NORDUGRID-MANUAL-4
14/12/2017

Extended Resource Specification Language

Reference Manual for ARC versions 0.8 and above

Contents

Chapter 1

Introduction

Computational tasks, such as data mining, data transformation, simulation of various conditions, execution of Monte Carlo algorihms and so on, are normally formalized as series of *jobs* submitted to computing resources. Such computing resources can be mainframe supercomputers, high-performance computing clusters, or even pools of regular desktop PCs. In a conventional approach, applicaton experts obtain user acconts at computing centers and learn peculiarities of their *batch systems* in order to formulate and submit computational jobs to the assigned resource. In the Grid infrastructure [7], experts get simultaneous access to a large number of very different computing resources, which are often widely distributed geographically. In order to abstract from the heterogeneous nature of Grid resources, a high-level job description language is necessary. The diversity of Grid resources implies special requirements for a proper description of a job, introducing many new options as opposed to a conventional high-performance computing center use case.

Globus Alliance [6] developed a Grid middleware toolkit, early versions of which made use of the *Resource Specification Language (RSL)* [2] to parse job options and definitions to resource management systems. The NorduGrid project [1] decided to re-use RSL when they developed the Advanced Resource Connector (ARC), – a solution [5] for a Grid facility, suitable for complex tasks, like, for example, High Energy Physics data processing. To match the complexity of such tasks, this solution requires certain extensions to the RSL.

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

> **User-side RSL**, i.e., the set of attributes specified by a user in a job-specific file. This file is interpreted by a *Client* [4], and after the necessary modifications is passed to the ARC computing service: the *Grid Manager (GM)* [10]
>
> **GM-side RSL**, i.e., the set of attributes pre-processed by a client, and ready to be interpreted by the GM. Effectively, this is an *internal* job representation of ARC.

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 client tools; therefore developers of such tools must take care of converting user-submitted RSL to the internal job description (GM-side RSL).

In what follows, 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"

*One can also use a Grid-standard JSDL language; however, it is less versatile

Usage examples are given for the standard ARC command-line client (also known as *User Interface*) [4]; other clients may introduce additional features, please refer to their respective documentation for details.

Chapter 2

xRSL syntax and rules

For a complete description of Globus® RSL, see reference [2]. xRSL uses the same syntax conventions, although changes the meaning and interpretation of some attributes.

2.1 Syntax

A Grid task is described by means of xRSL attributes, which can be either passed via a command-line, or, more conveniently, be collected in a so-called xRSL-file (suggested extension *.xrsl*). Such a file contains a plain list of attribute-value pairs and boolean operators "&" (for AND) and "|" (for OR). Attribute names are case-insensitive.

If the attributes are specified from the command line, the entire description must be enclosed either in single or in double quotes. Single quotes enclosure is completely analogous to xRSL-file usage, while double quotes imply standard shell expansion of the enclosed string. This has implications when strings contain local shell variables: they **will not be expanded** unless the task description is entered from the command line and is enclosed in double quotes.

An attribute-value pair is a key element of specification. It consists of an expression that assigns one or more values to an attribute, and is enclosed in round brackets:

```
(attribute="value")
(attribute="value1" "value2") (for multi-valued attributes)
```

Certain attributes do not have assigned value; instead, they contain a list of values that should be put in proper correspondence with each other:

```
(attribute=("value1" "value2")("value3" "value4"))
```

In the example above, `value1` and `value3` are put in correspondence to `value2` and `value4` respectively, according to the context of the attribute.

Values should be enclosed in quotes if they contain blank spaces or special characters. The **special characters** are:

```
+   &   |   (   )   =   <   >   !   "   '   ^   #   $
```

To quote a string containing special characters, you can use either single or double quotes. If your string, however, contains both such quotes, you can define any character as an own delimiter, by preceding it with the "carat" (^) character: `attribute=^*My "good" value^*` makes use of a carat-escaped asterisk as a delimiter.

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

```
&(attribute1=value1)(attribute2="value 2")...
```

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

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

> Only few selected attributes (indicated further in the document) can be requested by the user multiple times, like in the disjunct request example above. Most attributes **must be unique**, i.e., appear only once in the job description document.

In expressions, the following relational operators are allowed, in general:

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

However, most attributes can **only be used with equality operator "="**. For few attributes (as indicated in the document), some other operators can be used in client-side job description as well*.

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

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

Comments can not be nested.

Multiple job description in one file is realized via a standard Globus® RSL multi-request operator "+", 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)*. There are also some additional URL *options* that can be used.

Depending on the installed ARC components some or all of the following transfer protocols and metadata services are supported:

*Job description document received by the server must only contain "=" operators

`ftp`	ordinary *File Transfer Protocol (FTP)*
`gsiftp`	GridFTP, the Globus® -enhanced FTP protocol with security, encryption, etc. developed by The Globus Alliance [6]
`http`	ordinary *Hyper-Text Transfer Protocol (HTTP)* with PUT and GET methods using multiple streams
`https`	HTTP with SSL v3
`httpg`	HTTP with Globus® GSI
`ldap`	ordinary *Lightweight Data Access Protocol (LDAP)* [13]
`srm`	Storage Resource Manager (SRM) service [12]
`root`	Xrootd [9] protocol (available in ARC 2.0.0 and later (read-only), 4.2.0 and later (full functionality))
`rucio`	Rucio [8] – the next generation ATLAS data management system (read only, available in ARC 4.1.0 and later)
`acix`	ARC Cache Index (read only, available in ARC 4.1.0 and later)
`s3`	Amazon S3 (available in ARC 5.1.0 and later)
`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 or take advantage of various features, it can have additional options:

