NORDUGRID

NORDUGRID-MANUAL-1

6/11/2009

ARC User Interface

User's Manual

Contents

Chapter 1

Introduction

The command line user interface of ARC consists of a set of commands necessary for job submission and manipulation and data management. A special utility also exists for test purposes. Several third-party commands (Globus [1], VOMS [2]) are available as well. All these are described in the following sections.

Chapter 2

Commands

2.1 Job submission and management

The following commands are used for job submission and management, such as status check, results retrieval, cancellation, re-submission and such. The jobs must be described using a special language: xRSL or JSDL [3].

2.1.1 ngsync

It is advised to start every grid session by running `ngsync`, especially when changing workstations. The reason is that your job submission history is cached on your machine, and if you are using ARC client installations on different machines, your local lists of submitted jobs will be different. To synchronise these lists with the information in the Information System, use the `ngsync` command.

ngsync [options]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-f, -force`		don't ask for confirmation
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

The ARC ARC client keeps a local list of jobs in the user's home directory (see section 2.1.11). If this file is lost, corrupt, or the user wants to recreate the file on a different workstation, the `ngsync` command will recreate this file from the information available in the Information System.

Since the information about a job in the system can be slightly out of date in case the user submitted or removed a job very recently, a warning is issued when this command is run. The `-f` option disables this warning.

As the Information System has a certain latency, do not use `ngsync` immediately after submitting or killing a job, first wait a few minutes.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.2 ngsub

The `ngsub` command is the most essential one, as it is used for submitting jobs to the Grid resources. . `ngsub` matches user's job description to the information collected from the Grid, and the optimal site is being selected for job submission. The job description is then being forwarded to that site, in order to be submitted to the Local Resource Management System (LRMS), which can be, e.g., PBS or Condor or SGE etc.

ngsub [options] <task ...>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server (GIIS)
`-G, -giislist`	*filename*	list of GIIS URLs
`-e, -xrsl`	*filename*	string describing the job to be submitted
`-f, -file`	*filename*	file describing the job to be submitted
`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the job description
`-dumpxrsl`		do not submit – dump transformed job description to stdout
`-U, -unknownattr`		allow unknown attributes in job description – pass them through
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`task ...`	strings or files describing the jobs to be submitted

Option `-e` was introduced in ARC $\geq$ 0.5.49:
– in ARC $\leq$ 0.5.48, strings describing the job should not be pre-pended with option `-e`
– in ARC $\geq$ 0.5.49, files describing the job don't have to be pre-pended with option `-f`

A simple *"Hello World"* job would look like[*]:

```
    ngsub -e '&(executable="/bin/echo")(arguments="Hello
World")(stdout="hello.txt")'
```

Use single quotes for the xRSL string and double quotes for attribute values.

Such a request would submit the task to any available site, as there are no specific requirements specified. The job will be executed in a batch mode, and the standard output will be written to a file `hello.txt` at the execution site. You will have to retrieve this file manually, using the `ngget` command.

If a job is successfully submitted, a **job identifier** (*job ID*) is printed to standard output.

The job ID uniquely identifies the job while it is being executed. A typical job ID looks like follows:

```
    gsiftp://site.it.uni.org:2812/jobs/10308913211503407485
```

You should use this as a handle to refer to the job when doing other job manipulations, such as querying job status (`ngstat`), killing it (`ngkill`), re-submitting (`ngresub`), or retrieving the result (`ngget`).

Every job ID is a valid URL for the job session directory. You can always use it to access the files related to the job, by using data management tools (see Chapter 2.2).

The job description in xRSL or JSDL format can be given either as an argument on the command line, as in the example above, or can be **read from a file**[†]. Several jobs can be requested at the same time by giving more than one filename argument, or by repeating the `-f` or `-e` options. It is possible to mix `-e` and `-f` options in the same `ngsub` command.

To **validate** your job description without actually submitting a job, use the `-dryrun` option: it will capture possible syntax or other errors, but will instruct the site not to submit the job for execution.

If the `-o` option is given, the job identifier is also written to a file with the specified filename. This file can later be used with the corresponding `-i` option of the other job manipulating ARC client commands.

```
    ngsub -o my_jobid_list myjob.xrsl
    ngkill -i my_jobid_list
```

The `-c` option can be used to **force** a job to be submitted to a particular site (cluster), or to reject submission to a site. The matching is done by string match to the site name (i.e. hostname) as defined in the Information System, or to an alias, as defined in user configuration file (see Section 4). The `-c` option can be repeated several times, for example:

```
    ngsub -c grid.nbi.dk -c grid.tsl.uu.se myjob.xrsl
```

This will submit a job to either `grid.nbi.dk` or `grid.tsl.uu.se`. To submit a job to any site except `badsite.abc.org`, use `-` sign in front of the name:

```
    ngsub -c -badsite.abc.org myjob.xrsl
```

For convenience, you may list the sites in a file, and use the `-C` option to refer to the whole list:

```
    ngsub -C preferred_sites myjob.xrsl
```

[*]When using ARC $\leq$ 0.5.48 client, omit `-e`

[†]When using ARC $\leq$ 0.5.48, one must prepend file name with `-f`

If a cluster name or file name is preceded with a minus sign ("-"), this site (or the list) will be avoided during the submission. This gives a possibility to blacklist unwanted sites:

```
ngsub -C -blacklist myjob.xrsl
```

The `ngsub` command locates the available sites by querying the Information System. By default, a list of the ARC Information System **servers** (GIIS) is distributed with the middleware and is stored in `$NORDUGRID_LOCATION/etc/giislist`. A user is free to choose any other set of GIIS servers, either by listing them in the configuration file[‡], or by specifying them via `-g` option:

```
ngsub -g ldap://hostname[:port]/basedn myjob.xrsl
```

