# 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