NORDUGRID-MANUAL-4
7/1/2006

Extended Resource Specification Language

Reference Manual

Contents

Chapter 1

Introduction

Execution of jobs at batch clusters is the major part of the data analysis, production and other computational tasks and applications in many research areas. In the Grid infrastructure [1], such clusters are not necessarily local to a job submission site, but can be widely distributed. This implies extra requirements for the proper description of job options. The Globus® project [2] developed a Grid middleware toolkit, which makes use of the *Resource Specification Language (RSL)* [3] to parse job options and definitions to resource management systems. The NorduGrid project [4] developed the Advanced Resource Connector (ARC), – a solution [5, 6] for a Grid facility, suitable for complex tasks, like, for example, a High Energy Physics production. To match the complexity of such tasks, this solution requires certain extensions to the RSL.

To describe a task to be submitted to the ARC-enabled resources, an extended version of the Globus® RSL is used. The extensions concern not only introduction of new attributes, but also differentiation between two levels of the job options specifications:

> **User-side RSL**, i.e., the set of attributes specified by a user in a job-specific file. This file is interpreted by the *User Interface (UI)* [7], and after the necessary modifications is parsed to the *Grid Manager (GM)* [8]
>
> **GM-side RSL**, i.e., the set of attributes pre-processed by the UI, and ready to be interpreted by the GM

A user only has to know the user-side part, and utilize it to describe the Grid tasks. The Grid Manager, however, uses slightly different notations, supplied by the User Interface.

In what follows, the description of the NorduGrid-extended RSL, further denoted as **xRSL**, is given, using the following notations:

`<xxxx>`	parameter to be substituted with a corresponding string or a number
`[xxxx]`	optional parameter
`xxx|yyy|zzz`	list of possible values of a parameter
`-"-`	"same as above"

Chapter 2

xRSL syntax and rules

For a complete description of Globus® RSL, see reference [3]. xRSL uses the same syntax conventions, although changes the meaning and interpretation of some attributes.

2.1 Syntax

A Grid task is described by the mean of xRSL attributes, which can be either parsed via a command-line, or, more conveniently, be collected a so-called xRSL-file (suggested extension *.xrsl*). Such a file contains a plain list of attribute strings and boolean operands "&" (for AND) and "|" (for OR).

Typically, an xRSL job description starts with an ampersand ("&") , to indicate implicit **conjunction** of all the attributes:

```
&(attribute1=value1)(attribute2=value2)...
```

Whenever a **disjunct**-request of two or more attributes is needed, the following construction can be used:

```
(|(attribute=value1)(attribute=value2)...)
```

In expressions, the following operands are allowed:

```
=   !=   >   <   >=   <=
```

Commented lines should start with "(*" and be closed with "*)":

```
(*attribute=value1*)
```

Multiple job description in one file is realized via a standard Globus® RSL multi-request operand "+", which should precede multiple job description:

```
+(&(...))(&(...))(&(...))
```

The xRSL attributes can be written in a single string, or split in lines arbitrary; blank spaces between and inside `(attribute=value)` relations are ignored.

2.2 URLs

File locations in ARC can be specified both as local file names, and as Internet standard *Uniform Resource Locators (URL)*. However, there are some additional *options*, used by the Grid Manager. Detailed information is maintained in the ARC URL documentation [9].

The following transfer protocols and metadata servers are supported:

`ftp`	ordinary *File Transfer Protocol (FTP)*
`gsiftp`	GridFTP, the Globus® -enhanced FTP
`http`	ordinary *Hyper-Text Transfer Protocol (HTTP)*
`https`	HTTP with SSL v3
`httpg`	HTTP with Globus® GSI
`ldap`	ordinary *Lightweight Data Access Protocol (LDAP)* [10]
`rc`	Globus® *Replica Catalog (RC)* [11]
`rls`	Globus® *Replica Location Service (RLS)* [12]
`fireman`	Fireman indexing service of EGEE gLite [13]
`se`	ARC Smart Storage Element service [14]
`srm`	Storage Resource Manager (SRM) service [15]
`file`	local to the host file name with a full path

An URL can be used in a standard form, i.e.

```
<protocol>://host[:port]/<file>
```

Or, to enhance the performance, it can have additional options:

```
<protocol>://host[:port][;option[;option[...]]]/<file>
```

For a metadata service URL, construction is the following:

```
rc://rc://[location[|location[...]]@]<host>[:port]/<DN>/<lfn>
rls://[url[|url[...]]@]<host>[:port]/<lfn>
fireman://[url[|url[...]]@]<host>[:port]/<service_path>?<lfn>
```

For the Smart Storage Element service, the syntax is

```
se://host[:port][;options]/path[?file_id]
```

For the SRM service, the syntax is

```
srm://<host>[:port][;options]/[service_path?SFN=]<file_id>
```

Here the URL components are:

`location`	`<location_name_in_RC>[;option[;option[...]]]`
`host[:port]`	IP address of a server
`DN`	Distinguished Name (as in LDAP) of an RC collection
`lfn`	Logical File Name
`url`	URL of the file as registered in RLS/Fireman
`service_path`	End-point path of the Web service
`file`	local to the host file name with a full path

The following options are supported:

