NORDUGRID

NORDUGRID-MANUAL-?

14/8/2008

ARC Clients

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.

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 job description language. ARC supports the languages JSDL, xRSL and JDL [?].

2.1.1 arcsub

The `arcsub` command is the most essential one, as it is used for submitting jobs to the Grid resources. . `arcsub` 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.

arcsub [options] <task ...>

(ARC 0.9)

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
`-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

Some text can be in frames

A simple *"Hello World"* job would look like

arcstat [options] [job ...]

(ARC 0.9)

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 arcstat command returns the status of jobs submitted to the Grid. A job can be referred to either by the

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

2.1.2 arccat

It is often useful to monitor the job progress by checking what it prints on the standard output or error. The command `arccat` 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.

arccat [options] [job ...]

(ARC 0.9)

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, -gridlog`		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 `arccat` 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).

□

2.1.3 arcget

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

arcget [options] [job ...]

(ARC 0.9)

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 `arcsub` at submission time, or by its name, if the job description contained a job name attribute.

2.1.4 arckill

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

arckill [options] [job ...]

(ARC 0.9)

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.

2.1.5 arcresub

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 `arcresub` 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.

arcresub [options] [job ...]

(ARC 0.9)

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
`-v, -version`		print version information
`-h, -help`		print help page
Arguments:		
`job ...`		list of job IDs and/or jobnames

2.1.6 arcclean

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 `arcclean` to release the disk space and to remove the job ID from the list of submitted jobs and from the Information System.

arcclean [options] [job ...]

(ARC 0.9)

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.

2.1.7 arcrenew

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 `arcrenew` 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, or whithin 24 hours (or whatever is the expiration time set by the site) after the job end. In the latter case, the Grid Manager will attempt to finalize the job by uploading the output files to the desired location.

arcrenew [options] [job ...]

(ARC 0.9)

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 `arcrenew`, be sure to actually create the new proxy:

2.1.8 arcsync

If you are using User Interface 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 `arcsync` command.

arcsync [options]

(ARC 0.9)

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 User Interface keeps a local list of jobs in the user's home directory (see section 2.1.9). If this file is lost, corrupt, or the user wants to recreate the file on a different workstation, the `arcsync` command will recreate this file from the information available in the Information System.

2.1.9 Auxilliary files

User Interface 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 2.1.8 about the `arcsync` command.

`$HOME/.arc/history` contains the `jobIDs` 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 arcls

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

arcls [options] <URL>

(ARC 0.9)

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 and temporary cached locations
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
```

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

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

2.2.2 arccp

The `arccp`[†] 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.

arccp [options] <source> <destination>

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

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)
`-Y`	*cache_data_path*	path for cache data (if different from -y)
`-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)
`-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.

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

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)

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

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 [?] for access control.

ngacl [options] get|put <URL>

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

Options:		
`-d, -debug`	*debuglevel*	debug level: 0 = some, 1 = more, 2 = a lot
`-v`		print version information

[‡]In ARC $\leq$ 0.5.28 was called `ngremove`

`-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)

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

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
```

[§]In ARC ≤ 0.5.28, `set` shoud be used instead of `put`
[¶]In ARC ≤ 0.5.28, was called `ngrequest`

2.3 Test suite

2.4 Third-party commands

Chapter 3

URLs

Chapter 4

Configuration

4.1 ARC Client Configuration