



JSDL: Job Submission Description Language and its Extensions

Balazs Konya (Lund University)



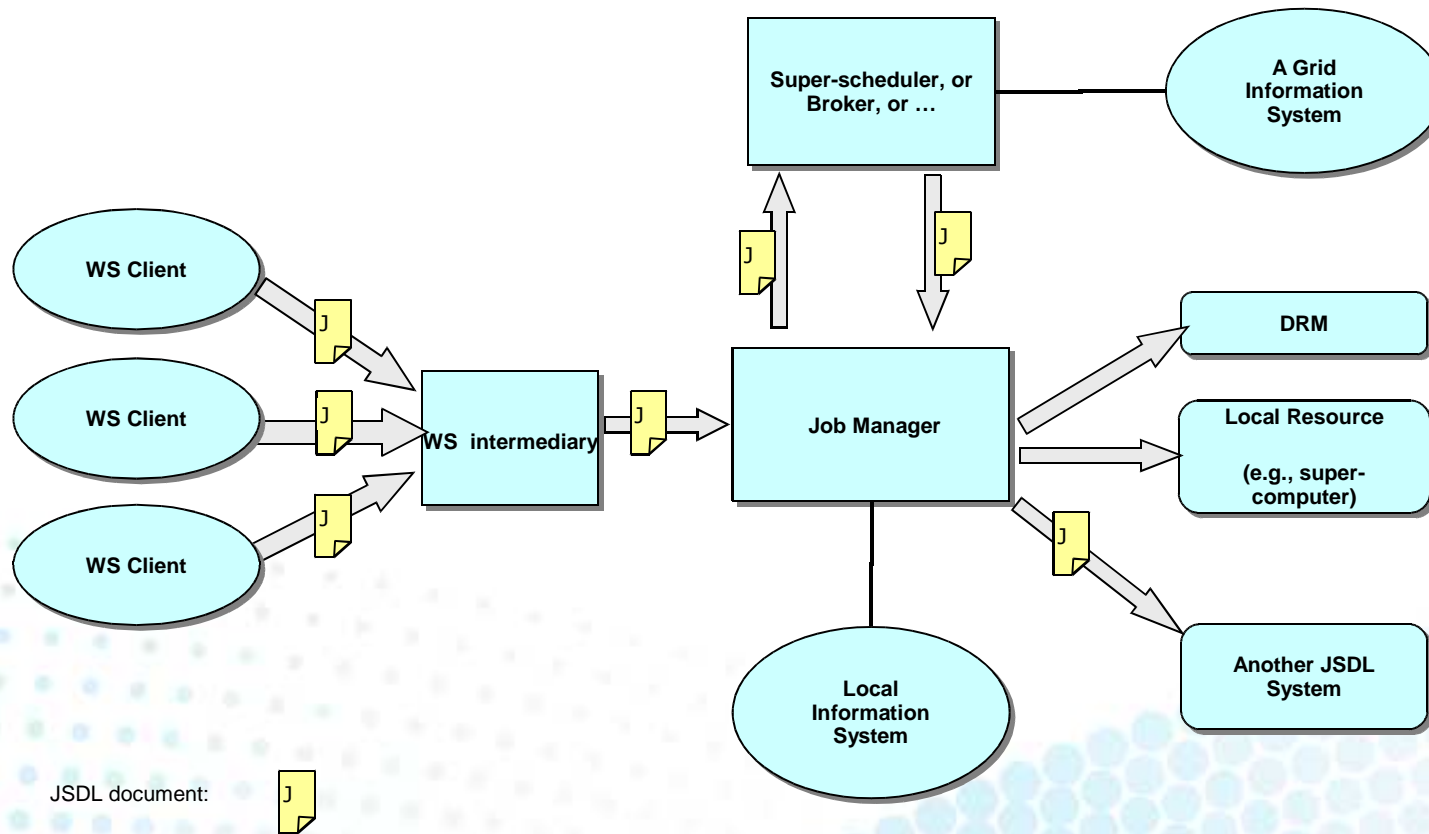
JSDL

- Job Submission Description Language
- Most recent specification is Version 1.0
 - Version 1.1 being worked out
 - Version 1.0 available at
<http://www.gridforum.org/documents/GFD.56.pdf>
- JSDL Working Group
<https://forge.gridforum.org/projects/jsdl-wg/>