`threads=<number>`	specifies number of parallel streams to be used by GridFTP or HTTP(s,g); default value is 1, maximal value is 10
`cache=yes|no`	indicates whether the GM should cache the file; default for input files is `yes` (NB: cached files can not be modified by jobs)
`readonly=yes|no`	for transfers to `file://` destinations, specifies whether the file should be read-only (unmodifiable) or not; default is `yes`
`secure=yes|no`	indicates whether the GridFTP data channel should be encrypted; default is `no`
`blocksize=<number>`	specifies size of chunks/blocks/buffers used in GridFTP or HTTP(s,g) transactions; default is protocol dependent
`checksum=cksum|md5`	specifies the algorithm for checksum to be computed (ev. provided to the indexing server)
`exec=yes|no`	means the file should be treated as executable
`preserve=yes|no`	specify if file must be uploaded to this destination even if job processing failed (default is `no`)
`pattern=<pattern>`	defines file matching pattern; currently works for file listing requests sent to an `se://` endpoint
`guid=yes|no`	make software use GUIDs instead of LFNs while communicating to indexing services; meaningful for `rls://` only
`overwrite=yes|no`	make software try to overwrite existing file(s), i.e. before writing to destination, tools will try to remove any information/content associated with specified URL

Local files are referred by specifying either location relative to the job submission working directory, or by an absolute path (the one that starts with "/"), preceded with a `file://` prefix.

Examples of URLs are:

```
http://grid.domain.org/dir/script.sh
gsiftp://grid.domain.org:2811;threads=10;secure=yes/dir/input_12378.dat
ldap://grid.domain.org:389/lc=collection1,rc=Nordugrid,dc=nordugrid,dc=org
rc://grid.domain.org/lc=collection1,rc=Nordugrid,dc=nordugrid,dc=org/zebra/f1.zebra
rls://gsiftp://se.domain.org/datapath/file25.dat@grid.domain.org:61238/myfile02.dat
fireman://fireman_host:8443/glite-data-catalog-interface/FiremanCatalog?data.root
file:///home/auser/griddir/steer.cra
```

Chapter 3

Attributes

Most of the attributes introduced originally by Globus® RSL 1.0 are supported, some with modifications as indicated in this document. Many new attributes are introduced, of which some are to be specified in the user's script, and others are internal for the GM (are added or modified by the UI).

Attribute names are case-insensitive, although assigned values may well be case-sensitive, if they represent file names, environment variables etc..

3.1 User-side attributes

The following attributes can be specified in a user's xRSL script. Some are to be modified by the UI before being passed to the GM. If this is the case, corresponding GM input is described.

executable

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(executable=<string>)`

GM input: for ARC $\leq$ 0.5.25: `(executable=/bin/echo)`

Example: `(executable=''local_to_job.exe'')`

The executable to be submitted as a main task to a Local Resource Management System (LRMS).

`string` file name (including path), local to the computing element (CE)

Executable is a file that has to be executed as the main process of the task. It could be either a pre-compiled binary, or a script. Users may transfer their own executables, or use the ones known to be already installed on the remote system (CE).

If an executable has to be transferred to the destination site (CE) from some source, it has to be specified in the `inputFiles` list. If it is not specified in `inputFiles`, the source is expected to be local to the user (client) and will be added as such to the `inputFiles` list by the UI.

If the file name starts with a leading slash ("`/`"), it is considered to be **the full path to the executable at the destination site (CE)**; otherwise the location of the file is **relative** to the session directory (where job input and files are stored). If the file name starts with an environment variable ("`$...`"), the value of this variable is resolved locally, but if it is enclosed in double quotes, it will be resolved at the remote computing element:
`(executable=$ROOT_DIR/myprog.exe)` – `$ROOT_DIR` is resolved locally (*will cause errors if the path does not exist at the execution machine*)
`(executable=''$ROOT_DIR/myprog.exe'')` – `$ROOT_DIR` will be resolved remotely

For ARC $\leq$ 0.5.25, UI created a dummy executable attribute for internal GM use, with e.g., "`/bin/echo`" as an argument (unless the specified executable itself started with "`/`"). The actual executable was

transferred by the UI to `arguments`, `inputFiles` and `executables` lists in the pre-processed XRLS script.

arguments

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(arguments=<string> [string] ... )`

GM input: `(arguments=<executable> <string> [string] ... )`

Example: `(arguments="10000" $(ATLAS)/input.dat)`

List of the arguments for the executable.

`string`	an argument
`executable`	the executable to be run by LRMS, taken by the UI from the user-specified `executable` attribute

For ARC $\leq$ 0.5.25, UI added the name specified in `executable` to the `arguments` list in the pre-processed XRSL script.

inputFiles

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(inputFiles=(<filename> <location>) ... )`

GM input:
```
(inputFiles=(<filename> <URL>)
        (<filename> [size][.checksum]) ... )
```

Example:
```
(inputFiles=(``local_to_job'' ``gsiftp://se1.lu.se/p1/remote_one'')
        (``local_to_job.dat'' ``/scratch/local_to_me.dat'')
        (``same_name_as_in_my_current_dir'' ``''))
```

List of files to be copied to the computing element before the execution.