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

For a metadata service URL, construction is the following:

```
protocol://[url[|url[...]]@]host[:port][;option[;option[...]]]
     /lfn[:metadataoption[:metadataoption[...]]]
```

where the nested URL(s) are physical replicas. Options are passed on to all replicas, but if it is desired to use the same option with a different value for all replicas, the option can be specified as a common option using the following syntax:

```
protocol://[;commonoption[;commonoption]|][url[|url[...]]@]host[:port]
     [;option[;option[...]]/lfn[:metadataoption[:metadataoption[...]]]
```

In user-level tools, URLs may be expressed using this syntax, or there may be simpler ways to construct complex URLs. In particular, command line tools such as `arccp`, and the xRSL and JSDL job description languages provide methods to express URLs and options in a simpler way.

For the SRM service, the syntax is

```
srm://host[:port][;options]/[service_path?SFN=]file[:metadataoptions]
```

Versions 1.1 and 2.2 of the SRM protocol are supported. The default *service_path* is `srm/managerv2` when the server supports v2.2, `srm/managerv1` otherwise.

For Rucio two forms of URLs are supported:

```
1. rucio://rucio-lb-prod.cern.ch/replicas/scope/lfn
2. rucio://rucio-lb-prod.cern.ch/objectstores/s3+rucio://oshost:port/bucket/scope:lfn/RSE/operation
```

1. This URL is used to look up replicas of the given file

2. This URL is for using Rucio as a proxy for accessing objectstores. *operation* is "read" or "write", and *RSE* is the Rucio Storage Element for the objectstore with the given host, port and bucket.

The Rucio authorisation URL can be specified with the environment variable `$RUCIO_AUTH_URL`. The Rucio account to use can be specified either through the `rucioaccount` URL option or `$RUCIO_ACCOUNT` environment variable. If neither are specified the account is taken from the VOMS `nickname` attribute.

For ACIX the URLs look like

```
acix://cacheindex.ndgf.org:6443/data/index?url=http://host.org/file1
```

S3 authentication is done through keys which must be set by the environment variables `$S3_ACCESS_KEY` and `$S3_SECRET_KEY`.

The URL components are:

`host[:port]`	Hostname or IP address [and port] of a server
`lfn`	Logical File Name
`url`	URL of the file as registered in indexing service
`service_path`	End-point path of the web service
`file`	File name with full path
`option`	URL option
`commonoption`	URL option for all replicas
`metadataoption`	Metadata option

The following URL 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
`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`)
`cache=yes|no|renew|copy|check|invariant`	indicates whether the file should be cached; default for input files in A-REX is `yes`. `renew` forces a download of the file, even if the cached copy is still valid. `copy` forces the cached file to be copied (rather than linked) to the session directory, this is useful if for example the file is to be modified. `check` forces a check of the permission and modification time against the original source. `invariant` disables checking the original source modification time. (`check` option is available in ARC 2.0.0 and above, `invariant` option is available in ARC 3.0.0 and above).
`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|adler32|no`	specifies the algorithm for checksum to be computed (for transfer verification or provided to the indexing server). This is overridden by any metadata options specified (see below). If this option is not provided, the default for the protocol is used. `checksum=no` disables checksum calculation.

`overwrite=yes|no`	makes software trying (or not) to overwrite existing file(s); if `yes`, the tool will try to remove any information/content associated with the specified URL before writing to the destination.
`protocol=gsi|gssapi|ssl|tls|ssl3`	distinguishes between different kinds of `HTTPS`/`HTTPG` and `SRM` protocols. Here `gssapi` stands for `HTTPG` implementation using only GSSAPI functions to wrap data and `gsi` uses additional headers as implemented in Globus IO. The `ssl` and `tls` options stand for the usual `HTTPS` and are specifically usable only if used with the `SRM` protocol. The `ssl3` option is mostly the same as the `ssl` one but uses SSLv3 handshakes while establishing `HTTPS` connections. The default is `gssapi` for `SRM` connections, `tls` for `HTTPS` and `gssapi` for `HTTPG`. In the case of `SRM`, if default fails, `gsi` is tried.
`spacetoken=<pattern>`	specifies a space token to be used for uploads to SRM storage elements supporting SRM version 2.2 or higher
`autodir=yes|no`	specifies whether before writing to the specified location the software should try to create all directories mentioned in the specified URL. Currently this applies to FTP and GridFTP only. Default value for these protocols is `yes`
`tcpnodelay=yes|no`	controls the use of the `TCP_NODELAY` socket option (which disables Nagle's algorithm). Applies to HTTP(S) only. Default is `no` (supported only in `arcls` and other `arc*` tools)
`transferprotocol=protocols`	specifies transfer protocols for meta-URLs such as SRM. Multiple protocols can be specified as a comma-separated list in order of preference.
`rucioaccount=account`	specifies the Rucio account to use when authenticating with Rucio.
`httpputpartial=yes|no`	while storing a file on a HTTP(S) server, the software will try to send it in chunks/parts. If the server reports error for the partial PUT command, the software will fall back to transferring the file in a single piece. This behavior is non-standard and not all servers report errors properly. Hence the default is a safer `no`.
`httpgetpartial=yes|no`	while retrieving a file from a HTTP(S) server, the software will try to read it in chunks/parts. If the server does not support the partial GET command, it usually ignores requests for partial transfer range and the file is transfered in one piece. Default is `yes`.
`failureallowed=yes|no`	if set to `yes` for a job input or output file, then a failure to transfer this file will not cause a failure of the job. Default is `no`.
`relativeuri=yes|no`	if set to `yes`, HTTP operations will use the path instead of the full URL. Default is `no`.

