NORDUGRID-MANUAL-?
28/8/2009

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. A special utility also exists for test purposes. These commands replace the commands `ng*` (`ngsub, ngget` etc.) found in ARC versions previous to 0.9, and provide enhanced functionality.

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 [1].

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]

(ARC 0.9)

Options:

`-c, --cluster`	`[-]`*url*	explicitly select or reject a specific site (cluster)
`-i, --indexurl`	*url*	URL of an 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 the job IDs will be stored
`-D, --dryrun`		add dryrun option to the job description
`-x, --dumpdescription`		do not submit – dump transformed job description to stdout
`-t, --timeout`	*time*	timeout for queries in seconds (default 20)
`-d, --debug`	*debuglevel*	debug level, FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE - default WARNING
`-U, --unknownattr`		allow unknown attributes in the job description
`-v, --version`		print version information
`-?, --help`		print help page

The -c and -i arguments accept “urls” of the format GRID:URL,
e.g. for index servers:
ARC0:ldap://grid.tsl.uu.se:2135/mds-vo-name=sweden,O=grid
CREAM:ldap://cream.grid.upjs.sk:2170/o=grid
or clusters:
ARC0:ldap://grid.tsl.uu.se:2135/nordugrid-cluster-name=grid.tsl.uu.se,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.tsl.uu.se:2135/nordugrid-cluster-name=grid.tsl.uu.se,Mds-Vo-name=local,o=grid -f job.jsdl

Assuming that the url represents an ARC0 cluster that the user is authorized to submit to and job.jsdl is the file 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>
```

2.1.2 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
`-q, --queues`		show information about clusters and queues
`-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
`-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.

2.1.3 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
`-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).

2.1.4 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
`-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.

2.1.5 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
`-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.

2.1.6 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
`-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
`-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.

2.2 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 is one of FATAL, ERROR, WARNING, INFO, DEBUG or VERBOSE
`-l`		detailed listing
`-L`		detailed listing including URLs from which file can be downloaded and temporary cached locations
`-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 rls://rc.host:38203/logical_file_name
arcls -l gsiftp://lscf.nbi.dk:2811/jobs/1323842831451666535
arcls -L srm://grid.uio.no:58000/srm/managerv1/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.

2.2.2 arccp

The `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 \
          rc://;threads=4@grid.uio.no/lc=Collection,rc=Catalog/zebra4.f
arccp http://www.nordugrid.org/data/somefile gsiftp://hathi.hep.lu.se/data/
```

2.2.3 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 (RC, 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 rc://grid.uio.no/lc=Collection,rc=Catalog/badfile
```

2.2.4 arcacl

This command retrieves or modifies access control information associated with a stored object if service supports GridSite GACL language [2] 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

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
```

2.2.5 arctransfer

The `arctransfer` This command is not implemented.

2.3 Test suite

2.4 Third-party commands

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

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 [3]
`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)* [4]
`rc`	Globus® *Replica Catalog (RC)* [5]
`rls`	Globus® *Replica Location Service (RLS)* [6]
`fireman`	Fireman indexing service of EGEE gLite [7]
`lfc`	LFC catalog and indexing service of EGEE gLite [7]
`se`	ARC Smart Storage Element service [8]
`srm`	Storage Resource Manager (SRM) service [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:

```
rc://rc://[location[|location[...]]@]<host>[:port]/<DN>/<lfn>
rls://[url[|url[...]]@]<host>[:port]/<lfn>
fireman://[url[|url[...]]@]<host>[:port]/<service_path>?<lfn>
lfc://[url[|url[...]]@]<host>[:port]/<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
`url`	URL of the file as registered in RLS/Fireman
`service_path`	End-point path of the Web service
`file`	local to the host file name with a full path

The following options are supported 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`	specifies the algorithm for checksum to be computed (ev. 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` in a source URL will disable checksum verification.
`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
`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

Bibliography

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

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

[3] 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.

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

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

[6] 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.

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

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

[9] 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