JSDL maps

draft v 0.3.2

by Claudio Cacciari[1]
UniGrids project - Uniform Interface to Grid Services

1 CINECA - High Performance Computing Department

Contents

1. JSDL – AJO map

1.1 Introduction

One way to write a submission request for Unicore/GS 1.0 is through the JSDL language.
It is developed by the GGF JSDL-WG and is a specification for an abstract standard about job submission that is independent of language bindings.
A JSDL request is an XML document complying with a normative XML schema corresponding to the JSDL specification.
However some Unicore components, in the Unicore/GS 1.0 architecture, need to comply with AJOs. These ones are java objects, conform to an internal standard, which describe all Unicore actions (execute tasks, file transfers, jobs dependencies and so on).
In order to preserve the compatibility with these components we have developed tools to convert JSDL documents into AJOs and viceversa.
In this paper are considered the JSDL specification version 1.0 and the AJO specification version 4.6 .
The next schema shows a possible mapping between the two specifications.

1.2 JSDL to AJO mapping

JSDL	AJO
JobDefinition	
JobDescription	-> AbstractAction.ActionGroup.AbstractJob
JobIdentification{0,1}	
JobName{0,1}	-> AbstractAction.ActionGroup.AbstractJob.name
Description{0,1}	
JobAnnotation{0,n}	-> AbstractAction.notes
JobProject{0,n}	-> Utility.UserAttributesConverter.project
Application{0,1}	
ApplicationName{0,1}	-> Resource.CapabilityResource.SoftwareResource.Application.name | AbstractAction.ActionGroup.AbstractJob.JSDLImpl application.name
ApplicationVersion{0,1}	-> Resource.CapabilityResource.SoftwareResource.Application.version | AbstractAction.ActionGroup.AbstractJob.JSDLImpl application.version AbstractAction.ActionGroup.AbstractJob.JSDLImpl
Description{0,1}	-> application.description | userApplication.description
POSIXApplication{0,1}	
Executable{0,1}	-> userApplication.executable
Argument{0,n}	-> application.arguments | userApplication.arguments
Input{0,1}	-> application.stdin | userApplication.stdin
Output{0,1}	-> application.stdout | userApplication.stdout
Error{0,1}	-> application.stderr | userApplication.stderr
WorkingDirectory{0,1}	AbstractAction.ActionGroup.AbstractJob.JSDLImpl
Environment{0,n}	-> environmentVariable Resource.CapabilityResource.Limit

```
        WallTimeLimit{0,1}          ->      limit & type.Wall_time
                                        | AbstractAction.ActionGroup.AbstractJob.JSDLImpl
                                            resources.wall_time_limit
                                        Resource.CapabilityResource.Limit
        FileSizeLimit{0,1}          ->      limit & type.File_size
        CoreDumpLimit{0,1}          ->      limit & type.Core_dump
        DataSegmentLimit{0,1}       ->      limit & type.Data_segment
        LockedMemoryLimit{0,1}      ->      limit & type.Locked_memory
        MemoryLimit{0,1}            ->      limit & type.Memory_rss
        OpenDescriptorsLimit{0,1}   ->      limit & type.Open_descriptors
        PipeSizeLimit{0,1}          ->      limit & type.Pipe_size
        StackSizeLimit{0,1}         ->      limit & type.Stack_size
        CPUTimeLimit{0,1}           ->      limit & type.CPU_time
                                        | AbstractAction.ActionGroup.AbstractJob.JSDLImpl
                                            resources.cpu_time_limit
        ProcessCountLimit{0,1}      ->      limit & type.Process_count
        VirtualMemoryLimit{0,1}     ->      limit & type.Virtual_memory
                                        | AbstractAction.ActionGroup.AbstractJob.JSDLImpl
                                            resources.virtual_memory_limit
        ThreadCountLimit{0,1}       ->      limit & type.Thread_count
        UserName{0,1}               -> Utility.UserAttributesConverter.xlogin
        GroupName{0,1}

Resources{0,1}
    CandidateHosts{0,1}
        HostName{1,n}               -> Vsite
    FileSystem{0,n}                 -> Resource.CapacityResource.Storage.PathedStorage
                                            AlternativeUspace
                                            | Home
                                            | Root
                                            | StorageServer
                                            | Temp
                                            | USpace
        Description{0,1}
        MountPoint{0,1}
        MountSource{0,1}
        DiskSpace{0,1}              ->              request
        FileSystemType{0,1}
    ExclusiveExecution{0,1}
    OperatingSystem{0,1}
        OperatingSystemType{0,1}
            OperatingSystemName{1,1}
        OperatingSystemVersion{0,1}
    CPUArchitecture{0,1}
        CPUArchitectureName{1,1}
    IndividualCPUSpeed{0,1}
    IndividualCPUTime{0,1}
                                        AbstractAction.ActionGroup.AbstractJob.JSDLImpl
    IndividualCPUCount{0,1}         ->      resources.cpu_count
    IndividualNetworkBandwidth{0,1} -> Resource.CapacityResource.PerformanceResource.Network
                                            performance
    IndividualPhysicalMemory{0,1}
    IndividualVirtualMemory{0,1}    -> AbstractAction.ActionGroup.AbstractJob.JSDLImpl
                                            resources.virtual_memory_limit
                                        | Resource.CapacityResource
                                            Memory.request
    IndividualDiskSpace{0,1}
    TotalCPUTime{0,1}
                                        AbstractAction.ActionGroup.AbstractJob.JSDLImpl
    TotalCPUCount{0,1}              ->      resources.cpu_count
    TotalPhysicalMemory{0,1}
    TotalVirtualMemory{0,1}         -> AbstractAction.ActionGroup.AbstractJob.JSDLImpl
                                            resources.virtual_memory_limit
                                        | Resource.CapacityResource.Memory.request
    TotalDiskSpace{0,1}
    TotalResourceCount{0,1}         -> Resource.CapacityResource.Node

DataStaging{0,n}
                                        AbstractAction.ActionGroup.AbstractJob.JSDLImpl
    FileName{1,1}                   ->      stageInFile.local_name
                                            | stageOutFile.local_name
    FileSystemName{0,1}             ->      stageInFile.local_file_system_name
                                            | stageOutFile.local_file_system_name
    CreationFlag{1,1}               ->      stageInFile.overwrite
```