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

URLs also support metadata options which can be used for registering additional metadata attributes or querying the service using metadata attributes. These options are specified at the end of the LFN and consist of name and value pairs separated by colons. The following attributes are supported:

`checksumtype`	Type of checksum. Supported values are `cksum` (default), `md5` and `adler32`
`checksumvalue`	The checksum of the file

The checksum attributes may also be used to validate files that were uploaded to remote storage.

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
file:///home/auser/griddir/steer.cra
srm://srm.domain.org/griddir/user/file1:checksumtype=adler32:checksumvalue=12345678[1]
srm://srm.domain.org;transferprotocol=https/data/file2[2]
```

[1]This is a destination URL. The file will be copied to `srm.domain.org` at the path `griddir/user/file1` and the checksum will be compared to what is reported by the SRM service after the transfer.

[2]This is a source or destination URL. When getting a transport URL from SRM, the HTTPS transfer protocol will be requested.

Chapter 3

Attributes

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

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

It is possible to use unsupported attributes in job description. Standard ARC client submission commands (`arcsub` and `arcresub`) must be used with a command line option "-U" in order to accept unknown attributes. Without this command line option, the client tool will concider job description invalid if it contains unsupported attributes.

3.1 User-side attributes

The following attributes can be specified in a user's xRSL script. Some have to be modified by the client tool before being passed to the GM. If this is the case, the corresponding modified GM input is described in this document as well.

executable

Unique:	yes
Operators:	`=`
User input:	`(executable=<string>)`
GM input:	`-"-`
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 ARC Client.

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 xRSL string is entered from the command line and is enclosed in double quotes, standard shell expansion of variables takes place. That is, if the file name contains an environment variable ("`$...`"), the value of this variable is resolved locally, but if the name itself is also enclosed in double quotes, it will be resolved at the remote computing element:
`(executable=$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

arguments

Unique: yes
Operators: `=`
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 ARC Client from the user-specified `executable` attribute

inputFiles

Unique: yes
Operators: `=`
User input: `(inputFiles=(<filename> <source> [option] ... ) ... )`
GM input:
```
(inputFiles=(<filename> <URL>)
        (<filename> [size][.checksum]) ... )
```
Example:
```
(inputFiles=("local_to_job" "gsiftp://se1.lu.se/p1/remote.1" "threads=5")
        ("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 job execution.

`filename`	destination file name, local to the computing element and always relative to the session directory
`source`	source of the file: (remote URLs, or a path, local to the submission node). If void (""", use the quotes!), the input file is taken from the submission directory.
`option`	(ARC $\geq$ 1.0) URL options for source. See Section 2.2 for possible values. The ARC Client converts `source` and any options given here to a URL with the syntax described in Section 2.2.
`URL`	URL of the file (see Section 2.2)
`size`	file size in bytes
`checksum`	file checksum (as returned by `cksum`)

If the `inputFiles` list does not contain the standard input file (as specified by `stdin`) and/or the executable file (as specified by `executable`), an ARC client must append these files to the list. If the `<source>` is a URL, any options given by `option` are added to it, then it is passed by the ARC Client to the GM as shown in the example above. GM recognizes all URLs except `file:///`.

Internally, the client must forward the `(<filename> <source> [option] ...)` request to the execution service without changes, unless `<source>` is a local path, void ("") or `file:///`. In case `<source>` is a local path, void ("") or `file:///`, the client must extract file size and checksum, and substitute the `<source>` string with `[size][.checksum]`. In the unlikely case when it is impossible to extract file size, the `<source>` string must be substituted by a void one ("").

> Please note that the `inputFiles` attribute is not meant to operate with directories, for reasons of access control and checksum verifications. You must specify a pair `("<local_to_job>" "<source>")` for each file.

executables

Unique:	yes
Operators:	`=`
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 if relative to the session directory) is not in the `executables` list, it will be added to the list by the ARC Client.

cache

Unique:	yes
Operators:	`=`
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 affects all input files, even those described by `executables`.

If not specified, default value is "yes".

> Cached files can not be modified by jobs by default. If your job has to modify input files, please use the `(readonly="no")` URL option for those files. This option does not affect whether or not the file is cached.

outputFiles

Unique:	yes
Operators:	`=`
User input:	`(outputFiles=(<string> <URL> [option] ... ) ... )`
GM input:	`(outputFiles=(<string> <URL>) ... )`
Example:	`(outputFiles=("local_to_job.dat" "gsiftp://se1.uio.no/stored.dat")`
	`("local_to_job_dir/" ""))`

List of files to be retrieved by the user or uploaded by the GM and optionally indexed (registered) in a data indexing service.

`string` file name, local to the *Computing Element (CE)*. If this string ends with a backslash "/" and `<URL>` is empty, the entire directory will be kept at the execution site. If however this string ends with a backslash "/" but the `<URL>` is a remote location, the contents of the directory are transferred to the destination.

`URL` destination 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.

`option` (ARC $\geq$ 1.0) URL options for destination URL. See Section 2.2 for possible values. When the destination is an indexing service, a physical file location may be specified by the additional option "location". This option can be given multiple times. The CE will attempt to upload the file to the specified locations in the order they are given until one succeeds. Options specified after a location option only affect that location. Before passing to the GM, the ARC Client adds to `URL` any options and locations given here, using the syntax described in Section 2.2.

