GWD-R (draft-ggf-jsdl-spec-1.0 (draft 19))
Job Submission Description Language (JSDL) Specification
http://forge.gridforum.org/projects/jsdl-wg

Authors:
Ali Anjomshoaa, EPCC
Fred Brisard, CA
An Ly, CA
Stephen McGough, LeSC
Darren Pulsipher, Ovoca LLC
Andreas Savva, Fujitsu (Editor)

27 May 2005

Job Submission Description Language (JSDL) Specification, Version 1.0 (draft 19)

Status of this Memo

This document provides information to the Grid community regarding the specification of the Job Submission Description Language (JSDL). Distribution is unlimited.

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/

Copyright Notice

Abstract

This specification document details the semantics and structure of the Job Submission Description Language (JSDL). JSDL 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.

Contents

1 Introduction

The Job Submission Description Language (JSDL) is a language for describing the submission requirements of jobs to the Grid. The JSDL language contains a vocabulary that facilitates the expression of those requirements as a set of XML elements.

The JSDL is motivated by two main scenarios. First, many organizations operate a variety of job management systems. Each system has its own language for describing job submission requirements. In order to utilize all the different systems in their organization they have to prepare and maintain a number of different job submission documents, one for each system and all describing the same submission. A standardized language that can be easily mapped to the various systems would clearly alleviate this problem.

Second, Grid environments involve the interaction of a number of different systems in a heterogeneous environment. Figure 1 illustrates one possible view of a Grid environment. The description of a job submission (shown as a JSDL document in the figure) on the Grid may be transformed by intermediaries or refined further by information not available to the initial submitter. It may be passed on between systems or instantiated on resources matching the resource requirements. All these automatic interactions can be facilitated by a standard language that can be easily transformed. In addition such a language should also have well defined mappings to other more system specific languages.

Note that Figure 1 simply illustrates the places where a JSDL document may be used within a Grid environment. Usage of a JSDL document is not be limited to just these situations and many others may exist. For example, accounting systems, security systems, archiving systems, provenance (audit) systems may also consume JSDL documents.

The JSDL 1.0 provides the core set of vocabulary for describing a job for submission to the Grid. It is envisioned that other more complex functionality will be provided through standardized and non-standardized extensions to the core JSDL 1.0 specification.

JSDL version 1.0 elements fall broadly into the following categories:

- Job identification requirements;
- Resource requirements; and
- Data requirements.

The set of JSDL version 1.0 elements is restricted to the description of requirements at submission time. In other words, there are no elements defined in JSDL version 1.0 that contain information specific to a job after it has been submitted. Such information (e.g., unique job identifiers or job status values) typically maintained by underlying job management systems, may be available in separate documents or added to JSDL 1.0 compliant documents through extensions. Section 3 describes the rationale behind the decision to restrict the scope of JSDL version 1.0 in this way.

The JSDL version 1.0 language elements are defined in section 4 and 5 and the normative XML schema is provided in Appendix 2.

JSDL version 1.0 is extensible. Section 6 describes the extension mechanism. A normative extension is defined in section 7.

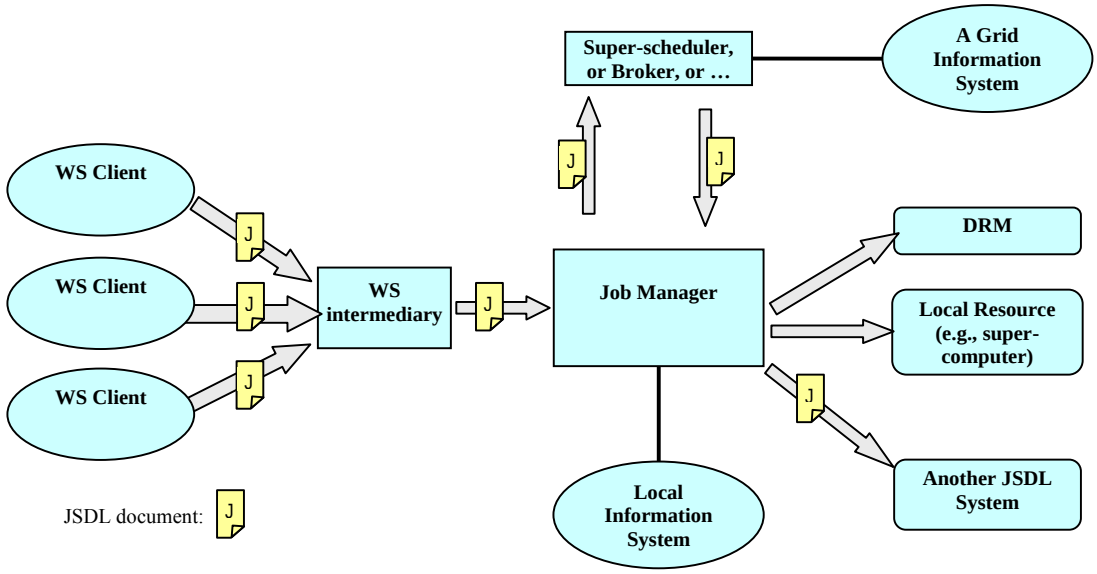

Figure 1: JSDL consumers in a Grid environment

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] .

Pseudo-schemas are provided for each component, before the description of the component. They use BNF-style conventions for attributes and elements: ? denotes optionality (i.e. zero or one occurrences), * denotes zero or more occurrences, + one or more occurrences, [and] are used to form groups, | represents choice. Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema.

```
<!-- sample pseudo-schema -->
<defined_element
       required_attribute_of_type_string="xs:string"
       optional_attribute_of_type_int="xs:int"? >
   <required_element />
   <optional_element />?
   <one_or_more_of_these_elements />+
   [ <choice_1 /> | <choice_2 /> ]*
</defined_element>
```

This specification uses namespace prefixes throughout; they are listed in Table 2-1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 2-1: Prefixes and namespaces used in this specification.

Prefix	**Namespace**
xsd	http://www.w3.org/2001/XMLSchema

jsdl	http://schemas.ggf.org/jsdl/2005/05/jsdl
jsdl-posix	http://schemas.ggf.org/jsdl/2005/05/jsdl-posix

The terms *JSDL element* and *JSDL attribute* indicate that the corresponding language construct is represented as an XML element and XML attribute in the normative JSDL schema.

The term *JSDL document* refers to a well formed XML document that can be validated against the normative JSDL Schema definition contained in Appendix 1.

The key word *present,* when used to refer to a JSDL element, means that an instance of that element appears in a JSDL document.

The key word *support,* with respect to a JSDL consuming system supporting the JSDL specification and the language constructs, refers to the ability of that system to parse a JSDL document. That is, the consuming system MUST be able to interpret the language constructs and assign to them the semantics described in this specification and the values assigned to them in that JSDL document. All JSDL compliant consuming systems must, therefore, support all of the JSDL language constructs.

Note, however, that the result of submitting a JSDL document to a consuming system is out of scope of the JSDL specification.

3 JSDL Scope

JSDL is a language for describing the submission requirements of individual jobs It does not attempt to address the entire lifecycle of a job or the relationship between individual jobs. Therefore when managing and running jobs that were described and submitted using JSDL, particularly in a Grid environment, a number of other languages and protocols are also required. For example, a workflow language should be used to describe the relationship between individual jobs described using JSDL, and in turn, the relationship of those jobs with the data they consume and produce. As another example, consider the negotiation required to put a job in an optimal resource environment according to its requirements. In this case, use of appropriate negotiation and agreement protocols are required. An example of such a protocol is that defined in the GGF's WS-Agreement specification. Figure 2 outlines some of the other elements that are required to submit, manage, and run jobs, particularly in a Grid. Many of these elements are languages and protocols that are not yet available and are either currently being developed or need to be specified and standardized.

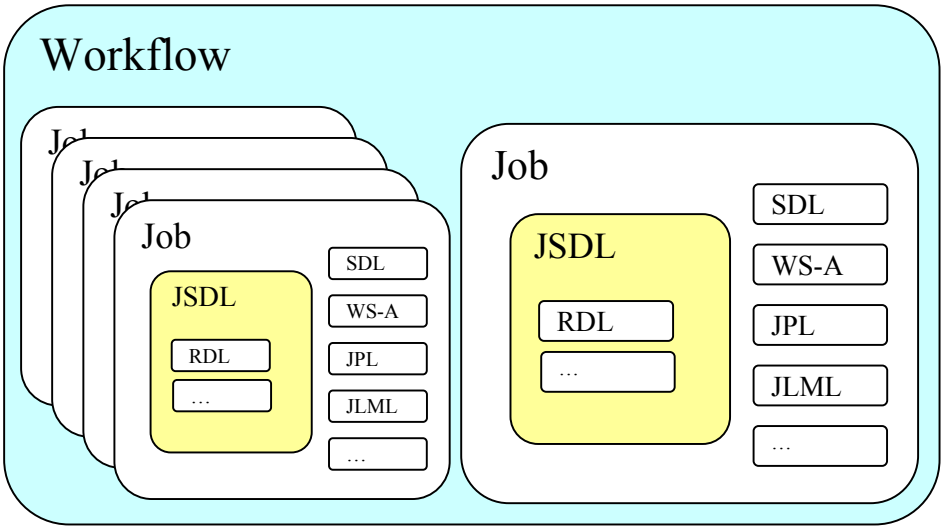

Figure 2: Relationship between JSDL and other valuable specifications.
(RDL – Resource Description Language,
SDL – Scheduling Description Language,
WS-A – Web Service Agreement,
JPL – Job Policy Language,
JLML – Job Lifetime Management Language)

3.1 On Resource Description Language (RDL)

The JSDL specification could make use of a language for describing the resource requirements for a Job. We call this language a Resource Requirements Language (RRL). The definition of the RRL would be a full specification in its own right and is thus beyond the scope of the JSDL. However, in the absence of a suitable RRL, we provide here a core set of elements that should be contained in the RRL. It is likely that future versions of the JSDL specification will use a standard RRL specification when one is available.

3.2 On Scheduling Description Language (SDL)

The scheduling of jobs, and the description of scheduling requirements, are complex tasks. Scheduling requirements fall into many categories. These include:

- **Temporal scheduling**, i.e. job start and end times, validity windows for running a job, etc.;
- **Data-dependent scheduling**, i.e. scheduling jobs based on the expected availability or validity of input data, etc.; and
- **Workflow-dependent scheduling**, i.e. scheduling jobs according to the order of jobs in a workflow.

In addition, there may be dependencies between these categories for scheduling requirements for a job. For example, there may be a dependency between the defined order of jobs in a workflow and the input data required for the job being scheduled.

Due to the complexity of the description of scheduling requirements for a job, the decision has been made to put Scheduling requirements out of scope for JSDL. We do, however, realize the importance of a Scheduling Description Language (SDL) for job submission onto resources, and the management and running of those jobs in potentially complex environments, such as Grids.

3.3 On Web Service Agreement (WS-AG)

Building up an agreement between a resource's Web Service front-end and a job submission Web Service is currently the scope of the GGF's WS-Agreement (WS-AG) specification. We see WS-Agreement as an important emerging standard for the submission of a job to resources. However, many systems currently in use do not have a Web Service front-end and may not require such functionality. Though it is important that JSDL be usable with other standards, it is just as important it can be used independently as well. In other words a JSDL description of a job can be deployed and used with or without WS-AG. (Note that WS-AG is based on the Web Services Resource Framework (WSRF) set of specifications.)

3.4 On Job Policy Language (JPL)

Job policy is a very broad subject and policies may be applied to any aspect of job execution. Users may wish to apply policies to their jobs and resource owners may also wish to apply their policies to jobs using their resources. Expressing such policy constraints requires a specialized language and therefore the definition of such policies is outside the scope of the JSDL. A Job Policy Language (JPL) could be seen as a sister language to JSDL.

