JSDL Working Group Use Case Document

# Abstract

After its publication, the JSDL 1.0 specification has become one of the most utilized OGF specifications the organization has produced. Part of numerous interoperability fests (both directly related to JSDL and indirectly through other specifications like BES and HPC-BP), and the subject of multiple extensions, JSDL has become the defacto means of describing compute jobs that a user wants to execute. However, in the time since its first publication, a number of grid middleware provides have identified areas of weakness in the JSDL specification that, thus far, have been addressed using proprietary solutions. The purpose of this document is to provide a set of use cases that motivate the need for a set of new JSDL extensions necessary to address current shortcomings.

This document consists of three individual use cases, each comprising a common format and capable of standing alone (that is, separable from the other use cases). The use cases are merely combined together into this one document for convenience.

# Presentation of the Use Case

## Use Case Name

*Express Resource Requirements of Data Processing Jobs*

## Version

Use case taking into account lessons about JDL, JSDL and GLUE gathered by OGF GIN – V2.0

## Goal

The goal is *GLUE compliant RRL* : For Users wishing to submit data processing Jobs, JSDL (Job Submission Description Language) must contain an RRL (Resource Requirements Language) permitting to express Resource Requirements with syntax and semantics in compliance with GLUE 2.0. Since the Execution Service managing the jobs receives the description of all resources in compliance with GLUE 2.0, this is the only method permitting the Execution service to easily match the Resource Requirements with the resource descriptions.

## Summary

Inside a Job description document, a User describes a Job using JSDL, and in particular specifies Resource Require­ments expressed in compliance with GLUE 2.0. The User submits this Job description document to an Execution Service, which matches the Resource Requirements with the description of all resources published in compliance with GLUE 2.0, and sends the Job to the computing resource which 'best' matches the Resource Requirements.

# Detailed Description of the Use Case

## System

The System interacting with the User is the whole Job execution infrastructure (as small as a single core, or as large as a Service Grid). The interface is the Execution Service, providing Job execution services independently of the peculiarities of the underlying resources. So, the description of resource capabilities must be expressed in the same stan­dardized manner by the resources themselves (that is GLUE 2.0) and by the RRL section of the Job description document.

## Actors (or Stakeholders, or Participants, or Roles)

A User wishing to submit a data processing Job.

## Preconditions

Execution Service regularly receives updated descriptions of all resources expressed in compliance with GLUE 2.0.

## Triggers

A User wishes to submit a data processing Job.

## Basic course of events

The User describes the Job inside a Job description document using JSDL, with Resource Requirements expressed in compliance with GLUE 2.0.

The User submits this Job description document to an Execution Service of a Job execution infrastructure.

The Execution Service matches the Resource Requirements with the description of all resources published in compliance with GLUE 2.0, and selects the computing resource associated to the 'best' matching score.

The Execution Service sends the Job to the computing resource selected just above.

## Postconditions

Logs of the Job execution infrastructure should contain persistent records indicating which User submitted which Job when, and where this Job was executed.

# Additional Information for the Use Case

## Alternative paths or Extensions

If the Resource Requirements specified inside the Job description document do not fully comply with GLUE 2.0, then the Execution Service has to perform syntactic and semantic translation if possible, or else reject the Job.

## Business rules

Most Job execution infrastructures already agree to use the GLUE 2.0 standard to describe resources.

## Notes

<http://www.ogf.org/documents/GFD.147.pdf> is the GLUE 2.0 specification.

<http://www.ogf.org/documents/GFD.136.pdf> is the current JSDL 1.0 specification.

<http://forge.gridforum.org/sf/wiki/do/viewPage/projects.pgi-wg/wiki/ReqJD3> describes OGF PGI requirement 100 corresponding to this Use Case.

<http://forge.gridforum.org/sf/go/doc16166> shows a proposal of implementation.

<http://forge.gridforum.org/sf/go/doc16171> provides matching between JSDL 1.0 RRL and GLUE 2.0.

## Author and date

Etienne URBAH 2011-03-15

# Presentation of the Use Case

## Use Case Name

Adaptable Job Description

## Version

1.0

## Goal

The goal is to adapt user-side job description attributes depending on chosen execution target.