`filename`	file name, local to the computing element and always relative to the session directory
`location`	location of the file (`gsiftp`, `https`, `ftp`, `http` URLs, or a path, local to the submission node). If void (””, use the quotes!), the input file is taken from the submission directory.
`URL`	URL of the file (see Section 2.2)
`size`	file size in bytes
`checksum`	file checksum (as returned by `cksum`)

If the list does not contain the standard input file (as specified by `stdin`) and/or the executable file (as specified by `executable` if the given name), the UI appends these files to the list. If the `<location>` is a URL, it is passed by the UI to the GM without changes; if it is a local path (or void, “”), the UI converts it to `<size>` and `<checksum>` and uploads those files to the execution cluster.

> Please note that the `inputFiles` attribute is not meant to operate with directories: you must specify a pair `("<local_to_job>" "[source]")` for each file.

For ARC $\leq$ 0.5.25, UI added the name specified in `executable` to the `inputFiles` list in the pre-processed XRSL script.

executables

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(executables=<string> [string] ...)`

GM input: `-''-`

Example: `(executables=myscript.sh myjob.exe)`

List of files from the `inputFiles` set, which will be given executable permissions.

`string` file name, local to the computing element and relative to the session directory

If the executable file (as specified in `executable` and is relative to the session directory) is not listed, it will be added to the list by the UI.

cache

(ARC $\geq$ 0.3.34, ARC 0.4, ARC 0.5)

User input: `(cache=yes|no)`

GM input: `-''-`

Example: `(cache=''yes'')`

Specifies whether input files specified in the `inputFiles` should be placed by default in the cache or not. This does not affect files described by the `executable`, which will be placed in the session directory always.

If not specified, default value is "yes".

> Cached files can not be modified by jobs. If your job has to modify input files, please specify `(cache=''no'')`.

outputFiles

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(outputFiles=(<string> <URL>) ... )`

GM input: `-''-`

Example:
```
(outputFiles=(``local_to_job.dat'' ``gsiftp://se1.uo.no/stored.dat'')
    (``local_to_job.dat'' ``rc://rc1.uc.dk/set1/stored_and_indexed.dat'')
    (``local_to_job_dir/'' ``''))
```

List of files to be retrieved by the user or uploaded by the GM and indexed (registered) in a data indexing service, e.g. Globus® RLS or Fireman.

`string` file name, local to the *Computing Element (CE)*. If this string ends with a backslash "/" and `<URL>` is empty, the entire directory is being kept at the execution site. If however this string ends with a backslash "/" but the `<URL>` starts with `gsiftp` or `ftp`, the whole directory is transfered to the destination.

`URL` URL of the remote file (see Section 2.2); if void (""), use the quotes!), the file is kept for manual retrieval. Note that this can not be a local `file://` URL.

Using a Replica Catalog pseudo-URL (see Section 2.2), you should make sure that the location is already defined in the Replica Catalog. If more than one such location is specified, only those found in the Replica

Catalog for this collection will be used. Grid Manager will store output files in **one** location only: the first randomly picked location that exists. If no locations are specified, all those found in the Replica Catalog for this collection will be tried.

If in a `rc://` pseudo-URL the server component `host[:port]/DN` is not specified, server specified in the `replicaCollection` attribute will be used.

If the list does not contain standard output, standard error file names and GM log-files directory name (as specified by `stdout`, `stderr` and `gmlog`), the UI appends these items to the `outputFiles` list. If the `<URL>` is not specified (void, ""), use the quotes!), files will be kept on SE and should be downloaded by the user via the UI. If specified name of file ends with "/", the entire directory is kept.

A convenient way to keep the entire job directory at the remote site for a manual retrieval is to specify `(outputfiles=("/" ""))`.

> Please note that the `outputFiles` attribute is not meant to operate with directories: you must specify a pair `("<local_to_job>" "[destination]")` for each file. One exception is when you want to preserve an entire directory at the remote computer for later **manual** download via `ngget`. In that case, simply add the trailing backslash "/" at the end of the remote directory name. You can not upload a directory to a URL location, only to your local disk.

cpuTime

(ARC $\geq$ 0.3.17*, ARC 0.4, ARC 0.5)

User input: `(cpuTime=<time>)`

GM input: `(cpuTime=<tttt>)`

Example: `(cpuTime=240)`

Maximal CPU time request for the job.

`time`	time (minutes)
`tttt`	time converted by the UI from `time` to minutes (ARC $\leq$ 0.5.26) or seconds (ARC $\geq$ 0.5.27)

If only number is specified, the time is assumed to be minutes (ARC $\leq$ 0.5.26) or seconds (ARC $\geq$ 0.5.27). Otherwise, a free format is accepted, i.e., any of the following will be interpreted properly (make sure to enclose such strings in quotes!):

```
''1 week''
''3 days''
''2 days, 12 hours''
''1 hour, 30 minutes''
''36 hours''
''9 days''
''240 minutes''
```

If both `cpuTime` and `wallTime` are specified, the UI converts them both to seconds. `cpuTime` can not be specified together with `gridTime` or `benchmarks`.

> This attribute should be used to direct the job to a system with sufficient CPU resources, typically, a batch queue with the sufficient upper time limit. Jobs exceeding this maximum most likely will be **terminated** by remote systems! If time limits are not specified, the limit is not set and jobs can run as long as the system settings allow (note that in this case you can not avoid queues with too short time limits).

