Table of Contents

CREAM JDL guide

1 Introduction

The Job Description Language (JDL) is a high-level, user-oriented language based on Condor classified advertisements (classads) for describing jobs to be submitted to the CREAM CE service. Being the JDL an extensible language the user is allowed to use whatever attribute for the description of a request without incurring in errors from the JDL parser. However, only a certain set of attributes, that we will refer as "supported attributes" from now on, is taken into account by the CREAM CE service.

Some of the attributes in the JDL are mandatory. If the user does not specify them, CREAM cannot handle the request. For the other attributes the system may find a default value if they are necessary for processing the request.

Before starting with the detailed attribute description please note that a request description is composed by entries that are strings having the format

```
attribute = expression;
```

and are terminated by the semicolon character. The whole description has to be included between square brackets, i.e. `[ ]`. The termination with the semicolon is not mandatory for the last attribute before the closing square bracket].

Attribute expressions can span several lines provided the semicolon is put only at the end of the whole expression. Comments must have a sharp character (#) or a double slash (//) at the beginning of each line. Comments spanning multiple lines can be specified enclosing the text between "/*" and "*/".

Please note that since CREAM exposes a publicly available WSDL interface, no assumption is made in the document (unless explicitly specified) about the client tool used to submit the JDL description of the job.

2 Request and Job Types

2.1 Type

This a string representing the type of the request described by the JDL, e.g.:

```
Type = "Job";
```

For the time being the only possible value is: `Job`

The value for this attribute is case insensitive. If this attribute is not specified in the JDL description, the default value ("Job") will be considered.

Default: "Job"

3 Job Attributes Description

This section reports the detailed description of the JDL attributes that can be specified for describing Job requests. A sub-section for each attribute is provided.

3.1 JobType

This a string representing the type of the job described by the JDL, e.g.:

```
JobType = "Normal";
```

At least for the time being the only possible value is: `Normal`.

This attribute only makes sense when the Type attribute equals to "Job". The value for this attribute is case insensitive. If not specified in the JDL, it will be set to "Normal"

Default: "Normal"

3.2 Executable

This is a string representing the executable/command name. The user can specify an executable that lies already on the remote CE and in this case the absolute path, possibly including environment variables referring to this file should be specified, e.g.:

```
Executable = "usr/local/java/j2sdk1.4.0_01/bin/java";
```

Or:

```
Executable = "$JAVA_HOME/bin/java";
```

The other possibility is to provide either an executable located on a remote gridFTP server accessible by the user (HTTPS servers are also supported but this requires to have the GridSite `htcp` client command installed on the WN). In both cases the executable file will be staged from the original location to the Computing Element WN. In both cases only the file name has to be specified as executable. The URI of the executable should be then listed in the InputSandbox attribute expression to make it be transferred. E.g. respectively:

```
Executable = "cms_sim.exe";
InputSandbox = {"file:///home/edguser/sim/cms_sim.exe", ...};
```

Or:

```
Executable = "cms_sim.exe";
InputSandbox = {"gsiftp://neo.datamat.it:5678/tmp/cms_sim.exe", ...};
```

Also descriptions as follows:

```
Executable = "cms_sim.exe";
InputSandbox = {"/home/edguser/sim/cms_sim.exe", ... };
```

are accepted and interpreted as in the first case, i.e. the executable file is available on the local file system.

It is important to remark that if the job needs for the execution some command line arguments, they have to be specified through the `Arguments` attribute. This attribute is mandatory.

Mandatory: Yes

Default: No

3.3 Arguments

This is a string containing all the job command line arguments. E.g. an executable sum that has to be started as:

```
$ sum  N1 N2 -out result.out
```

is described by:

```
Executable = "sum";
Arguments = "N1 N2 -out result.out";
```

If you want to specify a quoted string inside the Arguments then you have to escape quotes with the \ character. E.g. when describing a job like:

```
$ grep -i "my name" *.txt
```

you will have to specify:

```
Executable = "/bin/grep";
Arguments = "-i \"my name\" *.txt";
```

Analogously, if the job takes as argument a string containing a special character (e.g. the job is the tail command issued on a file whose name contains the ampersand character, say file1&file2), since on the shell line you would have to write:

```
$ tail -f file1\&file2
```

in the JDL you'll have to write:

```
Executable = "/usr/bin/tail";
Arguments = "-f file1\\\&file2";
```

i.e. a \ for each special character. In general, special characters such as:

```
&, |, >, <
```

are only allowed if specified inside a quoted string or preceded by triple \. The character ` cannot be specified in the JDL.