		| stageOutFile.overwrite
DeleteOnTermination{0,1}		
Source{0,1}		
URI{1,1}	->	stageInFile.vsite
		| stageInFile.file_system_name
		| stageInFile.file_name
Target{0,1}		
URI{1,1}	->	stageOutFile.vsite
		| stageOutFile.file_system_name
		| stageOutFile.file_name

The JobDefinition's attribute id is ignored.
For each JobDescription element there is one and only one AbstractAction.ActionGroup.AbstractJob .
Each AbstractJob can't contain more than three ActionGroup : one for the import tasks, one for the application task and one for the export tasks.
The AbstractJob can't contain more than one application.

1.2.1 JobIdentification

The JobName element is mapped to the corresponding AbstractAction.ActionGroup.AbstractJob.name .
The Description element is ignored.
While the content of JobAnnotation and JobProject elements is concatenated in a unique (for each of the two elements) string and mapped respectively to AbstractAction.notes and Utility.UserAttributesConverter.project .
In general all the elements with a multiplicity bigger than one, related to java object with multiplicity one, are managed in this way: the content of each instance is concatenated in a unique string, separated with single space and mapped to the corresponding ajo element. So, for example :

```
<jsdl:JobAnnotation>first release</jsdl:JobAnnotation>
<jsdl:JobAnnotation>by Claudio Cacciari</jsdl:JobAnnotation>
<jsdl:JobAnnotation>12/12/2004</jsdl:JobAnnotation>
```

becomes :

```
“first release by Claudio Cacciari 12/12/2004”
```

1.2.2 Application

The ApplicationName element can be mapped to Resource.CapabilityResource.SoftwareResource.Application.name or to the class AbstractAction.ActionGroup.AbstractJob.JSDLImpl, method setApplication, parameter name. The same path is used for *ApplicationVersion, while Description* is associated only to the *JSDLImpl* class.
This last one is the implementation of an AJO created from a JSDL template.
The choice to map *ApplicationName* and *ApplicationVersion* as a software resource or as an application depends on the presence of the explicit path to the executable.
If the executable path is explicit the job description is interpreted as a user defined application and *ApplicationName* is translated to a software resource whose presence on the target machine is mandatory, while if it is missing, *ApplicationName* is considered as a symbolic name and the job will be associated to a specific application on the target

machine.
Also the elements under POSIXApplication are related to the executable if it is declared or to the abstract application otherwise.
The Argument, Input, Output, Error and Environment elements are mapped to the corresponding parameters of the *JSDLImpl* class, but Environment don't respect the rule above mentioned about the mapping of elements with multiplicity bigger than one. Each element Environment is a couple (name,value), thus the content of more instances can't be concatenated and each element is mapped to one environmentVariable of the *JSDLImpl* class.
The WorkingDirectory element is ignored because Unicore assigns a working directory to the user.
All these last elements, Executable included, can have the attribute filesystemName . If it is present, the element's content string must be interpreted as a filename relative to the mount point of the named file system.
But Unicore doesn't allow to specify the mount point from the client side, it is fixed on the server side, so the client can use only symbolic names as file systems references.
In order to solve the problem the content of filesystemName is interpreted as a Unicore symbolic name.
Besides Unicore executes the job only under the uspace (the Unicore working directory), therefore all files requested by the job before the execution have to be copied into the uspace, generating stage in tasks which are not explicit in the JSDL job description.
So, for example, writing :

```
<jsdl-posix:Executable filesystemName="HOME">
      test1.sh
</jsdl-posix:Executable>
```

is the same as :

```
<jsdl-posix:Executable>
      test1.sh
</jsdl-posix:Executable>

[...]

<jsdl:DataStaging>
      <jsdl:FileName>test1.sh</jsdl:FileName>
      <jsdl:FileSystemName>USPACE</jsdl:FileSystemName>
      <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
      <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
      <jsdl:Source>
            <jsdl:URI>file:///test1.sh#home</jsdl:URI>
      </jsdl:Source>
</jsdl:DataStaging>
```

