency' of the 'network connection' required by the 'job' in the 'execution environment'. This 'latency' is specified in milliseconds. This property may be used to specify an exact number, a minimum number, or a maximum number.

6.2.1.5 FileSystemSpace

Definition: A string specifying the 'file system' required by the 'job' in the 'execution environment'. This attribute specifies a mount point, directory path or drive letter.

Multiplicity: {0..n}

Usage: The 'FileSystemSpace' attribute contains the following properties:

DiskSpace - A string specifying the 'amount' of 'disk space' on the 'file system' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification (K, M, G, T). This property may be used to specify an exact amount, a minimum amount, or a maximum amount.

SwapSpace – A string specifying the 'amount' of 'swap space' on the 'file system' required by the 'job' in the 'execution environment'. This 'amount is specified as a value combined with a units specification (K, M, G, T). This property may be used to specify an exact amount, a minimum amount, or a maximum amount.

TemporarySpace - A string specifying the 'amount' of 'temporary space' on the 'file system' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification (K, M, G, T). This property may be used to specify an exact amount, a minimum amount, or a maximum amount.

Example:

```
<FileSystemSpace>
      <DiskSpace 4000 Units=M Constraint=Min</DiskSpace>
      <SwapSpace 2000 Units=M Constraint=Min</SwapSpace>
      <TemporarySpace 4000 Units=M Constraint=Min</TemporarySpace>
</FileSystemSpace>
```

6.2.1.6 HostUsage

Definition: Specifies whether this job shares Hosts with other jobs.

Multiplicity: {0..1}

Usage: The 'HostUsage' attribute MAY be used for resource allocations.

Usage: {shared | not_shared | slice_not_shared}
shared - Specifies that nodes can be shared with other jobs. (default)
not_shared - Specifies that nodes are not shared: no other jobss are scheduled on this node during its execution. Exclusive execution
slice_not_shared - Specifies that a job will not share nodes with other jobs when it is running in its own time-slice. When the time slice ends, other jobs can run.

[AS: Discuss the need for reservation period, and in the case of already having a reservation, a reservation id.]

6.2.2 Environment Variables

6.2.2.1 EnvironmentVariable

Definition: A complex type specifying the 'name' and 'value' of an 'environment variable' that will be defined in the 'execution environment'.

Multiplicity: {0..n}

Usage: It is RECOMMENDED that only one instance of the 'EnvironmentVariable' attribute appear for each unique 'name'.

6.2.3 Software Requirements

6.2.3.1 OperatingSystem

Definition: A string specifying the 'name' of 'operating system' required by the 'job' in the 'execution environment'.

Multiplicity: {0…n}

Usage: The 'OperatingSystem' attribute contains the following properties:

Description – A string specifying a 'description' of the 'operating system' required by the 'job' in the 'execution environment'. The 'user' or 'execution environment' may use this property to provide more descriptive information about the 'operating system'. Also, this property may be used to further qualify the 'operating system' required by the 'job'.

Version - A string specifying the 'version' of 'operating system' required by the 'job' in the 'execution environment'.

6.2.3.2 VirtualMemory

Definition: A string specifying the 'amount' of 'virtual memory' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification.

Multiplicity: {0..1}

Usage: The 'VirtualMemory' attribute may be used to specify an exact amount, a minimum amount, or a maximum amount.

6.2.3.3 DataSegmentSize

Definition: A string specifying the 'amount' of 'data segment memory' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification.

Multiplicity: {0..1}

Usage: The 'DataSegmentSize' attribute may be used to specify an exact amount, a minimum amount, or a maximum amount.

6.2.3.4 Priority

Definition: A string specifying the 'priority' required by the 'job' in the 'execution environment'.

Multiplicity: {0..1}

Usage: The 'Priority' attribute SHOULD be any string that is understood by an 'execution environment', that MAY enable that environment to assign a 'priority' to the 'job'.

6.2.3.5 CoreDumpSize

Definition: A string specifying the maximum 'size' of any 'core dump file' created by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification.