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 ARC Client appends these items to the `outputFiles` list. If the `<URL>` is not specified (void, ""), use the quotes!), files will be kept on the CE and should be downloaded by the user via the ARC Client. 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=("/" ""))`.

In some cases, the list of output files may only be known after the job has completed. ARC allows a user to specify a list of output files dynamically in a file or files in the session directory as part of their job. The file(s) containing the output file information can be defined in the xRSL script as the path to the file relative to the session directory preceeded by '@'. The format of these files is lines of 2 values separated by a space. The first value contains name of the output file relative to the session directory and the second value is a URL to which the file will be uploaded.

Example: `(outputFiles=("@output.files" "")`

output.files is generated by the user and contains

```
file1 gsiftp://grid.domain.org/file1
file2 gsiftp://grid.domain.org/file2
```

After the job completes, the file output.files in the session directory will be read and any files described within will be uploaded to the given URLs.

cpuTime

Unique: yes

Operators: `=`

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

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

Example: `(cpuTime="240")`

Maximal CPU time request for the job. For a multi-processor job, this is a sum over all requested processors.

`time` time (in minutes if no unit is specified)

`tttt` time converted by the ARC Client from `time` to seconds.

The client converts time specified in the user-side XRSL file to seconds. If no time unit is specified, the client assumes the time given in minutes. Otherwise, a text 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 ARC Client converts them both. `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).

wallTime

Unique:	yes
Operators:	`=`
User input:	`(wallTime=<time>)`
GM input:	`(wallTime=<tttt>)`
Example:	`(wallTime="240")`

Maximal wall clock time request for the job.

`time`	time (in minutes if no unit is specified)
`tttt`	time converted by the ARC Client to seconds

The client converts time specified in the user-side XRSL file seconds. If no time unit is specified, the client assumes the time given in minutes. Otherwise, a text 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 ARC Client converts them both. `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 ARC Client 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

Unique: yes
Operators: `=`
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 (in minutes if no unit is specified)

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 ARC Client and added to the job description.

benchmarks

Unique: yes
Operators: `=`
User input: `(benchmarks=(<string> <value> <time>) ... )`
GM input:
Example: `(benchmarks=("mybenchmark" "10" "1 hour, 30 minutes"))`

Evaluate a job's `cpuTime` based on benchmark values.

`string` benchmark name
`value` benchmark value of reference machine
`time` the `cpuTime` the job requires on the reference machine

`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 ARC Client and added to the job description.

memory

Unique: yes
Operators: `=`
User input: `(memory=<integer>)`
GM input: `-"-`
Example: `(memory>="500")`

Memory required for the job, per count for parallel jobs.

`integer` size (Mbytes)

Similarly to `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

Unique: no
Operators: `=` `!=` `>` `<` `>=` `<=`
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

Unique: no
Operators: `=` `!=` `>` `<` `>=` `<=`
User input: `(runTimeEnvironment=<string>)(runTimeEnvironment=<string>)`
GM input: only `=` is allowed
Example: `(runTimeEnvironment>="APPS/HEP/ATLAS-10.0.1")`

Required runtime environment.

`string` environment name

The site to submit the job to will be chosen by the ARC Client 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.

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

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.

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 and use "`>`" or "`>=`" operators to select the highest version.

middleware

Unique: no
Operators: = != > < >= <=
User input: (middleware=<string>)
GM input: only = is allowed
Example: (middleware="nordugrid-arc-0.5.99")

Required middleware version. Make sure to specify full name and version number.

`string` Grid middleware name.

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

opsys

Unique: no
Operators: = != > < >= <=
User input: (opsys=<string>)
GM input: only = is allowed
Example: (opsys="FC3")

Required operating system.

`string` Operating system name and version.

The site to submit the job to will be chosen by the ARC Client 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

Unique: yes
Operators: =
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 ARC Client.

stdout

Unique: yes
Operators: =
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 ARC Client. If the standard output is not defined, ARC Client assigns a name.

stderr

Unique: yes
Operators: `=`
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 ARC Client. If the standard error is not defined, ARC Client assigns a name. If `join` is specified with value "yes", ARC Client adds `stderr` to the pre-processed xRSL script with the same value as `stdout`.

join

Unique: yes
Operators: `=`
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

Unique: yes
Operators: `=`
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 (ARC Client forces it to the list if `outputFiles`)

jobName

Unique: yes
Operators: =
User input: (jobName=<string>)
GM input: -"-
Example: (jobName="My Job nr. 1")

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 ARC Client. It is also available through the Information System.

ftpThreads

Unique: yes
Operators: =
User input: (ftpThreads=<integer>)
GM input: -"-
Example: (ftpThreads="4")

Defines how many parallel streams will be used by the GM during `gsiftp` and `http(s|g)`transfers of files.

`integer` a number from 1 to 10

If not specified, parallelism is not used.

acl

Unique: no
Operators: =
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 [11] 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

queue

Unique: yes
Operators: = !=
User input: (queue=<string>)
GM input: only = is allowed
Example: (queue="pclong")

The name of the remote batch queue.

> Use only when you are sure that the queue by this name does exist.

`string` known queue name

While users are not expected to specify `queue` in job descriptions, this attribute **must** be present in the GM-side xRSL. In fact, this is primarily an internal attribute, added to the job description by client tools after resource discovery and matchmaking. Still, users can specify this attribute to explicitly force job submission to a queue: when specified explicitly by the user, this value will not be overwritten by the ARC Client, and an attempt will be made to submit the job to the specified queue.

