NORDUGRID-MANUAL-1
4/11/2009

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. This manual replaces the older version of `NORDUGRID-MANUAL-1` and is valid for ARC versions 0.9 and above. Command line tools semantics is the same as in earlier versions of ARC, roughly following that of basic Linux commands and most common batch system commands. One obvious difference is change of the legacy prefix from "ng" to the more appropriate "arc". This is not only a cosmetic change: **behaviour of the commands also have changed**, as did their functionalities and options.

> Users are strongly discouraged from modifying their old scripts by simply replacing "ng" with "arc" – results may be unpredictable.

Chapter 2

Commands

2.1 Proxy utilities

ARC now comes complete with a set of utilities to create temporary user credentials (proxies) used to access Grid services.

2.1.1 arcproxy

In order to contact Grid services (submit jobs, copy data, check information etc), one has to present valid credentials. The are commonly formalized as so-called "proxy" certificates. There are many different types of proxy certificates, with different Grids and different services having own preferences. `arcproxy` is a powerful tool that can be used to generate most commonly used proxies. It supports the following types:

- pre-RFC GSI proxy
- RFC-compliant proxy (default)
- VOMS-extended proxy
- MyProxy delegation

`arcproxy` requires presence of user's private key and public certificate, as well as the public certificate of their issuer CA.

arcproxy [options]

(ARC 0.9)

Options:

`-P, --proxy`	*path*	path to the proxy file
`-C, --cert`	*path*	path to the certificate file
`-K, --key`	*path*	path to the key file
`-T, --cadir`	*path*	path to the trusted certificate directory, only needed for VOMS client functionality
`-V, --vomses`	*path*	path to the VOMS server configuration file
`-S, --voms`	*voms[:command]*	Specify VOMS server (more than one VOMS server can be specified like this: –voms VOa:command1 –voms VOb:command2)

		:command is optional, and is used to ask for specific attributes(e.g. roles). Command options are:
		all – put all of this DN's attributes into AC;
		list – list all of the DN's attribute,will not create AC extension;
		/Role=yourRole – specify the role, if this DN has such a role, the role will be put into AC
		/voname/groupname/Role=yourRole – specify the VO,group and role; if this DN has such a role, the role will be put into AC
`-G, --gsicom`		use GSI communication protocol for contacting VOMS services
`-O, --old`		use GSI proxy (default is RFC 3820 compliant proxy)
`-I, --info`		print all information about this proxy. In order to show the Identity (DN without CN as suffix for proxy) of the certificate, the 'trusted certdir' is needed.
`-U, --user`	*string*	username for MyProxy server
`-L, --myproxysrv`	*URL*	URL of MyProxy server
`-M, --myproxycmd`	*PUT|GET*	command to MyProxy server. The command can be PUT and GET.
		PUT/put – put a delegated credential to MyProxy server;
		GET/get – get a delegated credential from MyProxy server, credential (certificate and key) is not needed in this case.
`-c, --constraint`	*string*	proxy constraints
`-t, --timeout`	*seconds*	timeout in seconds (default 20 seconds)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

Supported constraints are:

- `validityStart=time` – e.g. 2008-05-29T10:20:30Z; time when certificate becomes valid. Default is now.
- `validityEnd=time` – time when certificate becomes invalid. Default is 43200 (12 hours) from start.
- `validityPeriod=time` – e.g. 43200 or 12h or 12H; for how long certificate is valid. If neither `validityPeriod` nor `validityEnd` are specified, default is 12 hours
- `vomsACvalidityPeriod=time` – e.g. 43200 or 12h or 12H; for how long the AC is valid. Default is the same as `validityPeriod`.
- `proxyPolicy=policy content` – assigns specified string to proxy prolicy to limit it's functionality.
- `proxyPolicyFile=policy file`

MyProxy functionality can be used together with VOMS functionality.

2.1.2 arcslcs

This utility generates short-lived credential based on the credential to IdP in SAML2SSO profile (normally the username/password to Shibboleth IdP).

arcslcs [options]

(ARC 0.9)

Options:

`-S, --ur;`	*URL*	URL of SLCS Service (e.g. https://127.0.0.1:60000/slcs)
`-I, --idp`	*URL*	the name of IdP (e.g. https://idp.testshib.org/idp/shibboleth)
`-U, --user`	*string*	User account to IdP
`-P, --password`	*string*	password for user accoutn to IdP
`-Z, --keysize`	*integer*	size of the private key, default is 1024
`-K, --keypass`		passphrase for protecting the private key; if not set, the private key file will not be protected by the passphrase.
`-L, --lifetime`	*hours*	life time of the credential (hours)), starting with current time
`-D, --storedir`	*path*	store directory of the credential
`-t, --timeout`	*seconds*	timeout in seconds (default 20 seconds)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-c, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

2.2 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 following languages: JSDL [1], xRSL [2] and JDL [3].