3.5 On Job Lifetime Management Language (JLML)

The lifetime management of jobs is an important aspect of the overall management of jobs. It allows the description of the state of the job and job control directives. We see a need for a Job Lifetime Management (JLM) language that can describe this aspect of job management. Since lifetime management is concerned with jobs after submission, the JLM is therefore outside the scope of JSDL.

3.6 On Workflow

The JSDL is designed for the description of job templates for the submission of individual jobs to resources. We realize that there is great interest in developing languages to describe parametric jobs, co-allocation, workflow and so on. Such languages could be JSDL-aware; in other words they can refer to the JSDL description of single jobs for each node in the activity. We see the definition of such a language, however, as a separate and independent effort that can build on top of a standard such as JSDL and is therefore outside the scope of JSDL.

4 The JSDL Structure

4.1 JSDL Document Layout

A JSDL document is described using XML and adheres to the normative XSD schema contained in §Appendix 1. The following sections describe the organization of a JSDL document.

The document is organized as follows: The root element JobDefinition contains a single mandatory child element named JobDescription. The JobDescription contains elements that describe the job: JobIdentification, Application, Resource, and DataStaging. The pseudo schema definition follows:

```
<JobDefinition>
    <JobDescription>
        <JobIdentification ... />?
        <Application ... />
        <Resource ... />*
        <DataStaging ... />*
    </JobDescription>
```

```
   <xsd:any##other/>*
</JobDefinition>
```

4.2 JSDL Element Type Information

The JSDL specification adopts a number of normative xsd types. It also defines a number of types specific to the description of job requirements.

The JSDL-WG has based the definition of a number of types on existing standards, in particular CIM and POSIX. Since there is no available normative XML schema definition of these types one is provided here. The Operating Systems types (see §4.2.2.3) is defined based on CIM[RFC 2396] T. Berners-Lee and R. Fielding and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax.* . Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt
[CIM].

A number of types define a special value 'other'. This value can be used to introduce elements from other specifications. See §6.3 for more details.

4.2.1 Normative XML Schema Types

The JSDL specification adopts the following normative XML Schema (xsd) types:

Table 4-1 Normative XML Schema Types

XSD types			
xsd:string	xsd:normalizedString	xsd:positiveInteger	xsd:boolean
xsd:NCName	xsd:anyType	xsd:token	xsd:any##other
Complex			

4.2.2 JSDL types

The following additional types are defined by the JSDL schema version 1.0.

4.2.2.1 Processor Architectures (jsdl:ProcessorArchitectureEnumeration)

The processor architecture is defined by the ISA processor architecture value for the machine. The following are the values that MUST be supported.

Table 4-2 Processor Architectures

Normative JSDL Name	Definition
sparc	A SPARC architecture processor
powerpc	A PowerPC architecture processor.
x86	A 32-bit Intel Architecture processor.
parisc	A PARISC architecture processor
mips	A MIPS architecture processor
ia64	A 64 bit Intel Architecture processor
arm	An ARM processor
other	A value not defined by this enumeration

This list of types is extensible. Extensions of this token set MUST NOT be in the JSDL namespace.

4.2.2.2 Filesystem Types (jsdl:FileSystemTypeEnumeration)

The following are the defined filesystem types that MUST be supported.

Table 4-3 FileSystem Types

Normative JSDL Name	Definition
swap	Conventional swap space for paging out memory
temporary	Conventional temporary space for a file that is periodically removed. The space is unavailable after the job completes
spool	Temporary space for a file that may persist after the completes
normal	This is a normal space for files to be written to and read from. Files are not deleted after the job except explicitly by the user

This list of types is extensible. Extensions of this token set MUST NOT be in the JSDL namespace.

4.2.2.3 Operating System Types (jsdl:OperatingSystemTypeEnumeration)

The following values MAY be supported by the consuming system. If the consuming system supports the following values they MUST interpret them accordingly.
The values listed here are from the OSType field of the CIM_OperatingSystem model[RFC 2396] T. Berners-Lee and R. Fielding and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax.* . Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt
[CIM].

Table 4-4 Operating System Types

Normative JSDL Names			
Unknown	WINNT	LINUX	HP_MPE
Other	WINCE	Lynx	NextStep
MACOS	NCR3000	XENIX	PalmPilot
ATTUNIX	NetWare	VM	Rhapsody
DGUX	OSF	Interactive_UNIX	Windows_2000
DECNT	DC_OS	BSDUNIX	Dedicated
Tru64_UNIX	Reliant_UNIX	FreeBSD	OS_390
OpenVMS	SCO_UnixWare	NetBSD	VSE

HPUX	SCO_OpenServer	GNU_Hurd	TPF
AIX	Sequent	OS9	Windows_R_Me
MVS	IRIX	MACH_Kernel	Caldera_Open_UNIX
OS400	Solaris	Inferno	OpenBSD
OS_2	SunOS	QNX	Not_Applicable
JavaVM	U6000	EPOC	Windows_XP
MSDOS	ASERIES	IxWorks	z_OS
WIN3x	TandemNSK	VxWorks	
WIN95	TandemNT	MiNT	
WIN98	BS2000	BeOS	

This list of types is extensible. Extensions of this token set MUST NOT be in the JSDL namespace.

4.2.2.4 File Creation Flags (jsdl:CreationFlagEnumeration)

This element is an enumeration of the following values:

Table 4-5 File Creation Flags

Normative JSDL Name	Definition
Overwrite	Overwrite an existing file with the same name; create a new one otherwise
DontOverwrite	Do not overwrite an existing file with the same name
Append	Append to an existing file with the same name; create a new one otherwise

4.2.2.5 Range Value (jsdl:RangeValueType)

A range value is a complex type that allows the definition of exact values (with an optional epsilon argument), left-open or right-open intervals and ranges. All numbers given are of type xsd:double. UpperBoundedRanges and LowerBoundedRanges are limited to the upper or lower bound, respectively. Ranges may be "unlimited" to either negative or positive infinity, subject to the consuming system's capabilities. For example, expressed in Java, an implementation of "infinity" could be java.lang.Double.NEGATIVE_INFINITY and java.lang.Double.POSITIVE_INFINITY, respectively.

The optional attribute "exclusiveBound" has the default value of "false" if not specified. If the optional attribute "epsilon" is not specified it defaults to the value of 0 and an exact match MUST

be assumed. If an exact match cannot be provided in such case, the document MUST be rejected by the consuming system.

RangeValues that specify intersecting ranges MAY be collapsed by the consuming system in order to match the given job description, but the JSDL document MUST NOT be changed.

This type MUST be supported by the consuming system.

4.2.2.5.1 Matching Semantics

The following matching semantics MUST be used. A numeric value, *N*, matches a RangeValue, *R*, if and only if *at least one* of the following conditions is true:

- *R* contains an UpperBoundedRange, *U*, with a false exclusiveBound attribute and where $N \leq U$.
- *R* contains an UpperBoundedRange, *U*, with a true exclusiveBound attribute and where $N < U$.
- *R* contains a LowerBoundedRange, *L*, with a false exclusiveBound attribute and where $N \geq L$.
- *R* contains a LowerBoundedRange, *L*, with a true exclusiveBound attribute and where $N > L$.
- *R* contains an Exact, *E*, with an epsilon attribute *e*, where $E-e \leq N \leq E+e$.
- *R* contains a Range with LowerBound, *L*, and UpperBound, *U*, such that both of the following are true:
    - *L* has a false exclusiveBound attribute and $N \geq L$, or *L* has a true exclusiveBound attribute and $N > L$.
    - *U* has a false exclusiveBound attribute and $N \leq U$, or *U* has a true exclusiveBound attribute and $N < U$.

4.2.2.5.2 Pseudo Schema

```
<UpperBoundedRange exclusiveBound="xsd:boolean"?>
   xsd:double
</UpperBoundedRange>?

<LowerBoundedRange exclusiveBound="xsd:boolean"?>
   xsd:double
</LowerBoundedRange>?

<Exact epsilon=”xsd:double”?>
   xsd:double
 </Exact>*

<Range>
   <LowerBound exclusiveBound="xsd:boolean"?>
      xsd:double
   <LowerBound>
   <UpperBound exclusiveBound="xsd:boolean"?>
      xsd:double
   </UpperBound>
</Range>*
```

4.2.2.5.3 Example

The expression "5, 6.7777, 7.0, [50.3,99.5), [100-" can be encoded in a RangeValue as follows:

```
  <jsdl:LowerBoundedRange> 100.0 </jsdl:LowerBoundedRange>
  <jsdl:Exact> 5.0 </jsdl:Exact>
  <jsdl:Exact epsilon="0.00001"> 6.7777 </jsdl:Exact>
  <jsdl:Exact> 7.0 </jsdl:Exact>
  <jsdl:Range>
    <jsdl:LowerBound> 50.3 </jsdl:LowerBound>
    <jsdl:UpperBound exclusiveBound="true"> 99.5 </jsdl:UpperBound>
  </jsdl:Range>
```

5 The JSDL Core Element Set

The JSDL core element set contains the semantics for elements that are defined by JSDL 1.0. All elements MUST be supported by JSDL 1.0 compliant consuming systems. (See also §2.)

5.1 Job Structure Elements

5.1.1 JobDefinition

5.1.1.1 Definition

This element describes the Job and it requirements. It contains a Job Description section. It is the root element of the JSDL document.

5.1.1.2 Multiplicity - 1

5.1.1.3 Type – Complex

This complex type MUST support the following elements:

- JobDescription

5.1.1.4 Attributes

- id – the id of the Job definition document. This is defines as a `xsd:ID` and is in the default namespace of the document. The id MAY be omitted.

5.1.1.5 Pseudo Schema

```
<JobDefinition id="xsd:ID"?>
   <JobDescription ... />
   <xsd:any##other/>*
</JobDefinition>
```

5.1.1.6 Example

```
<jsdl:JobDefinition id="gnuplot"?>
   <jsdl:JobDescription> ... </jsdl:JobDescription>
</jsdl:JobDefinition>
```

5.1.2 JobDescription

5.1.2.1 Definition

This element describes the Job and it requirements. It contains Job Identification, Application, Resources, and DataStaging elements.

5.1.2.2 Multiplicity - 1

5.1.2.3 Type – Complex

This complex type MUST support the following elements:

- JobIdentification
- Application
- Resource
- DataStaging

5.1.2.4 Attributes — None

5.1.2.5 Pseudo Schema

```
<JobDescription>
   <JobIdentification ... />?
   <Application ... />?
   <Resource ... />*
   <DataStaging ... />*
   <xsd:any##other>*
</JobDescription>
```

5.1.3 Description Element

5.1.3.1 Definition

This element provides descriptive, human readable, information about its containing complex element. It can be present as a sub-element of a number of other JSDL elements: JobIdentification, Application, FileSystem, etc. If this is not present as a sub-element then no description is defined.

5.1.3.2 Multiplicity – 0-1

5.1.3.3 Type – xsd:string

5.1.3.4 Attributes — None

5.1.3.5 Pseudo Schema

```
<Description> xsd:string </Description>?
```

5.2 Job Identity Elements

5.2.1 JobIdentification

5.2.1.1 Definition

This element contains all other elements that identify the Job. It contains the following elements: Job Name, Job Description, Job Annotation, and Job Project. If this element is not present then its value, including all of its sub-elements, is undefined.

5.2.1.2 Multiplicity – 0-1

5.2.1.3 Type – Complex

This complex type MUST support the following elements:

- JobName,

- Description,
- JobAnnotation,
- JobProject,

5.2.1.4 Attributes — None

5.2.1.5 Pseudo Schema

```
<JobIdentification>
   <JobName ... />?
   <Description ... />?
   <JobAnnotation ... />*
   <JobProject ... />*
   <xsd:any##other>*
</JobIdentification>?
```

5.2.2 JobName

5.2.2.1 Definition

A string that MAY be specified by a user to name the job specified in this JSDL document. It may not be unique to a particular JSDL document, which means that a user MAY specify the same JobName for multiple JSDL documents. If this element is not present then it is not defined.