If for some reason (e.g. due to a client tool error) `queue` is absent from the GM-side xRSL, GM on the selected cluster will attempt to submit the job to the default queue if such is specified in the GM configuration.

startTime

Unique: yes
Operators: =
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 ARC Client from `time`)

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

lifeTime

Unique: yes
Operators: =
User input: (lifeTime=<time>)
GM input: (lifeTime=<tttt>)
Example: (lifeTime="2 weeks")

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

`time` time (in minutes if no unit is specified)

`tttt` time (seconds, converted by the ARC Client from `time`)

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

notify

Unique: yes

Operators: `=`

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][d] 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)

When no notification flags are specified, default value of "`eb`" will be used, i.e., notifications will be sent at the job's beginning and at its end.

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

rerun

Unique: yes

Operators: `=`

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 5. The job may be rerun after failure in any state for which reruning has sense. To initiate rerun user has to use the `arcresume` command.

architecture

Unique: no

Operators: `=` `!=`

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

GM input: none

Example: `(architecture="i686")`

Request a specific architecture.

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

nodeAccess

Unique: yes
Operators: `=`
User input: `(nodeAccess="inbound"|"outbound")`
GM input: none
Example: `(nodeAccess="inbound")`

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

dryRun

Unique: yes
Operators: `=`
User input: `(dryRun="yes"|"no")`
GM input: `-"-`
Example: `(dryRun="yes")`

If "yes", do dry-run: job description is sent to the optimal destination, input files are transferred, but no actual job submission to LRMS is made. Typically used for xRSL and communication validation.

rsl_substitution

Unique: no
Operators: `=`
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 define variables that simplify xRSL editing, e.g. when same path is used in several values, typically in `inputFiles`. 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`.

After the substitution is defined, it should be used in a way similar to shell variables in scripts: enclosed in round brackets, preceded with a dollar sign, **without quotes**:
`(inputfiles=("myfile" $(ATLAS)/data/somefile))`

Unlike the `environment` attribute, `rsl_substitution` definition is only used by the client and is valid inside xRSL script. It can not be used to define environment or shell variable at the execution site.

environment

Unique: no
Operators: =
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. Unlike the rsl_substitution attribute, it can not be used to define variables on the client side.

count

Unique: yes
Operators: =
User input: (count=<integer>)
GM input: -"-
Example: (count="4")

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

countpernode

(ARC $\geq$ 3.0)

Unique: yes
Operators: =
User input: (countpernode=<integer>)
GM input: -"-
Example: (countpernode="2")

Specifies amount of sub-jobs per node to be submitted for parallel tasks. Note: The count attribute must be specified when this attribute is specified.

exclusiveexecution

(ARC $\geq$ 3.0)

Unique: yes
Operators: =
User input: (exclusiveexecution="yes"|"no")
GM input: -"-
Example: (exclusiveexecution="yes")

Specifies whether the node should be allocated for exclusive use by the job.

jobreport

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

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

`URL` URL

It is up to users to make sure the requested accounting service accepts reports from the set of clusters they intend to use.

credentialserver

Unique:	yes
Operators:	=
User input:	(credentialserver=<URL>)
GM input:	-"-
Example:	(credentialserver="myproxy://myproxy.nordugrid.org;username=user")

Specifies an URL which Grid Manager may contact to renew/extend delegated proxy of job. Only MyProxy servers are supported.

`URL` URL of MyProxy server

It is up to a user to make sure the specified MyProxy server will accept requests from Grid Manager to renew expired credentials. `URL` may contain options `username` and `credname` to specify user name and credentials name which Grid Manager should pass to MyProxy server. If `username` is not specified DN of user credentials is used instead.

priority

Unique:	yes
Operators:	=
User input:	(priority=<integer>)
GM input:	-"-
Example:	(priority="80")

Specifies priority given to this job during staging of input and output files when the new data staging framework is used by A-REX. Values are limited to between 1 (lowest priority) and 100 (highest priority). Default if this attribute is not specified is 50.

3.2 GM-side attributes

The following attributes are a part of the internal ARC job representation, and must be provided by ARC client tools and passed to the GM. Users may even specify them manually (**not advised!**), while developers of new ARC client tools and utilities must make sure these attributes are added to the user job description before it is submitted to a GM.

sstdin

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

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

Example: `(action="request")`

Action to be taken by the gatekeeper: submit the job, cancel job execution, clear the results of the job (also cancels the job), renew the proxy of the job, or restart the job from a previous failed state.

savestate

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

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

Example: `(lrmstype="pbs")`

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

`string` LRMS type

hostName

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

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

Name (e.g. as returned by the Linux `hostname` command) of the client machine from which the submission was made.

`string` client host name, as passed by the ARC client

jobid

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

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

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, eiher an error or a warning should be issued by the client tool, and the corresponding attribute should be 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)`

*Standard ARC client will throw an error, unless option "-U" is specified

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>)

Chapter 4

XRSL and JSDL

ARC has adopted JSDL [3] as second format for describing job requests*. ARC also adds own extensions to JSDL. Below supported JSDL elements are listed together with corresponding XRSL attributes. Here *jsdl* stands for jsdl namespace, *jsdlPOSIX* – for POSIX extensions, and *jsdlARC* – for ARC extensions.