Some other guidelines for the management of special characters:

- If the arguments string contains one or more spaces, the string must be specified in single quotes
- If the arguments string contains single quotes, then the string must be specified in double quotes (which must be escaped)
- If the arguments string contains spaces and single quotes, then the string must be specified in double quotes (which must be escaped)
- the number of the backquote characters must be an even number

Some examples:

```
Arguments = "\"Problematic character is: '\" " ;
Arguments = "-o '/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=romanov/CN=427293/CN=Vladimir Romano
```

```
Arguments = "-o \"'option'\"";
```

Mandatory: No

Default: No

3.4 StdInput

This is a string representing the standard input of the job. . This means that the job is run as follows:

```
$ executable < <standard input file>
```

It can be an absolute path possibly including environment variables (wild cards are instead not allowed), i.e. it is already available on the CE, e.g.

```
StdInput = "/var/tpm/jobInput";
```

or just a file name, e.g.

```
StdInput = "myjobInput";
```

and this means that file needs to be made available on the WN where the job is run. Therefore the standard input file has to be added to the InputSandbox file list so that it will be downloaded on the WN. The same rules described for the Executable attribute apply to StdInput.

Mandatory: No

Default: No

3.5 StdOutput

This is a string representing the file name where the standard output of the job is saved. The user can specify either a file name or a relative path (with respect to the job working directory on the WN), e.g.:

```
StdOutput = "myjobOutput";
```

```
StdOutput = "event1/myjobOutput";
```

Wild cards are not allowed. The value specified for StdError can be the same as the one for StdOutput: this means that the two standard streams of the job are saved in the same file. The user can choose to have this file staged automatically on a GridFTP server specifying a URI for that file in the OutputSandbox attribute expression. E.g.:

```
StdOutput = "myjobOutput";
OutputSandbox = {
"gsiftp://fox.infn.it:5678/home/gftp/myjobOutput",
...
};
```

indicates that `myjobOutput` when the job has completed its execution has to be transferred on `gsiftp://fox.infn.it:5678` in the `/home/gftp` directory.

Mandatory: No

Default: No

3.6 StdError

This is a string representing the file name where the standard error of the job is saved. The user can specify either a file name or a relative path (with respect to the job working directory on the WN), e.g.:

```
StdError = "myjobError";
```

```
StdError = "event1/myjobError";
```

Wild cards are not allowed. The value specified for StdError can be the same as the one for StdOutput: this means that the two standard streams of the job are saved in the same file. The user can choose to have this file staged automatically on a GridFTP server specifying a URI for that file in the OutputSandboxDestURI attribute expression The same rules as for the StdOutput apply to StdError.

Mandatory: No

Default: No

3.7 InputSandbox

This is a string or a list of strings identifying the list of files available on the file system of the client (UI) machine and/or on an accessible gridFTP server (HTTPS servers are also supported but this requires to have the GridSite htcp client command installed on the WN) needed by the job for running. These files hence have to be transferred to the WN before the job is started. Wildcards and environment variables are admitted in the specification of this attribute only if the submission takes place through a client able to resolve them locally before passing the JDL to the CREAM service (e.g. this is the case for the CREAM CLI). Admitted wildcard patterns are the ones supported by the Linux glob function. One can remove the special meaning of the characters:

```
'? ', '*', ' and ['
```

by preceding them by a backslash.

File names can be provided as URI on a gridFTP/HTTPS server, simple file names, absolute paths and relative paths with respect to the current UI working directory. The InputSandbox file list cannot contain two or more files having the same name (even if in different paths) as when transferred in the job's working directory on the WN they would overwrite each other. This attribute can also be used to accomplish executable and standard input staging to the CE where job execution takes place as explained above. The InputSandbox attribute meaning is strictly coupled with the value of the InputSandboxBaseURI defined in the following that specifies a common location on a gridFTP/HTTPS server where files in the InputSandbox not specified as URI are located.