5.2.2.2 Multiplicity – 0-1

5.2.2.3 Type – xsd:string

5.2.2.4 Attributes — None

5.2.2.5 Pseudo Schema

```
<JobName> xsd:string </JobName>?
```

5.2.2.6 Example

May be used for search and sort purposes and for job management.

```
<jsdl:JobName> visualization </jsdl:JobName>
```

5.2.3 JobAnnotation

5.2.3.1 Definition

A string that MAY be specified by a user to annotate the job. If this element is not present then it is not defined.

5.2.3.2 Multiplicity – 0-n

5.2.3.3 Type – xsd:string

5.2.3.4 Attributes — None

5.2.3.5 Pseudo Schema

```
<JobAnnotation> xsd:string </JobAnnotation>*
```

5.2.3.6 Example

May be used for search and sort purposes and for job management.

```
<jsdl:JobAnnotation> uuid:ashjasuwhqsoi </jsdl:JobAnnotation>
```

5.2.4 JobProject

5.2.4.1 Definition

A string specifying the project to which the job belongs. The project could be used by accounting systems or access control systems. The interpretation of the JobProject elements is left to the implementation of the consuming system. If this element is not present then it is not defined.

5.2.4.2 Multiplicity – 0-n

5.2.4.3 Type – xsd:string

5.2.4.4 Attributes — None

5.2.4.5 Pseudo Schema

```
<JobProject> xsd:string </JobProject>*
```

5.2.4.6 Example

The job belongs to the Project group "wcdma".

```
<jsdl:JobProject> wcdma </jsdl:JobProject>
```

5.3 Application Requirements

5.3.1 Application Element

5.3.1.1 Definition

This element describes the Application and its requirements. It contains ApplicationName, ApplicationVersion and Description It serves as a high level generic container for specific application definitions that are most likely to be understood by only a small fraction of JSDL implementors. A POSIX compliant normative extension is given in §7.1.1. Used without any extension, it uniformly describes an application by its name and version number. If this is not present then this job definition does not define an application to execute. This could be a data staging job, or a null job. See also §5.3.2 and §7.1.2.

5.3.1.2 Multiplicity – 0-1

5.3.1.3 Type – Complex

This complex type MUST support the following elements:

- ApplicationName
- ApplicationVersion
- Description

5.3.1.4 Attributes — None

5.3.1.5 Pseudo Schema

```
<Application>
   <ApplicationName ... />?
```

```
   <ApplicationVersion ... />?
   <Description ... />?
   <xsd:any##other/>*
</Application>?
```

5.3.2 ApplicationName Element

5.3.2.1 Definition

This element is the name of the application and is used to identify the application independent of its execution on a host or system. If this is not present then it is not defined and a null job is assumed if there is no application extension element present that defines the application to execute. See also §5.3.1 and §7.1.2.

5.3.2.2 Multiplicity – 0-1

5.3.2.3 Type – xsd:string

5.3.2.4 Attributes — None

5.3.2.5 Pseudo Schema

```
<ApplicationName> xsd:string </ApplicationName>?
```

5.3.2.6 Example

The BLAST application:

```
<jsdl:ApplicationName> BLAST </jsdl:ApplicationName>
```

5.3.3 ApplicationVersion Element

5.3.3.1 Definition

This element defines the version of the application to execute. The consuming system MUST use exact textual match to select the version of the application. If this element is not present then it is not defined and any version of the application MAY be executed.

5.3.3.2 Multiplicity – 0-1

5.3.3.3 Type – xsd:string

5.3.3.4 Attributes — None

5.3.3.5 Pseudo Schema

```
<ApplicationVersion> xsd:string </ApplicationVersion>?
```

5.3.3.6 Example

The BLAST application version 1.4:

```
<jsdl:Application>
  <jsdl:ApplicationName> BLAST </jsdl:ApplicationName>
  <jsdl:ApplicationVersion> 1.4 </jsdl:ApplicationVersion>
  ...
</jsdl:Application>
```

5.4 Resource Requirements

The resource elements describe the resource requirements of the job. At most one Resources element may be used.Additional resource requirements MAY be defined as extensions.

5.4.1 Resources

5.4.1.1 Definition

This element describes the resource requirements of the job. If this element is not present then the consuming system MAY choose any set of resources to execute the job.

5.4.1.2 Multiplicity – 0-1

5.4.1.3 Type – Complex

This complex type MUST support the following elements:

- CandidateHosts
- FileSystem
- ExclusiveExecution
- OperatingSystem
- CPUArchitecture
- IndividualCPUSpeed
- IndividualCPUTime
- IndividualCPUCount
- IndividualNetworkBandwidth
- IndividualPhysicalMemory
- IndividualVirtualMemory
- IndividualDiskSpace
- TotalCPUTime
- TotalCPUCount
- TotalPhysicalMemory
- TotalVirtualMemory
- TotalDiskSpace
- TotalResourceCount

5.4.1.4 Attributes — None

5.4.1.5 Pseudo Schema

```
<Resource>
   <CandidateHosts ... />?
   <FileSystem .../>*
   <ExlusiveExecution .../>?
   <OperatingSystem .../>?
   <CPUArchitecture .../>?
   <IndividualCPUSpeed .../>?
   <IndividualCPUTime .../>?
   <IndividualCPUCount .../>?
   <IndividualNetworkBandwidth .../>?
   <IndividualPhysicalMemory .../>?
   <IndividualVirtualMemory .../>?
   <IndividualDiskSpace .../>?
   <TotalCPUTime .../>?
```

```
   <TotalCPUCount .../>?
   <TotalPhysicalMemory .../>?
   <TotalVirtualMemory .../>?
   <TotalDiskSpace .../>?
   <TotalResourceCount .../>?
   <xsd:any##other>*
</Resource>*
```

5.4.2 CandidateHosts

5.4.2.1 Definition

This element is a complex type specifying the set of named hosts which may be selected for running the job. If this element is present then one or more hosts from this list MUST be chosen to run the job. If this is not present then it is not defined. A named host may be a single host (e.g., a machine name), a logical group of hosts (e.g., a named logical group or cluster), a virtual machine, and so on.

5.4.2.2 Multiplicity – 0-1

5.4.2.3 Type – Complex

This complex type MUST support the following elements:

- HostName

5.4.2.4 Attributes — None

5.4.2.5 Pseudo Schema

```
<CandidateHosts>
  <HostName .../>+
</CandidateHosts>?
```

5.4.2.6 Example

The machine bach.example.com should be used:

```
<jsdl:CandidateHosts>
  <jsdl:HostName> bach.example.com </jsdl:HostName>
</jsdl:CandidateHosts>
```

The list of machines that may be used:

```
<jsdl:CandidateHosts>
  <jsdl:HostName> bach.example.com </jsdl:HostName>
  <jsdl:HostName> mozart.example.com </jsdl:HostName>
  <jsdl:HostName> beethoven.example.com </jsdl:HostName>
</jsdl:CandidateHosts>
```

Any or all of these machines may be used.

5.4.3 HostName

5.4.3.1 Definition

This element is a simple type containing a single name of a host which may be selected for running a job. The name may refer to a single host (e.g., a machine name), a logical group of hosts (e.g., a named logical group or cluster), a virtual machine, and so on.

5.4.3.2 Multiplicity – 1-n

5.4.3.3 Type – xsd:string

5.4.3.4 Attributes — None

5.4.3.5 Pseudo Schema

```
<HostName> xsd:string </HostName>+
```

5.4.3.6 Example

The machine "bach" must be used:

```
<jsdl:HostName> bach.example.com </jsdl:HostName>
```

Machines from the logical group with name "the-composers" must be used:

```
<jsdl:HostName> the-composers </jsdl:HostName>
```

5.4.4 CPUArchitecture

5.4.4.1 Definition

This element is a string specifying the CPU architecture required by the job in the execution environment. If this is not present then it is not defined and the consuming system MAY choose any value. Values not defined by the JSDL ProcessorArchitectureEnumeration (see §4.2.2.1) MAY be used by specifying the special token 'other' and including the value as an extension (see §6.3). See also example below.

5.4.4.2 Multiplicity – 0-1

5.4.4.3 Type – Complex

This complex type MUST support the following elements:

- ***CPUArchitectureName***

5.4.4.4 Attributes — None

5.4.4.5 Pseudo Schema

```
<CPUArchitecture>
  <CPUArchitectureName .../>
   <xsd:any##other>*
</CPUArchitecture>?
```

5.4.4.6 Example

A SPARC architecture machine:

```
<jsdl:CPUArchitecture>
  <jsdl:CPUArchitectureName>jsdl:sparc</jsdl:CPUArchitectureName>
</jsdl:CPUArchitecture>
```

A CELL architecture machine:

```
<jsdl:CPUArchitecture>
  <jsdl:CPUArchitectureName> other </jsdl:CPUArchitectureName>
  <tns:OtherCPUArchitectures> tns:cell </tns:OtherCPUArchitectures>
</jsdl:CPUArchitecture>
```

5.4.5 CPUArchitectureName

5.4.5.1 Definition

This element is a token specifying the CPU architecture required by the job in the execution environment.

5.4.5.2 Multiplicity – 1

5.4.5.3 Type – jsdl:ProcessorArchitectureEnumeration

5.4.5.4 Attributes — None

5.4.5.5 Pseudo Schema

```
<CPUArchitectureName> jsdl:ProcessorArchitectureEnumeration
</CPUArchitectureName>
```

5.4.5.6 Example

A SPARC architecture machine:

```
<jsdl:CPUArchitectureName>jsdl:sparc</jsdl:CPUArchitectureName>
```

5.4.6 IndividualCPUSpeed

5.4.6.1 Definition

This element is a range value specifying the speed of each CPU required by the job in the execution environment. The IndividualCPUSpeed is given in multiples of Hertz. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.6.2 Multiplicity – 0-1

5.4.6.3 Type – jsdl:RangeValueType

5.4.6.4 Attributes — None

5.4.6.5 Pseudo Schema

```
<IndividualCPUSpeed> jsdl:RangeValueType </IndividualCPUSpeed>?
```

5.4.6.6 Example

A CPU that has speed of at least 1GHz.

```
<jsdl:IndividualCPUSpeed>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
</jsdl:IndividualCPUSpeed>
```

5.4.7 IndividualCPUTime

5.4.7.1 Definition

This element is a range value specifying total number of CPU seconds required on each resource to run to execute the job. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.7.2 Multiplicity – 0-1

5.4.7.3 Type – jsdl:RangeValueType

5.4.7.4 Attributes — None

5.4.7.5 Pseudo Schema

```
<IndividualCPUTime> jsdl:RangeValueType </IndividualCPUTime>?
```

5.4.7.6 Example

At most 60 CPU seconds:

```
<jsdl:IndividualCPUTime>
  <jsdl:UpperBoundedRange>60.0</jsdl:UpperBoundedRange>
</jsdl:IndividualCPUTime>
```

5.4.8 TotalCPUTime

5.4.8.1 Definition

This element is a range value specifying total number of CPU seconds required, across all CPUs used to execute the job. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.8.2 Multiplicity – 0-1

5.4.8.3 Type – jsdl:RangeValueType

5.4.8.4 Attributes — None

5.4.8.5 Pseudo Schema

```
<TotalCPUTime> jsdl:RangeValueType </TotalCPUTime>?
```

5.4.8.6 Example

At most 600 CPU seconds across all CPUs.

```
<jsdl:TotalCPUTime>
  <jsdl:UpperBoundedRange>600.0</jsdl:UpperBoundedRange>
</jsdl:TotalCPUTime>
```