Multiplicity: {0..1}

6.2.3.6 FileSizeLimit

Definition: A string specifying the maximum 'size' of any 'file' created by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification.

Multiplicity: {0..1}

6.2.3.7 CPUTimeLimit

Definition: A string specifying the maximum 'amount' of 'processor time' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification.

Multiplicity: {0..1}

6.2.3.8 WallTimeLimit

Definition: A string specifying the maximum 'amount' of 'wall clock time' required by the 'job' in the 'execution environment'. This 'amount' is specified as a value combined with a units specification

Multiplicity: {0..1}

6.2.3.9 ProcessQueue

Definition: A string specifying the 'name' of the 'queue' required by the 'job' in the 'execution environment'.

Multiplicity: {0..n}.

Usage: The 'ProcessQueue' attribute SHOULD be any string that is understood by the 'execution environment', that MAY enable that environment to assign the 'job' to a specific 'queue'.

6.2.4 Application Requirements

6.2.4.1 Executable

Definition: A string specifying the 'command' required by the 'job' in the 'execution environment'. In most cases, this attribute refers to a simple command. However, it is possible to specify a 'complex executable' using a combination of the properties below.

Multiplicity: The 'Executable' attribute occurs {1}.

Usage: The 'Executable' attribute contains the following properties:

Type – A string specifying the 'type' of 'command' required by the 'job' in the 'execution environment'. The default 'type' indicates a simple command. This property may contain different values for 'execution systems' that support many 'types' of 'execution'.

Description – A string specifying a 'description' of the 'command' required by the 'job' in the 'execution environment'. A 'complex executable' may use this property to provide a full 'description' of the 'steps' necessary for the 'execution' of the 'job'.

6.2.4.2 Argument

Definition: A string specifying the 'argument' required by the 'job' in the 'execution environment'. In most cases, this attribute refers to a simple command-line argument. However, it

is possible to specify any string that may represent a ‘complex argument’ for the ‘complex executable’ described in the ‘Executable’ attribute section.

Multiplicity: The ‘Argument’ attribute occurs {0..n}. This should satisfy the ability of the ‘job’ to accept multiple ‘arguments’.

6.2.4.3 StandardInput

Definition: A string specifying the ‘name’ of the ‘standard input file’ or ‘device’ required by the ‘job’ in the ‘execution environment’.

Multiplicity: The ‘StandardInput’ attribute occurs {0..1}.

6.2.4.4 StandardOutput

Definition: A string specifying the ‘name’ of the ‘standard output file’ or ‘device’ required by the ‘job’ in the ‘execution environment’.

Multiplicity: The ‘StandardOutput’ attribute occurs {0..1}.

6.2.4.5 StandardError

Definition: A string specifying the ‘name’ of the ‘standard error file’ or ‘device’ required by the ‘job’ in the ‘execution environment’.

Multiplicity: The ‘StandardError’ attribute occurs {0..1}.

6.2.4.6 InitialDirectory

Definition: A string specifying the starting ‘directory’ required by the ‘job’ in the ‘execution environment’.

Multiplicity: The ‘InitialDirectory’ attribute occurs {0..1}.

6.2.4.7 Shell

Definition: A string specifying the starting ‘command shell’ required by the ‘job’ in the ‘execution environment’.

Multiplicity: The ‘Shell’ attribute occurs {0..1}.

Usage: The ‘Shell’ attribute SHOULD be any string that is understood by the ‘execution environment’, that MAY enable that environment to provide the ‘command shell’ required by the ‘job’.

6.2.4.8 Log

Definition: A string specifying the ‘log’ required by the ‘job’ in the ‘execution environment’. In most cases, this attribute specifies a ‘log’ file.

Multiplicity: The ‘Log’ attribute occurs {0..1}.

Usage: The ‘Log’ attribute SHOULD be any string that is understood by the ‘execution environment’, that MAY enable that environment to provide the ‘log’ required by the ‘job’.

6.2.5 Resource Attributes Summary

6.3 Data Staging Attributes

Data attributes define the files that should be moved to the execution host (stage in) and the files that should be moved from the execution host (stage out). Files are staged in before the job starts executing. Files are staged out after the job terminates.