2.2.1 arcsync

It is advised to start every grid session by running `arcsync`, 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 `arcsync` command.

arcsync [options]

(ARC 0.9)

Options:

`-c, --cluster`	`[-]`*name*	explicitly select or reject a specific site

`-i, --index`	*url*	explicitly select or reject (-) a specific index server
`-j, --joblist`	*filename*	file where user's job information will be stored
`-f, --force`		don't ask for confirmation
`-T, --truncate`		truncate the job list before synchronising
`-t, --timeout`	*seconds*	timeout in seconds (default 20)
`-d, --debug`	*debuglevel*	debug level, FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE - default WARNING
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

The ARC client keeps a local list of jobs in the user's home directory. 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.

Since the information about a job retrieved from a cluster can be slightly out of date if the user very recently submitted or removed a job, a warning is issued when this command is run. The `--force` option disables this warning.

If the job list is not empty when invoking syncronisation, the old jobs will be merged with the new jobs, unless the `--truncate` option is given, in which case the job list will first be truncated and then the new jobs will be added.

2.2.2 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] [filename]

(ARC 0.9)

Options:

`-c, --cluster`	`[-]`*url*	explicitly select or reject (-) a specific site
`-i, --index`	*url*	explicitly select or reject (-) a specific index server
`-e, --jobdescrstring`	*filename*	string describing the job to be submitted
`-f, --jobdescrfile`	*filename*	file describing the job to be submitted
`-j, --joblist`	*filename*	file where user's job information will be stored
`-x, --dumpdescription`		do not submit – dump transformed job description to stdout
`-t, --timeout`	*seconds*	timeout in seconds (default 20)
`-d, --debug`	*debuglevel*	debug level, FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE - default WARNING
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-b, --broker`	*string*	select broker method (default is Random)
`-n, --dolocalsandbox`		store job descriptions in local sandbox (useful for eventual resubmission/migration)
`-v, --version`		print version information

`-h, --help`	print help page
Arguments:	
`filename`	file describing the job(s) to be submitted

The `-c` and `-i` arguments accept meta-URLs of the format `GRID:URL`, where `GRID` indicates a Grid middleware flavour. Possible flavours are `ARC0`, `ARC1`, `CREAM` and `UNICORE`. For example, for index servers:

```
ARC0:ldap://index.ng.org:2135/mds-vo-name=sweden,O=grid
CREAM:ldap://cream.glite.org:2170/o=grid
```

or clusters:

```
ARC0:ldap://ce.ng.eu:2135/nordugrid-cluster-name=ce.ng.eu,Mds-Vo-name=local,o=grid
```

As a shorthand `-f` can be omitted if the job description file is put last on the commandline.

A simple *"Hello World"* job could look like:

```
arcsub -c ARC0:ldap://grid.ng.org:2135/nordugrid-cluster-name=grid.ng.org,Mds-Vo-name=local,o=grid -f job.jsdl
```

A user has to have valid credentials (see Section 2.1) and be authorised at the specified site. The test file `job.jsdl` is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition
 xmlns="http://schemas.ggf.org/jsdl/2005/11/jsdl"
 xmlns:posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
 <JobDescription>
   <JobIdentification>
     <JobName>Hello World job</JobName>
   </JobIdentification>
   <Application>
     <posix:POSIXApplication>
       <posix:Executable>/bin/echo</posix:Executable>
       <posix:Argument>'Hello World'</posix:Argument>
       <posix:Output>out.txt</posix:Output>
       <posix:Error>err.txt</posix:Error>
     </posix:POSIXApplication>
   </Application>
   <DataStaging>
     <FileName>out.txt</FileName>
     <CreationFlag>overwrite</CreationFlag>
     <DeleteOnTermination>false</DeleteOnTermination>
   </DataStaging>
   <DataStaging>
     <FileName>err.txt</FileName>
     <CreationFlag>overwrite</CreationFlag>
     <DeleteOnTermination>false</DeleteOnTermination>
   </DataStaging>
 </JobDescription>
</JobDefinition>
```

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. Job IDs differ strongly between Grid flavours, but basically they have a form of a URL. You should use Job ID as a handle to refer to the

job when doing other job manipulations, such as querying job status (`arcstat`), killing it (`arckill`), re-submitting (`arcresub`), or retrieving the result (`arcget`).

> 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 2.3).

The job description in xRSL or JSDL format can be given either as an argument on the command line, 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 `arcsub` command.

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:

```
arcsub -c alias1 -c alias2 myjob.xrsl
```

This will submit a job to either `alias1` or `alias2`. To submit a job to any site except `badsite`, use `-` sign in front of the name:

```
arcsub -c -badsite myjob.xrsl
```

If option `-c` is not given, the `arcsub` command locates the available sites by querying the Information System. Default index services for the Information System are specified in the configuration template distributed with the middleware, and can be overwritten both in the user's configuration (see Section 4) and from the command line using option `-i`. Different Grids use different notation for such index services.

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:

```
arcsub -d DEBUG 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 in the user's configuration file.

It often happens that some sites that `arcsub` 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 20 seconds) with the `-t` option:

```
arcsub -t 5 myjob.xrsl
```

A default value for the timeout can be set in the user's configuration file.

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

Possible broker values for the `arcsub` command line option `-b` are:

- `Random` – ranks targets randomly (default)
- `FastestQueue` – ranks targets according to their queue length
- `Benchmark[:name]` – ranks targets according to a given benchmark, as specified by the `name`. If no benchmark is specified, CINT2000 * is used
- `Data` – ranks targets according the amount of megabytes of the requested input files that are already in the computing resources cache.

*http://www.spec.org/cpu2000/CINT2000/

- `Python:<module>.<class>[:arguments]` – ranks targets using any user-supplied custom Python broker module, optionally with broker arguments. Such module can reside anywhere in user's `PYTHONPATH`
- `<otherbroker>[:arguments]` – ranks targets using any user-supplied custom C++ broker plugin, optionally with broker arguments. Default location for broker plugins is `/usr/lib/arc` (may depend on the operating system), or the one specified by the `ARC_PLUGIN_PATH`.

To write a custom broker in C++ one has to write a new specialization of the `Broker` base class and implement the `SortTargets` method in the new class. The class should be compiled as a loadable module that has the proper ARC plugin descriptor for the new broker. For example, to build a broker plugin "MyBroker" one executes:

```
g++ -I /arc-install/include \
    -L /arc-install/lib \
    `pkg-config --cflags glibmm-2.4 libxml-2.0` \
    -o libaccmybroker.so -shared MyBroker.cpp
```

For more details, refer to *libarclib* documentation [4].

2.2.3 arcstat

arcstat [options] [job ...]

(ARC 0.9)

Options:

`-a, --all`		all jobs
`-i, --joblist`	*filename*	file containing a list of jobIDs
`-c, --cluster`		show information about a site (cluster)
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-i, --indexurl`	*url*	URL of an index service
`-l, --long`		long format (extended information)
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-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. Then `-c` and `-i` accept arguments in the GRID:URL notation explained in the description of `arcsub`.

Different sites may report slightly different job states, depending on the installed software version.

2.2.4 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, --cluster`		show information about clusters
`-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
`-l, --gmlog`		show the grid manager's error log of the job
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-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).

2.2.5 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)
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-D, --dir`	*dirname*	download directory (the job directory will be created in this directory)
`-k, --keep`		keep files on gatekeeper (do not clean)
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-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.

2.2.6 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
`-j, --joblist`	*filename*	file containing a list of jobIDs
`-c, --cluster`		show information about clusters
`-s, --status`	*statusstr*	only select jobs whose status is *statusstr*
`-k, --keep`		keep files on gatekeeper (do not clean)
`-t, --timeout`	*time*	timeout for queries (default 20 sec)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-v, --version`		print version information
`-h, --help`		print help page

Arguments:

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

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

2.2.7 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
`-j, --joblist`	*filename*	file containing a list of jobIDs

`-c, --cluster`	`[-]`textemname	explicitly select or reject a specific site (cluster)
`-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 20 sec)
`-d, --debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-z, --conffile`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-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.

2.3 Data manipulation

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

2.3.1 arcls

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

arcls [options] <URL>

(ARC 0.9)

Options:		
`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-l`		detailed listing
`-L`		detailed listing including URLs from which file can be downloaded
`-m`		display all available metadata
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:

```
arcls -L rls://rls.nordugrid.org:38203/logical_file_name
arcls -l gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535
```

```
arcls srm://grid.uio.no:8446/srm/managerv2?SFN=/johndoe/log2
```

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.