*Was called `maxCpuTime` for ARC $\leq$ 0.3.16.

wallTime

(ARC $\geq$ 0.5.27)

User input: `(wallTime=<time>)`
GM input: `(wallTime=<tttt>)`
Example: `(wallTime=240)`

Maximal wall clock time request for the job.

`time` time (minutes)
`tttt` time converted by the UI from `time` to seconds

If only number is specified, the time is assumed to be seconds. Otherwise, a free format is accepted, i.e., any of the following will be interpreted properly (make sure to enclose such strings in quotes!):

```
''1 week''
''3 days''
''2 days, 12 hours''
''1 hour, 30 minutes''
''36 hours''
''9 days''
''240 minutes''
```

If both `cpuTime` and `wallTime` are specified, the UI converts them both to seconds. `wallTime` can not be specified together with `gridTime` or `benchmarks`. If only `wallTime` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the UI and added to the job description.

> This attribute should be used to direct the job to a system with sufficient CPU resources, typically, a batch queue with the sufficient upper time limit. Jobs exceeding this maximum most likely will be **terminated** by remote systems! If time limits are not specified, the limit is not set and jobs can run as long as the system settings allow (note that in this case you can not avoid queues with too short time limits).

gridTime

(ARC $\geq$ 0.3.31, ARC 0.4, ARC 0.5)

User input: `(gridTime=<time>)`
GM input: none
Example: `(gridTime=''2 h'')`

Maximal CPU time request for the job scaled to the 2.8 GHz Intel® Pentium® 4 processor.

`time` time (minutes)

The attribute is completely analogous to `cpuTime`, except that it will be recalculated to the actual CPU time request for each queue, depending on the published processor clock speed.

`gridTime` can not be specified together with `cpuTime` or `wallTime`. If only `gridTime` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the UI and added to the job description.

benchmarks

(ARC $\geq$ 0.5.14)

User input: `(benchmarks=(<string> <value>) ... )`

GM input:

Example: `(benchmarks=(''mybenchmark'' ''10''))`

Request a cluster with a specified benchmark value.

`string`	benchmark name
`value`	benchmark value, in seconds

`benchmarks` can not be specified together with `cpuTime` or `wallTime`. If only `benchmarks` is specified, but not `cpuTime`, the corresponding `cpuTime` value is evaluated by the UI and added to the job description.

memory

(ARC $\geq$ 0.3.17[†], ARC 0.4, ARC 0.5)

User input: `(memory=<integer>)`

GM input: `-''-`

Example: `(memory>=500)`

Memory required for the job.

`integer`	size (Mbytes)

> Just like `cpuTime`, this attribute should be used to direct a job to a resource with a sufficient capacity. Jobs exceeding this memory limit will most likely be **terminated** by the remote system.

disk

(ARC $\geq$ 0.3.17[‡], ARC 0.4, ARC 0.5)

User input: `(disk=<integer>)`

GM input: none

Example: `(disk=500)`

Disk space required for the job.

`integer`	disk space, Mbytes

> This attribute is used at the job submission time to find a system with sufficient disk space. However, it **does not guarantee** that this space will be available at the end of the job, as most known systems do not allow for disk space allocation. Eventually, a remote system can terminate a job that exceeds the requested disk space.

runTimeEnvironment

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(runTimeEnvironment=<string>)(runTimeEnvironment=<string>)`

GM input: For ARC $\leq$ 0.5.39: `(runTimeEnvironment=''<string>'' ''<string>'')`

Example: `(runTimeEnvironment>=''APPS/HEP/ATLAS-10.0.1'')`

[†]Was called `maxMemory` for ARC $\leq$ 0.3.16
[‡]Was called `maxDisk` for ARC $\leq$ 0.3.16

Required runtime environment

`string` environment name

The site to submit the job to will be chosen by the UI among those advertising specified runtime environments. Before starting the job, the GM will set up environment variables and paths according to those requested. Runtime environment names are defined by Virtual Organizations, and tend to be organized in name spaces.

To request several environments, repeat the attribute string:
`(runTimeEnvironment=ENV1)(runTimeEnvironment=ENV2)` etc. In ARC $\leq$ 0.5.39, before being submitted to the GM, a grouping was done by the UI, such that GM input was, e.g., `(runTimeEnvironment=''ENV1'' ''ENV2'')`.

To make a disjunct-request, use a boolean expression:
`(|(runTimeEnvironment=env1)(runTimeEnvironment=env2))`.

Runtime environment string interpretation is case-insensitive. If a runtime environment string consists of a name and a version number, a partial specification is possible: it is sufficient to request only the name.

You can use "`>=`" or "`<=`" operators: job will be submitted to any suitable site that satisfies such requirements, and among the available at the sites runtime environments, the highest version satisfying a requirement will be requested in the pre-processed XRSL script.

middleware

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(middleware=<string>)`

GM input: `-''-`

Example: `(middleware=NorduGrid-0.3.99)`

Required middleware.

`string` Grid middleware name.

The site to submit the job to will be chosen by the UI among those advertising specified middleware. Usage is identical to that of the `runTimeEnvironment`. Use the "`>=`" operator to request a version "equal or higher". Request `(middleware=nordugrid)` defaults to `(middleware>=nordugrid-0.0.0.0)`.

opsys

(ARC $\geq$ 0.3.21, ARC 0.4, ARC 0.5)

User input: `(opsys=<string>)`

GM input: `-''-`

Example: `(opsys=''FC3'')`

Required operating system.