Support for file transfer from gridftp servers running using user credentials instead of host credentials is also provided1. In this case the distinguish name of such user credentials must be specified in the URI using:

```
 ?DN=<distinguish name>
```

as shown in the example below.

Here below follows an example of InputSandbox setting:

```
InputSandbox = {
"/tmp/ns.log",
"mytest.exe",
"myscript.sh",
"data/event1.txt",
"gsiftp://neo.datamat.it:5678/home/fpacini/cms_sim.exe ",
"file:///tmp/myconf",
"gsiftp://lxsgaravatto.pd.infn.it:47320/etc/fstab?DN=/C=IT/O=INFN/OU=Personal Certificate/L=Padov
};
InputSandboxBaseURI = "gsiftp://matrix.datamat.it:5432/tmp";
```

It means that:

- `/tmp/ns.log` is located on the UI machine local file system
- `mytest.exe` , `myscript.sh` and `data/event1.txt` are available on `gsiftp://matrix.datamat.it:5432` in the `/tmp` directory
- `/tmp/myconf` is located on the user local file system (explicitly specified using the file:// prefix)
- `/etc/fstab` is available on `gsiftp://lxsgaravatto.pd.infn.it:47320` which is a gridftp server running using user credentials (with the specified distinguish name)

If the InputSandboxBaseURI is not specified than also `mytest.exe`, `myscript.sh` and `data/event1.txt` would be interpreted as located on the user local file system

Mandatory: No

Default: No

3.8 InputSandboxBaseURI

This is a string representing the URI on a gridFTP server (HTTPS servers are also supported but this requires to have the GridSite htcp client command installed on the WN) where the InputSandbox files that have been specified as simple file names and absolute/relative paths are available for being transferred on the WN before the job is started. E.g.

```
InputSandbox = {
 ...
 "data/event1.txt",
 ...
 };
InputSandboxBaseURI = "gsiftp://matrix.datamat.it:5432/tmp";
```

makes CREAM consider

```
"gsiftp://matrix.datamat.it:5432/tmp/data/event1.txt"
```

for the transfer on the WN.

Support for file transfer from gridftp servers running using user credentials instead of host credentials is also provided1. In this case the distinguish name of such user credentials must be specified in the URI using:

```
 ?DN=<distinguish name>
```

as shown in the example below.

E.g.

```
InputSandbox = {
 ...
 "data/event2.txt",
 ...
 };
InputSandboxBaseURI  = "gsiftp://lxsgaravatto.pd.infn.it:47320/tmp?DN=/C=IT/O=INFN/OU=Personal Ce
```

makes CREAM consider

```
"gsiftp://lxsgaravatto.pd.infn.it:47320/tmp/data/event2.txt"
```

for the transfer on the WN, where that gridftp server has been started using user credentials (with the specified distinguish name)

Mandatory: No

Default: No

3.9 OutputSandbox

This is a string or a list of strings identifying the list of files generated by the job on the WN at runtime, which the user wants to save. This attribute can be combined with the OutputSandboxDestURI or the OutputSandboxBaseDestURI to have, upon job completion, the output directly copied to specified locations running a gridFTP server (HTTPS servers are also supported but this requires to have the GridSite htcp client command installed on the WN).

Wildcards are admitted in the specification of this attribute only if the OutputSandboxBaseDestURI attribute is used along with the OutputSandbox attribute. Admitted wildcard patterns are the ones supported by the Linux glob function. One can remove the special meaning of the characters:

```
 '? ', '*' and ' ['
```

by preceding them by a backslash.

File names can be provided as simple file names or relative paths with respect to the current working directory on the executing WN. The OutputSandbox file list should not contain two or more files having the same name (even if in different paths).

Mandatory: No

Default: No

3.10 OutputSandboxDestURI

This is a string or a list of strings representing the URI(s) on a gridFTP/HTTPS server where the files listed in the OutputSandbox attribute have to be transferred at job completion.