XRSL attribute	**JSDL element**	**Comments**
executable	jsdlPOSIX:Executable	
cpuTime	jsdlPOSIX:CPUTimeLimit jsdl:Resources jsdl:TotalCpuTime jsdl:IndividualCpuTime	
arguments	jsdlPOSIX:Argument	
inputFiles	jsdl:DataStaging jsdl:Source jsdl:FileName jsdl:URI jsdl-arc:URIOption	jsdl:DataStaging without URI in Source are treated like user-uploadable files
outputFiles	jsdl:DataStaging jsdl:Target jsdl:FileName jsdl:URI jsdl-arc:URIOption jsdl-arc:Location jsdl:URI	jsdl:DataStaging without Target and Source or without URI in Target are treated like user-downloadable files
wallTime	jsdlPOSIX:WallTimeLimit	
memory	jsdlPOSIX:MemoryLimit jsdl:Resources jsdl:TotalPhysicalMemory jsdl:IndividualPhysicalMemory	
disk	jsdl:Resources jsdl:TotalDiskSpace jsdl:IndividualDiskSpace	
stdin	jsdlPOSIX:Input	
stdout	jsdlPosix:Output	
stderr	jsdlPOSIX:Error	
jobName	jsdl:JobIdentification	
architecture	jsdl:Resources jsdl:CPUArchitectureName jsdl:CPUArchitecture	

*Available in releases $\geq$ 0.5.37, ARC 0.6, ARC 0.8

count	jsdl:Resources jsdl:TotalCpuCount jsdl:IndividualCpuCount	
executables	jsdl:DataStaging jsdl-arc:IsExecutable	
gridTime	jsdl:Resources jsdl-arc:GridTimeLimit	
runTimeEnvironment	jsdl:Resources jsdl-arc:RunTimeEnvironment	Only "Exact" condition with one Version is supported so far
middleware	jsdl:Resources jsdl-arc:Middleware	Only "Exact" condition with one Version is supported so far
gmlog	jsdl-arc:LocalLogging jsdl-arc:Directory	
acl	jsdl-arc:AccessControl Type=GACL Content	
cluster	jsdl:Resources jsdl-arc:CandidateTarget jsdl-arc:HostName	
queue	jsdl:Resource jsdl-arc:CandidateTarget jsdl-arc:QueueName	
startTime	jsdl-arc:ProcessingStartTime	
lifeTime	jsdl:Resources jsdl-arc:SessionFileTime	
notify	jsdl-arc:Notify jsdl-arc:Type=Email	
rerun	jsdl-arc:Reruns	
jobreport	jsdl-arc:RemoteLogging jsdl-arc:URL	
credentialserver	jsdl-arc:CredentialServer jsdl-arc:URL	

Conditional requests are supported through recursive jsdl:JobDescription - jsdl:JobDescriptin inside jsdl:JobDescription.

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/johndoe"))
     (rsl_substitution=("NGTEST" $(TOPDIR)/ngtest))
     (rsl_substitution=("BIGFILE" "/scratch/johndoe/100mb.tmp"))
(* some environment variables, to be used by the job *)
     (environment=("ATLAS" "/opt/atlas") ("CERN" "/cern"))
(* the main executable file to be staged in and submitted to the PBS *)
     (executable="checkall.sh")
(* the arguments for the executable above *)
     (arguments="pal")
(* files to be staged in before the execution *)
     (inputFiles = ("be_kaons" "")
     ("file1" gsiftp://grid.uio.no$(TOPDIR)/remfile.txt)
      ("bigfile.dat" $(BIGFILE) )    )
(* files to be given executable permissions after staging in *)
     (executables="be_kaons")
(* files to be staged out after the execution *)
     (outputFiles=
        ("file1" "gsiftp://grid.tsl.uu.se/tmp/file1.tmp")
        ("100mb.tmp" "rls://rls.nordugrid.org:39281/test/bigfile")
        ("be_kaons.hbook"  gsiftp://ce1.grid.org$(NGTEST)/kaons.hbook)   )
(* 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 john.doe@gmail.com jane.doe@mail.org")
(* 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, per count, 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="APPS/HEP/Atlas-1.1")
(* number of re-runs, in case of a system failure *)
     (rerun="2")
```

A.2 GM-side xRSL script

Note that a client tool must do matchmaking and modify correspondingly the job document before submitting it to the matching resource. Specifically, a client tool has to:

- expand all the `rsl_substitution` values
- add double quotes to **all** strings
- insert `queue` attribute in case such is missing
- make sure every logical or comparison operator is expanded and replaced with a deterministic "=" statement
- streamline blank spaces

Comment lines are removed from the GM-side xRSL; below they are shown to explain details.