JSDL: Motivations

- Many organizations accommodate a variety of job management systems
 - Each system has its own language for describing job submission requirements.
- Grid environments may involve the interaction of a number of different types of job management systems
 - The description of a job may be transformed by intermediaries

Job Submission Scenario



Source: *JSDL Specification*, version 1.0

JSDL Scope

- What JSDL *is*
 - JSDL is a language for describing the submission requirements of individual jobs.
- What JSDL *is not*
 - Does not address the lifecycle of a job
 - Does not address job management (BES does that)
 - Does not consider Workflows
 - Does not mandate how resource matching should be performed

JSDL Document Structure

JobDefinition

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

- Describes the job and its requirements
- This is the root of the JSDL document

JSDL Document Structure JobDescription

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

- Describes the job and its requirements
- This is the actual content of the JSDL document

JSDL Document Structure

JobIdentification

- Is a complex XML type and contains elements to identify the job

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

JSDL Document Structure Application

- OS-independent description of the application to be executed

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

JSDL Document Structure Resources

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

- Describes the resource requirements of the job
- Only a limited set of core requirements are supported
 - Additional requirements can be defined through extensions

JSDL Document Structure

DataStaging

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

- Describes the data requirements of the job
 - Which files are used as input (stage in)
 - Which files are produced as output (stage out)

Example

```
<JobDefinition id="HelloWorldJob">
  <JobDescription>
    <JobIdentification>
      <JobName>Hello World</JobName>
      <Description>This job prints the Hello World
message</Description>
      <JobProject>OMII-EU</JobProject>
      <JobAnnotation>uuid:090fe040e0</JobAnnotation>
    </JobIdentification>
    <Application>
      <ApplicationName>Echo</ApplicationName>
      <ApplicationVersion>1.0</ApplicationVersion>
      <Description>This application prints the Hello
World message to the standard output</Description>
    </Application>
  </JobDescription>
</JobDefinition>
```

JSDL Core Element Set: Resources Element

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

JSDL Core Element Set: Resource Element

```
<Resources>
  <CandidateHosts ... />?
  <FileSystem .../>*
  <ExclusiveExecution .../>?
  <OperatingSystem .../>?
  <CPUArchitecture .../>?
  <IndividualCPUSpeed .../>?
  <IndividualCPUTime .../>?
  <IndividualCPUCount .../>?
  <IndividualNetworkBandwidth .../>?
  <IndividualPhysicalMemory .../>?
  <IndividualVirtualMemory .../>?
  <IndividualDiskSpace .../>?
  <TotalCPUTime .../>?
  <TotalCPUCount .../>?
  <TotalPhysicalMemory .../>?
  <TotalVirtualMemory .../>?
  <TotalDiskSpace .../>?
  <TotalResourceCount .../>?
  <xsd:any##other>*
</Resources>?
```

- 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.

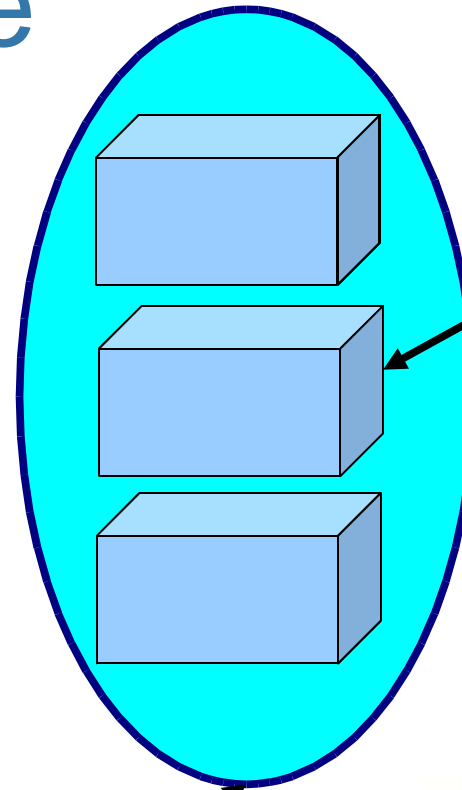
FileSystem element examples

```
<jsd1:FileSystem name="HOME">  
  <jsd1:Description>Ali's home</jsdl:Description>  
  <jsd1:MountPoint>/home/ali</jsdl:MountPoint>  
  <jsd1:DiskSpace>  
    <jsd1:LowerBoundedRange>1073741824.0</jsdl:LowerBoundedRange>  
  </jsdl:DiskSpace>  
  <jsd1:FileSystemType>normal</jsdl:FileSystemType>  
</jsdl:FileSystem>
```

```
<jsd1:FileSystem name="ROOT">  
  <jsd1: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>  
  <jsd1:FileSystemType>normal</jsdl:FileSystemType>  
</jsdl:FileSystem>
```