2.3.2 arccp

`arccp` is a powerful tool to copy files over the Grid. It is a part of the A-REX, but can be used by the User Interface as well.

arccp [options] <source> <destination>

(ARC 0.9)

Options:

`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or 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)
`-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.

`arccp` 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.

Usage examples of `arccp` are:

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

2.3.3 arcrm

The `arcrm` command allows users to erase files at any location specified by a valid URL.

arcrm [options] <source>

(ARC 0.9)

Options:		
`-h`		short help
`-v`		print version information
`-d`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-c`		continue with meta-data even if it failed to delete real file
Arguments:		
`source`		source URL

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

Here is an `arcrm` example:

```
arcrm lfc://grid.uio.no/grid/atlas/AOD_0947.pool.root
```

2.3.4 arcacl

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

arcacl [options] get|put <URL>

(ARC 0.9)

Options:		
`-d, -debug`	*debuglevel*	debug level is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-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

The 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:

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

2.3.5 arctransfer

The `arctransfer` command is not implemented.

2.3.6 chelonia

`chelonia` is a client tool for accessing the Chelonia storage system. With it you can create, remove and list file collections, upload, download and remove files, and move and stat collections and files, using Logical Names (LN).

chelonia [options] <method> [arguments]

(ARC 0.9)

Options:

`-b`	*URL*	URL of Bartender to connect
`-x`		print SOAP XML messages
`-v`		verbose mode
`-z`	*filename*	configuration file (default $HOME/.arc/client.conf)
`-w`		allow to run without the ARC python client libraries (with limited functionality)

Methods:

`stat`	*LN [LN ...]*	get detailed information about an entry or several
`makeCollection, make, mkdir`	*LN*	create a collection
`unmakeCollection, unmake, rmdir`	*LN*	remove an empty collection
`list, ls`	*LN*	list the content of a collection
`move, mv`	*source target*	move entries within the namespace (both LNs)
`putFile, put`	*source target*	upload a file from a *source* to a *target* (both specified as LNs))
`getFile, get`	*source [target]*	download a file from a *source* to a *target*
`delFile, del, rm`	*LN [LN ...]*	remove file(s))
`modify, mod`	*string*	modify metadata
`policy, pol`	*string*	modify access policy rules. The string has a form <LN> <changeType> <identity> <action list>. <changeType> could be 'set', 'change' or 'clear' 'set': sets the action list to the given user overwriting the old one 'change': modify the current action list with adding and removing actions 'clear': clear the action list of the given user <identity> could be a '<UserDN>' or a 'VOMS:<VO name>'

[†]In ARC $\leq$ 0.5.28, `set` was used instead of `put`

		<action list> is a list actions prefixed with '+' or '-', e.g. '+read +addEntry -delete'; possible actions are: `read`, `addEntry`, `removeEntry`, `delete`, `modifyPolicy`, `modifyStates`, `modifyMetadata`
`unlink`	*string*	remove a link to an entry from a collection without removing the entry itself
`credentialsDelegation, cre`	*string*	delegate credentials for using gateway
`removeCredentials, rem`	*string*	remove previously delegated credentials
`makeMountPoint, makemount`	*string*	create a mount point

Without arguments, each method prints its own help.

Examples:

```
chelonia list /
chelonia put orange /
chelonia stat /orange
chelonia get /orange /tmp
chelonia mkdir /fruits
chelonia mkdir /fruits/apple
chelonia mv /orange /fruits
chelonia ls /fruits
chelonia rmdir /fruits/apple
chelonia rmdir /fruits
chelonia rm /fruits/orange
chelonia policy / change ALL +read +addEntry
chelonia modify /pennys-orange set states neededReplicas 2
```

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 [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)* [7]
`rls`	Globus® *Replica Location Service (RLS)* [8]
`lfc`	LFC catalog and indexing service of EGEE gLite [9]
`srm`	Storage Resource Manager (SRM) service [10]
`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:

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

For the SRM service, the syntax is

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

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:

`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
`metadataoption`	Metadata option for indexing service

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`)
`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
`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`
`tcpnodelay=yes|no`	controls the use of the TCP_NODELAY socket option (which disables the Nagle algorithm). Applies to http(s) only. 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.

