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

15 June 2005

Job Submission Description Language (JSDL) Specification, Version 1.0

Status of this Memo

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

Copyright Notice

Abstract

This document specifies 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 document includes the normative XML Schema for the JSDL, 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 requirements of computational jobs for submission to resources, particularly in Grid environments, though not restricted to the latter. The JSDL language contains a vocabulary and normative XML Schema that facilitate the expression of those requirements as a set of XML elements.

The JSDL specification is motivated by two main scenarios. First, many organizations accommodate a variety of job management systems, where each system has its own language for describing job submission requirements. This makes interoperability between these systems for job management difficult. In order to utilize all these different systems these organizations 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, such as JSDL, which can be easily mapped to the various systems, would clearly alleviate this problem.

Second, Grid environments involve the interaction of a number of different types of job management systems in a heterogeneous environment. Figure 1 illustrates one possible view of such a Grid environment. The description of a job submission (shown as a JSDL document in Figure 1) on the Grid, may be transformed by intermediaries or refined further by information not available to the initial submitter of that job. This description may be passed between systems or instantiated on resources matching the resource requirements for that job. All these interactions can be undertaken automatically, facilitated by a standard language that can be easily translated to each system's own language.

Figure 1 illustrates a small number of the systems where a JSDL document may be used within a Grid environment. Usage of a JSDL document is not limited to these systems alone and many other JSDL consumers may exist, including: accounting systems, security systems, archiving systems, and provenance (auditing) systems.

JSDL 1.0 provides the core vocabulary for describing a job for submission to Grid environments. This core vocabulary is informed by a number of existing systems: Condor[1], Globus Toolkit[2], Load Sharing Facility (LSF)[3], Portable Batch System (PBS)[4], (Sun) GridEngine (SGE)[5], and Uniform Interface to Computing Resources (Unicore)[6]. It is also informed by a number of existing standards or proposed standards: Distributed Resource Management Application API Specification 1.0[DRMAA], Common Information Model (CIM) 2.9.0[CIM], and Portable Operating System Interface (POSIX) 3[POSIX].

It is envisioned that more complex functionality will be provided through non-standardized extensions to the core JSDL 1.0 specification, some of which may become normative standard extensions to JSDL in due course.

JSDL 1.0 elements fall into the following general categories:

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

---

[1] Condor Project Homepage: http://www.cs.wisc.edu/condor/

[2] Globus Toolkit Homepage: http://www.globus.org/toolkit/

[3] Load Sharing Facility (LSF): http://www.platform.com/products/LSF/

[4] Portable Batch System (PBS): http://www.openpbs.org/

[5] (Sun) GridEngine (SGE): http://gridengine.sunsource.net/

[6] Uniform Interface to Computing Resources (UNICORE): http://unicore.sourceforge.net/

JSDL 1.0 elements are restricted to the description of requirements of jobs at submission time. There are no elements defined in JSDL 1.0, therefore, that contain any information about a job after it has been submitted. Information such as unique job identifiers or job status information are typically maintained by underlying job management systems, and may be available in separate documents or added to JSDL 1.0 compliant documents through extensions. Section 3 describes the rationale behind the scope of JSDL 1.0.

The JSDL 1.0 language elements are defined in Sections 4 and 6 and the normative XML Schema is provided in Appendix 1.

JSDL 1.0 is an extensible specification. Section 7 describes the extension mechanism for JSDL 1.0. Normative extensions are defined in Section 8.

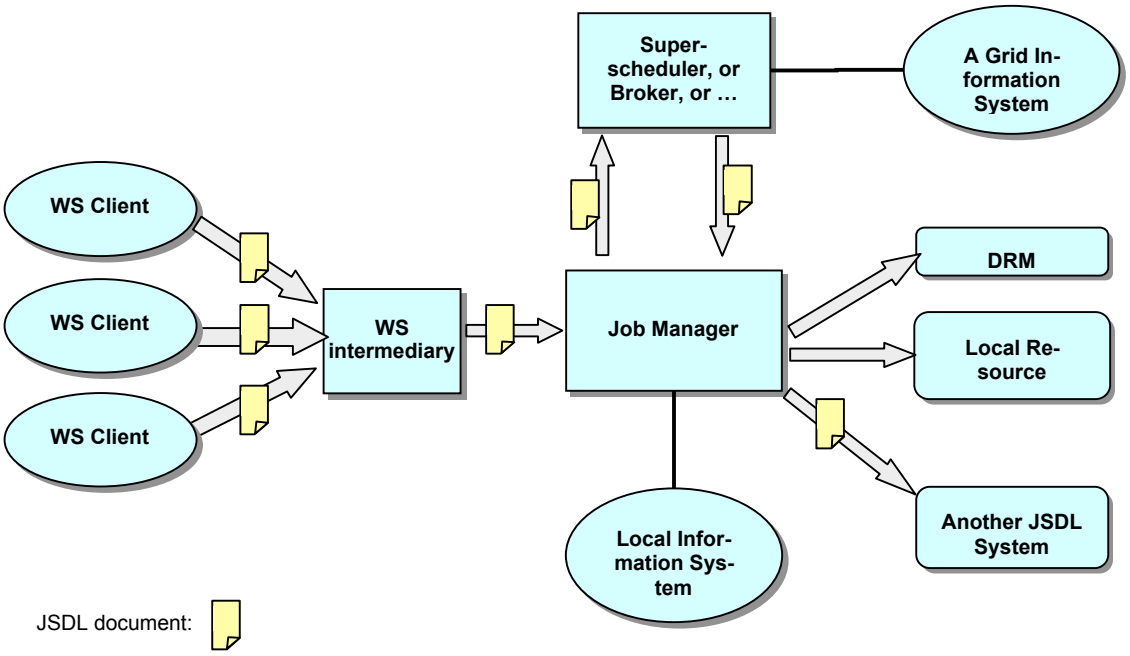

Figure 1: JSDL consumers in a Grid environment.

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 zero or one occurrences; ‘*’ denotes zero or more occurrences; ‘+’ denotes one or more occurrences. 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_this_element />+
</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/2005/06/jsdl
jsdl-posix	http://schemas.ggf.org/jsdl/2005/06/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.

The key word *satisfy*, with respect to a consuming system satisfying a JSDL document, means that the system can both support and satisfy all elements present in the document. An element is satisfied when the consuming system can provide an appropriate implementation of the value assigned to that element in the JSDL document.

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. It is therefore anticipated that, 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 WS-Agreement specification[WS-AG]. 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.

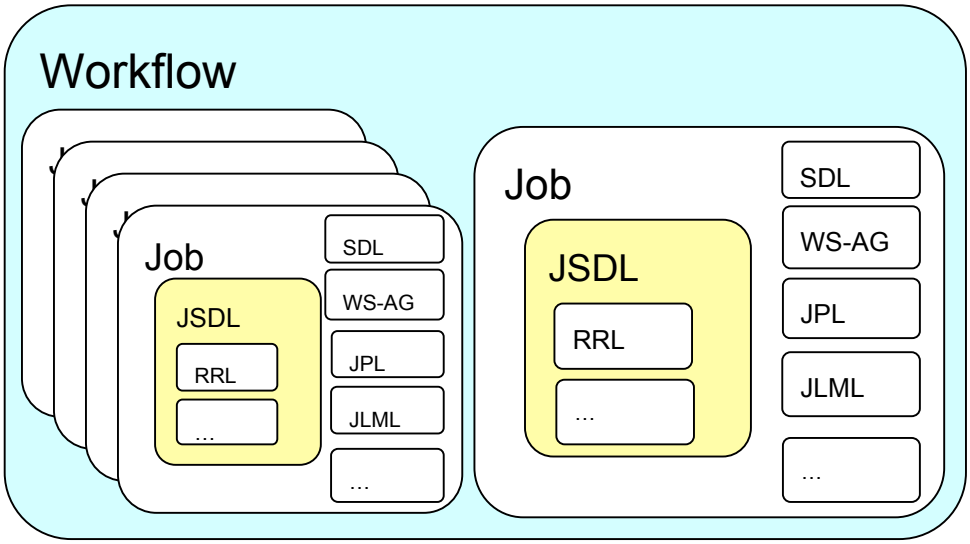

Figure 2: Relationship between JSDL and other specifications.