`string` Operating system name and version.

The site to submit the job to will be chosen by the UI among those advertising specified operating system. Usage is identical to that of `runTimeEnvironment` and `middleware`. Use the "`>=`" operator to request a version "equal or higher".

stdin

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(stdin=<string>)`

GM input: `-''-`

Example: `(stdin=myinput.dat)`

The standard input file.

`string` file name, local to the computing element

The standard input file should be listed in the `inputFiles` attribute; otherwise it will be forced to that list by the UI.

stdout

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(stdout=<string>)`

GM input: `-''-`

Example: `(stdout=myoutput.txt)`

The standard output file.

`string` file name, local to the computing element and relative to the session directory.

The standard output file should be listed in the `outputFiles` attribute; otherwise it will be forced to that list by the UI. If the standard output is not defined, UI assigns a name.

stderr

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(stderr=<string>)`

GM input: `-''-`

Example: `(stderr=myjob.err)`

The standard error file.

`string` file name, local to the computing element and relative to the session directory.

The standard error file should be listed as an `outputFiles` attribute; otherwise it will be forced to that list by the UI. If the standard error is not defined, UI assigns a name. If `join` is specified with value "yes", UI adds `stderr` to the pre-processed XRSL script with the same value as `stdout`.

join

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(join=yes|no)`

GM input:none

Example: `(join=yes)`

If "yes", joins `stderr` and `stdout` files into the `stdout` one. Default is `no`.

gmlog

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(gmlog=<string>)`

GM input: `-''-`

Example: `(gmlog=myjob.log)`

A name of the directory containing grid-specific diagnostics per job.

`string` a directory, local to the computing element and relative to the session directory

This directory is kept in the session directory to be available for retrieval (UI forces it to the list if `outputFiles`)

jobName

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(jobName=<string>)`

GM input: `-''-`

Example: `(jobName=MyJob)`

User-specified job name.

`string` job name

This name is meant for convenience of the user. It can be used to select the job while using the UI. It is also available through the Information System.

ftpThreads

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(ftpThreads=<integer>)`

GM input: `-''-`

Example: `(ftpThreads=4)`

Defines how many parallel streams will be used by the GM during `gsiftp` transfers of files.

`integer` a number from 1 to 10

If not specified, parallelism is not used.

acl

(ARC $\geq$ 0.5.12)

Available in ARC v0.5.12 and higher.

User input: `(acl=<xml>)`

GM input: `-''-`

Example:
```
(acl="<?xml version=""1.0""?>
    <gacl version=""0.0.1""><entry><any-user></any-user>
    <allow><write/><read/><list/><admin/></allow></entry></gacl>")
```

Makes use of GACL [16] rules to list users who are allowed to access and control job in addition to job's owner. Access and control levels are specified per user. `any-user` tag refers to any user authorized at the execution cluster. To get more information about GACL please refer to `http://www.gridsite.org`.

`xml` a GACL-compliant XML string defining access control list

Following job control levels can be specified via `acl`:

- `write` – allows to modify contents of job data (job directory) and control job flow (cancel, clean, etc.)
- `read` – allows to read content of job data (contents of job directory)
- `list` – allows to list files available for the job (contents of job directory)
- `admin` – allows to do everything – full equivalence to job ownership

cluster

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(cluster=<string>)`

GM input: `-''-`

Example: `(cluster=nbi)`

The name of the execution cluster.

`string` known cluster name, or a substring of it

Use this attribute to explicitly force job submission to a cluster, or to avoid such. The job will not be submitted if the cluster does not satisfy other requirements of the job. Disjunct-requests of the kind `(|(cluster=clus1)(cluster=clus2))` are supported. To exclude a cluster, use `(cluster!=clus3)`.

queue

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(queue=<string>)`

GM input: `-''-`

Example: `(queue=pclong)`

The name of the remote batch queue.

`string` known queue name

Use this attribute to explicitly force job submission to a queue.

startTime

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(startTime=<time>)`

GM input: `(startTime=<tttt>)`

Example: `(startTime="2002-05-25 21:30")`

Time to start job processing by the Grid Manager, such as e.g. start downloading input files.

`time` time string, YYYY-MM-DD hh:mm:ss

`tttt` time string, YYYYMMDDhhmmss[Z] (converted by the UI from `time`)

> Actual job processing on a worker node starts depending on local scheduling mechanisms, but not sooner than `startTime`.

lifeTime

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(lifeTime=<time>)`

GM input: `(lifeTime=<tttt>)`

Example: `(lifeTime=60)`

Maximal time to keep job files (the session directory) on the gatekeeper upon job completion.

`time` time (days)

`tttt` time (seconds, converted by the UI from `time`)

Typical life time is 1 day (24 hours). Specified life time can not exceed local settings.

notify

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(notify=<string> [string] ... )`

GM input: `-''-`

Example: `(notify="be your.name@your.domain.com")`

Request e-mail notifications on job status change.