This solution doesn't work for the Environment element because if we try to write :

```
<jsdl-posix:Environment name=”TMPDIR” filesystemName=”TMP”/>
```

we would export the variable TMPDIR=TMP because Unicore doesn't replace the content of an environment variable. Therefore, in this case, the attribute filesystemName will be

ignored.
All the Limit elements are translated to the corresponding Resource.CapabilityResource.Limit and three of them also to the respective limit parameter of the *JSDLImpl* class.
The only important element related to the user's identity is UserName which is associated to Utility.UserAttributesConverter.xlogin and whose value is the local user name of the job owner under the target machine where the job will be executed.
The GroupName element is ignored.

1.2.3 Resources

The element HostName under CandidateHosts is mapped to a Vsite object, that is a collection of resources on which a job runs. A typical configuration is a Unicore gateway for each organization (usite) and a vsite for each machine of an organization, so to determine univocally the job target machine is necessary to specify the URL of the gateway and the symbolic name of the vsite.
In order to express this information as a unique URI, the path field of the URI is interpreted as the vsite symbolic name. It will be :

ssl://gateway_hostname:gateway_port/vsite_name

If there is more than one HostName only the last one is considered.
The element FileSystem is associated to the Resource.CapacityResource.Storage.PathedStorage class, whose subtypes represent different type of file storage. They are Home, Root, Temp, USpace, AlternativeUspace and StorageServer, where AlternativeUspace is the working directory of another job (on the same Vsite and owned by the same User) and StorageServer represents various resource such as bulk storage and tape archives and backup facilities as well as conventional file systems not modelled by other classes.
The elements Description, MountPoint, MountSource and FileSystemType are ignored.
The DiskSpace one is mapped to the parameter request for each storage subtypes.
However this parameter sets the amount of disk space of a particular storage resource requested by the containing Unicore task (that is an import or export operation or the execution of an application).
If the Unicore task doesn't perform operations on the file system whose disk space has been requested the job fails.
This means, for example, that if a job contains two export tasks, one copies a file to the home space, requiring 10 MB and the other to the temp space, requiring 20 MB, the two disk space requests have to be associated to the respective tasks, not to the whole job. This is not a mapping problem, but a JSDL document parsing and AJO building issue.
Besides the DiskSpace's content is a range of values, while the request parameter is a java double. There is more than one possibility to extract a value from a range. For all the elements with a range type content the weakest constraint was chosen that is the minimum value of the range.
The elements ExclusiveExecution, OperatingSystem, CPUArchitecture, IndividualCPUTime, IndividualPhysicalMemory, IndividualDiskSpace, IndividualCPUSpeed, TotalPhysicalMemory and TotalDiskSpace are ignored.
The element IndividualCPUCount is related to the cpu_count parameter of the *JSDLImpl* class and interpreted as the the number of CPUs for each of the resources to be allocated to the

job submission. And the total number of resources, TotalResourceCount, is considered as the number of nodes of a cluster.
In Unicore CPUs and memory requests are always managed as per node requests.
Therefore also the TotalCPUCount element is mapped to the cpu_count parameter, but divided by the TotalResourceCount's content and rounded off to the superior integer.
The possible cases are :

IndividualCPUCount	TotalCPUCount	TotalResourceCount	->	CPUs	nodes
n	m	k		n	k
	m	k		ceil(m/k)	k
n		k		n	k
n	m			n	1
n				n	1
	m			m	1
		k		1	k

The same solution is adopted for the IndividualVirtualMemory and the TotalVirtualMemory elements.
Another aspect is that one related to the measurement units.
The content of the elements IndividualVirtualMemory, TotalVirtualMemory, IndividualNetworkBandwidth and DiskSpace is expressed in bytes, while the respective AJO resources are in megabytes so it is necessary a conversion.
It is the same for the Limit elements related to memory and disk space.