Legend:

RRL—Resource Requirements Language

SDL—Scheduling Description Language

WS-AG—Web Services Agreement

JPL—Job Policy Language

JLML—Job Lifetime Management Language

3.1 On Resource Requirements Language (RRL)

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.* Job start and end times, validity windows for running a job, etc.;
- *Data-dependent scheduling.* Scheduling jobs based on the expected availability or validity of input data, etc.; and
- *Workflow-dependent scheduling.* 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 WS-Agreement (WS-AG) specification[WS-AG]. 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 submission 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 being developed by the OASIS WSRF Techical Committee[7].)

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 JSDL Document Structure

A JSDL document is described using XML and adheres to the normative XML schema contained in Appendix 1.

A JSDL 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, Resources, and DataStaging. The pseudo schema definition follows:

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

[7] OASIS Web Services Resource Framework TC: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

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

Complete examples of JSDL documents are in Appendix 3.

All elements present in a JSDL document MUST be satisfied for the entire document to be satisfied. A JSDL element is satisfied when a consuming system can support it and can provide an appropriate implementation of the value assigned to that element in the JSDL document.

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

5 The JSDL Element Types

The JSDL specification uses a number of normative XML Schema 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 the Common Information Model (CIM) and the Portable Operating System Interface (POSIX). Since there is no available normative XML Schema definition of these types one is provided here. The Operating Systems types (see §5.2.3) is defined based on CIM[CIM].

A number of types define a special value “other.” This value can be used to introduce elements from other specifications. See §7.3 for more details.

5.1 Normative XML Schema Types

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

Table 5-1 Normative XML Schema Types

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

5.2 JSDL types

The following types are defined by JSDL 1.0: ProcessorArchitectureEnumeration, FileSystemTypeEnumeration, OperatingSystemTypeEnumeration, CreationFlagEnumeration, and RangeValue_Type.

5.2.1 ProcessorArchitectureEnumeration Type

The processor architecture enumeration is based on the Instruction Set Architecture (ISA) names of a small number of common processor architectures. The following are the values that MUST be supported.

Table 5-2 Processor Architectures (jsdl:ProcessorArchitectureEnumeration)

Normative JSDL Name	Definition
sparc	A SPARC architecture processor

powerpc	A PowerPC architecture processor
x86	An Intel Architecture processor derived from the 8086 chip set
x86_32	An x86 processor capable of 32-bit processing mode
x86_64	An x86 processor capable of 64-bit processing mode
parisc	A PARISC architecture processor
mips	A MIPS architecture processor
ia64	An Intel Architecture 64-bit 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 name-space.

5.2.2 FileSystemTypeEnumeration Type

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

Table 5-3 FileSystem Types (jsdl:FileSystemTypeEnumeration)

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 job completes
normal	This is a normal space for files to be written to and read from. Files are not deleted after the job completes unless the user explicitly re-quests it

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

5.2.3 OperatingSystemTypeEnumeration Type

The following values MUST be supported by the consuming system. The values are based on the OSType field of the CIM_OperatingSystem model[CIM].

Table 5-4 Operating System Types (jsdl:OperatingSystemTypeEnumeration)

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 name-space.

5.2.4 CreationFlagEnumeration Type

This following file creation flags MUST be supported.

Table 5-5 File Creation Flags (jsdl:CreationFlagEnumeration)

Normative JSDL Name	Definition
overwrite	Overwrite an existing file with the same name; create a new one otherwise

	erwise
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

5.2.5 RangeValue_Type Type

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 carried out 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. The formal matching semantics are defined in the next section (§5.2.5.1).

5.2.5.1 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 $N < U$.

5.2.5.2 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>*
```

5.2.5.3 Examples

The pseudo-expression “5, 6.7777, 7.0, [50.3,99.5), [100-” (which indicates a disjoint range consisting of the values 5, 6.7777, 7.0, all values inclusive of from 50.3 up to, but not including, 99.5 and all values from 100 on up) 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>
```

6 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.)

6.1 Job Structure Elements

6.1.1 JobDefinition Element

6.1.1.1 Definition

This element describes the job and its requirements. It contains a JobDescription section. It is the root element of the JSDL document.

6.1.1.2 Multiplicity

The multiplicity of this element is one.

6.1.1.3 Type

This is a complex type. It MUST support the following elements:

- JobDescription

6.1.1.4 Attributes

The following attributes are defined:

- id—the id of the Job definition document. This is defined as a “xsd:ID” and is in the default namespace of the document. The id MAY be omitted.

6.1.1.5 Pseudo Schema

```
<JobDefinition id=”xsd:ID”?>
```

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

6.1.1.6 Examples

A document with the id “gnuplot”:

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

6.1.2 JobDescription Element

6.1.2.1 Definition

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

6.1.2.2 Multiplicity

The multiplicity of this element is one.

6.1.2.3 Type

This is a complex type. It MUST support the following elements:

- JobIdentification
- Application
- Resources
- DataStaging

6.1.2.4 Attributes

No attributes are defined.

6.1.2.5 Pseudo Schema

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

6.1.3 Description Element

6.1.3.1 Definition

This element provides descriptive, human readable, information about its containing complex element. It MAY 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.

6.1.3.2 Multiplicity

The multiplicity of this element is zero or one.

6.1.3.3 Type

The type of this element is xsd:string.

6.1.3.4 Attributes

No attributes are defined.

6.1.3.5 Pseudo Schema

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

6.2 Job Identity Elements

6.2.1 JobIdentification Element

6.2.1.1 Definition

This element contains all elements that identify the job: JobName, Description, JobAnnotation, and JobProject. If this element is not present then its value, including all of its sub-elements, is undefined.

6.2.1.2 Multiplicity

The multiplicity of this element is zero or one.

6.2.1.3 Type

This is a complex type. It MUST support the following elements:

- JobName
- Description
- JobAnnotation
- JobProject

6.2.1.4 Attributes

No attributes are defined.

6.2.1.5 Pseudo Schema

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

6.2.2 JobName Element

6.2.2.1 Definition

This element is a string that MAY be specified by a user to name the job specified in the 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.

6.2.2.2 Multiplicity

The multiplicity of this element is zero or one.

6.2.2.3 Type

The type of this element is xsd:string.

6.2.2.4 Attributes

No attributes are defined.

6.2.2.5 Pseudo Schema

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

6.2.2.6 Examples

The name may be used for search and sort purposes and for job management.

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

6.2.3 JobAnnotation Element

6.2.3.1 Definition

This element is a string that MAY be specified by a user to annotate the job. If this element is not present then it is not defined. In contrast to the Description element, JobAnnotation MAY contain information that is intended for use by the consuming system.

6.2.3.2 Multiplicity

The multiplicity of this element is zero or more.

6.2.3.3 Type

The type of this element is xsd:string.

6.2.3.4 Attributes

No attributes are defined.

6.2.3.5 Pseudo Schema

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

6.2.3.6 Examples

The annotation may be used to provide a unique identifier meaningful to the consuming system.

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

6.2.4 JobProject Element

6.2.4.1 Definition

This element is 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.

6.2.4.2 Multiplicity

The multiplicity of this element is zero or more.

6.2.4.3 Type

The type of this element is xsd:string.

6.2.4.4 Attributes

No attributes are defined.

6.2.4.5 Pseudo Schema

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

6.2.4.6 Examples

The job belongs to the project group “wcdma”.

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

6.3 Application Requirements

6.3.1 Application Element

6.3.1.1 Definition

This element describes the Application and its requirements. It contains ApplicationName, ApplicationVersion and Description elements. It serves as a high level generic container that is intended to hold more specific application definitions. One such definition is that of the POSIX compliant normative extension given in §8.1.1. 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. The JSDL document could be defining a data staging job, or a null job. See also §6.3.2 and §8.1.2.

6.3.1.2 Multiplicity

The multiplicity of this element is zero or one.