## Summary

In a heterogeneous Grid, computing resources can have strongly varying characteristics (operating system, network connectivity, available application environment etc). Job description often needs to be optimized in order to use resources efficiently: for example, requested execution time or amount of input data may depend on advertised capacity of a computing resource. This requires conditional statements (or a similar capability) in JSDL, as well as comparison and logical operators.

# Detailed Description of the Use Case

## System

The use case concerns JSDL which is to be processed by a metascheduler (broker). Nature of interfaces is of no particular importance, as long as the metascheduler can harvest reliable information about execution services.

## Actors

A user submitting a non uniquely defined job description (making use of conditional statements) to a heterogeneous Grid.

## Preconditions

Metascheduler must have up-to-date and sufficiently detailed information about the resources it can schedule on; it is also expected to be able to modify original job description.

## Triggers

A user submits a job described in such a manner that it changes its attributes depending on the properties of the offered target.

## Basic course of events

1. A user describes a job using conditional statements, and submits it to a metascheduler
2. Metascheduler discovers a set of matching targets for either of described job configurations
3. Metascheduler selects an optimal target (definition of “optimal” is out of scope of this use case)
4. Metascheduler creates a uniquely defined job description according to the selected target and submits the job

## Post Conditions

None

# Additional Information for the Use Case

## Alternative paths or Extensions

In step (4) of Basic Course of Events, metascheduler may submit the original job description, and job execution service will interpret it, resolving conditional statements. This course of events, however, requires a highly non-standard execution service interface

## Business rules

None

## Notes

The situation addressed by this use case arises when a user needs to utilize available resources in a most efficient manner, which often requires describing jobs differently for different targets. For example, a cluster without outbound connectivity will need extra input databases to be installed locally by the job; on the other hand, a cluster with pre-installed specific application environment will need less input data; user may also require different number of nodes per job depending on cluster characteristics, or even depending on its name. While it is possible to create several different job descriptions matching different combinations, it is much more convenient to have a single job description which handles these combinations using conditional statements of the kind “IF ClusterName=goodcluster THEN nodes=128 ELSE nodes=1”.

Other comparison operators ( != > < ) as well as logical operators (AND, OR) are also desirable in this context, e.g. “IF (bandwith>10Gb AND diskspace>1TB) THEN (stage in 100 files) ELSE (stage in 2 files)”.

This functionality was available in the original Globus Resource Specification Language, modeled on LDAP query language.

## Author and date

Oxana Smirnova

04/04/11

# Presentation of the Use Case

## Use Case Name

Schedule Resources Flexibly

## Version

1.0

## Goal

To be able to describe jobs in such a way that they could be scheduled on any set of resources that could meet the job’s requirements.

## Summary

The Genesis IIteam at UVa has found a strong need to be able to schedule grid jobs based off of things like Architecture and Operating system. We have also found classes of jobs that do not need to be unnecessarily constrained to one particular set of resources. Relying solely on the tools provided in JSDL 1.0 can potentially eliminate sets of valid resources. To that end, we would like the ability to relay the information necessary to indicate that jobs can run on any of a set of valid resources.

# Detailed Description of the Use Case

## System

This use case relates mostly to jobs submitted to metaschedulers but could also be applicable to jobs submitted to a BES backed by a set of heterogeneous resources (such as a pbs backed BES).

## Actors

A user requesting that their job(s) run on any of a set of machine classes.

## Preconditions

Metascheduler (or BES) must have up-to-date information about the resources it can schedule on.

## Triggers

A user submits a job to that can run on any of a set of machine classes.

## Basic course of events

1) A user submits a job(s)

2) A set of resources that could potentially match the job(s) requirements is determined

3) The job(s) are scheduled and sent to appropriate resources

## Post Conditions

NA

# Additional Information for the Use Case

## Alternative paths or Extensions

NA

## Business rules

NA

## Notes

Some examples of uses this use case seeks to address are:

* Attempting to run a Java program on any machine type that supports Mac OS X or Linux
* Attempting to run an R program for which the architecture is 32-bit and either Windows or Mac OS X
* Attempting to run an arbitrary executable from a set of pre-compiled binaries that will be staged in where the run is restricted to 32-bit windows machines with 2 G of available ram, or 64-bit windows machines with 4 G of ram.

## Author and date

Mark Morgan

Michael Saravo

03/28/11