The OutputSandboxDestURI list contains for each of the files specified in the OutputSandbox list the URI (including the file name) where it has to be transferred at job completion. Support for file transfer to gridftp servers running using user credentials instead of host credentials is also provided1. In this case the distinguish name of such user credentials must be specified in the URI using:

```
 ?DN=<distinguish name>
```

as shown in the example below. E.g.

```
OutputSandbox = {
"myjobOutput",
"run1/event1",
"run2/event2",
};

OutputSandboxDestURI = {
"gsiftp://matrix.datamat.it:5432/tmp/myjobOutput ",
"gsiftp://grid003.ct.infn.it:6789/home/cms/event1",
"gsiftp://lxsgaravatto.pd.infn.it:47320/tmp/event2?DN=/C=IT/O=INFN/
OU=Personal Certificate/L=Padova/CN=Massimo Sgaravatto"
};
```

makes CREAM transfer respectively:

- `myjobOutput` on `matrix.datamat.it` in the directory `/tmp`
- `event1` on `grid003.ct.infn.it` in the directory `/home/cms`
- `event2` on `lxsgaravatto.pd.infn.it` (gridftp server running using user credentials, with the specified distinguish name) in the directory `/tmp`

Specifying the URI `gsiftp://localhost`, the OutputSandboxFile is saved in the gridftp server of the CREAM CE, as shown in the following example:

```
OutputSandbox = {
"file1",
"file2",
};
OutputSandboxDestURI = {
"gsiftp://localhost/file1",
"gsiftp://grid003.ct.infn.it:6789/home/cms/file2"
};
```

In the above example `file1` is saved on the gridftp server of the CREAM CE, while `file2` is saved on `grid003.ct.infn.it` (directory `/home/cms`).

The OutputSandboxDestURI list must have the same cardinality as the OutputSandbox list, otherwise the JDL will be considered as invalid. Note that the file name specified in the OutputSandbox can be different from the corresponding destination file name specified in the OutputSandboxBaseDestURI. The OutputSandboxDestURI attribute and the OutputSandboxBaseDestURI cannot be specified together in the same JDL. One (and only one) among the OutputSandboxDestURI and OutputSandboxBaseDestURI attributes must be specified if the OutputSandbox attribute has been specified

Mandatory: No

Default: No

3.11 OutputSandboxBaseDestURI

The OutputSandboxBaseDestURI attribute is a string representing the base URI on a gridFTP server, i.e. a directory on the server, where the files listed in the OutputSandbox attribute have to be transferred at job completion. HTTPS servers are also supported but this requires to have the GridSite htcp client command installed on the WN. All the OutputSandbox files are transferred to the location specified by the URI with the same names (only names in a flat directory) as the ones specified in the OutputSandbox. E.g.:

```
OutputSandbox = {
"myjobOutput",
"run1/event1",
};
OutputSandboxBaseDestURI = "gsiftp://matrix.datamat.it:5432/tmp";
```

makes CREAM transfer both files in the `/tmp` directory of the gridFTP server `matrix.datamat.it` (note that `event1` will go in `/tmp` and not in `/tmp/run1`).

Support for file transfer to gridftp servers running using user credentials instead of host credentials is also provided1. In this case the distinguish name of such user credentials must be specified in the URI using:

```
 ?DN=<distinguish name>
```

as shown in the example below. E.g.

```
OutputSandbox = {
"myjobOutput",
"run1/event1",
};
OutputSandboxBaseDestURI = "gsiftp://lxsgaravatto.pd.infn.it:47320/tmp?DN=/C=IT/O=INFN/OU=Persona
```

makes CREAM transfer both files in the `/tmp` directory of the gridFTP server `lxsgaravatto.pd.infn.it` (running using user credentials, with the specified distinguish name).

Specifying the URI `gsiftp://localhost`, the OutputSandboxFile is saved in the gridftp server of the CREAM CE, as shown in the following example:

```
OutputSandbox = {
"file1",
"file2",
};
OutputSandboxBaseDestURI = "gsiftp://localhost";
```

In the above example `file1` and `file2` are saved on the gridftp server of the CREAM CE.

The OutputSandboxBaseDestURI attribute and the OutputSandboxDestURI cannot be specified together in the same JDL. One (and only one) among the OutputSandboxDestURI and OutputSandboxBaseDestURI attributes must be specified if the OutputSandbox attribute has been specified.

Mandatory: No

Default: No

3.12 Prologue

The Prologue attribute is a string representing the executable/script name of the prologue. The prologue is an executable run within the CREAM job wrapper before the user job is started. It can be used for purposes ranging from application-specific checks that the job environment has been correctly set on the WN to actions like data transfers, database updates or MPI pre script. If the prologue fails the job wrapper terminates. The rules for specification of the Prologue attributes and its relationship with the InputSandbox attribute are exactly the same already described for the Executable attribute.