6.3.1.3 Type

This is a complex type. It MUST support the following elements:

- ApplicationName
- ApplicationVersion
- Description

6.3.1.4 Attributes

No attributes are defined.

6.3.1.5 Pseudo Schema

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

6.3.2 ApplicationName Element

6.3.2.1 Definition

This element is a string that defines the name of the application and is used to identify the application independent of the location of its executable on a host or system. If this is not present then it is not defined and a null job is assumed unless there is an application extension element present that defines the application to execute. See also §6.3.1 and §8.1.2.

6.3.2.2 Multiplicity

The multiplicity of this element is zero or one.

6.3.2.3 Type

The type of this element is xsd:string.

6.3.2.4 Attributes

No attributes are defined.

6.3.2.5 Pseudo Schema

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

6.3.2.6 Examples

The BLAST application:

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

6.3.3 ApplicationVersion Element

6.3.3.1 Definition

This element is a string that 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.

6.3.3.2 Multiplicity

The multiplicity of this element is zero or one.

6.3.3.3 Type

The type of this element is xsd:string.

6.3.3.4 Attributes

No attributes are defined.

6.3.3.5 Pseudo Schema

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

6.3.3.6 Examples

The BLAST application version 1.4:

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

6.4 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.

6.4.1 Resources Element

6.4.1.1 Definition

This element contains 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.

Any combination of the listed Resources sub-elements MAY be present in the Resources element of a JSDL document. In particular, any combination of “Individual”, and “Total” elements of the same or different types MAY be present in a Resources element. But note that *all* elements present in a JSDL document MUST be satisfied for the entire document to be satisfied. (See also §4.)

6.4.1.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.1.3 Type

This is a complex type. It MUST support the following elements:

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

6.4.1.4 Attributes

No attributes are defined.

6.4.1.5 Pseudo Schema

```
<Resources>
   <CandidateHosts ... />?
   <FileSystem .../>*
   <ExlusiveExecution .../>?
   <OperatingSystem .../>?
   <CPUArchitecture .../>?
   <IndividualCPUSpeed .../>?
   <IndividualCPUTime .../>?
   <IndividualCPUCount .../>?
   <IndividualNetworkBandwidth .../>?
   <IndividualPhysicalMemory .../>?
   <IndividualVirtualMemory .../>?
   <IndividualDiskSpace .../>?
   <TotalCPUTime .../>?
   <TotalCPUCount .../>?
   <TotalPhysicalMemory .../>?
   <TotalVirtualMemory .../>?
   <TotalDiskSpace .../>?
```

```
   <TotalResourceCount .../>?
   <xsd:any##other>*
</Resources>?
```

6.4.2 CandidateHosts Element

6.4.2.1 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 the set 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.

6.4.2.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.2.3 Type

This is a complex type. It MUST support the following elements:

- HostName

6.4.2.4 Attributes

No attributes are defined.

6.4.2.5 Pseudo Schema

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

6.4.2.6 Examples

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.

6.4.3 HostName Element

6.4.3.1 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.

6.4.3.2 Multiplicity

The multiplicity of this element is one or more.

6.4.3.3 Type

The type of this element is xsd:string.

6.4.3.4 Attributes

No attributes are defined.

6.4.3.5 Pseudo Schema

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

6.4.3.6 Examples

The machine “bach.example.com” 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>
```

6.4.4 FileSystem Element

6.4.4.1 Definition

This element describes a filesystem that is required by the job. It is a complex type that may contain the location where the filesystem should be made available, the required amount of disk space and the type of the filesystem. The filesystem may be local to the resource (e.g., on a local disk), or may be remote (e.g., an NFS mount).

6.4.4.2 Multiplicity

The multiplicity of this element is zero or more.

6.4.4.3 Type

This is a complex type. It MUST support the following elements:

- Description
- MountPoint
- MountSource
- DiskSpace
- FileSystemType

6.4.4.4 Attributes

The following attributes are defined:

- name—the name of the FileSystem. Its type is xsd:NCName. This name is user defined and MUST be unique in a JSDL document. A small number of well known names of file-systems that are expected to be commonly supported are defined and may also be used here. See §6.4.4.6 for more details.

6.4.4.5 Pseudo Schema

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

6.4.4.6 Well-known FileSystem names

It is expected that consuming systems will typically support a small number of well-known File-Systems. 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 FileSystem, and so on.

The “HOME” FileSystem:

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

The “ROOT” 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>
```

The “SCRATCH” 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>
```

The “TMP” 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>
```

6.4.4.7 Examples

The FileSystem “HOME” must have at most 10 gigabytes 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 1 gigabyte 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 1 gigabyte 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, for example, high performance.

```
<jsdl:FileSystem name=”FASTSTORAGE”>
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
   <jsdl:FileSystemType>jsdl:normal</jsdl:FileSystemType>
   <tns:Performance>tns:high</tns:Performance>
</jsdl:FileSystem>
```

6.4.5 MountPoint Element

6.4.5.1 Definition

This is a string that describes a *local* location that MUST be made available in the allocated resources for the job.

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

6.4.5.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.5.3 Type

The type of this element is xsd:string.

6.4.5.4 Attributes

No attributes are defined.

6.4.5.5 Pseudo Schema

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

6.4.5.6 Examples

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>
```

For a POSIX application, 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:Application> ...
   <jsdl-posix:POSIXApplication>
      ...
      <jsdl-posix:Environment name=”HOME” filesystemName=”HOME”/>
   </jsdl-posix:POSIXApplication>
</jsdl:Application>

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

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
```

6.4.6 MountSource Element

6.4.6.1 Definition

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

6.4.6.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.6.3 Type

The type of this element is xsd:string.

6.4.6.4 Attributes

No attributes are defined.