If the port number is omitted, the default 2135 is used. Two different syntax for the base DN of the LDAP query are accepted, e.g. a NorduGrid top-level GIIS can be specified in either of the two following ways:

```
ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index1.nordugrid.org:2135/mds-vo-name=NorduGrid,O=Grid
```

Several GIISes can be specified by repeating the `-g` option:

```
ngsub -g ldap://url1 -g ldap://url2 ...
```

You may prefer to store the list of GIIS servers in a file other than the default one, and use the `-G` option to instruct `ngsub` to contact those servers instead:

```
ngsub -G my_GIIS_list myjob.xrsl
```

Note that LDAP URLs containing "Mds-Vo-name=local" refer not to GIISes, but to individual sites, for example:
`ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local`
This feature can be used to add "hidden" sites that do not register to any GIIS, or to list explicitly preferred submission sites.

If neither the `-giisurl` nor the `-giislist` option is given, a list of GIIS URLs is read from the first existing source in the following list:

1. `giis` options in `client` section in `$HOME/.arc/client.conf`
2. `giis` options in `common` section in `$HOME/.arc/client.conf`
3. `$HOME/.nggiislist` (one GIIS per line, ARC $\leq$ 0.5.48)
4. `giis` options in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `giis` options in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `$ARC_LOCATION/etc/giislist` (one GIIS per line)
7. `giis` options in `client` section in `/etc/arc.conf`
8. `giis` options in `common` section in `/etc/arc.conf`
9. `/etc/giislist` (one GIIS per line)

If you would like to get diagnostics of the process of resource discovery and requirements matching, a very useful option is `-d`. The following command:

[‡]For ARC $\leq$ 0.5.48, use `$HOME/.nggiislist` file

```
ngsub -d 2 myjob.xrsl
```

will print out the steps taken by the ARC client to find the best cluster satisfying your job requirements. A default value for the debug level can be set by the first existing of:

1. `debug` option in `client` section in `$HOME/.arc/client.conf`
2. `debug` option in `common` section in `$HOME/.arc/client.conf`
3. `NGDEBUG` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `debug` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `debug` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `debug` option in `client` section in `/etc/arc.conf`
7. `debug` option in `common` section in `/etc/arc.conf`

It often happens that some sites that `ngsub` has to contact are slow to answer, or are down altogether. This will not prevent you from submitting a job, but will slow down the submission. To speed it up, you may want to specify a shorter timeout (default is 40 seconds) with the `-t` option:

```
ngsub -t 5 myjob.xrsl
```

A default value for the timeout can be set by the first existing of:

1. `timeout` option in `client` section in `$HOME/.arc/client.conf`
2. `timeout` option in `common` section in `$HOME/.arc/client.conf`
3. `NGTIMEOUT` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `timeout` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `timeout` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `timeout` option in `client` section in `/etc/arc.conf`
7. `timeout` option in `common` section in `/etc/arc.conf`

The user interface transforms input xRSL job description into a format that can be understood by the Grid Manager to which it is being submitted. By specifying the `-dumpxrsl` option, the transformed xRSL is written to stdout instead of being submitted to the remote site.

2.1.3 ngstat

The `ngstat` command is used for obtaining the status of jobs that have been submitted with ARC.

ngstat [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -clusters`		show information about sites (clusters)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*

`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-q, -queues`		show information about clusters and queues
`-l, -long`		long format (extended information)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`job ...`		list of job IDs and/or jobnames

The ngstat command returns the status of jobs submitted to the Grid. A job can be referred to either by the `jobID` that was returned by `ngsub` at submission time, or by its name if the job description specified a job name.

More than one `jobID` and/or job name can be given. If several jobs were submitted with the same job name, the status of all those jobs will be shown. If the `-i` option is used, the list of `jobID`s is read from a file with the specified name. By specifying the `-a` option, the status of all jobs of this user in the system will be shown.

`ngstat` uses local list of jobs (see Section 2.1.11). If you are using several different machines to submit jobs, make sure to issue `ngsync` command (Section 2.1.1) after changing the machine, to synchronise the list of known jobs.
`ngstat` can not return information about jobs that have been **erased**. The jobs are normally erased by the sites if they were not retrieved within 24 hours after job end.

A standard output produced by `ngstat` is rather short, informing the user only about the job status. Use `ngstat -l` to receive the complete job information as stored in the system.

The `-s` option can be used to select jobs in a specific state. For finished jobs, the `-s` option will match `FINISHED` or `FAILED` depending on whether the job finished successfully or not. These options can be repeated several times. For example, to check the status of jobs being on a particular stage of execution (e.g., only running jobs), use:

```
ngstat -s "INLRMS:R"
```

Diferent sites may report slightly diferent job states, depending on the installed software version. A summary of essential job states is:

ARC 0.3, ARC 0.4	ARC 0.5, ARC0.6	Description
	`ACCEPTING`	job has reached the site
`ACCEPTED`	`ACCEPTED`	job submitted but not yet processed
`PREPARING`	`PREPARING`	input files are being retreived
	`PREPARED`	input files are retreived
`SUBMITTING`	`SUBMITTING`	interaction with LRMS ongoing
`INLRMS: Q`	`INLRMS:Q`	job is queued by LRMS
`INLRMS: R`	`INLRMS:R`	job is running
	`INLRMS:S`	job is suspended
	`INLRMS:E`	job is finishing in LRMS