```
&
(* saves RSL in a temporary file if "yes" *)
     ("savestate" = "yes" )
(* job submission to be performed if action is "request" *)
     ("action" = "request" )
(* submission host name *)
     ("hostname" = "ce1.grid.org" )
(* client software version *)
     ("clientsoftware" = "nordugrid-arc-0.6.0.3" )
(* walltime value added by the client, in seconds *)
     ("walltime" = "3600" )
(* test run: if "yes", only submits RSL without actual job start *)
     ("dryRun" = "no" )
(* some local variables defined for further convenience *)
     ("rsl_substitution" = ("TOPDIR" "/home/johndoe" ) )
     ("rsl_substitution" = ("NGTEST" "/home/johndoe/ngtest" ) )
     ("rsl_substitution" = ("BIGFILE" "/scratch/johndoe/100mb.tmp" ) )
(* some environment variables, to be used by the job *)
     ("environment" = ("ATLAS" "/opt/atlas" ) ("CERN" "/cern" ) )
(* executable *)
     ("executable" = "checkall.sh" )
(* arguments *)
     ("arguments" = "pal" )
(* files to be staged in before the execution *)
     ("inputfiles" = ("checkall.sh" "279320" )
        ("myinput.dat" "39806" )
        ("be_kaons" "8807" )
        ("file1" "gsiftp://grid.uio.no/home/johndoe/remfile.txt" )
        ("bigfile.dat" "104857600" )
     )
(* 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" "rls://rls.nordugrid.org:39281/test/bigfile" )
        ("be_kaons.hbook" "gsiftp://ce1.grid.org/home/johndoe/ngtest/kaons.hbook" )
        ("myoutput.dat" "" )
        ("myerror.dat" "" )
    )
(* 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 john.doe@gmail.com jane.doe@mail.org" )
(* specific queue to submit the job *)
     ("queue" = "atlas" )
(* CPU time required for the job, converted into seconds *)
     ("cputime" = "3600" )
(* maximal time for the session directory to exist on the remote node, seconds *)
     ("lifetime" = "604800" )
(* memory required for the job, per count, 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" = "APPS/HEP/Atlas-1.1" )
(* number of re-runs, in case of a system failure *)
     ("rerun" = "2" )
(* original client xRSL with expanded string substitutions; shortened here *)
     ("clientxrsl" = "&(""dryrun"" = ""no"" )(... )(""rerun"" = ""2"" )" )
```

A.3 JSDL script

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition
  xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl"
  xmlns:posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
  xmlns:arc="http://www.nordugrid.org/ws/schemas/jsdl-arc">
  <JobDescription>
    <!-- Generic job description (jobname in XRSL) -->
    <JobIdentification>
        <JobName>JSDL Hello World job</JobName>
        <Description>
          This is a simple Hello World job test
        </Description>
    </JobIdentification>
    <!-- Standard application (same as executable, arguments in XRSL) -->
    <Application>
        <ApplicationName>Echo</ApplicationName>
        <posix:POSIXApplication>
```

```
          <posix:Executable>/bin/echo</posix:Executable>
          <posix:Argument>Hello world</posix:Argument>
        </posix:POSIXApplication>
    </Application>
    <!-- Get the GM diagnostics (same as gmlog in XRSL) -->
    <arc:LocalLogging>
      <arc:Directory>gmlog</arc:Directory>
    </arc:LocalLogging>
 </JobDescription>
</JobDefinition>
```

Appendix B

JSDL-ARC schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.nordugrid.org/ws/schemas/jsdl-arc"
    xmlns:jsdl-arc="http://www.nordugrid.org/ws/schemas/jsdl-arc"
    targetNamespace="http://www.nordugrid.org/ws/schemas/jsdl-arc">

 <xsd:simpleType name="GMState_Type">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="ACCEPTED"/>
   <xsd:enumeration value="PREPARING"/>
   <xsd:enumeration value="SUBMIT"/>
   <xsd:enumeration value="INLRMS"/>
   <xsd:enumeration value="FINISHING"/>
   <xsd:enumeration value="FINISHED"/>
   <xsd:enumeration value="DELETED"/>
   <xsd:enumeration value="CANCELING"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:complexType name="Version_Type">
  <xsd:sequence>
   <xsd:element name="UpperExclusive" type="xsd:string" minOccurs="0"/>
   <xsd:element name="LowerExclusive" type="xsd:string" minOccurs="0"/>
   <xsd:element name="Exact" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="Exclusive" type="xsd:boolean" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:simpleType name="SessionType_Type">
  <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
  <!-- xsd:element ref="SessionType" minOccurs="0"/ -->
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="INTERNAL"/>
   <xsd:enumeration value="LIMITED"/>
   <xsd:enumeration value="READONLY"/>
   <xsd:enumeration value="FULL"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="IsExecutable_Type">
  <xsd:documentation> For jsdl:DataStaging_Type (default: false) </xsd:documentation>
  <!-- xsd:element ref="IsExecutable" minOccurs="0"/ -->
  <xsd:restriction base="xsd:boolean"/>
 </xsd:simpleType>
```

```
<xsd:simpleType name="FileParameters_Type">
 <xsd:documentation> For jsdl:DataStaging_Type </xsd:documentation>
 <!-- xsd:element ref="IsExecutable" minOccurs="0"/ -->
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="JoinOutputs_Type">
 <xsd:documentation> For jsdl:JobDescription_Type (default: false) </xsd:documentation>
 <!-- xsd:element ref="JoinOutputs" minOccurs="0"/ -->
 <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>

<xsd:simpleType name="Reruns_Type">
 <xsd:documentation> For jsdl:JobDescription_Type (default: false) </xsd:documentation>
 <!-- xsd:element ref="Reruns" minOccurs="0"// -->
 <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>

<xsd:complexType name="RunTimeEnvironment_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="RunTimeEnvironment" minOccurs="0" maxOccurs="unbounded"/ -->
 <xsd:sequence>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Version" type="Version_Type" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Middleware_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="Middleware" minOccurs="0" maxOccurs="unbounded"/ -->
 <xsd:sequence>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Version" type="Version_Type" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="RemoteLogging_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="RemoteLogging" minOccurs="0" maxOccurs="3"/ -->
 <xsd:sequence>
  <xsd:element name="URL" minOccurs="1" maxOccurs="1" type="xsd:anyURI"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CredentialServer_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="CredentialServer" minOccurs="0"/ -->
 <xsd:sequence>
  <xsd:element name="URL" minOccurs="1" maxOccurs="1" type="xsd:anyURI"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="LocalLogging_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="LocalLogging" minOccurs="0" maxOccurs="1"/ -->
 <xsd:sequence>
  <xsd:element name="Directory" minOccurs="1" maxOccurs="1" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AccessControlType_Type">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="GACL"/>
```