5.4.9 IndividualCPUCount

5.4.9.1 Definition

This element is a range value specifying number of CPUs for each of the resources to be allocated to the job submission. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.9.2 Multiplicity – 0-1

5.4.9.3 Type – jsdl:RrangeValueType

5.4.9.4 Attributes — None

5.4.9.5 Pseudo Schema

```
<IndividualCPUCount> jsdl:RangeValueType </IndividualCPUCount>?
```

5.4.9.6 Example

Each resource provided should have two CPUs available for this job.

```
<jsdl:IndividualCPUCount>
  <jsdl:exact>2.0</jsdl:exact>
</jsdl:IndividualCPUCount>
```

5.4.10 TotalCPUCount

5.4.10.1 Definition

This element is a range value specifying the total number of CPUs required for this job submission. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.10.2 Multiplicity – 0-1

5.4.10.3 Type – jsdl:RrangeValueType

5.4.10.4 Attributes — None

5.4.10.5 Pseudo Schema

```
<TotalCPUCount> jsdl:RangeValueType </TotalCPUCount>?
```

5.4.10.6 Example

A total of two CPUs are required.

```
<jsdl:TotalCPUCount><jsdl:exact>2.0</jsdl:exact></jsdl:TotalCPUCount>
```

5.4.11 TotalResourceCount

5.4.11.1 Definition

This element is a range value specifying the total number of resources required by the job. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.11.2 Multiplicity – 0-1

5.4.11.3 Type – jsdl:RangeValueType

5.4.11.4 Attributes — None

5.4.11.5 Pseudo Schema

```
<TotalResourceCount> jsdl:RangeValueType </TotalResourceCount>?
```

5.4.11.6 Example

Five resources:

```
<jsdl:Resources>
      ...
      <jsdl:TotalResourceCount>
            <jsdl:exact>5.0</jsdl:exact>
      </jsdl:TotalResourceCount>
</jsdl:Resources>
```

Five resources each with 2 CPUS for the job:

```
<jsdl:Resources>
      ...
      <jsdl:IndividualCPUCount>
            <jsdl:exact>2.0</jsdl:exact>
      </jsdl:IndividualCPUCount>
      <jsdl:TotalResourceCount>
            <jsdl:exact>5.0</jsdl:exact>
      </jsdl:TotalResourceCount>
</jsdl:Resources>
```

5.4.12 IndividualPhysicalMemory

5.4.12.1 Definition

This element is a range value specifying the required amount of physical memory required on each individual resource. The amount is specified as multiple of Bytes.If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.12.2 Multiplicity – 0-1

5.4.12.3 Type – jsdl:RangeValueType

5.4.12.4 Attributes — None

5.4.12.5 Pseudo Schema

```
<IndividualPhysicalMemory> jsdl:RangeValueType
</IndividualPhysicalMemory>?
```

5.4.12.6 Example

Each resource should have at least 1GB of physical memory:

```
<jsdl:IndividualPhysicalMemory>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
</jsdl:IndividualPhysicalMemory>
```

5.4.13 TotalPhysicalMemory

5.4.13.1 Definition

This element is a range value specifying the required amount of physical memory for the entire job across all resources. The amount is specified as multiple of Bytes.If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.13.2 Multiplicity – 0-1

5.4.13.3 Type – jsdl:RangeValueType

5.4.13.4 Attributes — None

5.4.13.5 Pseudo Schema

```
<TotalPhysicalMemory> jsdl:RangeValueType </TotalPhysicalMemory>?
```

5.4.13.6 Example

A total of 10GB of physical memory across the resources:

```
<jsdl:TotalPhysicalMemory>
      <jsdl:LowerBoundedRange>10737418240.0</jsdl:LowerBoundedRange>
</jsdl:TotalPhysicalMemory>
```

5.4.14 IndividualVirtualMemory

5.4.14.1 Definition

This element is a range value specifying the required amount of virtual memory for each of the resources to be allocated for this job submission. The amount is specified as multiple of Bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.14.2 Multiplicity – 0-1

5.4.14.3 Type – jsdl:RangeValueType

5.4.14.4 Attributes — None

5.4.14.5 Pseudo Schema

```
<IndividualVirtualMemory> jsdl:RangeValueType
</IndividualVirtualMemory>?
```

5.4.14.6 Example

A resource with at least 1Gigabyte of virtual memory:

```
<jsdl:IndividualVirtualMemory>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
</jsdl:IndividualVirtualMemory>
```

5.4.15 TotalVirtualMemory

5.4.15.1 Definition

This element is a range value specifying the required total amount of virtual memory for the job submission. The amount is specified as multiple of Bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.15.2 Multiplicity – 0-1

5.4.15.3 Type – jsdl:RangeValueType

5.4.15.4 Attributes — None

5.4.15.5 Pseudo Schema

```
<TotalVirtualMemory> jsdl:RangeValueType </TotalVirtualMemory>?
```

5.4.15.6 Example

A resource with at least 1Gigabyte of virtual memory:

```
<jsdl:TotalVirtualMemory>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
</jsdl:TotalVirtualMemory>
```

5.4.16 IndividualNetworkBandwidth

5.4.16.1 Definition

This element is a range value specifying the bandwidth requirements of each individual resource. The amount is specified as multiple of Bits per second. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.16.2 Multiplicity – 0-1

5.4.16.3 Type – jsdl:RangeValueType

5.4.16.4 Attributes — None

5.4.16.5 Pseudo Schema

```
<IndividualNetworkBandwidth> xsd:RangeValueType
</IndividualNetworkBandwidth>?
```

5.4.16.6 Example

A resource with at least 100 Megabit per second network bandwidth:

```
<jsdl:IndividualNetworkBandwidth>
  <jsdl:LowerBoundedRange>104857600.0</jsdl:LowerBoundedRange>
</jsdl:IndividualNetworkBandwidth>
```

5.4.17 IndividualDiskSpace

5.4.17.1 Definition

This is a range value that describes the amount of free writable space for each resource allocated to the job. The amount of disk space is given in bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.17.2 Multiplicity – 0-1

5.4.17.3 Type – jsdl:RangeValueType

5.4.17.4 Attributes — None

5.4.17.5 Pseudo Schema

```
<IndividualDiskSpace> jsdl:RangeValueType </IndividualDiskSpace>?
```

5.4.17.6 Example

Each resource should have at least 1GB of free space:

```
<jsdl:Resources>
   ...
   <jsdl:IndividualDiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:IndividualDiskSpace>
</jsdl:Resource>
```

Note that in this example a separate configuration mechanism might be necessary to prepare the resource for the job.

5.4.18 TotalDiskSpace

5.4.18.1 Definition

This is a range value that describes the total amount of free writable space that should be allocated to the job. The amount of disk space is given in bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.18.2 Multiplicity – 0-1

5.4.18.3 Type – jsdl:RangeValueType

5.4.18.4 Attributes — None

5.4.18.5 Pseudo Schema

```
<TotalDiskSpace> jsdl:RangeValueType </TotalDiskSpace>?
```

5.4.18.6 Example

The job should be given a total of at least 10GB of free space disk space:

```
<jsdl:Resources>
   ...
   <jsdl:TotalDiskSpace>
      <jsdl:LowerBoundedRange>10737418240.0</jsdl:LowerBoundedRange>
   </jsdl:TotalDiskSpace>
</jsdl:Resource>
```

Note that in this example a separate configuration mechanism might be necessary to prepare the resources for the job.

5.4.19 FileSystem

5.4.19.1 Definition

This is a complex type that describes a location and capacity of storage required for the job.

5.4.19.2 Multiplicity – 0-n

This complex type MUST support the following elements:

- Description
- MountPoint
- MountSource
- DiskSpace
- FileSystemType

5.4.19.3 Type – Complex

This complex type MUST support the following elements:

- Description
- MountPoint
- MountSource
- DiskSpace
- FileSystemType

5.4.19.4 Attributes

- name – xsd:NCName – The name of the Filesystem. This name is user defined and MUST be unique in a JSDL document. A small number of well known names of filesystems that are expected to be commonly supported are defined and may also be used here. See §5.4.19.6 for more details.

5.4.19.5 Pseudo Schema

```
<FileSystem name=″xsd:NCName″>
   <Description ... />?
   <MountPoint ... />?
   <MountSource ... />?
   <DiskSpace ... />?
   <FileSystemType ... />?
   <xsd:any##other/>*
</FileSystem>*
```

5.4.19.6 Well-known Filesystem names

It is expected that consuming systems will typically support a small number of well-known FileSystems. The names and semantics of these well-known FileSystems are listed here together with minimal example definitions. These FileSystems are not available by default; they must be defined in a JSDL document if needed for job execution.

For actual submission these example declarations may be further specialized by defining the mount point or available space of any file-system, etc.

```
<jsdl:FileSystem name="HOME">
   <jsdl:Description>
      The user's home directory.
   </jsdl:Description>
   <jsdl:FileSystemType> jsdl:normal </jsdl:FileSystemType>
```

```
</jsdl:FileSystem>
```

```
<jsdl:FileSystem name="ROOT">
   <jsdl:Description>
      The root of the main system filesystem.
      Not safe to assume that it is writeable.
      The actual root of the filesystem depends on the OS.
   </jsdl:Description>
   <jsdl:FileSystemType> jsdl:normal </jsdl:FileSystemType>
</jsdl:FileSystem>
```

```
<jsdl:FileSystem name="SCRATCH">
   <jsdl:Description>
      Temporary space that persists some time after the job completes
      and which is capable of taking relatively large files.
   </jsdl:Description>
   <jsdl:FileSystemType> jsdl:spool </jsdl:FileSystemType>
</jsdl:FileSystem>
```

```
<jsdl:FileSystem name="TMP">
   <jsdl:Description>
      Temporary space that does not necessarily persist after the job
      terminates and which might not be shared between resources, but
      which will be fast.
   </jsdl:Description>
   <jsdl:FileSystemType> jsdl:tmp </jsdl:FileSystemType>
</jsdl:FileSystem>
```

5.4.19.7 Example

The filesystem “HOME” must have at most 10GB capacity. The execution system is free to choose an appropriate MountPoint since no value is provided.

```
<jsdl:FileSystem name=”HOME”>
   <jsdl:Description> Steve’s home </jsdl:Description>
      <jsdl:DiskSpace>
         <jsdl:UpperBoundedRange>10737418240.0</jsdl:UpperBoundedRange>
      </jsdl:DiskSpace>
   <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
</jsdl:FileSystem>
```

The filesystem “HOME” must have at least 1GB capacity and must be provided in the location contained in MountPoint.

```
<jsdl:FileSystem name=”HOME”>
   <jsdl:Description> Ali’s home </jsdl:Description>
   <jsdl:MountPoint>/home/ali</jsdl:MountPoint>
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
   <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
</jsdl:FileSystem>
```

The filesystem “TMP” must have at least 1GB capacity. The execution system is free to choose an appropriate MountPoint.

```
<jsdl:FileSystem name=”TMP”>
```

```
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
   <jsdl:FileSystemType>jsdl:tmp</jsdl:FileSystemType>
</jsdl:FileSystem>
```

The filesystem “FASTSTORAGE” must be made available. The execution system is free to choose an appropriate MountPoint. An extension specifies additional requirements, e.g., high performance.

```
<jsdl:FileSystem>
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
   <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
   <tns:Performance>tns:high</tns:Performance>
</jsdl:FileSystem>
```

5.4.20 MountPoint of FileSystem

5.4.20.1 Definition

This is a string that describes a *local* location that MUST be made available in the containing resource element for the job.

If MountPoint is not defined the consuming system MUST choose the local location in which to provide the requested FileSystem.