	`INLRMS:O`	job is in any other LRMS state
`CANCELING`	`KILLING`	job is being cancelled by user request
	`EXECUTED`	job is completed in LRMS
`FINISHING`	`FINISHING`	output files are being transferred
`FINISHED`	`FINISHED`	job is finished
	`FAILED`	job is finished with an error
	`KILLED`	job is cancelled by user request
`DELETED`	`DELETED`	job is removed due to expiration time

Detailed information on job states can be found in the Information System manual [4]. For ARC versions 0.3 and 0.4, most of the states may be prepended with the "`PENDING:`" message, if the job can not move from the state – for example, if the limit of jobs in the next state is reached. That is, `PENDING:PREPARING` means that the job **exited** the `PREPARING` state and is pending transfer to the next state, `SUBMITTING`. The `FINISHED` state may be followed by an error message if the job failed, starting with "`: FAILURE`" string. In ARC versions 0.5 and 0.6, new states are added instead, to better describe the situaton.

If the `-q` option is used, `ngstat` returns the status of the sites and queues available rather than of the jobs submitted. In this case the `-c` and `-C` options can be used to select or reject clusters for which the status should be returned.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.4 ngcat

It is often useful to monitor the job progress by checking what it prints on the standard output or error. The command `ngcat` assists here, extracting the corresponding information from the execution cluster and pasting it on the user's screen. It works both for running tasks and for the finished ones. This allows a user to check the output of the finished task without actually retreiving it.

ngcat [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of job IDs
`-c, -clusters`		show information about clusters
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-o, -stdout`		show the stdout of the job (default)
`-e, -stderr`		show the stderr of the job
`-f, -follow`		show tail of the requested file and follow its changes
`-l, -gmlog`		show the grid error log of the job
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

The `ngcat` command can return the standard output of a job (`-o` option), the standard error (`-e` option) and the errors reported by the Grid Manager (`-l` option).

> `ngcat` can only operate on jobs where either `stdout`, `stderr` and/or `gmlog` files were specified in the job description, respectively.

Option `-f` is handy for very long output files: it prints out only the last lines of the specified output, and follows its updates.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.5 ngget

To retrieve the results of a finished job, the `ngget` command should be used. It will download the files specified by the `outputfiles` attribute of job description to the user's computer.

ngget [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-dir`	*dirname*	download directory (the job directory will be created in this directory)
`-j, -usejobname`		use the jobname instead of the digital ID as the job directory name
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Only the results of jobs that have finished can be downloaded. The job can be referred to either by the `jobID` that was returned by `ngsub` at submission time, or by its name, if the job description contained a job name attribute.

The files to be retrieved by `ngget` should be described by the xRSL as follows:

```
    (outputfiles=(file1 "")(file2 ""))
```

That is, while the filenames must be listed, no output destination should be requested. This will tell the Grid Manager not to erase the files after job's end and allow the ARC client to download them.

Files specified as `stdout`, `sterr` and the `gmlog` directory are always treated as `outputfiles` by the system, which means you don't have to add them to the output files list.

By default, `ngget` will create in your current directory a new folder, by the same name as the remote session directory (typically, a numerical string). This new directory will contain all the files listed for download in your job description. If you would like to store the files in another location, use the `-dir` option. The option `-j` will assign your job name to the local directory with the output files (be careful not to call all the jobs same name when using this option).

UI **does not clean up** such download areas, so please take care of doing this yourself.

A default value for the download directory other than the current working directory can be set by the first existing of:

1. `downloaddir` option in `client` section in `$HOME/.arc/client.conf`
2. `downloaddir` option in `common` section in `$HOME/.arc/client.conf`
3. `NGDOWNLOAD` option in `$HOME/.ngrc` (ARC $\leq$ 0.5.48)
4. `downloaddir` option in `client` section in `$ARC_LOCATION/etc/arc.conf`
5. `downloaddir` option in `common` section in `$ARC_LOCATION/etc/arc.conf`
6. `downloaddir` option in `client` section in `/etc/arc.conf`
7. `downloaddir` option in `common` section in `/etc/arc.conf`

If your job produces many small output files, consider archiving them into a single file. Transfering a lot of small files at a time may temporarily consume all TCP ports allocated for data transfer on some servers, resulting in an error.

When the job result is retrieved, the session directory is **erased** from the execution machine, and the job ID is removed from your list of submitted jobs. If you however want to keep the job for a while, use the `-k` option.

Beware that the job results will be **removed** by the Grid Manager at the execution machine 24 hours after job completion independently of whether you retrieved the results or not.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.6 ngkill

It happens that a user may wish to cancel a job. This is done by using the `ngkill` command. A job can be killed amost on any stage of processing through the Grid.

ngkill [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -clusters`		show information about clusters
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`job ...`		list of job IDs and/or jobnames

When option `-j` is used, and several jobs have the same name, **ALL such jobs will be cancelled!**.

Job cancellation is an asynchronous process, such that it may take a few minutes before the job is actually cancelled.

When a job is successfully killed, it is **erased** from the remote site. Use `-keep` option to keep the output for eventual retrieval with `ngget` or `ngcat`.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.7 ngresub

Quite often it happens that a user would like to re-submit a job, but has difficulties recovering the original job description xRSL file. This happens when xRSL files are created by scripts on-fly, and matching of xRSL to the job ID is not straightforward. The utility called `ngresub` helps in such situations, allowing users to resubmit jobs known only by their job IDs.