`string` string of the format: `[b][q][f][e][c] user1@domain1 [user2@domain2] ...`
here flags indicating the job status are:
`b` – begin (PREPARING)
`q` – queued (INLRMS)
`f` – finalizing (FINISHING)
`e` – end (FINISHED)
`c` – cancellation (CANCELLED)
`d` – deleted (DELETED)

No more than 3 e-mail addresses per status change accepted.

replicaCollection

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(replicaCollection=<URL>)`

GM input: `-''-`

Example: `(replicaCollection="ldap://grid.uio.no:389/lc=TestCollection,rc=NorduGrid,nordugrid,dc=org")`

Location of a logical collection in the Replica Catalog.

`URL` LDAP directory specified as an URL (`ldap://host[:port]/dn`)

rerun

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(rerun=<integer>)`

GM input: `-''-`

Example: `(rerun=2)`

Number of reruns (if a system failure occurs).

`integer` an integer number

If not specified, the default is 0. Default maximal allowed value is 2.

architecture

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(architecture=<string>)`

GM input:

Example: `(architecture=i686)`

Request a specific architecture.

`string` architecture (e.g., as produced by uname -a)

nodeAccess

(ARC $\geq$ 0.3.24, ARC 0.4, ARC 0.5)

User input: `(nodeAccess=inbound|outbound)`

GM input:

Example: `(nodeAccess=inbound)`

Request cluster nodes with inbound or outbound IP connectivity. If both are needed, a conjunct request should be specified.

dryRun

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(dryRun=yes|no)`

GM input: `-''-`

Example: `(dryRun=yes)`

If "yes", do dry-run: RSL is parsed, but no job submission to LRMS is made. Should be used for xRSL validation.

rsl_substitution

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(rsl_substitution=(<string1> <string2>))`

GM input: `-''-`

Example: `(rsl_substitution=(ATLAS /opt/atlas))`

Substitutes `<string2>` with `<string1>` for **internal** RSL use.

`string1` new internal RSL variable

`string2` any string, e.g., existing combination of variables or a path

Use this attribute to simplify xRSL editing. Only one pair per substitution is allowed. To request several substitution, concatenate such requests. Bear in mind that substitution must be defined **prior** to actual use of a new variable `string1`.

environment

(ARC 0.3, ARC 0.4, ARC 0.5)

User input: `(environment=(<VAR> <string>) [(<VAR> <string>)] ... )`

GM input: `-''-`

Example:
```
(environment=(ATLSRC /opt/atlas/src)
             (ALISRC /opt/alice/src))
```

Defines execution shell environment variables.

`VAR`	new variable name
`string`	any string, e.g., existing combination of variables or a path

Use this to define variables at an execution site.

count

(ARC ≥ 0.3.12, ARC 0.4, ARC 0.5)

User input:	`(count=<integer>)`
GM input:	`-''-`
Example:	`(count=4)`

Specifies amount of sub-jobs to be submitted for parallel tasks.

jobreport

(ARC ≥ 0.3.34, ARC 0.4, ARC 0.5)

User input:	`(jobreport=<URL>)`
GM input:	`-''-`
Example:	`(jobreport="https://grid.uio.no:8001/logger")`

Specifies an URL for a logging service to send reports about job to. The default is set up in the cluster configuration.

`URL`	URL

It is up to a user to make sure the requested logging service accepts reports from the set of clusters she intends to use.

3.2 GM-side attributes

The following attributes are constructed by the UI and are parsed to the GM, although users may specify them manually (**not advised!**).

sstdin

(ARC 0.3, ARC 0.4, ARC ≤ 0.5.25)

User input:	
GM input:	`(sstdin=<filename>)`
Example:	`(sstdin=myinput.dat)`

Internal attribute for the standard input. Can also be spelled `stdinput`. Only needed for GRAM compatibility, not used by ARC as such.

`filename`	standard input file name

action

(ARC 0.3, ARC 0.4, ARC 0.5)

User input:

GM input: `(action=request|cancel|clean)`

Example: `(action=request)`

Action to be taken by the gatekeeper: submit the job, cancel job execution, or clear the results of the job (also cancels the job).

savestate

(ARC 0.3, ARC 0.4, ARC $\leq$ 0.5.25)

User input:

GM input: `(savestate=yes|no)`

Example: `(savestate=yes)`

If "yes", input RSL is stored in a temporary file at the gatekeeper. Must be always set as "yes" in the current implementation. Only needed for GRAM compatibility, not used by ARC as such.

lrmstype

(ARC $\leq$ 0.3.15)

User input:

GM input: `(lrmstype=<string>)`

Example: `(lrmstype=pbs)`

LRMS type, indicating which submission script is to be invoked.

`string` LRMS type

hostName

(ARC 0.3, ARC 0.4, ARC 0.5)

User input:

GM input: `(hostname=<string>)`

Example: `(hostName=grid.quark.lu.se)`

Submission host name.

`string` host name, as passed by the UI

jobid

(ARC 0.3, ARC 0.4, ARC 0.5)

User input:

GM input: `(jobid=<string>)`

Example: `(jobid=grid.quark.lu.se:2119/jobmanager-ng/157111017133827)`

Unique job identification string, needed for cancellation and clean-up.

`string` global job ID

It can also be provided during submission of the job and should be unique to a computing element (cluster).

clientxrsl

(ARC $\geq$ 0.3.28, ARC 0.4, ARC 0.5)

User input:

GM input: `(clientxrsl=<string>)`

Example: `(clientxrsl=''&(executable=/bin/echo)(arguments=boo)'')`

Job description XRSL string as submitted by the user, before being pre-processed by the client.