5.4.20.2 Multiplicity – 0-1

5.4.20.3 Type – xsd:string

5.4.20.4 Attributes — None

5.4.20.5 Pseudo Schema

```
<MountPoint> xsd:string </MountPoint>?
```

5.4.20.6 Example

The filesystem “HOME” must be made available in “/home/darren”

```
<jsdl:FileSystem name=”HOME”>
      ...
      <jsd:MountPoint>/home/darren</jsdl:MountPoint>
</jsdl:FileSystem>
```

The filesystem “HOME” must be made available in “c:\Documents and Settings\fred”

```
<jsdl:FileSystem name=”HOME”>
      ...
      <jsdl:MountPoint>c:\Documents and Settings\fred</jsdl:MountPoint>
</jsdl:FileSystem>
```

The filesystem “HOME” was requested with no MountPoint element defined. The environment variable “HOME” is defined to hold the actual value of the mount point.

```
<jsdl:FileSystem name="HOME">
      ...
   <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
</jsdl:FileSystem>

<jsdl:Application>
...
<jsdl-posix:Environment name="HOME" filesystemName="HOME"/>
</jsdl:Application>
```

Suppose that this filesystem was made available at the local location "/usr/x00a10". The job's environment will contain the following environment variable:

```
HOME=/usr/x00a10
```

5.4.21 MountSource of FileSystem

5.4.21.1 Definition

This is a string that describes a *remote* location that MUST be made available for the job.

5.4.21.2 Multiplicity – 0-1

5.4.21.3 Type – xsd:string

5.4.21.4 Attributes — None

5.4.21.5 Pseudo Schema

```
<MountSource> xsd:string </MountSource>?
```

5.4.21.6 Example

The filesystem "HOME" must have at most 10GB capacity. The execution system is free to choose an appropriate MountPoint since no value is provided. This should mount the NFS export sputnick.ggf.org:/home/steve

```
<jsdl:FileSystem name="HOME" >
  <jsdl:Description> Steve's home </jsdl:Description>
  <jsdl:MountSource>nfs:sputnick.ggf.org/home/steve</jsdl:MountSource>
    <jsdl:DiskSpace>
      <jsdl:UpperBoundedRange>10737418240.0</jsdl:UpperBoundedRange>
    </jsdl:DiskSpace>
    <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
  </jsdl:FileSystem>
```

5.4.22 DiskSpace of File System

5.4.22.1 Definition

This is a range value that describes the amount of free writable space on the containing FileSystem element for the job. The amount of disk space is given in bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.22.2 Multiplicity – 0-1

5.4.22.3 Type – jsdl:RangeValueType

5.4.22.4 Attributes — None

5.4.22.5 Pseudo Schema

```
<DiskSpace> jsdl:RangeValueType </DiskSpace>?
```

5.4.22.6 Example

A filesystem with at least 1GB of free space:

```
<jsdl:FileSystem name="HOME">
   ...
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
</jsdl:FileSystem>
```

A filesystem with at most 1GB of free space:

```
<jsdl:FileSystem name="TMP">
   ...
   <jsdl:DiskSpace>
      <jsdl:UpperBoundedRange>1073741824.0</jsdl:UpperBoundedRange>
   </jsdl:DiskSpace>
</jsdl:FileSystem>
```

5.4.23 FileSystemType of File System

5.4.23.1 Definition

This is a token that describes the type of filesystem of the containing FileSystem element. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.23.2 Multiplicity – 0-1

5.4.23.3 Type – jsdl:FileSystemTypeEnumeration

5.4.23.4 Attributes — None

5.4.23.5 Pseudo Schema

```
<FileSystemType> jsdl:FileSystemTypeEnumeration </FileSystemType>?
```

5.4.23.6 Example

```
<jsdl:FileSystem name="HOME">
      ...
      <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
</jsdl:FileSystem>
```

5.4.24 ExclusiveExecution

5.4.24.1 Definition

This is a boolean that designates whether the job must be the only job run at a time on the machine by the consuming system. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.24.2 Multiplicity – 0-1

5.4.24.3 Type – xsd:boolean

- True – run exclusively on the resource
- False – others can run at the same time.

5.4.24.4 Attributes — None

5.4.24.5 Pseudo Schema

```
<ExclusiveExecution> xsd:boolean </ExclusiveExecution>?
```

5.4.24.6 Example

Exclusive execution:

```
<jsdl:ExclusiveExecution> true </jsdl:ExclusiveExecution>
```

5.4.25 OperatingSystem

5.4.25.1 Definition

This is a complex type that contains a description, version, and type of operating system. If this is not present then it is not defined and the consuming system MAY choose any value.

5.4.25.2 Multiplicity – 0-1

5.4.25.3 Type – Complex

- Description
- Version
- Type

5.4.25.4 Attributes — None

5.4.25.5 Pseudo Schema

```
<OperatingSystem>
   <OperatingSystemType ... />?
   <OperatingSystemVersion ... />?
   <Description ... />?
   <xsd:any##other/>*
</OperatingSystem>?
```

5.4.26 OperatingSystemType

5.4.26.1 Definition

This is an element that contains the name of the operating system. If this is not present then it is not defined and the consuming system MAY choose any value. Values not defined by the JSDL

OperatingSystemTypeEnumeration (see §4.2.2.3) may be used by specifying the special token 'other' and including the value as an extension (see §6.3). See example below.

5.4.26.2 Multiplicity – 0-1

5.4.26.3 Type – Complex

- ***OperatingSystemName***

5.4.26.4 Attributes — None

5.4.26.5 Pseudo Schemas

```
<OperatingSystemType>
  <OperatingSystemName>
   <xsd:any##other/>*
</OperatingSystemType>?
```

5.4.26.6 Example

The Inferno OS:

```
<jsdl:OperatingSystemType>
  <jsdl:OperatingSystemName>jsdl:Inferno<jsdl:OperatingSystemName>
</jsdl:OperatingSystemType>
```

The Windows 2003 Server OS:

```
<jsdl:OperatingSystemType>
  <jsdl:OperatingSystemName>jsdl:other<jsdl:OperatingSystemName>
  <tns:MSWindows>tns:Windows_2003_Server</tns:MSWindows>
</jsdl:OperatingSystemType>
```

5.4.27 OperatingSystemName

5.4.27.1 Definition

This is a token type that contains the name of the operating system.

5.4.27.2 Multiplicity – 1

5.4.27.3 Type – jsdl:OperatingSystemTypeEnumeration

5.4.27.4 Attributes — None

5.4.27.5 Pseudo Schemas

```
<OperatingSystemName> jsdl:OperatingSystemTypeEnumeration
</OperatingSystemName>
```

5.4.27.6 Example

```
  <jsdl:OperatingSystemName>jsdl:Inferno<jsdl:OperatingSystemName>
```

5.4.28 OperatingSystemVersion

5.4.28.1 Definition

This element is the version of the operating system that the job is to be executed on. The consuming system MUST use exact textual match to select the version of the operating system. If this is not present then any version of the operating system MAY be used.

5.4.28.2 Multiplicity – 0-1

5.4.28.3 Type – xsd:string

5.4.28.4 Attributes — None

5.4.28.5 Pseudo Schema

```
<OperatingSystemVersion> xsd:string </OperatingSystemVersion>?
```

5.4.28.6 Example

```
<jsdl:OperatingSystemVersion> 5.01 </jsdl:OperatingSystemVersion>
```

5.4.29 Additional Resources

It is possible to extend JSDL (see §6) to describe additional resources. The kind of resources that could be described this way are licenses, named resources, software libraries, software packages, special hardware, and so on.

5.5 Data Staging Elements

5.5.1 DataStaging Element

5.5.1.1 Definition

Data staging defines 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.

If a directory is specified in the FileName Element or Source Element then a recursive copy will be performed. If the execution environment does not support recursive copying an error should be reported. The specification of this error, including how or when it is raised, is out of scope of the JSDL specification.

It is possible to stage out the same file to more than once by specifying the same FileName (on the same FileSystem) in multiple stage out DataStaging elements.

It is also possible, but deprecated, to use the same FileName in separate DataStaging elements to *stage in* to the same local file. The result is unspecified.

The CreationFlag determines whether the staged file should append or overwrite an existing file. This element MUST be present in a DataStaging element.

The ordering of the DataStaging elements in the JSDL document is not significant. That is, the order of the DataStaging elements in the document does not imply any ordering, besides the ordering already mentioned concerning job execution and carrying out the different stage in (or stage out) operations.

More complex file transfers, for example, conditional transfers based on job termination status are out of scope.

Permission and Access control for the staged files should be handled by the implementation and is out of scope of the JSDL specification.

More complicated deployment scenarios than the file staging described here (e.g., deployment and configuration of the execution environment itself) are out of scope of the JSDL specification.

5.5.1.2 Multiplicity – 0-n

5.5.1.3 Type – Complex

This complex type MUST support the following elements:

- FileName
- FileSystemName
- CreationFlag
- DeleteOnTermination
- Source
- Target

5.5.1.4 Attributes

- name – xsd:NCName – An optional name for the DataStaging element. This name is user defined and MUST be unique in a JSDL document.

5.5.1.5 Pseudo Schema

```
<DataStaging name=″xsd:NCName″?>
   <FileName ... />
   <FileSystemName ... />?
   <CreationFlag ... />
   <DeleteOnTermination ... />?
   <Source ... />*
   <Target ... />*
   <xsd:any##other/>*
</DataStaging>*
```

5.5.1.6 Example

A simple example of staging in a file:

```
<jsdl:DataStaging>
      <jsdl:FileName>control.txt</jsdl:FileName>
      <jsdl:Source>
        <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
      </jsdl:Source>
      <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
</jsdl:DataStaging>
```

A simple example of staging in a file relative to a FileSystem

```
<jsdl:FileSystem> ... </jsdl:FileSystem>
...
<jsdl:DataStaging>
      <jsdl:FileName>control.txt</jsdl:FileName>
      <jsdl:FileSystemName>HOME</jsdl:FileSystemName>
```

```
      <jsdl:Source>
        <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
      </jsdl:Source>
      <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
</jsdl:DataStaging>
```

Stage In and Stage Out – It is possible to define both Source and Target elements so that the file is first staged in before the job starts execution and staged out after the job finishes.

```
<jsdl:DataStaging>
      <jsdl:FileName>state.txt</jsdl:FileName>
      <jsdl:Source>
        <jsdl:URI>http://node1/~me/state.txt</jsdl:URI>
      </jsdl:Source>
      <jsdl:Target>
        <jsdl:URI>http://node2/~me/state.txt</jsdl:URI>
      </jsdl:Target>
      <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
</jsdl:DataStaging>
```

Multiple stage out operations may be specified by using the same FileName (on the same FileSystem) in separate DataStaging Elements.

```
<jsdl:DataStaging>
      <jsdl:FileName>result.txt</jsdl:FileName>
      <jsdl:Target>
        <jsdl:URI>http://node1/~me/result.txt</jsdl:URI>
      </jsdl:Target>
      ...
</jsdl:DataStaging>
...
<jsdl:DataStaging>
      <jsdl:FileName>result.txt</jsdl:FileName>
      <jsdl:Target>
        <jsdl:URI>http://node2/~me/result.txt</jsdl:URI>
      </jsdl:Target>
      ...
</jsdl:DataStaging>
```

It is also possible to use the same FileName in separate DataStaging elements to *stage in* to the same local file. The result of staging in to the same FileName on the same FileSystem are unspecified, however, since no the order is implied.

5.5.2 FileName Element

5.5.2.1 Definition

This element is a string specifying the *local* name of the file (or directory) on the execution host. The FileName MUST be a relative path (see §5.5.3). In other words it MUST NOT start with an initial '/'. The FileName MAY be a hierarchical directory path of the form `<directory>/<directory>/.../<name>`. The only delimiter allowed MUST be `/`. The `<name>` MAY be either a directory or a file.