Only jobs where the `gmlog` attribute was specified in the job description can be resubmitted.

ngresub [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-k, -kluster`	`[-]`textemname	explicitly select or reject a specific site (cluster) as re-submission target
`-K, -Klustlist`	`[-]`textemfilename	list of clusters to select or reject as re-submission target
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the xRSL
`-dumpxrsl`		do not submit – dump transformed xRSL to stdout
`-keep`		keep files on gatekeeper (do not clean)
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-U, -unknownattr`		allow unknown attributes in job description – pass them through
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`job ...`		list of job IDs and/or jobnames

If the original job description contained input file locations specified as relative paths (non-URLs), the command must be issued in the same directory as the original `ngsub` instruction.

It is important to distinguish `-c` and `-k` options (as well as `-C` and `-K`). The former stands for `-cluster` and is used to instruct `ngresub` to look for jobIDs and descriptions only at a given site (cluster), or exclude a cluster. This is convenient to use together with the `-a` option, in case you would like to re-submit all the jobs which were originally sent to a specific site (cluster). The latter option, `-k`, stands for `-kluster`, and should be used to specify preferred re-submission target. **Important** examples:

```
ngresub -c fire.ii.uib.no myjob1
```

will resubmit all jobs on `fire.ii.uib.no` and the job `myjob1` to any cluster (except the cluster at which the job is currently), while

```
ngresub -k fire.ii.uib.no myjob1
```

will resubmit the job `myjob1` from wherever it was to `fire.ii.uib.no` (note the usage of `-c` and `-k` options).

```
ngresub -c grid1.ua.org -k grid2.ub.org
```

will resubmit all your jobs from cluster `grid1.ua.org` to `grid2.ub.org`.

If the re-submission was successful, the original job will be removed from the remote cluster unless you specify the `-keep` option.
Old Grid Manager log (`gmlog`) records are not preserved and are not transferred to the new site..

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-o`, `-dryrun`, `-dumpxrsl`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.8 ngclean

If a job fails, or you are not willing to retrieve the results for some reasons, a good practice for users is not to wait for the Grid Manager to clean up the job leftovers, but to use `ngclean` to release the disk space and to remove the job ID from the list of submitted jobs and from the Information System.

ngclean [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-f, -force`		removes the job ID from the local list even if the job is not found on the Grid
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Only jobs that have finished can be cleaned.

Use `-f` option to remove jobs from your local list of jobs (`.ngjobs` file), if they can not be found in the information system and you are confident they are not needed anymore.

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.9 ngrenew

Quite often, the user proxy expires while the job is still running (or waiting in a queue). In case such job has to upload output files to a Grid location (Storage Element), it will fail. By using the `ngrenew` command, users can upload a new proxy to the job. This can be done while a job is still running, thus preventing it from failing.

If a job has failed in file upload due to expired proxy, `ngrenew` can be issued whithin 24 hours (or whatever is the expiration time set by the site) after the job end, which must be followed by `ngresume`. The Grid Manager will then attempt to finalize the job by uploading the output files to the desired location. In ARC versions prior to 0.6, `ngrenew` automatically resumed jobs failed in upload.

ngrenew [options] [job ...]

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

Prior to using `ngrenew`, be sure to actually create the new proxy, e.g.:

```
grid-proxy-init -valid 24:00
```

Here `-valid` specifies for how long the proxy will be valid. You can renew a VOMS-extended proxy as well (use `voms-proxy-init`).

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.10 ngresume

In some cases a user may want to restart a failed job, for example, when input files become available, or the storage element for the output files came back online, or when a proxy is renewed with `ngrenew`. This can be done using the `ngresume` command.

Make sure your proxy is still valid, or when uncertain, run `ngrenew` before `ngresume`. The job will be resumed from the state where it has failed.

ngresume [options] [job ...]

(ARC 0.6, ARC 0.8)

Options:

`-a, -all`		all jobs
`-i, -joblist`	*filename*	file containing a list of jobIDs
`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-C, -clustlist`	`[-]`textemfilename	list of sites (clusters) to select or reject
`-s, -status`	*statusstr*	only select jobs whose status is *statusstr*
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Arguments:

`job ...`	list of job IDs and/or jobnames

For descriptions of `-a`, `-i` and `-s` options, see `ngstat` section 2.1.3.

For descriptions of the `-c`, `-C`, `-g`, `-G`, `-t` and `-d` options, refer to the `ngsub` description in Section 2.1.2.

2.1.11 Auxilliary files

ARC client keeps local job lists in two files: `$HOME/.ngjobs` and `$HOME/.arc/history`[§].

`$HOME/.ngjobs` is a local list of the user's active jobs. When a job is successfully submitted, it is added to this list, and when it is removed from the remote site, it is removed from this list. This list is used as the list of all active jobs when the user specifies `-a` option to the various ARC user interface commands. For information about how to reconstruct this file in case it is damaged or you relocate to a different workstation, see section 2.1.1 about the `ngsync` command.

`$HOME/.arc/history` contains the `jobID`s of the jobs the user has submitted together with the time of submission. This file is purely informational.

2.2 Data manipulation

ARC provides basic data management tools, which are simple commands for file copy and removal, with eventual use of data indexing services.

2.2.1 ngls

The `ngls` is a simple utility that allows to list contents and view some attributes of objects of a specified (by an URL) remote directory.

ngls [options] <URL>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

[§]In ARC $\leq$ 0.5.48, `$HOME/.nghistory`

Options:

`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-l`		detailed listing
`-L`		detailed listing including URLs from which file can be downloaded
`-m`		list all available metadata attributes
`-r`	*level*	list recursively to given level

Arguments:

`URL`	file or directory URL

This tool is very convenient not only because it allows to list files at a Storage Element or records in an indexing service, but also because it can give a quick overview of a job's working directory, which is explicitly given by job ID.