`string` original XRSL description submitted by the user

This attribute is added by the User Interface during pre-processing, and is used for job re-submission in order to repeat brokering and matchmaking.

clientsoftware

(ARC $\geq$ 0.3.35, ARC 0.4, ARC 0.5)

User input:

GM input: `(clientsoftware=<string>)`

Example: `(clientsoftware=''nordugrid-arc-0.5.39'')`

Version of ARC client used to submit the job.

`string`

This attribute is added by the User Interface during pre-processing.

3.3 Unsupported Globus RSL attributes

The following Globus® attributes are not supported by the ARC middleware. Whenever they are specified, a warning is issued by the UI, and the corresponding attribute is ignored.

3.3.1 Unsupported RSL 1.0 attributes

- `(resourceManagerContact=<string>)`
- `(directory=<string>)`
- `(maxCpuTime=<time>)`
- `(maxWallTime=<time>)`
- `(maxTime=<time>)`
- `(maxMemory=<memory>)`
- `(minMemory=<memory>)`
- `(gramMyJob=independent|collective)`
- `(project=<string>)`
- `(hostCount=<number>)`
- `(label=<string>)`
- `(subjobCommsType=blocking-join|independent)`
- `(subjobStartType=strict-barrier|loose-barrier|no-barrier)`

3.3.2 Unsupported GRAM RSL attributes

- (directory=<string>)
- (fileCleanUp=<array>)
- (fileStageIn=<array>)
- (fileStageInShared=<array>)
- (fileStageOut=<array>)
- (gassCache=<path>)
- (gramMyJob=independent|collective)
- (hostCount=<number>)
- (jobType=<string>)
- (libraryPath=<path>)
- (maxCpuTime=<time>)
- (maxWallTime=<time>)
- (maxTime=<time>)
- (maxMemory=<memory>)
- (minMemory=<memory>)
- (project=<string>)
- (remoteIoUrl=<string>)
- (scratchDir=<string>)

Appendix A

Examples

A.1 User-side xRSL script

```
&
(* test run: if "yes", only submits RSL without actual job start *)
     (dryRun=no)
(* some local variables defined for further convenience *)
     (rsl_substitution=(TOPDIR /home/oxana))
     (rsl_substitution=(NGTEST $(TOPDIR)/examples/ngtest))
     (rsl_substitution=(BIGFILE /scratch/oxana/100mb.tmp))
(* some environment variables, to be used by the job *)
     (environment=(ATLAS /opt/atlas) (CERN /cern))
(* the main executable file to be staged in and submitted to the PBS *)
     (executable=checkall.sh)
(* the arguments for the executable above *)
     (arguments=pal)
(* files to be staged in before the execution *)
     (inputFiles = (be_kaons "")
     (file1 gsiftp://grid.uio.no$(TOPDIR)/remfile.txt)
      (bigfile.dat $(BIGFILE) )    )
(* files to be given executable permissions after staging in *)
     (executables=be_kaons)
(* files to be staged out after the execution *)
     (outputFiles=
        (file1 gsiftp://grid.tsl.uu.se/tmp/file1.tmp)
        (100mb.tmp rc://grid.fi.uib.no/bigfile)
        (be_kaons.hbook  gsiftp://grid.quark.lu.se$(NGTEST)/kaons.hbook)   )
(* location of the Replica collection *)
   (replicaCollection="ldap://grid.uio.no:389/lc=Col1,rc=NorduGrid,dc=nordugrid,dc=org")
(* user-specified job name *)
     (jobName=NGtest)
(* standard input file *)
     (stdin="myinput.dat")
(* standard output file *)
     (stdout="myoutput.dat")
(* standard error file *)
     (stderr="myerror.dat")
(* GM logs directory name *)
     (gmlog="gmlog")
(* flag whether to merge stdout and stderr *)
     (join="no")
(* request e-mail notification on status change *)
```

```
     (notify="bqfe oxana.smirnova@quark.lu.se oxana.smirnova@cern.ch")
(* specific queue to submit the job *)
     (queue=atlas)
(* maximal CPU time required for the job, minutes for PBS*)
     (CpuTime=60)
(* maximal time for the session directory to exist on the remote node, days *)
     (lifeTime=7)
 (* memory required for the job, Mbytes *)
     (Memory=200)
(* wall time to start job processing *)
     (startTime="2002-04-28 17:15:00")
(* disk space required for the job, Mbytes *)
     (Disk=500)
(* required architecture of the execution node *)
     (architecture=i686)
(* required run-time environment *)
     (runTimeEnvironment="Atlas-1.1")
(* number of re-runs, in case of a system failure *)
     (rerun=2)
```