6.4.6.5 Pseudo Schema

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

6.4.6.6 Examples

The FileSystem “HOME” must have at most 10 gigabytes 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>
```

6.4.7 DiskSpace Element

6.4.7.1 Definition

This is a range value that describes the required amount of disk space on the containing File-System 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.

6.4.7.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.7.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.7.4 Attributes

No attributes are defined.

6.4.7.5 Pseudo Schema

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

6.4.7.6 Examples

A FileSystem with at least 1 gigabyte of free space:

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

A FileSystem with at most 1 gigabyte of free space:

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

6.4.8 FileSystemType Element

6.4.8.1 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.

6.4.8.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.8.3 Type

The type of this element is jsdl:FileSystemTypeEnumeration.

6.4.8.4 Attributes

No attributes are defined.

6.4.8.5 Pseudo Schema

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

6.4.8.6 Examples

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

6.4.9 ExclusiveExecution Element

6.4.9.1 Definition

This is a boolean that designates whether the job must have exclusive access to the resources allocated to it by the consuming system. If this is not present then it is not defined and the consuming system MAY choose any value.

6.4.9.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.9.3 Type

The type of this element is xsd:boolean.

- True—run exclusively on the resources
- False—others can run at the same time

6.4.9.4 Attributes

No attributes are defined.

6.4.9.5 Pseudo Schema

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

6.4.9.6 Examples

Exclusive execution:

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

6.4.10 OperatingSystem Element

6.4.10.1 Definition

This is a complex type that defines the operating system required by the job. It may contain Description, OperatingSystemVersion, and OperatingSystemType elements. If this is not present then it is not defined and the consuming system MAY choose any value.

6.4.10.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.10.3 Type

This is a complex type. It MUST support the following elements:

- OperatingSystemType
- OperatingSystemVersion
- Description

6.4.10.4 Attributes

No attributes are defined.

6.4.10.5 Pseudo Schema

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

6.4.11 OperatingSystemType Element

6.4.11.1 Definition

This is a complex type 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 §5.2.3) may be used by specifying the special token "other" and including the value as an extension (see §7.3). See examples below.

6.4.11.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.11.3 Type

This is a complex type. It MUST support the following elements:

- OperatingSystemName

6.4.11.4 Attributes

No attributes are defined.

6.4.11.5 Pseudo Schema

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

6.4.11.6 Examples

The Inferno OS:

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

The Windows 2003 Server OS (using the "jsdl:other" extension mechanism):

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

6.4.12 OperatingSystemName Element

6.4.12.1 Definition

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

6.4.12.2 Multiplicity

The multiplicity of this element is one.

6.4.12.3 Type

The type of this element is jsdl:OperatingSystemTypeEnumeration.

6.4.12.4 Attributes

No attributes are defined.

6.4.12.5 Pseudo Schema

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

6.4.12.6 Examples

The Inferno OS:

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

6.4.13 OperatingSystemVersion Element

6.4.13.1 Definition

This element is a string that defines the version of the operating system required by the job. 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.

6.4.13.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.13.3 Type

The type of this element is xsd:string.

6.4.13.4 Attributes

No attributes are defined.

6.4.13.5 Pseudo Schema

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

6.4.13.6 Examples

The operating system with version number “5.01”:

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

6.4.14 CPUArchitecture Element

6.4.14.1 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 §5.2.1) MAY be used by specifying the special token “other” and including the value as an extension (see §7.3). See also examples below.

6.4.14.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.14.3 Type

This is a complex type. It MUST support the following elements:

- CPUArchitectureName

6.4.14.4 Attributes

No attributes are defined.

6.4.14.5 Pseudo Schema

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

6.4.14.6 Examples

A SPARC architecture machine:

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

A CELL architecture machine (using the “jsdl:other” extension mechanism):

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

6.4.15 CPUArchitectureName Element

6.4.15.1 Definition

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

6.4.15.2 Multiplicity

The multiplicity of this element is one.

6.4.15.3 Type

The type of this element is jsdl:ProcessorArchitectureEnumeration.

6.4.15.4 Attributes

No attributes are defined.

6.4.15.5 Pseudo Schema

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

6.4.15.6 Examples

A SPARC architecture machine:

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

6.4.16 IndividualCPUSpeed Element

6.4.16.1 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.

6.4.16.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.16.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.16.4 Attributes

No attributes are defined.

6.4.16.5 Pseudo Schema

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

6.4.16.6 Examples

A CPU that has speed of at least 1 gigahertz:

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

6.4.17 IndividualCPUTime Element

6.4.17.1 Definition

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

6.4.17.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.17.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.17.4 Attributes

No attributes are defined.

6.4.17.5 Pseudo Schema

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

6.4.17.6 Examples

At most 60 CPU seconds:

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

6.4.18 IndividualCPUCount Element

6.4.18.1 Definition

This element is a range value specifying the 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.

6.4.18.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.18.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.18.4 Attributes

No attributes are defined.

6.4.18.5 Pseudo Schema

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

6.4.18.6 Examples

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

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

6.4.19 IndividualNetworkBandwidth Element

6.4.19.1 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.

6.4.19.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.19.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.19.4 Attributes

No attributes are defined.

6.4.19.5 Pseudo Schema

```
<IndividualNetworkBandwidth>
   jsdl:RangeValue_Type
</IndividualNetworkBandwidth>?
```

6.4.19.6 Examples

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

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

6.4.20 IndividualPhysicalMemory Element

6.4.20.1 Definition

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

6.4.20.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.20.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.20.4 Attributes

No attributes are defined.

6.4.20.5 Pseudo Schema

```
<IndividualPhysicalMemory>
```

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

6.4.20.6 Examples

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

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

6.4.21 IndividualVirtualMemory Element

6.4.21.1 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 given in bytes. If this is not present then it is not defined and the consuming system MAY choose any value.

6.4.21.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.21.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.21.4 Attributes

No attributes are defined.

6.4.21.5 Pseudo Schema

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

6.4.21.6 Examples

A resource with at least 1 gigabyte of virtual memory:

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

6.4.22 IndividualDiskSpace Element

6.4.22.1 Definition

This is a range value that describes the required amount of disk 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.

6.4.22.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.22.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.22.4 Attributes

No attributes are defined.

6.4.22.5 Pseudo Schema

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

6.4.22.6 Examples

Each resource should have at least 1 gigabyte of disk 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.

6.4.23 TotalCPUTime Element

6.4.23.1 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.

6.4.23.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.23.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.23.4 Attributes

No attributes are defined.

6.4.23.5 Pseudo Schema

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

6.4.23.6 Examples

At most 600 CPU seconds across all CPUs:

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

6.4.24 TotalCPUCount Element

6.4.24.1 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.

6.4.24.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.24.3 Type

The type of this element is jsdl:RangeValueType.

6.4.24.4 Attributes

No attributes are defined.

6.4.24.5 Pseudo Schema

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

6.4.24.6 Examples

A total of two CPUs are required.

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

6.4.25 TotalPhysicalMemory Element

6.4.25.1 Definition

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

6.4.25.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.25.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.25.4 Attributes

No attributes are defined.

6.4.25.5 Pseudo Schema

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

6.4.25.6 Examples

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

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

6.4.26 TotalVirtualMemory Element

6.4.26.1 Definition

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

6.4.26.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.26.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.26.4 Attributes

No attributes are defined.

6.4.26.5 Pseudo Schema

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

6.4.26.6 Examples

A resource with at least 1 gigabyte of virtual memory:

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

6.4.27 TotalDiskSpace Element

6.4.27.1 Definition

This is a range value that describes the required total amount of disk 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.

6.4.27.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.27.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.27.4 Attributes

No attributes are defined.

6.4.27.5 Pseudo Schema

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

6.4.27.6 Examples

The job should be given a total of at least 10 gigabytes of 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.

6.4.28 TotalResourceCount Element

6.4.28.1 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.

6.4.28.2 Multiplicity

The multiplicity of this element is zero or one.

6.4.28.3 Type

The type of this element is jsdl:RangeValue_Type.

6.4.28.4 Attributes

No attributes are defined.

6.4.28.5 Pseudo Schema

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

6.4.28.6 Examples

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>
```

6.4.29 Additional Resources

It is possible to extend JSDL (see §7) to describe additional resources. For example, licenses, named resources, software libraries, software packages, and special hardware may be described using extensions.

6.5 Data Staging Elements

6.5.1 DataStaging Element

6.5.1.1 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 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 DeleteOnTermination element may be used to delete a file after the job terminates. If the file is to be staged out the deletion is done after the stage out completes.

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 or pre-warming of grid-enabled file-systems 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, for example, deployment and configuration of the execution environment itself, are out of scope of the JSDL specification.

6.5.1.2 Multiplicity

The multiplicity of this element is zero or more.

6.5.1.3 Type

This is a complex type. It MUST support the following elements:

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

6.5.1.4 Attributes

The following attributes are defined:

- name—an optional name for the DataStaging element. Its type is xsd:NCName. This name is user defined and MUST be unique in a JSDL document.

6.5.1.5 Pseudo Schema

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

6.5.1.6 Examples

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>
```

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 File-System) 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>
```

6.5.2 FileName Element

6.5.2.1 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 §6.5.3)[8] and the only delimiter allowed MUST be '/'. In other words the FileName MUST NOT start with an initial '/'. The FileName MAY be a hierarchical directory path of the form <directory>/<directory>/.../<name>. The "<name>" MAY be either a directory or a file.

6.5.2.2 Multiplicity

The multiplicity of this element is one.

[8] Note that the FileName can be relative to a FileSystem that is forced to correspond to the root of the underlying system's file-system tree, so this is not a serious restraint on expressibility.

6.5.2.3 Type

The type of this element is xsd:string.

6.5.2.4 Attributes

No attributes are defined.