Usage examples can be as follows:

```
ngls rls://rc.host:38203/logical_file_name
ngls -l gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535
ngls -L srm://grid.uio.no:58000/srm/managerv1/johndoe/log2
```

Examples of URLs accepted by this tool can be found in Section 3, though `ngls` won't be able to list a directory at an HTTP server, as they normally do not return directory listings.

2.2.2 ngcp

The `ngcp`[¶] is a powerful tool to copy files over the Grid. It is a part of the Grid Manager, but can be used by the User Interface as well.

ngcp [options] <source> <destination>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: from -3 = quiet to 3 = verbose
`-y`	*cache_path*	path to local cache (use to put file into cache)
`-p`		use passive transfer (does not work if secure is on, default if secure is not requested)
`-n`		do not try to force passive transfer
`-i`		show progress indicator
`-u`		use secure transfer (insecure by default)
`-r`	*recursion_level*	operate recursively (if possible) up to specified level (0 - no recursion)
`-R`	*number*	how many times to retry transfer of every file before failing
`-t`	*time*	timeout in seconds (default 20)

[¶]In ARC ≤ 0.5.28 was called `ngcopy`

`-f`	if the destination is an indexing service and not the same as the source and the destination is already registered, then the copy is normally not done. However, if this option is specified the source is assumed to be a replica of the destination created in an uncontrolled way and the copy is done like in case of replication
`-T`	do not transfer file, just register it - destination must be non-existing meta-url

Arguments:

`source`	source URL
`destination`	destination URL

This command transfers contents of a file between 2 end-points. End-points are represented by URLs or meta-URLs. For supported endpoints please refer to Section 3.

`ngcp` can perform multi-stream transfers if `threads` URL option is specified and server supports it.

Source URL can end with `"/"`. In that case, the whole fileset (directory) will be copied. Also, if the destination ends with `"/"`, it is extended with part of source URL after last `"/"`, thus allowing users to skip the destination file or directory name if it is meant to be identical to the source.

> Since the job ID is in fact a `gsiftp://` URL of the job top directory, you can use `ngcp` to copy files from the job directory at any time.

Usage examples of `ngcp` are:

```
ngcp gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535/job.out \
        file:///home/myname/job2.out
ngcp gsiftp://aftpexp.bnl.gov;threads=10/rep/my.file \
        rc://;threads=4@grid.uio.no/lc=Collection,rc=Catalog/zebra4.f
ngcp http://www.nordugrid.org/data/somefile gsiftp://hathi.hep.lu.se/data/
```

2.2.3 ngrm

The `ngrm`[‖] command allows users to erase files at any location specified by a valid URL.

ngrm [options] <source>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:

`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-c`		continue with meta-data even if it failed to delete real file
`-C`	*cache_data_path*	store cached data

Arguments:

`source`	source URL

[‖]In ARC ≤ 0.5.28 was called `ngremove`

A convenient use for `ngrm` is to erase the files in a data indexing catalog (RC, RLS or such), as it will not only remove the physical instance, but also will clean up the database record.

Here is an `ngrm` example:

```
ngrm rc://grid.uio.no/lc=Collection,rc=Catalog/badfile#\\
```

2.2.4 ngacl

This command retrieves or modifies access control information associated with a stored object if service supports GridSite GACL language [5] for access control.

ngacl [options] get|put <URL>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-d, -debug`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-v`		print version information
`-h`		short help
Arguments:		
`get`	*URL*	get Grid ACL for the object
`put`	*URL*	set Grid ACL for the object
`URL`		object URL; curently only gsiftp and sse URLs are supported

ACL document (an XML file) is printed to standard output when `get` is requested, and is acquired from standard input when `set` is specified[**]. Usage examples are:

```
ngacl get gsiftp://se1.ndgf.csc.fi/ndgf/tutorial/dirname/filename
ngacl set gsiftp://se1.ndgf.csc.fi/ndgf/tutorial/dirname/filename $<$ myacl
```

2.2.5 ngtransfer

The `ngtransfer` command[††] initiates direct transfer of data between 2 servers (known as *third-party transfer*).

ngtransfer [options] <destination>

(ARC 0.3, ARC 0.4, ARC 0.5, ARC 0.6, ARC 0.8)

Options:		
`-s, -source`	*URL*	source file URL
`-d, -debug`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-v`		print version information
`-h`		short help
Arguments:		
`destination`		destination URL; currently only se:// and (gsi)ftp:// are supported

[**] In ARC $\leq$ 0.5.28, `set` shoud be used instead of `put`

[††] In ARC $\leq$ 0.5.28, was called `ngrequest`

This command initiates file copy from multiple source instances to a destination URL. Destination of `(gsi)ftp` type accepts only similar kinds of sources. Destination can also be URL of an Indexing Service. In such a case, real destinations with suitable protocols are chosen, when available. Requests are sent to the corresponding services/servers to initiate file transfer from one of the sources.

> Absence of `-s` option is treated as a file replication request. In this case, destination must be an Indexing service or an SRM.

Following source URL types are supported: `http`, `https`, `httpg`, `ftp`, `gsiftp`, `rc`, `rls`, `se`, `srm`, `fireman` (see Section 3 for URL details).