5.5.2.2 Multiplicity – 1

5.5.2.3 Type – xsd:string

5.5.2.4 Attributes — None

5.5.2.5 Pseudo Schema

```
<FileName> xsd:string </FileName>
```

5.5.2.6 Example

A filename:

```
<jsdl:DataStaging>
      <jsdl:FileName>control.txt</jsdl:Filename>
      ...
</jsdl:DataStaging>
```

A hierarchical directory path specifying a file:

```
<jsdl:DataStaging>
      <jsdl:FileName>job1/input/control.txt</jsdl:FileName>
      ...
<jsdl:DataStaging>
```

A hierarchical directory path specifying a directory:

```
<jsdl:DataStaging>
      <jsdl:FileName>job1/input</jsdl:FileName>
      ...
</jsdl:DataStaging>
```

5.5.3 FileSystemName Element

5.5.3.1 Definition

If the `FileSystemName` is specified then the `FileName` is relative to the specified `FileSystem` declaration referenced by the name. In this case there MUST also be a `FileSystem` Element with the same name. If this element is not present then it is not defined. In this case the `FileName` is relative to the working job directory as specified by `WorkingDirectory` element. If the `WorkingDirectory` is also not specified then the base location is determined by the consuming system.

5.5.3.2 Multiplicity – 0-1

5.5.3.3 Type – xsd:NCName

5.5.3.4 Attributes — None

5.5.3.5 Pseudo Schema

```
<FileSystemName> xsd:NCName </FileSystemName>?
```

5.5.3.6 Example

Staging a file to a specific filesystem:

```
<jsdl:FileSystem > ... </jsdl:FileSystem>
...

<jsdl:DataStaging>
      <jsdl:FileSystemName>HOME</jsdl:FileSystemName>
      ...
</jsdl:DataStaging>
```

5.5.4 CreationFlag Element

5.5.4.1 Definition

This element determines whether the file created on the local execution system can overwrite or append to an existing file. A typical value for this element, expected to be commonly supported, is overwrite.

5.5.4.2 Multiplicity – 1

5.5.4.3 Type – jsdl:CreationFlagEnumeration

5.5.4.4 Attributes — None

5.5.4.5 Pseudo Schema

```
<CreationFlag> jsdl:CreationFlagEnumeration </CreationFlag>
```

5.5.4.6 Example

```
<jsdl:DataStaging>
      <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
      ...
</jsdl:DataStaging>
```

5.5.5 DeleteOnTermination Element

5.5.5.1 Definition

If true the file is deleted after the job terminates or after the file has been staged out. Otherwise the file remains on the execution host (subject to the persistency of the `FileSystem` it is on). If not present, behavior is unspecified and depends on the consuming system.

5.5.5.2 Multiplicity – 0-1

5.5.5.3 Type – xsd:boolean

- True – Remove the file after the job terminates.
- False – Do not remove the file after the job terminates.

5.5.5.4 Attributes — None

5.5.5.5 Pseudo Schema

```
<DeleteOnTermination> xsd:boolean </DeleteOnTermination>?
```

5.5.5.6 Example

```
<jsdl:DataStaging>
      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
      ...
</jsdl:DataStaging>
```

5.5.6 Source Element

5.5.6.1 Definition

A Source element contains the location of the file or directory on the remote system. This file or directory MUST be staged in from the location specified by the URI before the job has started. If this element is not present then the file does not have to be staged in.

5.5.6.2 Multiplicity – 0-1

5.5.6.3 Type – Complex

- URI Element – a URI according to [RFC 2396].

5.5.6.4 Attributes — None

5.5.6.5 Pseudo Schema

```
<Source>
   <URI ... />
   <xsd:any##other>*
</Source>?
```

5.5.6.6 Example

The source is a single file:

```
<jsdl:DataStaging>
      <jsdl:Source>
        <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
      </jsdl:Source>
      ...
</jsdl:DataStaging>
```

The source is a directory:

```
<jsdl:DataStaging>
      <jsdl:Source>
        <jsdl:URI>http://foo.bar.com/~me/job1/input</jsdl:URI>
      </jsdl:Source>
      ...
</jsdl:DataStaging>
```

The URI may specify any protocol, not just http:

```
<jsdl:DataStaging>
      <jsdl:Source>
        <jsdl:URI>ftp://foo.bar.com/~me/job1/input</jsdl:URI>
      </jsdl:Source>
      ...
</jsdl:DataStaging>
```

or

```
<jsdl:DataStaging>
      <jsdl:Source>
        <jsdl:URI>rsync://foo.bar.com/~me/job1/input</jsdl:URI>
      </jsdl:Source>
      ...
</jsdl:DataStaging>
```

5.5.7 URI Element

5.5.7.1 Definition

This is a URI [RFC 2396]. It MAY specify location (and protocol) that can be used to stage in or out a file.

5.5.7.2 Multiplicity – 1

5.5.7.3 Type – xsd:anyURI

5.5.7.4 Attributes — None

5.5.7.5 Pseudo Schema

```
<URI> xsd:anyURI </URI>
```

5.5.8 Target Element

5.5.8.1 Definition

A Target element contains the location of the file or directory on the remote system. This file or directory MUST be staged out to the location specified by the URI after the job has terminated. If this element is not present then the file or directory does not have to be staged out.

5.5.8.2 Multiplicity – 0-1

5.5.8.3 Type – Complex

- URI Element – a URI according to [RFC 2396].

5.5.8.4 Attributes — None

5.5.8.5 Pseudo Schema

```
<Target>
   <URI ... />
   <xsd:any##other>*
</Target>?
```

5.5.8.6 Example

The target may be a single file:

```
<jsdl:DataStaging>
      <jsdl:Target>
       <jsdl:URI>http://foo.bar.com/~me/job1/output.txt</jsdl:URI>
      </jsdl:Target>
      ...
</jsdl:DataStaging>
```

Or a directory:

```
<jsdl:DataStaging>
      <jsdl:Target>
        <jsdl:URI>http://foo.bar.com/~me/job1/output</jsdl:URI>
      </jsdl:Target>
      ...
</jsdl:DataStaging>
```

6 Extending JSDL

JSDL provides the overall structure to define the submission requirements of jobs. This structure may be extended to best fit more specialized needs. JSDL provides two mechanisms for extension: using attributes and using elements. In general using any extension mechanism will limit interoperability and so these mechanisms should be used sparingly and only when necessary. It is recommended that people interested in extending JSDL should first verify that no other group has provided an extension which meets their requirements.

If elements or attributes defined by extensions are present in a JSDL document they MUST be supported in the same way as JSDL elements and attributes.

6.1 Attribute Extension

Every JSDL element allows for additional attributes, as many as needed, provided these attributes have a namespace other than the normative namespaces defined by JSDL.
The following example (shortened for brevity) introduces an attribute called “order” that defines the order of staging in files:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
    xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/05/jsdl"
    xmlns:o="http://www.example.org/order-of-execution">
    <jsdl:JobDescription>
        <jsdl:DataStaging o:order="1">
            <jsdl:FileName>foo</jsdl:FileName>
            <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
             <jsdl:Source><jsdl:URI>http://www.nowhere.com/foo-
file</jsdl:URI></jsdl:Source>
        </jsdl:DataStaging>
        <jsdl:DataStaging o:order="2">
            <jsdl:FileName>bar</jsdl:FileName>
            <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
             <jsdl:Source><jsdl:URI>http:://www.nowhere.com/bar-
file</jsdl:URI></jsdl:Source>
        </jsdl:DataStaging>
```

```
    </jsdl:JobDescription>
</jsdl:JobDefinition>
```

6.2 Element Extension

Where applicable within the overall structure of the document, the JSDL schema allows for additional elements that are not normatively defined by JSDL. As with attribute extension, these elements must assume a namespace different than any namespace normatively defined by JSDL. The following example (shortened for brevity) extends JSDL by introducing resource reservations:

```
...
  <jsdl:Resources>
    <jsdl:TotalCPUCount><jsdl:Exact>1.0</jsdl:Exact></jsdl:TotalCPUCoun
t>
    <jsdl:TotalDiskSpace> <!-- At least 1 GB disk space -->
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
    </jsdl:TotalDiskSpace>
    <ug:Reservation xmlns:ug="http://www.example.org/reservation">
      <ug:Ticket>h933fsolenri900wnmd90mm34</ug:Ticket>
    </ug:Reservation>
  </jsdl:Resources>
...
```

6.3 Semantics of jsdl:other

JSDL defines enumerations of values for processor architectures and operating system. Both enumerations are not assumed to be exhaustively complete. These enumerations are embedded in wrapper types to allow for extension, "jsdl:CPUArchitectureType" and "jsdl:OperatingSystemTypeType", respectively.

To extend the enumerations, the special keyword "other" MUST be used for the values of the elements "jsdl:CPUArchitectureName" and "jsdl:OperatingSystemName", respectiveely, and the element extension pattern as described above MUST be used.

For example, given there exists a CPU that natively executes Java Bytecode named "JavaCPU", this CPU could be specified as follows in the resource section of a JSDL instance document:

```
...
<jsdl:Resources>
    <jsdl:CPUArchitecture>
        <jsdl:CPUArchitectureName>jsdl:other</jsdl:CPUArchitectureName>
        <ex:NewCPUTypes  xmlns:ex="http://www.example.org/NewCPUTypes">
JavaCPU</ex:NewCPUTypes>
    </jsdl:CPUArchitecture>
</jsdl:Resources>
...
```

7 Normative Extensions

The following extension is normatively defined by JSDL version 1.0.

7.1 Executables on POSIX conformant hosts

This normative extension defines a schema describing an application executed on a POSIX compliant system.

The namespace prefix used for this schema in the specification is "jsdl-posix." The normative namespace for this schema is given in Table 2-1.

7.1.1 POSIXApplication Element

7.1.1.1 Definition

This element describes a POSIX style Application and its requirements. It contains Executable, Argument, Input, Output, Error, WorkingDirectory, Environment, various POSIX limits elements as well as User and Group names. If it is present as a sub-element of the Application element it MUST appear only once.

7.1.1.2 Multiplicity – 0-1

7.1.1.3 Type – Complex

This complex type MUST support the following elements:

- Executable
- Argument
- Input
- Output
- Error
- WorkingDirectory
- Environment
- WallTimeLimit
- FileSizeLimit
- CoreDumpLimit
- DataSegmentLimit
- LockedMemoryLimit
- MemoryLimit
- OpenDescriptorsLimit
- PipeSizeLimit
- StackSizeLimit
- CPUTimeLimit
- ProcessCountLimit
- VirtualMemoryLimit
- ThreadCountLimit
- UserName
- GroupName

7.1.1.4 Attributes

- name – the name of this definition. Its type is xsd:NCName so that it can be reused and referred to from outside the containing JSDL document.

7.1.1.5 Pseudo Schema

```
<POSIXApplication name="xsd:NCName"?>
    <Executable ... />?
    <Argument ... />*
    <Input ... />?
    <Output ... />?
    <Error ... />?
```

```
    <WorkingDirectory ... />?
    <Environment ... />*
    <WallTimeLimit ... />?
    <FileSizeLimit ... />?
    <CoreDumpLimit ... />?
    <DataSegmentLimit ... />?
    <LockedMemoryLimit ... />?
    <MemoryLimit ... />?
    <OpenDescriptorsLimit ... />?
    <PipeSizeLimit ... />?
    <StackSizeLimit ... />?
    <CPUTimeLimit ... />?
    <ProcessCountLimit ... />?
    <VirtualMemoryLimit ... />?
    <ThreadCountLimit ... />?
    <UserName ... />?
    <GroupName ... />?
</POSIXApplication>?
```

7.1.2 Executable Element

7.1.2.1 Definition

A string specifying the command to execute. If ApplicationName is not specified and the Executable element is also not specified then the JSDL document defines a null job or a data staging. See also §5.3.1 and §5.3.2.

7.1.2.2 Multiplicity – 0-1

7.1.2.3 Type – xsd:string

7.1.2.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Executable element's content string MUST be interpreted as a filename relative to the mount point of the named filesystem.

7.1.2.5 Pseudo Schema