Example

```
<Resources>  
  <!-- ... -->  
  <IndividualCPUSpeed>  
    <Exact> 1000000000 </Exact>  
  </IndividualCPUSpeed>  
  <IndividualCPUCount>  
    <LowerBound> 4.0 </LowerBound>  
  </IndividualCPUCount>  
  <TotalCPUCount>  
    <UpperBound> 16.0  
  </UpperBound>  
  </TotalCPUCount>  
</Resources>?
```



Each resource has *at least* 4 CPUs at 1Ghz each

All the resources must provide *at most* 16 CPUs

Beware of unsatisfiable requirements!

- Here we request:
 - *At most 8 resources...*
 - ...each one providing *exactly 1 CPU...*
 - ...with a *total CPU count of exactly 16*

```
<Resources>  
  <!-- ... -->  
  <IndividualCPUCount>  
    <Exact> 1.0 </Exact>  
  </IndividualCPUCount>  
  <TotalCPUCount>  
    <Exact> 16.0 </Exact>  
  </TotalCPUCount>  
  <TotalResourceCount>  
    <UpperBound> 8.0 </UpperBound>  
  </TotalResourceCount>  
</Resources>?
```

JSDL Core Element Set: DataStaging Element

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

Data Staging Elements

- Define the files that should be moved to the execution host (stage in) and the files that should be moved from the execution host (stage out).
- Permission and access control for the staged files should be handled by the implementation and is out of scope of the JSDL specification.

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>overwrite</jsdl:CreationFlag>  
  <jsdl>DeleteOnTermination>true</jsdl>DeleteOnTermination>  
</jsdl:DataStaging>
```

Examples

- Staging in a file relative to a filesystem

```
<jsd1:FileSystem> ... </jsdl:FileSystem>
...
<jsd1:DataStaging>
  <jsd1:FileName>control.txt</jsdl:FileName>
  <jsd1:FileSystemName>HOME</jsdl:FileSystemName>
  <jsd1:Source>

  <jsd1:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
  </jsdl:Source>
  <jsd1:CreationFlag>overwrite</jsdl:CreationFlag>
  <jsd1>DeleteOnTermination>true</jsdl>DeleteOnTermination>
</jsdl:DataStaging>
```

Examples

- 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>overwrite</jSDL:CreationFlag>  
  
<jSDL>DeleteOnTermination>true</jSDL>DeleteOnTermination>  
</jSDL>DataStaging>
```

Examples

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

```
<jsd1:DataStaging>  
  <jsd1:FileName>result.txt</jsdl:FileName>  
  <jsd1:Target>  
  
    <jsd1:URI>http://node1/~me/result.txt</jsdl:URI>  
    </jsdl:Target>  
    ...  
  </jsdl:DataStaging>  
  ...  
  <jsd1:DataStaging>  
    <jsd1:FileName>result.txt</jsdl:FileName>  
    <jsd1:Target>  
  
      <jsd1:URI>http://node2/~me/result.txt</jsdl:URI>  
      </jsdl:Target>  
      ...  
    </jsdl:DataStaging>
```