Example:

```
ngtransfer -s http://www.host1.org/dat1  -s gsiftp://host2.org/dir/dat1 \
           se://se.host.org/se_service?new_file_lfn
```

2.2.6 ngstage

The `ngstage` command is used to request file staging from tapes to front-end disks. Since jobs can not read data from tapes directly, if an input file is on a tape, the job most ikely will fail for a timeout, because tape staging process is a rather lengthy one (can be few hours). To avoid these timeout failures, `ngstage` should be used before submitting jobs that are known to use data archived on tapes.

ngstage [options] <URL(s)>

(ARC 0.6, ARC 0.8)

Options:

`-q`	*requestID*	query the status of a request. Use `-d 1` in addition to show a per-file status.
`-c`	*requestID*	cancel a request
`-D`		do a dry run, listing the files that would be staged
`-d`	*debuglevel*	debug level from 0 to 3 (default 0). Setting to 1 when listing a request status will show the status of each file in the request.
`-r`	*level*	if copying fileset/directory recurse to the specified level. If any given URL is a directory a level of at least 1 must be specified.
`-s`	*URL*	the endpoint of the staging service. This must be given with `-q`, `-c` and `-l` options.
`-t`	seconds	timeout in seconds. This is not a timeout for staging the file(s), but a timeout for calls to the service performing the staging. A timeout of this kind does not necessarily mean that the staging request has failed.
`-v`		print version information
`-h`		short help

Arguments:

`URL`		file URL; currently only srm:// meta-protocol is supported

The command will return a request ID that should be used to monitor the status (with `-q` option) or cancel the request (`-c` option). A list of currently active requests can be found using the `-l` option. If either of these three options are used, URL of the file server performing the staging must also be specified, through the `-s` option, this usually consists of the protocol (`srm://`) and hostname of the URL(s) passed in the inital request.

> Every URL in the request must have the same host name, as the command sends request to one file server at a time.

Wildcards are allowed at the end of a URL, and URLs can be directories, but `-r` must be used if the contents of a directory are to be staged. If a given URL is a directory, `-r 1` will stage all files in that directory.

Example:

```
ngstage srm://srm.ndgf.org/pnfs/ndgf.org/data/atlas/tape/file.1\\
ngstage -q -2139909111 -s srm://srm.ndgf.org
```

This requests a file to be staged and queries request status.

> After the first time a successful request is queried, it is internally marked as aborted by the system. For any subsequent queries it is not possible to obtain the per-file status.

2.3 Test suite

ARC comes with a fairly complete test-suite that tests different functionalities of the middleware. The test-suite consists of one utility, called `ngtest`, that contains a variety of test examples which can be executed. This utility can be used either for testing a newly installed cluster or by a new user that wants to try out a few simple things on the Grid for the first time. The utility can print helpful output about installed user credentials, user authorization on computing resources and storage elements, VO membership information, and it can also be used to submit various test-jobs, testing either the software installation on the client or some server.

The utility comes with the `nordugrid-arc-client` and the `nordugrid-arc-standalone` packages.

2.3.1 ngtest

ngtest [options]

(ARC $\geq$ 0.5.49[‡‡])

Options:

`-E, -certificate`		prints information about installed user- and CA-certificates
`-j, -job`	*number*	submit test job given by the number (1,2,3 possible)
`-R, -resources`		print user authorization information for clusters and storage elements
`-O, -configuration`		print user configuration
`-V, -vo`		print VO membership information for the user
`-r, -runtime`	*time*	run testjob 1 for the specified number of minutes (default 5)

[‡‡]`ngtest` was available in ARC 0.4 and earlier 0.5.x versions, but had a substantially different functionality

`-c, -cluster`	`[-]`textemname	explicitly select or reject a specific cluster
`-C, -clustlist`	`[-]`textemfilename	list of clusters to select or reject
`-g, -giisurl`	*url*	URL of a central Information System server
`-G, -giislist`	*filename*	list of GIIS URLs
`-o, -joblist`	*filename*	file where the job IDs will be stored
`-dryrun`		add dryrun option to the xRSL
`-dumpxrsl`		do not submit – dump transformed xRSL to stdout
`-t, -timeout`	*time*	timeout for queries (default 40 sec)
`-d, -debug`	*debuglevel*	debug level, from -3 (quiet) to 3 (verbose) - default 0
`-x, -anonymous`		use anonymous bind for queries (default)
`-X, -gsi`		use GSI-GSSAPI bind for queries
`-v, -version`		print version information
`-h, -help`		print help page

Default values of some options (e.g. `timeout` or `giislist`) are shared with other client commands and can be specified in the user configuration file (Section 4).

Usage examples:

- discover resources available for the user:
  `ngtest -R`
- print out certificate information:
  `ngtest -E`
- submit a test job:
  `ngtest -J 1`
- submit a test job to a selected site:
  `ngtest -J 1 -c grid.place.org`
- submit a test job with dianostics printout :
  `ngtest -J 1 -d 2`

Test job descriptions:

1. This test-job calculates prime-numbers for 2 minutes and saves the list. The source-code for the prime-number program, the Makefile and the executable is downloaded to the cluster chosen from various servers and the program is compiled before the run. In this way, the test-job constitute a fairly comprehensive test of the basic setup of a grid cluster.
2. This test-job does nothing but print `hello, grid` to stdout.
3. This test-job tests download capabilities of the chosen cluster by downloading 8 inputfiles over different protocols – HTTP, FTP, gsiFTP and RLS.

2.4 Third-party commands

A set of third-party commands is bundled with ARC client. Currently, these are a subset of Globus [1], VOMS [2] and LDAP [6] utilities.

2.4.1 Globus utilities

A set of Globus Replica Catalog (RC) utilities contains the following commands:

```
rc-add-file
rc-add-location
rc-add-logical
rc-add-physical
rc-delete-location
rc-delete-logical
rc-delete-physical
rc-list
rc-wrapper
```

With the `standalone` ARC package, some additional Globus commands are available.