A.2 GM-side xRSL script

```
&
(* saves RSL in a temporary file if "yes"*)
     (savestate=yes)
(* job submission to be performed if action is "request" *)
     (action=request)
(* LRMS type, so far only pbs *)
     (lrmstype=pbs)
(* submission host name *)
     (hostName="grid.quark.lu.se")
(* test run: if "yes", only submits RSL without actual job start *)
     (dryRun=no)
(* some local variables defined for further convenience *)
     (rsl_substitution=(TOPDIR /home/oxana))
     (rsl_substitution=(NGTEST $(TOPDIR)/examples/ngtest))
     (rsl_substitution=(BIGFILE /scratch/oxana/100mb.tmp))
(* some environment variables, to be used by the job *)
     (environment=(ATLAS /opt/atlas) (CERN /cern))
(* a dummy executable *)
     (executable=/bin/echo)
(* the main executable itself, and its arguments *)
     (arguments=checkall.sh pal)
(* files to be staged in before the execution *)
     (inputFiles = (checkall.sh 210.279320915)
        (myinput.dat 14.3980640923)
        (be_kaons 880788.2035414420)
        (file1 gsiftp://grid.uio.no$(TOPDIR)/remfile.txt)
        (bigfile.dat 104857600.4087847787)
     )
(* files to be given executable permissions after staging in *)
     (executables=checkall.sh be_kaons)
(* files to be staged out after the execution *)
     (outputFiles=(file1 gsiftp://grid.tsl.uu.se/tmp/file1.tmp)
        (100mb.tmp rc://grid.fi.uib.no/bigfile)
        (be_kaons.hbook  gsiftp://grid.quark.lu.se$(NGTEST)/kaons.hbook)
```

```
        (myoutput.dat  gsiftp://grid.quark.lu.se/examples/ngtest/myoutput.dat)
        (myerror.dat  gsiftp://grid.quark.lu.se/examples/ngtest/myerror.dat)
    )
(* location of the Replica collection *)
  (replicaCollection="ldap://grid.uio.no:389/lc=Col1,rc=NorduGrid,dc=nordugrid,dc=org")
(* user-specified job name *)
     (jobName=NGtest)
(* standard input file *)
     (stdin="myinput.dat")
(* standard output file *)
     (stdout="myoutput.dat")
(* standard error file *)
     (stderr="myerror.dat")
(* flag whether to merge stdout and stderr *)
     (join="no")
(* request e-mail notification on status change *)
     (notify="bqfe oxana.smirnova@quark.lu.se oxana.smirnova@cern.ch")
(* specific queue to submit the job *)
     (queue=pc)
(* CPU time required for the job, minutes for PBS*)
     (CpuTime=60)
(* maximal time for the session directory to exist on the remote node, seconds *)
     (lifeTime=604800)
(* memory required for the job, Mbytes *)
     Memory=200)
(* wall time to start job processing *)
     (startTime=20020128171500)
(* disk space required for the job, Mbytes *)
     (Disk=500)
(* required architecture of the execution node *)
     (architecture=i686)
(* required run-time environment *)
     (runTimeEnvironment="Atlas-1.1")
(* number of re-runs, in case of a system failure *)
     (rerun=2)
(* job ID - needed for cancellation and cleanup *)
     (*(jobid=157111017133827)*)
```

Bibliography

[1] I. Foster and C.Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[2] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128, 1997.

[3] The Globus Resource Specification Language RSL v1.0. [Online]. Available: http://www-fp.globus.org/gram/rsl˙spec1.html

[4] The NorduGrid Collaboration. [Online]. Available: http://www.nordugrid.org

[5] A.Wäänänen *et al.*, "An Overview of an Architecture Proposal for a High Energy Physics Grid," in *PARA '02: Proceedings of the 6th International Conference on Applied Parallel Computing Advanced Scientific Computing*. Springer-Verlag, 2002, pp. 76–88, lecture Notes In Computer Science, Vol. 2367.

[6] M. Ellert *et al.*, "The NorduGrid project: Using Globus toolkit for building Grid infrastructure," *Nucl. Instr. and Methods* ***A***, vol. 502, pp. 407–410, 2003.

[7] M. Ellert, *The NorduGrid toolkit user interface*, The NorduGrid Collaboration, NORDUGRID-MANUAL-1.

[8] A. Konstantinov, *The NorduGrid Grid Manager And GridFTP Server: Description And Administrator's Manual*, The NorduGrid Collaboration, NORDUGRID-TECH-2.

[9] ——, *Protocols, Uniform Resource Locators (URL) and Extensions Supported in ARC*, The NorduGrid Collaboration, NORDUGRID-TECH-7.

[10] M. Smith and T. A. Howes, *LDAP : Programming Directory-Enabled Applications with Lightweigt Directory Access Protocol*. Macmillan, 1997.

[11] H. Stockinger *et al.*, "File and Object Replication in Data Grids," *Cluster Computing*, vol. 5, no. 3, pp. 305–314, July 2002.

[12] A. L. Chervenak *et al.*, "Performance and Scalability of a Replica Location Service," in *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC'04)*. IEEE Computer Society Press, 2004, pp. 182–191.

[13] gLite, Lightweight Middleware for Grid Computing. [Online]. Available: http://glite.web.cern.ch/glite/

[14] A. Konstantinov, *The NorduGrid Smart Storage Element*, The NorduGrid Collaboration, NORDUGRID-TECH-10.

[15] Storage Resource Management Working Group. [Online]. Available: http://sdm.lbl.gov/srm-wg/

[16] A. McNab, "The GridSite Web/Grid security system: Research Articles," *Softw. Pract. Exper.*, vol. 35, no. 9, pp. 827–834, 2005.

Index