```
<Executable filesystemName="xsd:NCName"?> xsd:string </Executable>?
```

7.1.2.6 Example

Explicitly named executable:

```
<jsdl-posix:Executable> /usr/local/bin/gnuplot </jsdl-posix:Executable>
```

Handle in a consumer-defined way (usually by looking up the executable on the PATH):

```
<jsdl-posix:Executable> gnuplot </jsdl-posix:Executable>
```

Executable in a location relative to a named filesystem, usually *~/scripts/myExample*:

```
<jsdl-posix:Executable filesystemName="HOME">
  scripts/myExample
</jsdl-posix:Executable>
```

7.1.3 Argument Element

7.1.3.1 Definition

This element is a constrained normalized string specifying an argument element for the application. Argument elements can be empty and MUST NOT be collapsed.

7.1.3.2 Multiplicity – 0-n

7.1.3.3 Type – xsd:normalizedString

7.1.3.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Argument element's content string MUST be interpreted as a filename or directory name relative to the mount point of the named filesystem. If the Argument has an empty body and a present filesystemName, the argument to the application must be the mount point of the named filesystem.

7.1.3.5 Pseudo Schema

```
<Argument filesystemName=”xsd:NCName”?> xsd:normalizedString
</Argument>*
```

7.1.3.6 Example

Specify CLASSPATH and the class to invoke for a Java application:

```
<jsdl-posix:Argument>-cp</jsdl-posix:Argument>
<jsdl-posix:Argument>./example.jar</jsdl-posix:Argument>
<jsdl-posix:Argument>org.example.Main</jsdl-posix:Argument>
```

would be translated to '`[java] -cp ./example.jar org.example.Main`'

Start the Apache2 service on a Windows box without being Administrator:

```
<jsdl-posix:Argument>/user:Administrator@WORKGROUP</jsdl-
posix:Argument>
<jsdl-posix:Argument>net start Apache2</jsdl-posix:Argument>
```

would be translated to '`[runas] /user:Administrator@WORKGROUP “net start Apache2”`'

Echo a concatenated list of arguments to stdout:

```
<jsdl-posix:Argument>foo</jsdl-posix:Argument>
<jsdl-posix:Argument>bar</jsdl-posix:Argument>
<jsdl-posix:Argument>baz</jsdl-posix:Argument>
```

would be translated to '`[echo]foo bar baz`'

```
<jsdl-posix:Argument>foo</jsdl-posix:Argument>
<jsdl-posix:Argument>bar</jsdl-posix:Argument>
<jsdl-posix:Argument></jsdl-posix:Argument>
<jsdl-posix:Argument>baz</jsdl-posix:Argument>
```

would be translated to '`[echo]foo bar “” baz`'

```
<jsdl-posix:Argument>foo</jsdl-posix:Argument>
<jsdl-posix:Argument>bar</jsdl-posix:Argument>
<jsdl-posix:Argument>baz</jsdl-posix:Argument>
<jsdl-posix:Argument></jsdl-posix:Argument>
```

would be translated to '`[echo]foo bar baz ""`'

Echo the location of the user's home directory to stdout:

```
<jsdl-posix:Argument filesystemName="HOME"/>
```

would be translated to '[echo] /home/darrenp' (assuming that HOME is mounted at /home/darrenp).

7.1.4 Input Element

7.1.4.1 Definition

This element is a string specifying the input (Standard Input) for the command. The Input element is a filename relative to the working directory or to the named file-system. If this element is not present then it is not defined and the consuming system MAY choose any value.

7.1.4.2 Multiplicity – 0-1

7.1.4.3 Type – xsd:string

7.1.4.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Input element's content string MUST be interpreted as a filename relative to the mount point of the named filesystem.

7.1.4.5 Pseudo Schema

```
<Input filesystemName="xsd:NCName"?> xsd:string </Input>?
```

7.1.4.6 Example

The standard input should be taken from ~/input.txt

```
...
     <jsdl-posix:Input filesystemName="HOME">
         input.txt
     </jsdl-posix:Input>
...
```

7.1.5 Output Element

7.1.5.1 Definition

This element is a string specifying the output (Standard Output) for the command. The Output element is a filename relative the working directory or to the named file-system. If this element is not present then it is not defined and the consuming system MAY choose any value.

7.1.5.2 Multiplicity – 0-1

7.1.5.3 Type – xsd:string

7.1.5.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Output element's content string MUST be interpreted as a filename relative to the mount point of the named filesystem.

7.1.5.5 Pseudo Schema

```
<Output filesystemName=”xsd:NCName”?> xsd:string </Output>?
```

7.1.5.6 Example

The standard output should be put in ~/output.txt

```
...
     <jsdl-posix:Input filesystemName="HOME">
         output.txt
     </jsdl-posix:Input>
...
```

7.1.6 Error Element

7.1.6.1 Definition

This element is a string specifying the error output (Standard Error) for the command. The Error element is a filename relative to the working directory or to the named file-system. If this element is not present then it is not defined and the consuming system MAY choose any value.

7.1.6.2 Multiplicity – 0-1

7.1.6.3 Type – xsd:string

7.1.6.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Error element's content string MUST be interpreted as a filename relative to the mount point of the named filesystem.

7.1.6.5 Pseudo Schema

```
<Error filesystemName=”xsd:NCName”?> xsd:string </Error>?
```

7.1.6.6 Example

The standard error should be put in ~/error.txt

```
...
     <jsdl-posix:Input filesystemName="HOME">
         error.txt
     </jsdl-posix:Input>
...
```

7.1.7 WorkingDirectory Element

7.1.7.1 Definition

This element is a string specifying the starting directory required by the job to execute. If this is not present then it is not defined and the consuming system MAY choose any value. In many cases the working directory MAY be related to a FileSystem element by specifying the WorkingDirectory's path to be the same as the local mount path of the FileSystem Element. The WorkingDirectory element MAY not have a required direct relationship to a particular FileSystem element.

7.1.7.2 Multiplicity – 0-1

7.1.7.3 Type – xsd:string

7.1.7.4 Attributes

- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the WorkingDirectory element's content string MUST be interpreted as a directory name relative to the mount point of the named filesystem.

7.1.7.5 Pseudo Schema

```
<WorkingDirectory filesystemName="xsd:NCName"?> xsd:string
</WorkingDirectory>?
```

7.1.7.6 Example

The job's working directory must be the user's home directory:

```
<WorkingDirectory filesystemName="HOME"/>
```

7.1.8 Environment Element

7.1.8.1 Definition

This element specifies the name and value of an environment variable that will be defined for the job in the execution environment.
As a special case an environment variable may be declared to contain the mount-point of a filesystem declared in a JSDL document. The linkage is done using the filesystemName attribute of the Environment element and the name of the FileSystem element. See the example section below. See also §5.4.17.

7.1.8.2 Multiplicity – 0-n

7.1.8.3 Type – xsd:string

This is the value of the environment variable.

7.1.8.4 Attributes

- name – The name of the environment variable.
- filesystemName – The name of a filesystem defined in a FileSystem element inside the JSDL document. If present, the Environment element's content string MUST be interpreted as a filename or directory name relative to the mount point of the named filesystem.

7.1.8.5 Pseudo Schema

```
<Environment name=″xsd:NCName″ filesystemName=″xsd:NCName″?> xsd:string
</Environment>*
```

7.1.8.6 Example

Set the SHELL to bash:

```
<Environment name=″SHELL″>/bin/bash</Environment>
```

Set the location of MAIL to ~/mailboxes/INBOX:

```
<jsdl-posix:Environment name="MAIL" filesystemName="HOME">
   mailboxes/INBOX
</jsdl-posix:Environment>
```

Set the environment variable TMPDIR to the root location of the filesystem with name “TMP”.

```
<jsdl:FileSystem name=″TMP″>...</jsdl:FileSystem>
...
<jsdl-posix:Environment name=″TMPDIR″ filesystemName=″TMP″/>
...
```

7.1.9 WallTimeLimit Element

7.1.9.1 Definition

This element is a positive integer that specifies the soft limit on the duration of the application’s execution, in seconds. If this element is not present then the consuming system MAY choose its default value.

7.1.9.2 Multiplicity – 0-1

7.1.9.3 Type – xsd:positiveInteger

7.1.9.4 Attributes — None

7.1.9.5 Pseudo Schema

```
<WallTimeLimit> xsd:positiveInteger </WallTimeLimit>?
```

7.1.9.6 Example

Set the duration to one minute:

```
<jsdl-posix:WallTimeLimit> 60 </jsdl-posix:WallTimeLimit>
```

7.1.10 FileSizeLimit Element

7.1.10.1 Definition

This element is a positive integer that describes the maximum filesize of any given file associated with this job. The file size is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.10.2 Multiplicity – 0-1

7.1.10.3 Type – xsd:positiveInteger

7.1.10.4 Attributes — None

7.1.10.5 Pseudo Schema

```
<FileSizeLimit> xsd:positiveInteger </FileSizeLimit>?
```

7.1.10.6 Example

Set the maximum file size to 1 GB:

```
<jsdl-posix:FileSizeLimit> 1073741824 </jsdl-posix:FileSizeLimit>
```

7.1.11 CoreDumpLimit Element

7.1.11.1 Definition

This element is a positive integer that describes the maximum size of core dumps a job may create. The size is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.11.2 Multiplicity – 0-1

7.1.11.3 Type – xsd:positiveInteger

7.1.11.4 Attributes — None

7.1.11.5 Pseudo Schema

```
<CoreDumpLimit> xsd:positiveInteger </CoreDumpLimit>?
```

7.1.11.6 Example

Disable core dumps by setting core dump size to 0:

```
<jsdl-posix:CoreDumpLimit> 0 </jsdl-posix:CoreDumpLimit>
```

7.1.12 DataSegmentLimit Element

7.1.12.1 Definition

This element is a positive integer that limits the data segment to the given size. The amount is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.12.2 Multiplicity – 0-1

7.1.12.3 Type – xsd:positiveInteger

7.1.12.4 Attributes — None

7.1.12.5 Pseudo Schema

```
<DataSegmentLimit> xsd:positiveInteger </DataSegmentLimit>?
```

7.1.12.6 Example

Limit the data segment to 32KB:

```
<jsdl-posix:DataSegmentLimit> 32768 </jsdl-posix:DataSegmentLimit>
```

7.1.13 LockedMemoryLimit Element

7.1.13.1 Definition

This element is a positive integer that describes the maximum amount of physical memory this job is allowed to lock. The amount is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.13.2 Multiplicity – 0-1

7.1.13.3 Type – xsd:positiveInteger

7.1.13.4 Attributes — None

7.1.13.5 Pseudo Schema

```
<LockedMemoryLimit> xsd:positiveInteger </LockedMemoryLimit>?
```

7.1.13.6 Example

Allow up to 8MB of locked memory:

```
<jsdl-posix:LockedMemoryLimit> 8388608 </jsdl-posix:LockedMemoryLimit>
```

7.1.14 MemoryLimit Element

7.1.14.1 Definition

This element is a positive integer that describes the maximum amount of physical memory (in bytes) that the job should use when executing. If this is not present then the consuming system MAY choose its default value.

[Note that this is really the Resident Set Size limit, and systems might impose a granularity of a kilobyte.]

7.1.14.2 Multiplicity – 0-1

7.1.14.3 Type – xsd:positiveInteger

7.1.14.4 Attributes — None

7.1.14.5 Pseudo Schema