Mandatory: No

Default: No

3.13 PrologueArguments

The PrologueArguments attribute is a string containing all the prologue executable command line arguments. All the rules reported in the description of the Arguments attribute also apply to the PrologueArguments attribute.

Mandatory: No

Default: No

3.14 Epilogue

The Epilogue attribute is a string representing the executable/script name of the epilogue. The epilogue is an executable/script run within the CREAM job wrapper after the user job completion. It can be used for purposes ranging from application-specific checks that the job performed correctly to actions like data transfers, database updates or MPI post script. The rules for specification of the Epilogue attributes and its relationship with the InputSandbox attribute are exactly the same already described for the Executable attribute.

Mandatory: No

Default: No

3.15 EpilogueArguments

The EpilogueArguments attribute is a string containing all the epilogue executable command line arguments. All the rules reported in the description of the Arguments attribute also apply to the EpilogueArguments attribute.

Mandatory: No

Default: No

3.16 Environment

This is a list of string representing environment settings that have to be performed on the execution machine and are needed by the job to run properly. The JobWrapper on the Worker Node performs these settings just before the job is started. Each item of the list is an equality `"VAR_NAME=VAR_VALUE"`. E.g.:

```
Environment  = {"JOB_LOG_FILE=/tmp/myjob.log",
"ORACLE_SID=edg_rdbms_1",
"JAVABIN=/usr/local/java"};
```

Mandatory: No

Default: No

3.17 PerusalFileEnable

The PerusalFileEnable attribute is a Boolean attribute that allows enabling the job file perusal support in CREAM. File perusal can be used when jobs are submitted to CREAM by the WMS, but it is possible to use this functionality also for jobs submitted directly to CREAM. When this attribute is set to true, i.e.

```
PerusalFileEnable = true;
```

the user can inspect, while the job is running, the files generated by the job on the WN. This is achieved by uploading on a location specified by the attribute PerusalFilesDestURI, at regular time intervals, chunks of the files (specified by the attribute PerusalListFileURI) generated by the job on the WN. The PerusalFileEnable attribute is not mandatory. If not specified in the JDL it is assumed to be set to false.

Mandatory: No

Default: false

3.18 PerusalTimeInterval

The PerusalTimeInterval attribute is a positive integer representing the difference in seconds between two subsequent saving (and upload on the location specified by the attribute PerusalFilesDestURI) of the job files generated by the job on the WN. Specifying e.g.

```
PerusalTimeInterval = 10;
```

makes the CREAM JobWrapper save the job files specified through the attribute PerusalListFileURI each 10 seconds and upload them on location specified by the attribute PerusalFilesDestURI, so that they can be inspected by the user.

Mandatory: No (unless PerusalFileEnable is true)

Default: No

3.19 PerusalFilesDestURI

The PerusalFilesDestURI attribute is a string representing the URI of the location on a gridFTP or HTTPS server (HTTPS servers are also supported but this requires to have the GridSite htcp command installed on the WN) where the chunks of files generated by the running job on the WN and specified through the attribute PerusalListFileURI have to be copied. E.g.

```
 PerusalFilesDestURI = gsiftp://ghemon.cnaf.infn.it/home/glite/peek
```

Mandatory: No (unless PerusalFileEnable is true)

Default: No

3.20 PerusalListFileURI

The PerusalListFileURI attribute is a string representing the URI of the file on a gridFTP server, containing the list of files (one for each line), that must be saved and uploaded on the location specified by the attribute PerusalFilesDestURI at regular time intervals. E.g.

```
PerusalListFileURI = "gsiftp://ghemon.cnaf.infn.it/peek/files2peek";
```

Mandatory: No (unless PerusalFileEnable is true)

Default: No

3.21 BatchSystem

This is a string representing the Local Resource Management System (LRMS), that is the batch system type (e.g. LSF, PBS, etc.) of the target CREAM CE. Here below follows an example for this attribute:

```
BatchSystem = "cms";
```

Mandatory: Yes (unless the CREAM CLI, since it retrieves this attribute from the CEID and automatically fills this attribute)

Default: No

3.22 QueueName

This is a string representing the queue name in the Local Resource Management System (LRMS) where the job has to be submitted on the target CREAM CE. Here below follows an example for this attribute:

```
QueueName = "long";
```

Mandatory: Yes (unless the CREAM CLI is used, since it retrieves this attribute from the CEID and automatically fills this attribute)

Default: No

3.23 CPUNumber

The CpuNumber attribute is an integer greater than 1 specifying the number of CPUs needed. This attribute can be used in particular for MPI jobs.