There is one declaration for each file that should be staged in or out. A file is staged in if a *source* declaration exists. A file is staged out if a *target* declaration exists. It is possible to stage in and then stage out a file by specifying both a Source and a Target.

6.3.1 File

The File attribute may appear {0..n} times. It consists of the following properties:

FileName: {1}. A string specifying the local name of the file (or directory) on the execution host. The FileName may be a file path. [TBD: Define format.]

FileSystemName: {0,1}. If the FileSystemName is specified then the FileName is relative to the specified FileSystem. (There must also be a separate matching FileSystem declaration.) Otherwise the FileName is relative to the initial job directory.

- This is a way to access local storage - not relative to the current path: eg: "scratch:/largeFile.dat" could be mapped to "z:\largeFile.dat" under Windows or "/tmp/largeFile.dat" under Linux

Source: {0..n}. Each source is a URL specifying location (and protocol) that can be used to stage in the file. If no source is specified then the file does not have to be staged in. If more than one source exists then any one may be used to retrieve the file. Sources should be tried in the order given.

Target: {0..n}. Each target is a URL specifying location (and protocol) that may be used to stage out the file. If no target is specified then the file does not have to be staged out. If more than one target exists then any one may be used to stage out the file. Targets should be tried in the order given.

Overwrite: {0,1}. Boolean. If true the file should be staged in regardless of whether a file with the same name exists. Otherwise the file is staged in only if it does not exist. If not defined behaviour is unspecified.

DeleteOnTermination: {0,1}. Boolean. If true the file is deleted after the job terminates or after it has been staged out. Otherwise the file remains on the execution host (subject to the persistency of the FileSystem it is on). If not defined behaviour is unspecified.

The ordering of the File attributes in the JSDL document is not significant. That is the order of the File attributes in the document does not imply any ordering in carrying out the different stage in (or stage out) operations.

A File declaration with the same FileName may appear more than once provided a different FileSystemName is used.

It is expected that in the XML schema definition the Source and Target have extensible content to allow for adding extra information, e.g., specifying an alternate identity for transfers.

More complex file transfers, for example, conditional transfers based on job termination status are out of scope. They may be supported through extensions.

[TBD:

- FileSet Attribute if needed
  - Probably not.
- C: Fred: Provisioning is more than just copying files, it's also installation and configuration.
  - Deployment and configuration is out of scope of JSDL. It is covered by CDDLM-WG. Add this text in an initial section on Relation of JSDL with other efforts..
- C: Recursive copying of files? Copy a directory means its contents also? (Yes)
- C: Shared file space for files and specific directory for others.[?]
- Q: Physical or logical filename? [?]
- C: Overwrite Attribute: file permissions/privileges for files when overwriting? Persist permissions of existing or staged file?
  - Discuss and decide on behaviour in simple case. Complex scenarios as extensions.
- Q: We need to address file priority for transfers.
  - Make sure schema can allow ordering of declaration; default is no ordering
- C: Require at least a "source" or a "target" on the File attribute.
- C: How does one express that one needs a file to be present but doesn't know if the file is there or not?
  - We could specify that this is the meaning associated with a declaration that only has a 'Filename' and no 'source.'
- Need to specify that a file exists or does not exist on a resource?
  - First part is covered by previous point. Second part seems to be going into data dependencies which are probably out of scope.]

6.4 Scheduling Attributes

[This section is under review.]

6.5 Security Attributes

[This section is TBD.]

7 The JSDL Extension Mechanism

[This section is TBD.]

8 Security Considerations

Security considerations are covered in the Security Attributes section.

9 Editor Information

Andreas Savva
Grid Computing and Bioinformatics Laboratory
Fujitsu Laboratories

4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Phone: +81-44-754-2628
Email: andreas.savva@jp.fujitsu.com

10 Contributors

We gratefully acknowledge the contributions made to this specification by Lee Cook, and Donald Fellows.

11 Acknowledgements

12 References

12.1 Normative References

12.2 Informative References

13 Appendix

13.1 Normative Schema

13.2 Translation Tables