Operations with Grid credentials:

```
grid-cert-info
grid-cert-request
grid-change-pass-phrase
grid-default-ca
grid-proxy-destroy
grid-proxy-info
grid-proxy-init
```

Globus installation information and setup:

```
globus-domainname
globus-hostname
globus-makefile-header
globus-sh-exec
globus-version
```

Globus Replica Location Service command line interface:

```
globus-rls-cli
```

Globus GridFTP file copy utility:

```
globus-url-copy
```

2.4.2 LDAP utilties

With the `standalone` ARC package, the following additional LDAP commands are available:

```
ldapadd
ldapdelete
ldapmodify
ldapmodrdn
ldappasswd
ldapsearch
ud
```

LDAP utilities can be used for low-level querying of the Information System or managing LDAP-based Virtual Organization databases.

2.4.3 VOMS utilities

With the `standalone` ARC package, the following additional VOMS commands are available:

```
voms-proxy-destroy
voms-proxy-fake
voms-proxy-info
voms-proxy-init
voms-proxy-list
```

VOMS proxies are used by Virtual Organizations (VO) to attach additional VO-specific information to a regular Grid proxy.

Chapter 3

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.

The following transfer protocols and metadata servers are supported:

`ftp`	ordinary *File Transfer Protocol (FTP)*
`gsiftp`	GridFTP, the Globus® -enhanced FTP protocol with security, encryption, etc. developed by The Globus Alliance [1]
`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)* [6]
`srm`	Storage Resource Manager (SRM) service [7]
`se`	ARC Smart Storage Element service [8]
`lfc`	LFC catalog and indexing service of EGEE gLite [9]
`rc`	Globus® *Replica Catalog (RC)* [10]
`rls`	Globus® *Replica Location Service (RLS)* [11]
`fireman`	Fireman indexing service of EGEE gLite [9]
`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:

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

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.

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 (LFN)
`url`	URL of the file as registered in LFC/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 for location URLs:

`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|renew|copy`	indicates whether the GM should cache the file; default for input files 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 dir, this is useful if for example the file is to be modified[*].
`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[†].
`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
`protocol=gsi|gssapi`	to distinguish between two kinds of `httpg`. `gssapi` stands for implemention using only GSSAPI functions to wrap data and `gsi` uses additional headers as implmented in Globus IO

[*] "copy" option is available in ARC 0.8.1 and above

[†] "adler32" and "no" options are available in ARC 0.8.1 and above

`spacetoken=<pattern>`	specify the space token to be used for uploads to SRM storage elements supporting SRM version 2.2 or higher
`autodir=yes|no`	specify if before writing to specified location software should try to create all directories mentioned in specified URL. Currently this applies to FTP and GridFTP only. Default for those protocols is `yes`

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.

Metadata service URLs also support metadata options which can be used for register additional metadata attributes or query 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:

`guid`	GUID of the file in the metadata service
`checksumtype`	Type of checksum. Supported values are cksum (default), md5 and ad (adler32 checksum)
`checksumvalue`	The checksum of the file

Currently these metadata options are only supported for lfc:// URLs.

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[1]
fireman://fireman_host:8443/glite-data-catalog-interface/FiremanCatalog?data.root
file:///home/auser/griddir/steer.cra
lfc://srm://srm.domain.org/griddir@lfc.domain.org//user/file1:guid=\
    bc68cdd0-bf94-41ce-ab5a-06a1512764dc:checksumtype=ad:checksumvalue=12345678[2]
lfc://;cache=no@lfc.domain.org/:guid=bc68cdd0-bf94-41ce-ab5a-06a1512764d[3]
```

[1]This is a destination URL. The file will be copied to the GridFTP server at `se.domain.org` with the path `datapath/file25.dat` and registered in the RLS indexing service at `grid.domain.org` with the LFN `myfile02.dat`.

[2]This is a destination URL. The file will be copied to `srm.domain.org` at the path `griddir/file1` and registered to the LFC service at `lfc.domain.org` with the LFN `/user/file1`. The given GUID and checksum attributes will be registered.

[3]This is a source URL. The file is registered in the LFC service at `lfc.domain.org` with the given GUID and can be copied or queried by this URL. Note that as URL options are part of the location (physical) URL, in meta service URLs the options must be part of the location URL, even if the location URL is empty.

Chapter 4

Configuration

4.1 ARC Client Configuration

Default behaviour of ARC client can be configured by specifying alternative values for some parameters in the client configuration file. The file is called `client.conf` and is located in directory `.arc` in user's home area:

```
$HOME/.arc/client.conf
```

If this file is not present or does not contain the relevant configuration information, the global configuration files (if exist) or default values are used instead.

The ARC configuration file consists of several configuration blocks. Each configuration block is identified by a keyword and contains the configuration options for a specific part of the ARC middleware.

The configuration file can be written in either of two different formats: plain text or XML. In the plain text format, client configuration block starts with its identifying keyword inside square brackets: `[client]`. Thereafter follows one or more attribute-value pairs written one on each line in the following format:

```
[client]
attribute1="value1"
attribute2="value2"
# comment line 1
# comment line 2
attribute2="value3"
...
```

In the XML format, the configuration file consists of a single "arc" XML element. Inside this element, client configuration block appears as a separate XML element: `<client>`. This block in turn contains attribute-value elements in the following format:

```
<arc>
 <client>
  <attribute1>value1</attribute1>
  <attribute2>value2</attribute2>
<!-- comment line -->
<!--
commented
block
```