6.5.2.5 Pseudo Schema

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

6.5.2.6 Examples

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>
```

6.5.3 FileSystemName Element

6.5.3.1 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 the FileSystemName element is not present then it is not defined. If the FileSystemName is not defined then the FileName is relative to the working job directory as specified by the WorkingDirectory element. If the WorkingDirectory is also not specified then the base location is determined by the consuming system.

6.5.3.2 Multiplicity

The multiplicity of this element is zero or one.

6.5.3.3 Type

The type of this element is xsd:NCName.

6.5.3.4 Attributes

No attributes are defined.

6.5.3.5 Pseudo Schema

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

6.5.3.6 Examples

Staging a file to a specific FileSystem:

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

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

6.5.4 CreationFlag Element

6.5.4.1 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."

6.5.4.2 Multiplicity

The multiplicity of this element is one.

6.5.4.3 Type

The type of this element is jsdl:CreationFlagEnumeration.

6.5.4.4 Attributes

No attributes are defined.

6.5.4.5 Pseudo Schema

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

6.5.4.6 Examples

Overwrite an existing file:

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

6.5.5 DeleteOnTermination Element

6.5.5.1 Definition

This is a boolean that determines whether the file should be deleted after the job terminates. If true the file is deleted after the job terminates or after the file has been staged out. Otherwise the file remains, subject to the persistency of the FileSystem it is on. If not present, behavior is un-specified and depends on the consuming system.

6.5.5.2 Multiplicity

The multiplicity of this element is zero or one.

6.5.5.3 Type

The type of this element is xsd:boolean.

- True—remove the file after the job terminates
- False—do not remove the file after the job terminates

6.5.5.4 Attributes

No attributes are defined.

6.5.5.5 Pseudo Schema

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

6.5.5.6 Examples

Delete file after the job terminates:

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

6.5.6 Source Element

6.5.6.1 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 (optional) URI before the job has started. If this element is not present then the file does not have to be staged in.

6.5.6.2 Multiplicity

The multiplicity of this element is zero or one.

6.5.6.3 Type

This is a complex type. It MUST support the following elements:

- URI element—a URI according to [RFC 2396].

6.5.6.4 Attributes

No attributes are defined.

6.5.6.5 Pseudo Schema

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

6.5.6.6 Examples

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>
```

6.5.7 URI Element

6.5.7.1 Definition

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

6.5.7.2 Multiplicity

The multiplicity of this element is zero or one.

6.5.7.3 Type

The type of this element is xsd:anyURI.

6.5.7.4 Attributes

No attributes are defined.

6.5.7.5 Pseudo Schema

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

6.5.8 Target Element

6.5.8.1 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 (optional) URI after the job has terminated. If this element is not present then the file or directory does not have to be staged out.

6.5.8.2 Multiplicity

The multiplicity of this element is zero or one.

6.5.8.3 Type

This is a complex type. It MUST support the following elements:

- URI element—a URI according to [RFC 2396].

6.5.8.4 Attributes

No attributes are defined.

6.5.8.5 Pseudo Schema

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

6.5.8.6 Examples

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>
```

7 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 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.

Also if elements or attributes defined by extensions are present in a JSDL document they MUST be satisfied for the entire document to be satisfied.

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

7.1 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/2005/06/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>
```

7.2 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:TotalCPUCount>
   <jsdl:TotalDiskSpace>
      <!-- At least 1 gigabyte disk space -->
      <jsdl:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>
   </jsdl:TotalDiskSpace>
   <res:Reservation xmlns:res="http://www.example.org/reservation">
      <res:Ticket>h933fsolenri900wnmd90mm34</res:Ticket>
   </res:Reservation>
</jsdl:Resources>
...
```

7.3 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:CPUArchitecture_Type" and "jsdl:OperatingSystemType_Type", respectively.

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

For example, suppose 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 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>
...
```

8 Normative Extensions

The following extension is normatively defined by JSDL 1.0.

8.1 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.

8.1.1 POSIXApplication Element

8.1.1.1 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 JSDL Application element it MUST appear only once.

8.1.1.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.1.3 Type

This is a complex type. It 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

8.1.1.4 Attributes

The following attributes are defined:

- name—an optional name for the POSIXApplication element. Its type is xsd:NCName so that it can be reused and referred to from outside the containing JSDL document.

8.1.1.5 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>?
```

8.1.2 Executable Element

8.1.2.1 Definition

This element is a string specifying the command to execute. If an ApplicationName element is not specified and the Executable element is also not specified then the JSDL document defines a null job or a data staging. If both ApplicationName and Executable elements are specified, Executable MUST specify the executable used to execute the application named, according to the contents of the ApplicationVersion element (if also present). See also §6.3.1 and §6.3.2.

8.1.2.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.2.3 Type

The type of this element is xsd:string.

8.1.2.4 Attributes

The following attributes are defined:

- 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.

8.1.2.5 Pseudo Schema

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

8.1.2.6 Examples

Explicitly named executable:

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

Handle in a consumer-defined way[9]:

[9] For example, by looking the executable up on the PATH.

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

Executable in a location relative to the “HOME” FileSystem, usually “~/scripts/myExample”:

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

8.1.3 Argument Element

8.1.3.1 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.

8.1.3.2 Multiplicity

The multiplicity of this element is zero or more.

8.1.3.3 Type

The type of this element is xsd:normalizedString.

8.1.3.4 Attributes

The following attributes are defined:

- 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.

8.1.3.5 Pseudo Schema

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

8.1.3.6 Examples

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', and

```
<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).

8.1.4 Input Element

8.1.4.1 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 FileSystem. If this element is not present then it is not defined and the consuming system MAY choose any value.

8.1.4.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.4.3 Type

The type of this element is xsd:string.

8.1.4.4 Attributes

The following attributes are defined:

- 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.

8.1.4.5 Pseudo Schema

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

8.1.4.6 Examples

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

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

8.1.5 Output Element

8.1.5.1 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 FileSystem. If this element is not present then it is not defined and the consuming system MAY choose any value.

8.1.5.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.5.3 Type

The type of this element is xsd:string.

8.1.5.4 Attributes

The following attributes are defined:

- 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.

8.1.5.5 Pseudo Schema

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

8.1.5.6 Examples

The standard output should be put in “~/output.txt”:

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

8.1.6 Error Element

8.1.6.1 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 FileSystem. If this element is not present then it is not defined and the consuming system MAY choose any value.

8.1.6.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.6.3 Type

The type of this element is xsd:string.

8.1.6.4 Attributes

The following attributes are defined:

- 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.

8.1.6.5 Pseudo Schema

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

8.1.6.6 Examples

The standard error should be put in “~/error.txt”:

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

8.1.7 WorkingDirectory Element

8.1.7.1 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.

8.1.7.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.7.3 Type

The type of this element is xsd:string.

8.1.7.4 Attributes

The following attributes are defined:

- 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.

8.1.7.5 Pseudo Schema

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

8.1.7.6 Examples

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

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

8.1.8 Environment Element

8.1.8.1 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 examples section below. See also §6.4.22.

8.1.8.2 Multiplicity

The multiplicity of this element is zero or more.

8.1.8.3 Type

The type of this element is xsd:string. This is the value of the environment variable.

8.1.8.4 Attributes

The following attributes are defined:

- 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 inter-

preted as a filename or directory name relative to the mount point of the named FileSystem.