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 adler32
`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
rls://gsiftp://se.domain.org/datapath/file25.dat@grid.domain.org:61238/myfile02.dat[1]
file:///home/auser/griddir/steer.cra
lfc://srm://srm.domain.org/griddir@lfc.domain.org/user/file1:guid=\
    bc68cdd0-bf94-41ce-ab5a-06a1512764dc:checksumtype=adler32:checksumvalue=12345678[2]
lfc://lfc.domain.org;cache=no/: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.

Chapter 4

ARC Client Configuration

Default behaviour of an 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. Some client tools may be able to create the default `$HOME/.arc/client.conf`, if it does not exist.

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

The configuration file is written in a plain text format known as INI. Configuration blocks start with identifying keywords inside square brackets. Typically, first comes a common block: `[common]`. Thereafter follows one or more attribute-value pairs written one on each line in the following format:

```
[common]
attribute1=value1
attribute2=value2
attribute3=value3 value4
# comment line 1
# comment line 2
...
```

Most attributes have counterpart command line options. Command line options always overwrite configuration attributes.

Two blocks are currently recognized, `[common]` and `[alias]`. Following sections describe supported attributes per block.

4.1 Block [common]

defaultservices

This attribute is multi-valued.

This attribute is used to specify default services to be used. Defining such in the user configuration file will override the default services set in the system configuration.

The value of this attribute should follow the format:

```
service_type:flavour:service_url
```

where `service_type` is type of service (e.g. `computing` or `index`), `flavour` specifies type of middleware plugin to use when contacting the service (e.g. ARC0, ARC1, CREAM, UNICORE, etc.) and `service_url` is the URL used to contact the service. Several services can be listed, separated with a blank space (no line breaks allowed).

Example:

```
defaultservices=index:ARC0:ldap://index1.ng.org:2135/Mds-Vo-name=testvo,o=grid
␣index:ARC1:https://index2.ng.org:50000/isis
␣computing:ARC1:https://ce.arc.org:60000/arex
␣computing:CREAM:ldap://ce.glite.org:2170/o=grid
␣computing:UNICORE:https://ce.unicore.org:8080/test/services/BESFactory?res=default_bes_factory
```

rejectservices

This attribute is multi-valued.

This attribute can be used to indicate that a certain service should be rejected ("blacklisted"). Several services can be listed, separated with a blank space (no line breaks allowed).

Example: `rejectservices=computing:ARC1:https://bad.service.org/arex`

verbosity

Default verbosity (debug) level to use for the ARC clients. Corresponds to the `-d` command line option of the clients. Default value is `WARNING`, possible values are `FATAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG` or `VERBOSE`.

Example: `verbosity=INFO`

timeout

Sets the period of time the client should wait for a service (information, computing, storage etc) to respond when communicating with it. The period should be given in seconds. Default value is 20 seconds. This attribute corresponds to the `-t` command line option.

Example: `timeout=10`

brokername

Configures which brokering algorithm to use during job submission. This attribute corresponds to the `-b` command line option. The default one is the `Random` broker that chooses targets randomly. Another possibility is, for example, the `FastestQueue` broker that chooses the target with the shortest estimated queue waiting time. For an overview of brokers, please refer to Section 2.2.2.

Example: `brokername=Data`

brokerarguments

This attribute is used in case a broker comes with arguments. This corresponds to the parameter that follows column in the `-b` command line option.

Example: `brokerarguments=cow`

joblist

Path to the job list file. This file will be used by commands such as `arcsub`, `arcstat`, `arcsync` etc. to read and write information about jobs. This attribute corresponds to the `-j` command line option. The default location of the file is in the $HOME/.arc/client.conf directory with the name `jobs.xml`.

Example:

```
joblist=/home/user/run/jobs.xml
joblist=C:\\run\jobs.xml
```

bartender

Specifies default *Bartender* services. Multiple Bartender URLs should be separated with a blank space. These URLs are used by the `chelonia` command line tool, the Chelonia FUSE plugin and by the data tool commands `arccp`, `arcls`, `arcrm`, etc..

Example: `bartender=http://my.bar.com/tender`

proxypath

Specifies a non-standard location of proxy certificate. It is used by `arcproxy` or similar tools during proxy generation, and all other tools during establishing of a secure connection. This attribute corresponds to the `-P` command line option of `arcproxy`.

Example: `proxypath=/tmp/my-proxy`

keypath

Specifies a non-standard location of user's private key. It is used by `arcproxy` or similar tools during proxy generation. This attribute corresponds to the `-K` command line option of `arcproxy`.