```
-->
  <attribute2>value3</attribute2>
  ...
 </client>
</arc>
```

If one has an ARC server installation as well, one can also use the `common` block, which can have the same contents as the `client` one, and is shared with the server.

For multi-valued attributes, several elements or attribute-value pairs have to be specified – one per each value.

The following attributes are allowed:

giis

This attribute is multi-valued.

Configures which GIISes (site indices) to use to discover computing and storage resources. The default is to use the four top level GIISes:

```
ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index2.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index3.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
ldap://index4.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid
```

Examples:

- plain text:
  ```
  giis="ldap://atlasgiis.nbi.dk:2135/O=Grid/Mds-Vo-name=Atlas"
  giis="ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid"
  giis="ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local"
  ```
- XML:
  ```
  <giis>ldap://odin.switch.ch:2135/O=Grid/Mds-Vo-name=Switzerland</giis>
  <giis>ldap://index1.nordugrid.org:2135/O=Grid/Mds-Vo-name=NorduGrid</giis>
  <giis>ldap://grid.tsl.uu.se:2135/O=Grid/Mds-Vo-name=local</giis>
  ```

Note that LDAP URLs containing "Mds-Vo-name=local" refer not to GIISes, but to individual sites. This feature can be used to add "hidden" sites that do not register to any GIIS, or to list explicitly preferred submission sites.

debug

Default debug level to use for the ARC clients. Corresponds to the `-d` command line switch of the clients. Default value is 0, possible range is from -3 to 3.

Examples:

- plain text:
  ```
  debug="2"
  ```
- XML:
  ```
  <debug>2</debug>
  ```

timeout

Timeout to use for interaction with LDAP servers, gridftp servers, etc. If a server, during e.g. job submission, does not answer in the specified number of seconds, the connection is timed out and closed. Default value is 20 seconds.

Examples:

- plain text:
  ```
  timeout="20"
  ```
- XML:
  ```
  <timeout>10</timeout>
  ```

downloaddir

Default download directory to download files to when retrieving output files from jobs using `ngget`. Default is the current working directory.

Examples:

- plain text:
  ```
  downloaddir="/home/johndoe/arc-downloads"
  ```
- XML:
  ```
  <downloaddir>/tmp/johndoe</downloaddir>
  ```

alias

This attribute is multi-valued.

Alias substitutions to perform in connection with the `-c` command line switch of the ARC clients. Alias definitions are recursive. Any alias defined in a block that is read before a given block can be used in alias definitions in that block. An alias defined in a block can also be used in alias definitions later in the same block.

Examples:

- plain text:
  ```
  alias="host1=somehost.nbi.dk"
  alias="host2=otherhost.uu.se"
  alias="myhosts=host1 host2"
  ```
- XML:
  ```
  <alias>host1=somehost.nbi.dk</alias>
  <alias>host2=otherhost.uu.se</alias>
  <alias>myhosts=host1 host2</alias>
  ```

With the example above, `ngsub -c myhosts` will resolve to `ngsub -c somehost.nbi.dk -c otherhost.uu.se`.

broker

Configures which brokering algorithm to use during job submission. The default one is the `FastestCpus` broker that chooses, among the possible targets, the target with the fastest CPUs. Another possibility is, for example, the `RandomSort` broker that chooses the target randomly among the targets surviving the job description matchmaking.

Examples:

- plain text:
  ```
  broker="RandomSort"
  ```
- XML:
  ```
  <broker>RandomSort</broker>
  ```

Deprecated configuration files

In ARC $\leq$ 0.5.48, configuration was done via files `$HOME/.ngrc`, `$HOME/.nggiislist` and `$HOME/.ngalias`.

The main configuration file `$HOME/.ngrc` could contain user's default settings for the debug level, the information system query timeout and the download directory used by `ngget`. A sample file could be the following:

```
# Sample .ngrc file
# Comments starts with #
NGDEBUG=1
NGTIMEOUT=60
NGDOWNLOAD=/tmp
```

If the environment variables NGDEBUG, NGTIMEOUT or NGDOWNLOAD were defined, these took precedence over the values defined in this configuration. Any command line options override the defaults.

The file `$HOME/.nggiislist` was used to keep the list of default GIIS server URLs, one line per GIIS (see `giis` attribute description above).

The file `$HOME/.ngalias` was used to keep the list of site aliases, one line per alias (see `alias` attribute description above).

Bibliography

[1] I. Foster and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit,” *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128, 1997, available at: http://www.globus.org.

[2] R. Alfieri *et al.*, “From gridmap-file to VOMS: managing authorization in a Grid environment,” *Future Gener. Comput. Syst.*, vol. 21, no. 4, pp. 549–558, 2005.

[3] O. Smirnova, *Extended Resource Specification Language*, The NorduGrid Collaboration, NORDUGRID-MANUAL-4. [Online]. Available: http://www.nordugrid.org/documents/xrsl.pdf

[4] B. Kónya, *The NorduGrid/ARC Information System*, The NorduGrid Collaboration, NORDUGRID-TECH-4. [Online]. Available: http://www.nordugrid.org/documents/arc_infosys.pdf

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

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

[7] A. Sim, A. Shoshani and others, “The Storage Resource Manager Interface (SRM) Specification v2.2,” May 2008, GFD-R-P.129. [Online]. Available: http://www.ggf.org/documents/GFD.129.pdf

[8] A. Konstantinov, *The NorduGrid Smart Storage Element*, The NorduGrid Collaboration, NORDUGRID-TECH-10. [Online]. Available: http://www.nordugrid.org/documents/SE.pdf

[9] “gLite, Lightweight Middleware for Grid Computing,” Web site. [Online]. Available: http://glite.web.cern.ch/glite/

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

[11] 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.

Index