8.1.8.5 Pseudo Schema

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

8.1.8.6 Examples

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”/>
...
```

8.1.9 WallTimeLimit Element

8.1.9.1 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.

8.1.9.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.9.3 Type

The type of this element is xsd:positiveInteger.

8.1.9.4 Attributes

No attributes are defined.

8.1.9.5 Pseudo Schema

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

8.1.9.6 Example

Set the duration to one minute:

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

8.1.10 FileSizeLimit Element

8.1.10.1 Definition

This element is a positive integer that describes the maximum size 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.

8.1.10.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.10.3 Type

The type of this element is xsd:positiveInteger.

8.1.10.4 Attributes

No attributes are defined.

8.1.10.5 Pseudo Schema

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

8.1.10.6 Examples

Set the maximum file size to 1 gigabyte:

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

8.1.11 CoreDumpLimit Element

8.1.11.1 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.

8.1.11.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.11.3 Type

The type of this element is xsd:positiveInteger.

8.1.11.4 Attributes

No attributes are defined.

8.1.11.5 Pseudo Schema

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

8.1.11.6 Examples

Disable core dumps by setting core dump size to 0:

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

8.1.12 DataSegmentLimit Element

8.1.12.1 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.

8.1.12.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.12.3 Type

The type of this element is xsd:positiveInteger.

8.1.12.4 Attributes

No attributes are defined.

8.1.12.5 Pseudo Schema

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

8.1.12.6 Examples

Limit the data segment to 32 kilobytes:

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

8.1.13 LockedMemoryLimit Element

8.1.13.1 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.

8.1.13.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.13.3 Type

The type of this element is xsd:positiveInteger.

8.1.13.4 Attributes

No attributes are defined.

8.1.13.5 Pseudo Schema

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

8.1.13.6 Examples

Allow up to 8 megabytes of locked memory:

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

8.1.14 MemoryLimit Element

8.1.14.1 Definition

This element is a positive integer that describes the maximum amount of physical memory that the job should use when executing. The amount is given in bytes. If this is not present then the consuming system MAY choose its default value[10].

8.1.14.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.14.3 Type

The type of this element is xsd:positiveInteger.

8.1.14.4 Attributes

No attributes are defined.

---

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

8.1.14.5 Pseudo Schema

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

8.1.14.6 Examples

Set the physical memory limit to 64 megabytes:

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

8.1.15 OpenDescriptorsLimit Element

8.1.15.1 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.

8.1.15.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.15.3 Type

The type of this element is xsd:positiveInteger.

8.1.15.4 Attributes

No attributes are defined.

8.1.15.5 Pseudo Schema

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

8.1.15.6 Examples

The maximum number of descriptors should be 16:

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

8.1.16 PipeSizeLimit Element

8.1.16.1 Definition

This element is a positive integer that describes the maximum size of pipelines created or and by the processing of the job. The amount is given in bytes. If this is not present then the consuming system MAY choose its default value[11].

8.1.16.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.16.3 Type

The type of this element is xsd:positiveInteger.

8.1.16.4 Attributes

No attributes are defined.

8.1.16.5 Pseudo Schema

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

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

8.1.16.6 Examples

Set a limit of 512 bytes:

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

8.1.17 StackSizeLimit Element

8.1.17.1 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.

8.1.17.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.17.3 Type

The type of this element is xsd:positiveInteger.

8.1.17.4 Attributes

No attributes are defined.

8.1.17.5 Pseudo Schema

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

8.1.17.6 Examples

Set a stack size of 1 megabyte:

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

8.1.18 CPUTimeLimit Element

8.1.18.1 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.

8.1.18.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.18.3 Type

The type of this element is xsd:positiveInteger.

8.1.18.4 Attributes

No attributes are defined.

8.1.18.5 Pseudo Schema

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

8.1.18.6 Examples

Allow 30 seconds of CPU time:

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

8.1.19 ProcessCountLimit Element

8.1.19.1 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.

8.1.19.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.19.3 Type

The type of this element is xsd:positiveInteger.

8.1.19.4 Attributes

No attributes are defined.

8.1.19.5 Pseudo Schema

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

8.1.19.6 Examples

At most 8 processes should be allowed:

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

8.1.20 VirtualMemoryLimit Element

8.1.20.1 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.

8.1.20.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.20.3 Type

The type of this element is xsd:positiveInteger.

8.1.20.4 Attributes

No attributes are defined.

8.1.20.5 Pseudo Schema

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

8.1.20.6 Examples

The maximum virtual memory limit should be 128 megabytes:

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

8.1.21 ThreadCountLimit Element

8.1.21.1 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.

8.1.21.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.21.3 Type

The type of this element is xsd:positiveInteger.

8.1.21.4 Attributes

No attributes are defined.

8.1.21.5 Pseudo Schema

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

8.1.21.6 Examples

Not more than 8 threads:

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

8.1.22 UserName Element

8.1.22.1 Definition

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

8.1.22.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.22.3 Type

The type of this element is xsd:string.

8.1.22.4 Attributes

No attributes are defined.

8.1.22.5 Pseudo Schema

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

8.1.22.6 Examples

A UNIX user name:

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

8.1.23 GroupName Element

8.1.23.1 Definition

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

8.1.23.2 Multiplicity

The multiplicity of this element is zero or one.

8.1.23.3 Type

The type of this element is xsd:string.

8.1.23.4 Attributes

No attributes are defined.

8.1.23.5 Pseudo Schema

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

8.1.23.6 Examples

A UNIX group name:

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

9 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 1.0 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, U.K.
Email: ali@epcc.ed.ac.uk

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

Michel Drescher
Fujitsu Laboratories of Europe
Hayes Park Central, Hayes End Road, Hayes, Middlesex UB4 8FE, U.K.
Email: Michel.Drescher@uk.fujitsu.com

Donal Fellows
University of Manchester
Oxford Road, Manchester M13 9PL, U.K.
Email: donal.k.fellows@manchester.ac.uk

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, U.K.
Email: asm@doc.ic.ac.uk

Darren Pulsipher
Ovoca LLC
Email: darren@pulsipher.org

Andreas Savva
Grid Computing and Bioinformatics Laboratory
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Email: andreas.savva@jp.fujitsu.com

Contributors

We gratefully acknowledge the contributions made to this specification by Karl Czajkowski, William Lee, Chris Smith, Kazushige Saga, David Snelling, Igor Sedukhin.

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) John Ainsworth, Sayaka Akioka, Alain Andrieux, Sven van de Berghe, Lee Cook, Asit Dan, Yuri Demchenko, Nathalie Furmento, Andreas Haas, Tomasz Haupt, Bill Harris, Hiro Kishimoto, Bill Nitzberg, Ariel Oleksiak, Jim Pruyne, Hrabri Rajic, Jennifer Schopf, Frank Siebenlist, Dan Templeton, Andrea Westerinen.

We would like to thank Cadence Design Systems and Xango for their generous technical support for our teleconferences.

Full Copyright Notice

Intellectual Property Statement

able; 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. Distributed Management Task Force, 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

[DRMAA] Brobst, R., Chan, W., Ferstl, F., Gardiner, J., Haas, A., Nitzberg, B., Tollefsrud, J. *Distributed Resource Management Application API Specification 1.0.* Global Grid Forum DRMAA-WG, GFD-R-P.022, June 2004. Available at http://www.ggf.org/documents/GFD.22.pdf

[WS-AG] 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

This section contains the full normative XML Schema definition for JSDL 1.0.

For convenience, an informative copy of this schema is also available online at https://forge.gridforum.org/projects/jsdl-wg/document/jsdl.xsd/en/14. If, however, there are any discrepancies between the definitions in this section and those portions described in other sections in the specification, or in the online schema then the definitions in this section MUST be considered normative.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl"
   xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/06/jsdl"
   targetNamespace="http://schemas.ggf.org/jsdl/2005/06/jsdl"
   elementFormDefault="qualified">
   <!--============================================================-->
   <!-- SIMPLE TYPES -->
   <xsd:simpleType name="ProcessorArchitectureEnumeration">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="sparc"/>
         <xsd:enumeration value="powerpc"/>
         <xsd:enumeration value="x86"/>
         <xsd:enumeration value="x86_32"/>
         <xsd:enumeration value="x86_64"/>
```

```
         <xsd:enumeration value="parisc"/>
         <xsd:enumeration value="mips"/>
         <xsd:enumeration value="ia64"/>
         <xsd:enumeration value="arm"/>
         <xsd:enumeration value="other"/>
      </xsd:restriction>
   </xsd:simpleType>
   <!--============================================================-->
   <xsd:simpleType name="OperatingSystemTypeEnumeration">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="Unknown"/>
         <xsd:enumeration value="MACOS"/>
         <xsd:enumeration value="ATTUNIX"/>
         <xsd:enumeration value="DGUX"/>
         <xsd:enumeration value="DECNT"/>
         <xsd:enumeration value="Tru64_UNIX"/>
         <xsd:enumeration value="OpenVMS"/>
         <xsd:enumeration value="HPUX"/>
         <xsd:enumeration value="AIX"/>
         <xsd:enumeration value="MVS"/>
         <xsd:enumeration value="OS400"/>
         <xsd:enumeration value="OS_2"/>
         <xsd:enumeration value="JavaVM"/>
         <xsd:enumeration value="MSDOS"/>
         <xsd:enumeration value="WIN3x"/>
         <xsd:enumeration value="WIN95"/>
         <xsd:enumeration value="WIN98"/>
         <xsd:enumeration value="WINNT"/>
         <xsd:enumeration value="WINCE"/>
         <xsd:enumeration value="NCR3000"/>
         <xsd:enumeration value="NetWare"/>
         <xsd:enumeration value="OSF"/>
         <xsd:enumeration value="DC_OS"/>
         <xsd:enumeration value="Reliant_UNIX"/>
         <xsd:enumeration value="SCO_UnixWare"/>