27/05/08



JSDL attribute extensions

- Example: define an attribute called “order” that defines the order of staging in files

```
<?xml version="1.0" encoding="UTF-8"?>
<jsd1:JobDefinition xmlns="http://www.example.org/"
  xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
  xmlns:o="http://www.example.org/order-of-execution">
  <jsd1:JobDescription>
    <jsd1:DataStaging o:order="1">
      <jsd1:FileName>foo</jsdl:FileName>
      <jsd1:CreationFlag>overwrite</jsdl:CreationFlag>
      <jsd1:Source>
        <jsd1:URI>http://www.nowhere.com/foo-
file</jsdl:URI>
      </jsdl:Source>
    </jsdl:DataStaging>
    <jsd1:DataStaging o:order="2">
      <jsd1:FileName>bar</jsdl:FileName>
      <jsd1:CreationFlag>overwrite</jsdl:CreationFlag>
      <jsd1:Source>
        <jsd1:URI>http://www.nowhere.com/bar-
file</jsdl:URI>
      </jsdl:Source>
    </jsdl:DataStaging>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

27/05/08



JSDL element extensions

- Similarly for elements

```
...  
<jsd1:Resources>  
  <jsd1:TotalCPUCount>  
    <jsd1:Exact>1.0</jsdl:Exact>  
  </jsdl:TotalCPUCount>  
  <jsd1:TotalDiskSpace>  
    <!-- At least 1 gigabyte disk space -->  
  
<jsd1: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>  
...
```

Normative extension: POSIX Application

- Defines a schema for applications executed on a POSIX compliant system.
 - 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.

JSDL Example / 1

```
<?xml version="1.0" encoding="UTF-8"?>
<jSDL:JobDefinition xmlns="http://www.example.org/"
  xmlns:jSDL="http://schemas.ggf.org/jSDL/2005/11/jSDL"
  xmlns:jSDL-posix="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <jSDL:JobDescription>
    <jSDL:JobIdentification>
      <jSDL:JobName>My Gnuplot invocation</jSDL:JobName>
      <jSDL:Description> ... </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>
    <!-- to be continued... -->
  </jSDL:JobDescription>
</jSDL:JobDefinition>
```



JSDL Example / 2

```
<!-- continues from previous slide -->
<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>
```

JSDL Extensions

- HPC Basic Profile (HPC-BP) v1.0 (GFD.114)
 - <http://www.ogf.org/documents/GFD.114.pdf>
- JSDL HPC Profile Application Extension, Version 1.0 (GDF.111)
 - <http://www.ogf.org/documents/GFD.111.pdf>
- JSDL SPMD Application Extension, Version 1.0 (GFD.115)
 - <http://www.ogf.org/documents/GFD.115.pdf>
- HPC File Staging Profile (currently in Public Comment)
 - http://www.ogf.org/Public_Comment_Docs/Documents/2008-02/HPC%20File%20Staging%20Profile.pdf

HPC Basic Profile

- Describes how a particular set of specifications are composed in order to solve a basic use case of High Performance Computing (HPC) systems.
- The Profile covers the coexistence of JSDL v1.0, the JSDL HPC Profile Application Extension v1.0 and BES

HPC Profile Application Extension

- Defines an extension to JSDL 1.0 for describing HPC applications made up of an executable file running within an operating system process.
- It shares much in common with the JSDL POSIXApplication
- Removes some of the features that present barriers to interoperability.

JSDL SPMD Application Extension

- Describes an SPMD (single program multiple data) parallel application and its requirements.
- Again, this is based on the (normative) POSIXApplication extension:

HPC File Staging Profile

- This profile addresses the limitations within BES/JSDL related to file staging
 - This profile is based on the HPC Profile
- It specifically addresses the credential provisioning problem for data access
- It also defines an extension of the BES state model

HPC File Staging Profile: Credential Provisioning

- Compliant implementations MUST at least recognize either Username Token or X509 Certificate Token elements

```
<DataStaging>
  <FileName>output.txt</FileName>
  <CreationFlag>overwrite</CreationFlag>
  <Target>
    <URI>ftp://server.inthe.sky:1234</URI>
  </Target>
  <Credential xmlns="http://schemas.ogf.org/hpcp/2007/01/fs">
    <UsernameToken
      xmlns="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <Username>demo</Username>
      <Password>pass</Password>
    </UsernameToken>
  </Credential>
  ...
</DataStaging>
```



HPC File Staging Profile: State Model