Please note that this attributes allows allocating the specified number of CPUs. Then it is up to the job using them to run the job (e.g. via mpistart)

An example of the JDL setting is:

```
CpuNumber = 5;
```

Mandatory: No

*Default: 1

3.24 SMPGranularity

The SMPGranularity attribute is an integer greater than 0 specifying the number of cores any host involved in the allocation has to dedicate to the considered job. This attribute can't be specified along with the Hostnumber attribute when WholeNodes is false.

Mandatory: No

Default: No

3.25 WholeNodes

The WholeNodes attribute is a boolean that indicates whether whole nodes should be used exclusively or not.

Mandatory: No

Default: False

3.26 HostNumber

HostNumber is an integer indicating the number of nodes the user wishes to obtain for his job. This attribute can't be specified along with the SMPGranularity attribute when WholeNodes is false. Please note that `HostNumber` shouldn't be greater than `CpuNumber`

Mandatory: No

Default: No

3.27 CERequirements

The CERequirements attribute is a Boolean ClassAd expression that uses C-like operators. It represents job requirements on resources. Properly configuring the BLAH software, it is possible to instruct the Local Resource Management System to select the most appropriate Worker Node to run the considered job. The CERequirements expression can contain attributes that describe the CE which are prefixed with `other.`.

This is an example of CERequirements expression:

```
CERequirements = "other.GlueCEPolicyMaxCPUTime >= 100 && other.GlueHostMainMemoryRAMSize > 2000";
```

The CERequirements attribute can be used when jobs are submitted directly to a CREAM based CE, while for jobs submitted to a CREAM based CE via the WMS, the CERequirements expression is automatically filled by the WMS considering the user's `Requirements` expression, and the value of the `CeForwardParameters` attribute in the WMS configuration file.

Detailed information about the forwarding of requirements to the batch system is available in the CREAM web site (http://grid.pd.infn.it/cream) under `Forward of requirements to the batch system`.

Mandatory: No

Default: No

3.28 MWVersion

The MWversion attribute is a string whose value is given to the environment value `EDG_MW_VERSION`, defined for the job immediately when it arrives on the WN (i.e. it is not defined in the job wrapper). There can be hooks on the WN which detect this (before any `/etc/profile.d` scripts are run) and set up the appropriate environment for a job.

Mandatory: No

Default: No

3.29 OutputData

This attribute allows the user to ask for the automatic upload and registration to the Replica Catalog of datasets produced by the job on the WN. Through this attribute it is possible to indicate for each output file the LFN (Logical Fine Name) to be used for registration and the SE (Storage Element) on which the file has to be uploaded. The OutputData attribute is not mandatory.

OutputData is a list of classads where each classad contains the following three attributes:

- `OutputFile`
- `StorageElement`
- `LogicalFileName`

These three attributes are only admitted if members of one of the classads composing OutputData. They cannot be specified independently in the job JDL.

Here below follows an example of the OutputData attribute:

```
OutputData = {
[
            OutputFile = "dataset_1.out ";
            LogicalFileName = "lfn:/test/result1";
 ],
[
            OutputFile = "dataset_2.out ";
            StorageElement = "se001.cnaf.infn.it";
 ],
]
            OutputFile = "cms/dataset_3.out";
            StorageElement = "se012.to.infn.it";
            LogicalFileName = "lfn:/cms/outfile1";
 ],
[
            OutputFile = "dataset_4.out ";
 ]
      };
```

If the attribute OutputData is found in the JDL then the JobWrapper at the end of the job calls the Data Management service that copies the file from the WN onto the specified SE and registers it with the given LFN. If the specified LFN is already in use, the DM service registers the file with a newly generated identifier GUID (Grid Unique Identifier).