         <xsd:enumeration value="SCO_OpenServer"/>
         <xsd:enumeration value="Sequent"/>
         <xsd:enumeration value="IRIX"/>
         <xsd:enumeration value="Solaris"/>
         <xsd:enumeration value="SunOS"/>
         <xsd:enumeration value="U6000"/>
         <xsd:enumeration value="ASERIES"/>
         <xsd:enumeration value="TandemNSK"/>
         <xsd:enumeration value="TandemNT"/>
         <xsd:enumeration value="BS2000"/>
         <xsd:enumeration value="LINUX"/>
         <xsd:enumeration value="Lynx"/>
         <xsd:enumeration value="XENIX"/>
         <xsd:enumeration value="VM"/>
         <xsd:enumeration value="Interactive_UNIX"/>
         <xsd:enumeration value="BSDUNIX"/>
         <xsd:enumeration value="FreeBSD"/>
         <xsd:enumeration value="NetBSD"/>
         <xsd:enumeration value="GNU_Hurd"/>
         <xsd:enumeration value="OS9"/>
         <xsd:enumeration value="MACH_Kernel"/>
         <xsd:enumeration value="Inferno"/>
```

```
         <xsd:enumeration value="QNX"/>
         <xsd:enumeration value="EPOC"/>
         <xsd:enumeration value="IxWorks"/>
         <xsd:enumeration value="VxWorks"/>
         <xsd:enumeration value="MiNT"/>
         <xsd:enumeration value="BeOS"/>
         <xsd:enumeration value="HP_MPE"/>
         <xsd:enumeration value="NextStep"/>
         <xsd:enumeration value="PalmPilot"/>
         <xsd:enumeration value="Rhapsody"/>
         <xsd:enumeration value="Windows_2000"/>
         <xsd:enumeration value="Dedicated"/>
         <xsd:enumeration value="OS_390"/>
         <xsd:enumeration value="VSE"/>
         <xsd:enumeration value="TPF"/>
         <xsd:enumeration value="Windows_R_Me"/>
         <xsd:enumeration value="Caldera_Open_UNIX"/>
         <xsd:enumeration value="OpenBSD"/>
         <xsd:enumeration value="Not_Applicable"/>
         <xsd:enumeration value="Windows_XP"/>
         <xsd:enumeration value="z_OS"/>
         <xsd:enumeration value="other"/>
      </xsd:restriction>
   </xsd:simpleType>
   <!--============================================================-->
   <xsd:simpleType name="FileSystemTypeEnumeration">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="swap"/>
         <xsd:enumeration value="temporary"/>
         <xsd:enumeration value="spool"/>
         <xsd:enumeration value="normal"/>
      </xsd:restriction>
   </xsd:simpleType>
   <!--============================================================-->
   <xsd:simpleType name="CreationFlagEnumeration">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="overwrite"/>
         <xsd:enumeration value="append"/>
         <xsd:enumeration value="dontOverwrite"/>
      </xsd:restriction>
   </xsd:simpleType>
   <!--============================================================-->
   <xsd:simpleType name="Description_Type">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <!--============================================================-->
   <!-- COMPLEX TYPES: Definitions for the RangeValueType -->
   <xsd:complexType name="Boundary_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:double">
            <xsd:attribute name="exclusiveBound" type="xsd:boolean"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
```

```
   <xsd:complexType name="Exact_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:double">
            <xsd:attribute name="epsilon" type="xsd:double"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <xsd:complexType name="Range_Type">
      <xsd:sequence>
         <xsd:element name="LowerBound" type="Boundary_Type"/>
         <xsd:element name="UpperBound" type="Boundary_Type"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <xsd:complexType name="RangeValue_Type">
      <xsd:sequence>
         <xsd:element name="UpperBoundedRange" type="Boundary_Type"
            minOccurs="0"/>
         <xsd:element name="LowerBoundedRange" type="Boundary_Type"
            minOccurs="0"/>
         <xsd:element name="Exact" type="Exact_Type" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:element name="Range" type="Range_Type" minOccurs="0"
            maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="JobDefinition_Type">
      <xsd:sequence>
         <xsd:element ref="jsdl:JobDescription"/>
         <xsd:any namespace="##other" minOccurs="0"
            maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="JobDescription_Type">
      <xsd:sequence>
         <xsd:element ref="jsdl:JobIdentification" minOccurs="0"/>
         <xsd:element ref="jsdl:Application" minOccurs="0"/>
         <xsd:element ref="jsdl:Resources" minOccurs="0"/>
         <xsd:element ref="jsdl:DataStaging" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:any namespace="##other" minOccurs="0"
            maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="JobIdentification_Type">
      <xsd:sequence>
         <xsd:element ref="jsdl:JobName" minOccurs="0"/>
```

```
      <xsd:element ref="jsdl:Description" minOccurs="0"/>
      <xsd:element ref="jsdl:JobAnnotation" minOccurs="0"
         maxOccurs="unbounded"/>
      <xsd:element ref="jsdl:JobProject" minOccurs="0"
         maxOccurs="unbounded"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<xsd:complexType name="Application_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:ApplicationName" minOccurs="0"/>
      <xsd:element ref="jsdl:ApplicationVersion" minOccurs="0"/>
      <xsd:element ref="jsdl:Description" minOccurs="0"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<!-- COMPLEX TYPES : Resource related types -->
<xsd:complexType name="Resources_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:CandidateHosts" minOccurs="0"/>
      <xsd:element ref="jsdl:FileSystem" minOccurs="0"
         maxOccurs="unbounded"/>
      <xsd:element ref="jsdl:ExclusiveExecution" minOccurs="0"/>
      <xsd:element ref="jsdl:OperatingSystem" minOccurs="0"/>
      <xsd:element ref="jsdl:CPUArchitecture" minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualCPUSpeed" minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualCPUTime" minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualCPUCount" minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualNetworkBandwidth"
         minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualPhysicalMemory"
         minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualVirtualMemory" minOccurs="0"/>
      <xsd:element ref="jsdl:IndividualDiskSpace" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalCPUTime" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalCPUCount" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalPhysicalMemory" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalVirtualMemory" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalDiskSpace" minOccurs="0"/>
      <xsd:element ref="jsdl:TotalResourceCount" minOccurs="0"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<xsd:complexType name="CandidateHosts_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:HostName" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<!--============================================================-->
<xsd:complexType name="CPUArchitecture_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:CPUArchitectureName"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<xsd:complexType name="FileSystem_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:FileSystemType" minOccurs="0"/>
      <xsd:element ref="jsdl:Description" minOccurs="0"/>
      <xsd:element ref="jsdl:MountPoint" minOccurs="0"/>
      <xsd:element ref="jsdl:DiskSpace" minOccurs="0"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:NCName"/>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<xsd:complexType name="OperatingSystem_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:OperatingSystemType" minOccurs="0"/>
      <xsd:element ref="jsdl:OperatingSystemVersion" minOccurs="0"/>
      <xsd:element ref="jsdl:Description" minOccurs="0"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<xsd:complexType name="OperatingSystemType_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:OperatingSystemName"/>
      <xsd:any namespace="##other" minOccurs="0"
         maxOccurs="unbounded"/>
   </xsd:sequence>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<!-- Complex types: Data staging -->
<xsd:complexType name="DataStaging_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl:FileName"/>
      <xsd:element ref="jsdl:FilesystemName" minOccurs="0"/>
      <xsd:element ref="jsdl:CreationFlag"/>
      <xsd:element ref="jsdl:DeleteOnTermination" minOccurs="0"/>
      <xsd:element ref="jsdl:Source" minOccurs="0"/>
      <xsd:element ref="jsdl:Target" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
   <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
<!--============================================================-->
<xsd:complexType name="SourceTarget_Type">
```

```
      <xsd:sequence>
         <xsd:element ref="URI" minOccurs="0"/>
         <xsd:any namespace="##other" minOccurs="0"
            maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:element name="JobDefinition" type="jsdl:JobDefinition_Type"/>
   <xsd:element name="JobDescription" type="jsdl:JobDescription_Type"/>
   <xsd:element name="JobIdentification"
      type="jsdl:JobIdentification_Type"/>
   <xsd:element name="JobName" type="xsd:string"/>
   <xsd:element name="Description" type="jsdl:Description_Type"/>
   <xsd:element name="JobAnnotation" type="xsd:string"/>
   <xsd:element name="JobProject" type="xsd:string"/>
   <xsd:element name="Application" type="jsdl:Application_Type"/>
   <xsd:element name="ApplicationName" type="xsd:string"/>
   <xsd:element name="ApplicationVersion" type="xsd:string"/>
   <xsd:element name="Resources" type="jsdl:Resources_Type"/>
   <xsd:element name="CandidateHosts" type="CandidateHosts_Type"/>
   <xsd:element name="HostName" type="xsd:string"/>
   <xsd:element name="OperatingSystem"
      type="jsdl:OperatingSystem_Type"/>
   <xsd:element name="OperatingSystemType"
      type="jsdl:OperatingSystemType_Type"/>
   <xsd:element name="OperatingSystemVersion" type="xsd:string"/>
   <xsd:element name="OperatingSystemName"
      type="jsdl:OperatingSystemTypeEnumeration"/>
   <xsd:element name="ExclusiveExecution" type="xsd:boolean"/>
   <xsd:element name="CPUArchitecture"
      type="jsdl:CPUArchitecture_Type"/>
   <xsd:element name="CPUArchitectureName"
      type="jsdl:ProcessorArchitectureEnumeration"/>
   <xsd:element name="FileSystem" type="jsdl:FileSystem_Type"/>
   <xsd:element name="FileSystemType"
      type="jsdl:FileSystemTypeEnumeration"/>
   <xsd:element name="MountPoint" type="xsd:string"/>
   <xsd:element name="DiskSpace" type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualCPUTime" type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualCPUSpeed" type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualCPUCount" type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualPhysicalMemory"
      type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualVirtualMemory"
      type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualNetworkBandwidth"
      type="jsdl:RangeValue_Type"/>
   <xsd:element name="IndividualDiskSpace" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalCPUTime" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalCPUCount" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalPhysicalMemory" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalVirtualMemory" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalDiskSpace" type="jsdl:RangeValue_Type"/>
   <xsd:element name="TotalResourceCount" type="jsdl:RangeValue_Type"/>
   <xsd:element name="DataStaging" type="jsdl:DataStaging_Type"/>
   <xsd:element name="FileName" type="xsd:string"/>
```

```
   <xsd:element name="FilesystemName" type="xsd:NCName"/>
   <xsd:element name="CreationFlag"
      type="jsdl:CreationFlagEnumeration"/>
   <xsd:element name="DeleteOnTermination" type="xsd:boolean"/>
   <xsd:element name="Source" type="jsdl:SourceTarget_Type"/>
   <xsd:element name="Target" type="jsdl:SourceTarget_Type"/>
   <xsd:element name="URI" type="xsd:anyURI"/>
</xsd:schema>
```