Example: `keypath=/home/username/key.pem`

certificatepath

Specifies a non-standard location of user's public certificate. It is used by `arcproxy` or similar tools during proxy generation. This attribute corresponds to the `-C` command line option of `arcproxy`.

Example: `certificatepath=/home/username/cert.pem`

cacertificatesdirectory

Specifies non-standard location of the directory containing CA-certificates. This attribute corresponds to the `-T` command line option of `arcproxy`.

Example: `cacertificatesdirectory=/home/user/cacertificates`

cacertificatepath

Specifies an explicit path to the certificate of the CA that issued user's credentials.

Example: `cacertificatepath=/home/user/myCA.0`

vomsserverpath

Specifies non-standard path to the file which contians list of VOMS services and associated configuration parameters. This attribute corresponds to the `-V` command line option of `arcproxy`.

Example: `vomsserverpath=/etc/voms/vomses`

username

Sets default username to be used for requesting credentials from Short Lived Credentials Service. This attribute corresponds to the `-U` command line option of `arcslcs`.

Example: `username=johndoe`

password

Sets default password to be used for requesting credentials from Short Lived Credentials Service. This attribute corresponds to the `-P` command line option of `arcslcs`.

Example: `password=secret`

keypassword

Sets default password to be used to encode the private key of credentials obtained from a Short Lived Credentials Service. This attribute corresponds to the `-K` command line option of `arcslcs`.

Example: `keypassword=secret2`

keysize

Sets size (strength) of the private key of credentials obtained from a Short Lived Credentials Service. Default value is 1024. This attribute corresponds to the `-Z` command line option of `arcslcs`.

Example: `keysize=2048`

certificatelifetime

Sets lifetime (in hours, starting from current time) of user certificate which will be obtained from a Short Lived Credentials Service. This attribute corresponds to the `-L` command line option of `arcslcs`.

Example: `certificatelifetime=12`

slcs

Sets the URL to the Short Lived Certificate Service. This attribute corresponds to the `-S` command line option of `arcslcs`.

Example: `slcs=https://127.0.0.1:60000/slcs`

storedirectory

Sets directory which will be used to store credentials obtained from a Short Lived Credential Servise. This attribute corresponds to the `-D` command line option of `arcslcs`.

Example: `storedirectory=/home/mycredentials`

idpname

Sets Identity Provider name (Shibboleth) to which user belongs. It is used for contacting Short Lived Certificate Services. This attribute corresponds to the `-I` command line option of `arcslcs`.

Example: `idpname=https://idp.testshib.org/idp/shibboleth`

4.2 Block [alias]

Users often prefer to submit jobs to a specific site; since contact URLs (and especially end-point references) are very long, it is very convenient to replace them with aliases. Block `[alias]` simply contains a list of alias-value pairs.

Alias substitutions is performed in connection with the `-c` command line switch of the ARC clients.

Aliases can refer to a list of services (separated by a blank space).

Alias definitions can be recursive. Any alias defined in a list that is read before a given list can be used in alias definitions in that list. An alias defined in a list can also be used in alias definitions later in the same list.

Examples:

```
[alias]

arc0=computing:ARC0:ldap://ce.ng.org:2135/nordugrid-cluster-name=ce.ng.org,Mds-Vo-name=local,o=grid
arc1=computing:ARC1:https://arex.ng.org:60000/arex
cream=computing:CREAM:ldap://cream.glite.org:2170/o=grid
unicore=computing:UNICORE:https://bes.unicore.org:8080/test/services/BESFactory?res=default_bes
crossbrokering=arc0 arc1 cream unicore
```

4.3 Deprecated configuration files

ARC configuration file in releases 0.6 and 0.8 has the same name and the same format. Only one attribute is preserved (`timeout`); other attributes unknown to newer ARC versions are ignored.

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

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

[3] F. Pacini and A. Maraschini, *Job Description Language attributes specification*, 2007, EGEE-JRA1-TEC-590869-JDL-Attributes-v0-8. [Online]. Available: https://edms.cern.ch/document/590869/1

[4] M. Ellert, B. Mohn, I. Márton, and G. Rőczei, *libarcclient – A Client Library for ARC*, The NorduGrid Collaboration, NORDUGRID-TECH-20. [Online]. Available: http://www.nordugrid.org/documents/client_technical.pdf

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

[6] 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.

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

[8] 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.

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

[10] A. Sim, A. Shoshani, *et al.*, "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

Index