```
 </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="AccessControl_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="AccessControl" minOccurs="0"/ -->
 <xsd:sequence>
  <xsd:element name="OwnerAlwaysAllowed" type="xsd:boolean" minOccurs="0"/>
  <xsd:element name="Type" type="AccessControlType_Type" minOccurs="0"/>
  <xsd:element name="Content" minOccurs="0" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="NotificationType_Type">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="Email"/>
 </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="Notify_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="Notify" minOccurs="0" maxOccurs="3"/ -->
 <xsd:sequence>
  <xsd:element name="Type" type="NotificationType_Type" minOccurs="0"/>
  <xsd:element name="Endpoint" minOccurs="0" type="xsd:string"/>
  <xsd:element name="State" minOccurs="1" maxOccurs="unbounded" type="GMState_Type"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SessionLifeTime_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="SessionLifeTime" minOccurs="0" maxOccurs="1"/ -->
 <xsd:restriction base="xsd:long"/>
</xsd:simpleType>

<xsd:simpleType name="GridTimeLimit_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="GridTimeLimit" minOccurs="0" maxOccurs="1"/ -->
 <xsd:restriction base="xsd:positiveInteger"/>
</xsd:simpleType>

<xsd:complexType name="CandidateTarget_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="jsdl-arc:CandidateTarget" minOccurs="0" maxOccurs="1"/ -->
 <xsd:sequence>
  <xsd:element name="HostName" minOccurs="0" type="xsd:string"/>
  <xsd:element name="QueueName" minOccurs="0" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="Time_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="ProcessingStartTime" minOccurs="0" maxOccurs="1"/ -->
 <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>

<!--=======================================================-->
<xsd:element name="IsExecutable" type="IsExecutable_Type"/>
<xsd:element name="FileParameters" type="FileParameters_Type"/>
<xsd:element name="RunTimeEnvironment" type="RunTimeEnvironment_Type"/>
<xsd:element name="Middleware" type="Middleware_Type"/>
<xsd:element name="RemoteLogging" type="RemoteLogging_Type"/>
<xsd:element name="LocalLogging" type="LocalLogging_Type"/>
<xsd:element name="AccessControl" type="AccessControl_Type"/>
<xsd:element name="Notify" type="Notify_Type"/>
```

```
 <xsd:element name="SessionLifeTime" type="SessionLifeTime_Type"/>
 <xsd:element name="SessionType" type="SessionType_Type"/>
 <xsd:element name="JoinOutputs" type="JoinOutputs_Type"/>
 <xsd:element name="Reruns" type="Reruns_Type"/>
 <xsd:element name="CredentialServer" type="CredentialServer_Type"/>
 <xsd:element name="GridTimeLimit" type="GridTimeLimit_Type"/>
 <xsd:element name="CandidateTarget" type="CandidateTarget_Type"/>
 <xsd:element name="ProcessingStartTime" type="Time_Type"/>

</xsd:schema>
```

Acknowledgements

This work was supported in parts by: the Nordunet 2 programme, the Nordic DataGrid Facility, the EU KnowARC project (Contract nr. 032691) and the EU EMI project (Grant agreement nr. 261611).

Bibliography

[1] The NorduGrid Collaboration. Web site. URL `http://www.nordugrid.org`.

[2] The Globus Resource Specification Language RSL v1.0. URL `http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html`.

[3] A. Anjomshoaa et al. Job Submission Description Language (JSDL) Specification, Version 1.0 (first errata update). GFD-R.136, July 2008. URL `http://www.gridforum.org/documents/GFD.136.pdf`.

[4] M. Ellert. *ARC User Interface*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/ui.pdf`. NORDUGRID-MANUAL-1.

[5] M. Ellert, M. Grønager, A. Konstantinov, et al. Advanced Resource Connector middleware for lightweight computational Grids. *Future Gener. Comput. Syst.*, 23(1):219–240, 2007. ISSN 0167-739X. doi: 10.1016/j.future.2006.05.008.

[6] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997. Available at: http://www.globus.org.

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

[8] V. Garonne, R. Vigne, G. Stewart, M. Barisits, T. Beermann, M. Lassnig, C. Serfon, L. Goossens, A. Nairz, and the Atlas Collaboration. Rucio – the next generation of large scale distributed system for atlas data management. *Journal of Physics: Conference Series*, 513(4):042021, 2014. URL `http://stacks.iop.org/1742-6596/513/i=4/a=042021`.

[9] A. Hanushevsky, A. Dorigo, and F. Furano. The Next Generation Root File Server. In J.Harvey A.Aimar and N.Knoors, editors, *Proc. of CHEP 2004, CERN-2005-002 vol.2*, page 680, 2005.

[10] A. Konstantinov. *The NorduGrid Grid Manager And GridFTP Server: Description And Administrator's Manual*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/GM.pdf`. NORDUGRID-TECH-2.

[11] A. McNab. The GridSite Web/Grid security system: Research Articles. *Softw. Pract. Exper.*, 35(9): 827–834, 2005. ISSN 0038-0644.

[12] A. Sim, A. Shoshani, et al. The Storage Resource Manager Interface (SRM) Specification v2.2. GFD-R-P.129, May 2008. URL `http://www.ggf.org/documents/GFD.129.pdf`.

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

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in roman refer to the pages where the entry is used.