```
<MemoryLimit> xsd:positiveInteger </MemoryLimit>?
```

7.1.14.6 Example

Set RSS to 64MB:

```
<jsdl-posix:MemoryLimit> 67108864 </jsdl-posix:MemoryLimit>
```

7.1.15 OpenDescriptorsLimit Element

7.1.15.1 Definition

This element is a positive integer that describes the maximum number of open file-descriptors (files, pipes, sockets) a job can have. If this element is not present then the consuming system MAY choose its default value.

7.1.15.2 Multiplicity – 0-1

7.1.15.3 Type – xsd:positiveInteger

7.1.15.4 Attributes — None

7.1.15.5 Pseudo Schema

```
<OpenDescriptorsLimit> xsd:positiveInteger </OpenDescriptorsLimit>?
```

7.1.15.6 Example

The maximum number of descriptors should be 16:

```
<jsdl-posix:OpenDescriptorsLimit> 16 </jsdl-posixOpenDescriptorsLimit>
```

7.1.16 PipeSizeLimit Element

7.1.16.1 Definition

This element is a positive integer that describes the maximum size of pipelines (in bytes) created or and by the processing of the job. If this is not present then the consuming system MAY choose its default value.

[Note that no system permits arbitrary sized pipe buffers, but their defaults are usually entirely acceptable. Systems typically impose a granularity of 512 bytes.]

7.1.16.2 Multiplicity – 0-1

7.1.16.3 Type – xsd:positiveInteger

7.1.16.4 Attributes — None

7.1.16.5 Pseudo Schema

```
<PipeSizeLimit> xsd:positiveInteger </PipeSizeLimit>?
```

7.1.16.6 Example

Set a limit of 512B:

```
<jsdl-posix:PipeSizeLimit> 512 </jsdl-posix:PipeSizeLimit>
```

7.1.17 StackSizeLimit Element

7.1.17.1 Definition

This element is a positive integer that describes maximum size of the execution stack for this job. The amount is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.17.2 Multiplicity – 0-1

7.1.17.3 Type – xsd:positiveInteger

7.1.17.4 Attributes — None

7.1.17.5 Pseudo Schema

```
<StackSizeLimit> xsd:positiveInteger </StackSizeLimit>?
```

7.1.17.6 Example

Set a stack size of 1MB:

```
<jsdl-posix:StackSizeLimit> 1048576 </jsdl-posix:StackSizeLimit>
```

7.1.18 CPUTimeLimit Element

7.1.18.1 Definition

This element is a positive integer that describes the number of CPU time seconds a job is allowed to consume before a SIGXCPU signal is sent to the job. The amount is given in seconds. If this element is not present then it is not defined and the consuming system MAY choose its default value..

7.1.18.2 Multiplicity – 0-1

7.1.18.3 Type – xsd:positiveInteger

7.1.18.4 Attributes — None

7.1.18.5 Pseudo Schema

```
<CPUTimeLimit> xsd:positiveInteger </CPUTimeLimit>?
```

7.1.18.6 Example

Allow 30 seconds of CPU time:

```
<jsdl-posix:CPUTimeLimit> 30 </jsdl-posix:CPUTimeLimit>
```

7.1.19 ProcessCountLimit Element

7.1.19.1 Definition

This element is a positive integer that describes the maximum allowed number of processes this job may spawn. If this element is not present then the consuming system MAY choose its default value.

7.1.19.2 Multiplicity – 0-1

7.1.19.3 Type – xsd:positiveInteger

7.1.19.4 Attributes — None

7.1.19.5 Pseudo Schema

```
<ProcessCountLimit> xsd:positiveInteger </ProcessCountLimit>?
```

7.1.19.6 Example

At most 8 processes should be allowed:

```
<jsdl-posix:ProcessCountLimit> 8 </jsdl-posix:ProcessCountLimit>
```

7.1.20 VirtualMemoryLimit Element

7.1.20.1 Definition

This element is a positive integer that describes the maximum allowed amount of virtual memory this job can allocate. The amount is given in Bytes. If this element is not present then the consuming system MAY choose its default value.

7.1.20.2 Multiplicity – 0-1

7.1.20.3 Type – xsd:positiveInteger

7.1.20.4 Attributes — None

7.1.20.5 Pseudo Schema

```
<VirtualMemoryLimit> xsd:positiveInteger </VirtualMemoryLimit>?
```

7.1.20.6 Example

The maximum virtual memory limit should be 128 MB:

```
<jsdl-posix:VirtualMemoryLimit>
  134217728
</jsdl-posix:VirtualMemoryLimit>
```

7.1.21 ThreadCountLimit Element

7.1.21.1 Definition

This element is a positive integer that describes the number of threads this job is allowed to create. If this element is not present then the consuming system MAY choose its default value.

7.1.21.2 Multiplicity – 0-1

7.1.21.3 Type – xsd:positiveInteger

7.1.21.4 Attributes — None

7.1.21.5 Pseudo Schema

```
<ThreadCountLimit> xsd:positiveInteger </ThreadCountLimit>?
```

7.1.21.6 Example

Not more than 8 threads:

```
<jsdl-posix:ThreadCountLimit> 8 </jsdl-posix:ThreadCountLimit>
```

7.1.22 UserName

7.1.22.1 Definition

The user name to be used when executing the application. This element has the type of a string. If it is not present then it is not defined and the consuming system MAY select a user name based on implementation.

7.1.22.2 Multiplicity – 0-1

7.1.22.3 Type – xsd:string

7.1.22.4 Attributes — None

7.1.22.5 Pseudo Schema

```
<UserName> xsd:string </UserName>?
```

7.1.22.6 Example

A UNIX user name:

```
<jsdl-posix:UserName>frank</jsdl-posix:UserName>
```

7.1.23 GroupName

7.1.23.1 Definition

The group name to be used when executing the application. This element has the type of a string. If it is not present then it is not defined and then the consuming system MAY select a group name based on implementation.

7.1.23.2 Multiplicity – 0-1

7.1.23.3 Type – xsd:string

7.1.23.4 Attributes — None

7.1.23.5 Pseudo Schema

```
<GroupName> xsd:string </GroupName>?
```

7.1.23.6 Example

A UNIX group name:

```
<jsdl-posix:GroupName>staff</ jsdl-posix:GroupName>
```

8 Security Considerations

This specification defines a language for describing the requirements of computational jobs for submission to Grids and other job management systems. It is assumed that job submission must be secured but such mechanisms are out of scope of this specification.

The ability to describe what rights are needed for job execution is very important. It is expected that JSDL version 1.0 instance documents should be composed with more specialized security or policy languages to express fine-grained delegation of rights when required.

Author Information

Ali Anjomshoaa
EPCC
University of Edinburgh
James Clerk Maxwell Building, Mayfield Road, Edinburgh EH9 3JZ, UK
Email: ali@epcc.ed.ac.uk

Fred Brisard
Computer Associates Intl, Inc.
Email: Fred.Brisard@ca.com

An Ly
Computer Associates Intl, Inc.
Email: an.ly@ca.com

Andrew Stephen McGough
London e-Science Center
Imperial College London
Department of Computing, 180 Queen's Gate, London SW7 2BZ, UK
Email: asm@doc.ic.ac.uk

Darren Pulsipher
Email: darren@pulsipher.org
Ovoca, LLC

Andreas Savva
Grid Computing and Bioinformatics Laboratory
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Phone: +81 (044) 777-1111
Email: andreas.savva@jp.fujitsu.com

Contributors

We gratefully acknowledge the contributions made to this specification by Michel Drescher, Donal Fellows, William Lee, Chris Smith, David Snelling, ….

Acknowledgements

We are grateful to numerous colleagues for discussions on the topics covered in this document, in particular (in alphabetical order, with apologies to anybody we've missed) Lee Cook, ….

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).

Normative References

[RFC 2119] Bradner, S. *Key words for use in RFCs to Indicate Requirement Levels*. Internet Engineering Task Force, RFC 2119, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt

[RFC 2396] T. Berners-Lee and R. Fielding and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax.* . Internet Engineering Task Force, August 1998. Available at http://www.ietf.org/rfc/rfc2396.txt

[CIM] Common Information Model (CIM) Specification, Version 2.9, January 2005. Available at http://www.dmtf.org/standards/cim/cim_schema_v29

[POSIX] ISO/IEC 9945:2003, Portable Operating System Interface, version 3.

Informative References

[WS-Agreement] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S. and Xu, M. Web Services Agreement Specification (WS-Agreement). Global Grid Forum GRAAP-WG, Draft, August 2004.

Appendix 1 JSDL Normative Schema

DEFER UPDATING here until final versions are out.
Corresponding version of schema available at:
https://forge.gridforum.org/projects/jsdl-wg/document/jsdl.xsd/en/13

Appendix 2 JSDL POSIXApplication Normative Schema

DEFER UPDATING here until final versions are out.
Corresponding version of schema available at:
https://forge.gridforum.org/projects/jsdl-wg/document/jsdl-posix.xsd/en/3

Appendix 3 JSDL Examples

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/05/jsdl" xmlns:jsdl-
posix="http://schemas.ggf.org/jsdl/2005/05/jsdl-posix"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <jsdl:JobDescription>
            <jsdl:JobIdentification>
                  <jsdl:JobName>My Gnuplot invocation</jsdl:JobName>
                  <jsdl:Description> Simple application invokation:
                User wants to run the application 'gnuplot' to produce
a plotted
                graphical file based on some data shipped in from
elsewhere
                (perhaps as part of a workflow).  A front-end
application will then
                build into an animation of spinning data.
                Front-end application knows URL for data file which
must be stage
                in. Front-end application wants to stage in a control
file that it
                specifies directly which directs gnuplot to produce the
output files.
                In case of error, messages should be produced on stderr
(also to be
                staged on completion) and no images are to be
transferred.
            </jsdl:Description>
            </jsdl:JobIdentification>
            <jsdl:Application>
                  <jsdl:ApplicationName>gnuplot</jsdl:ApplicationName>
                  <jsdl-posix:POSIXApplication>
                        <jsdl-
posix:Executable>/usr/local/bin/gnuplot</jsdl-posix:Executable>
                        <jsdl-posix:Argument>control.txt</jsdl-
posix:Argument>
                        <jsdl-posix:Input>input.dat </jsdl-posix:Input>
                        <jsdl-posix:Output>output1.png</jsdl-
posix:Output>
                  </jsdl-posix:POSIXApplication>
            </jsdl:Application>
            <jsdl:Resources>
                  <jsdl:TotalCPUCount>
                        <jsdl:Exact> 1.0 </jsdl:Exact>
                  </jsdl:TotalCPUCount>
                  <jsdl:IndividualPhysicalMemory>

      <jsdl:LowerBoundedRange>2097152.0</jsdl:LowerBoundedRange>
                  </jsdl:IndividualPhysicalMemory>
            </jsdl:Resources>
```

```
            <jsdl:DataStaging>
                  <jsdl:FileName>control.txt</jsdl:FileName>
                  <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>

      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                  <jsdl:Source>

      <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
                  </jsdl:Source>
            </jsdl:DataStaging>
            <jsdl:DataStaging>
                  <jsdl:FileName>input.dat</jsdl:FileName>
                  <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>

      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                  <jsdl:Source>

      <jsdl:URI>http://foo.bar.com/~me/input.dat</jsdl:URI>
                  </jsdl:Source>
            </jsdl:DataStaging>
            <jsdl:DataStaging>
                  <jsdl:FileName>output1.png</jsdl:FileName>
                  <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>

      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                  <jsdl:Target>

      <jsdl:URI>rsync://spoolmachine/userdir</jsdl:URI>
                  </jsdl:Target>
            </jsdl:DataStaging>
      </jsdl:JobDescription>
</jsdl:JobDefinition>
```