During this process the JobWrapper creates a file (named `DSUpload_<jobid>.out`) with the results of the operation. In case of submission to CREAM through the WMS this file is put automatically in the OutputSandbox attribute list by the UI (and can then be retrieved by the user with the `glite-wms-job-output` command). If instead the job was submitted directly to the CE, the file is put in the OSB directory of the CE node (and therefore can then be retrieved by the user with the `glite-ce-job-output` command) unless the `OutputSandboxBaseDestURI` attribute has been used (in this latter case the specified location is used to store the DSUpload_<jobid>.out file).

Mandatory: No

Default: No

3.29.1 OutputFile

This is a string attribute representing the name of the output file, generated by the job on the WN, which has to be automatically uploaded and registered by the WMS. Wildcards are not admitted in the specification of this attribute. File names can be provided as simple file names, absolute paths or relative paths with respect to the current working directory.

Mandatory: Yes (only if OutputData has been specified)

Default: No

3.29.2 StorageElement

This is a string representing the URI of the Storage Element where the output file specified in the corresponding OutputFile attribute has to be uploaded by the WMS.

Mandatory: No

Default: No

3.29.3 LogicalFileName

This is a string representing the logical file name (LFN) the user wants to associate to the output file when registering it to the Replica Catalogue. The specified name has to be prefixed by "lfn:" (lowercase). If this attribute is not specified then the corresponding output file is registered with a GUID that is assigned automatically by the Data Management services.

Mandatory: No

Default: No (If not specified a GUID is assigned by DM services)

4 JDL Examples

Simple examples of JDL describing different types of jobs and requests are reported in this section.

4.1 Example 1

[Type = "job"; JobType = "normal"; Executable = "/sw/command"; Arguments = "60"; StdOutput = "sim.out"; StdError = "sim.err"; OutputSandbox = { "sim.err", "sim.out" }; OutputSandboxBaseDestURI = "gsiftp://se1.pd.infn.it:5432/tmp"; InputSandbox = { "file:///home/user/file1", "gsiftp:///se1.pd.infn.it:1234/data/file2", "/home/user/file3", "file4" }; InputSandboxBaseURI = "gsiftp://se2.cern.ch:5678/tmp";]

With this JDL a "normal" (batch) job will be submitted.

Besides the specification of the executable (already available in the file system of the executing node, since not listed in the InputSandbox), and of the standard output/error files, it is specified that the files `file1`, `file2`, `file3`, `file4` will have to be staged on the executing node:

- `file1` and `file3` will be copied from the client (UI) file system
- `file2` will be copied from the specified GridFTP server (`gsiftp:///se1.pd.infn.it:1234/data/file2`)
- `file4` will be copied from the GridFTP server specified as InputSandboxBaseURI (`gsiftp://se2.cern.ch:5678/tmp`)

It is also specified that the file `sim.err` and `sim.out` (specified as OutputSandbox) must be automatically uploaded into `gsiftp://se1.pd.infn.it:5432/tmp` when job completes its execution.

4.2 Example 2

```
[
    Type = "job";
    JobType = "normal";
    Executable = "script.sh";
    Arguments = "60";
         StdOutput = "sim.out";
         StdInput = "sim.inp";
         StdError = "sim.err";
     OutputSandbox = {
              "sim.err",
              "sim.out"
             };
    OutputSandboxDestURI = {
"gsiftp://matrix.datamat.it:5432/tmp/sim.err",
"gsiftp://grid003.ct.infn.it:6789/home/cms/sim.out",
         };
    InputSandbox = {
                 "file:///home/user/file1",
                        "gsiftp:///se1.pd.infn.it:1234/data/file2",
                        "/home/user/file3",
                        "file4",
                        "script.sh",
                        "sim.inp"
              };
]
```

This JDL is very similar to the previous one. The only differences are the following:

- The executable and the standard input files have been included in the InputSandbox, and therefore they will be staged in the executing node
- Instead of specifying the URL to be used for all the files of the OutputSandbox, it is specified (via the attribute OutputSandboxDestURI) an URI for each file that have to be uploaded (the files listed as OutputSandbox)
- The attribute InputSandBoxBaseURI hasn't been specified, so the files `file4`, `script.sh` and `sim.inp` will be copied from the file system of the client (UI) machine.

-- MassimoSgaravatto - 2011-04-07

This topic: CREAM > JdlGuide
Topic revision: r12 - 2011-11-10 - MassimoSgaravatto