Appendix 2 JSDL POSIXApplication Normative Schema

This section contains the full normative XML Schema definition for JSDL POSIXApplication 1.0.

For convenience, an informative copy of this schema is also available online at https://forge.gridforum.org/projects/jsdl-wg/document/jsdl-posix.xsd/en/4. If, however, there are any discrepancies between the definitions in this section and those portions described in other sections in the specification, or in the online schema then the definitions in this section MUST be considered normative.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
   xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix"
   xmlns:jsdl-posix=http://schemas.ggf.org/jsdl/2005/06/jsdl-posix
   targetNamespace="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix"
   elementFormDefault="qualified">
   <!--============================================================-->
   <xsd:complexType name="Environment_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="name" type="xsd:NCName"/>
            <xsd:attribute name="filesystemName" type="xsd:NCName"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="Argument_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:normalizedString">
            <xsd:attribute name="filesystemName" type="xsd:NCName"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="FileName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="filesystemName" type="xsd:NCName"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
```

```
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="DirectoryName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="filesystemName" type="xsd:NCName"
               use="optional"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="Limits_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:positiveInteger">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="UserName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="GroupName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="POSIXApplication_Type">
      <xsd:sequence>
         <xsd:element ref="jsdl-posix:Executable" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:Argument" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:element ref="jsdl-posix:Input" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:Output" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:Error" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:WorkingDirectory"
            minOccurs="0"/>
         <xsd:element ref="jsdl-posix:Environment" minOccurs="0"
            maxOccurs="unbounded"/>
         <xsd:element ref="jsdl-posix:WallTimeLimit" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:FileSizeLimit" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:CoreDumpLimit" minOccurs="0"/>
         <xsd:element ref="jsdl-posix:DataSegmentLimit"
            minOccurs="0"/>
```

```
        <xsd:element ref="jsdl-posix:LockedMemoryLimit"
           minOccurs="0"/>
        <xsd:element ref="jsdl-posix:MemoryLimit" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:OpenDescriptorsLimit"
           minOccurs="0"/>
        <xsd:element ref="jsdl-posix:PipeSizeLimit" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:StackSizeLimit" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:CPUTimeLimit" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:ProcessCountLimit"
           minOccurs="0"/>
        <xsd:element ref="jsdl-posix:VirtualMemoryLimit"
           minOccurs="0"/>
        <xsd:element ref="jsdl-posix:ThreadCountLimit" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:UserName" minOccurs="0"/>
        <xsd:element ref="jsdl-posix:GroupName" minOccurs="0"/>
     </xsd:sequence>
     <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
     <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
  <!--============================================================-->
  <xsd:element name="POSIXApplication" type="POSIXApplication_Type"/>
  <xsd:element name="Executable" type="jsdl-posix:FileName_Type"/>
  <xsd:element name="Argument" type="jsdl-posix:Argument_Type"/>
  <xsd:element name="Input" type="jsdl-posix:FileName_Type"/>
  <xsd:element name="Output" type="jsdl-posix:FileName_Type"/>
  <xsd:element name="Error" type="jsdl-posix:FileName_Type"/>
  <xsd:element name="WorkingDirectory"
     type="jsdl-posix:DirectoryName_Type"/>
  <xsd:element name="Environment" type="jsdl-posix:Environment_Type"/>
  <xsd:element name="WallTimeLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="FileSizeLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="CoreDumpLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="DataSegmentLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="LockedMemoryLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="MemoryLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="OpenDescriptorsLimit"
     type="jsdl-posix:Limits_Type"/>
  <xsd:element name="PipeSizeLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="StackSizeLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="CPUTimeLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="ProcessCountLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="VirtualMemoryLimit"
     type="jsdl-posix:Limits_Type"/>
  <xsd:element name="ThreadCountLimit" type="jsdl-posix:Limits_Type"/>
  <xsd:element name="UserName" type="jsdl-posix:UserName_Type"/>
  <xsd:element name="GroupName" type="jsdl-posix:GroupName_Type"/>
</xsd:schema>
```

Appendix 3 Extended Informative JSDL Examples

The following example defines an invocation of the application “gnuplot.” The example uses all JSDL 1.0 main elements: JobIdentification, Application, Resources and DataStaging. It also uses the POSIXApplication extension. It features data staging for both input and output files.

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
      xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/06/jsdl"
```

```
    xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/06/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 invocation:
          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 staged-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:IndividualPhysicalMemory>
          <jsdl:LowerBoundedRange>2097152.0</jsdl:LowerBoundedRange>
       </jsdl:IndividualPhysicalMemory>
       <jsdl:TotalCPUCount>
          <jsdl:Exact>1.0</jsdl:Exact>
       </jsdl:TotalCPUCount>
    </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>
```