1.2.4 Data Staging

The DataStaging elements group is interpreted as a stage in operation if it is present the element Source and as a stage out operation if there is an element Target . For the same reason FileName, FileSystemName and URI are mapped to the addStageInFile or addStageOutFile method of the *JSDLImpl* class.
The CreationFlag element allows three different values : overwite, dontOverwrite and append, but only the first two are mapped, because Unicore doesn't support the append operation.
The element DeleteOnTermination is ignored.
All the stage in files are imported before the execution of the job, all the stage out ones are exported after. If a stage in or stage out operation fails, the job fails.
In order to express the information related to the file name and the file system name with a unique URI, the query part of a URI is considered as the file path and the fragment one as the file system name :

upl://gateway_hostname:gateway_port/vsite_name?path to file#file_system_name

In case of file transfer within the same machine (the same Unicore vsite), the path field of the URI is interpreted as file path :

file:///<path to file>#file_system_name

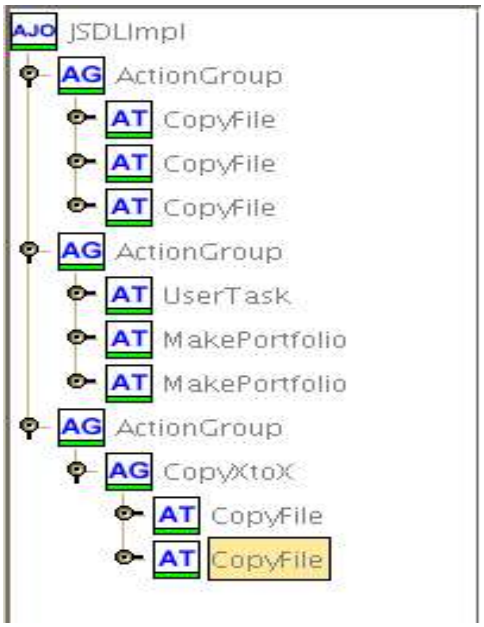
1.3 JSDL example

The following code is an example of JSDL job.

```
<?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/2005/05/jsdl-posix"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <jsdl:JobDescription>
            <jsdl:JobIdentification>
                  <jsdl:JobName>My Gnuplot invocation</jsdl:JobName>
                  <jsdl:Description> Simple application invokation:
                        User wants to run the application 'gnuplot' to
                        produce a plotted graphical file based on some data
                        shipped in from elsewhere(perhaps as part of a
                        workflow). A front-end application will then build
                        into an animation of spinning data.

                        Front-end application knows URI 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>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:CandidateHosts>
                        <jsdl:HostName>ssl://pc-cacciari.cineca.it:4433/PC-
CACCIARI</jsdl:HostName>
                  </jsdl:CandidateHosts>
                <jsdl:FileSystem name="ROOT">
                  <jsdl:Description>root dir</jsdl:Description>
            </jsdl:FileSystem>
                  <jsdl:IndividualVirtualMemory>
                        <jsdl:LowerBoundedRange>134217728.0</jsdl:LowerBoundedRange>
                  </jsdl:IndividualVirtualMemory>
                  <jsdl:TotalCPUCount>
                        <jsdl:Exact>1.0</jsdl:Exact>
                  </jsdl:TotalCPUCount>
            </jsdl:Resources>
            <jsdl:DataStaging>
```

```
                <jsdl:FileName>gnuplot</jsdl:FileName>
                <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
                <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                <jsdl:Source>
                      <jsdl:URI>file:///usr/bin/gnuplot#ROOT</jsdl:URI>
                </jsdl:Source>
          </jsdl:DataStaging>
          <jsdl:DataStaging>
                <jsdl:FileName>control.txt</jsdl:FileName>
                <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
                <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                <jsdl:Source>
                      <jsdl:URI>file:///control.txt#HOME</jsdl:URI>
                </jsdl:Source>
          </jsdl:DataStaging>
          <jsdl:DataStaging>
                <jsdl:FileName>input.dat</jsdl:FileName>
                <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
                <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                <jsdl:Source>
                      <jsdl:URI>file:///input.dat#HOME</jsdl:URI>
                </jsdl:Source>
          </jsdl:DataStaging>
          <jsdl:DataStaging>
                <jsdl:FileName>output1.png</jsdl:FileName>
                <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
                <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
                <jsdl:Target>
                      <jsdl:URI>file:///output1.png#HOME</jsdl:URI>
                </jsdl:Target>
          </jsdl:DataStaging>
      </jsdl:JobDescription>
</jsdl:JobDefinition>
```

and this is the corresponding AJO tasks tree :



2.Glossary

AJO : Abstract Job Object

GGF : Global Grid Forum

JSDL : Job Submission Description Language

JSDL-WG : Job Submission Description Language Working Group

URI : Uniform Resource Identifier

URL : Uniform Resource Locator

3.References

1. Ali Anjomshoaa, Fred Brisard, An Ly, Stephen McGough, Darren Pulsipher, Andreas Savva. Job Submission Description Language (JSDL) Specification Version 1.0.
2. http://unicore.sourceforge.net/