NORDUGRID-MANUAL-14
28/9/2011

Chelonia User's manual

Zsombor Nagy*
Jon Nilsen†
Salman Zubair Toor ‡

*zsombor@niif.hu
†j.k.nilsen@usit.uio.no
‡salman.toor@it.uu.se

Contents

1 Clients for Chelonia

1.1 chelonia

`chelonia` is a client tool for accessing the Chelonia storage system. It is capable of creating, removing and listing collections, uploading, downloading and removing files and moving and stating both, using Logical Names (LN). Collections contain files and other collections, creating a hierarchical namespace.

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
`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. Detailed explanation of each method is given below.

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
```

1.1.1 stat

With the `stat` method it is possible to get all the metadata about one or more entry (file, collection, etc.). The entries are specified with their Logical Name (LN).

chelonia stat <LN> [<LN> ...]

The output contains key-value pairs grouped in sections. The 'states' section contains the size and the checksum of a file, the number of needed replicas, and whether a collection is closed or not (a 'closed' collection should not be modified anymore, if it gets modified, its state becomes 'broken'); the 'entry' section contains the DN of the owner, the globally unique ID (GUID) of the entry, and the type of the entry (file, collection, etc.); the 'parents' section contains the GUID of the parent collection(s) of this entry, and the name of this entry in that collection separated with a '/'; the 'locations' sections contains the location of the replicas of a file, which contains of the ID (the URL) of the storage element, the ID of the replica within the storage element, and the state of the replica; the 'timestamps' section contains the creation time of the entry; the 'entries' section contains the name and GUID of the entries of a collection. Example stat of a file:

```
$ chelonia stat /thing
'/thing': found
  states
    checksumType: md5
    neededReplicas: 3
    size: 6
    checksum: a0186a90393bd4a639a1ce35d8ef85f6
  entry
    owner: /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Nagy Zsombor
    GUID: 398CBDEA-E282-4735-8DF6-2464CD00BE2D
    type: file
  parents
    0/thing: parent
  locations
    https://localhost:60000/Shepherd D519F687-EF65-4AEA-9766-E6E2D42166C4: alive
  timestamps
    created: 1257351119.3
```

Example stat of a collection:

```
$ chelonia stat /
'/': found
  states
    closed: no
  entry
    owner: /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Nagy Zsombor
    GUID: 0
    type: collection
  timestamps
    created: 1257351114.37
  entries
    thing: 398CBDEA-E282-4735-8DF6-2464CD00BE2D
```

1.1.2 makeCollection

With the `makeCollection` or `mkdir` method it is possible to create a new empty collection. The requested Logical Name (LN) should be specified.

chelonia makeCollection <LN>

The parent collection of the requested Logical Name must exist.

Example output of the method:

```
$ chelonia mkdir /newcoll
Creating collection '/newcoll': done

$ chelonia mkdir /nonexistent/newcoll
Creating collection '/nonexistent/newcoll': parent does not exist
```

1.1.3 unmakeCollection

With the `unmakeCollection` or `rmdir` method it is possible to delete an empty collection which is specified by its Logical Name (LN).

chelonia unmakeCollection <LN>

Example output of the method:

```
$ chelonia rmdir /newcoll
Removing collection '/newcoll': removed

$ chelonia rmdir /dir
Removing collection '/dir': collection is not empty
```

1.1.4 list

With the `list` or `ls` method it is possible to list the contents of one or more collections which are specified by their Logical Name (LN).

chelonia list <LN> [<LN> ...]

Example output of the method:

```
$ chelonia list / /newcoll
'/newcoll': collection
    empty.
'/': collection
  thing   <file>
  dir     <collection>
  newcoll <collection>
```

1.1.5 move

With the `move` or `mv` method it is possible to move a file or collection within the namespace of chelonia (including renaming the entry). The source path and the target path should be specified as Logical Names

chelonia move <sourceLN> <targetLN>

Example output of the method:

```
$ chelonia mv /thing /newcoll/
Moving '/thing' to '/newcoll/': moved

$ chelonia mv /newcoll/thing /newcoll/othername
Moving '/newcoll/thing' to '/newcoll/othername': moved
```

1.1.6 putFile

With the `putFile` or `put` method it is possible to upload a new file into the system creating a new Logical Name (LN). It can upload directories recursively using the `-r` flag. It is also possible the specify the number of needed replicas.

chelonia putFile [-r] <source filename> <target LN> [<number of replicas needed>]

Example output of the method:

```
$ chelonia put thing /newcoll/
'thing' (6 bytes) uploaded as '/newcoll/thing'.
```

1.1.7 getFile

With the `getFile` or `get` method it is possible to download a file specified with its Logical Name (LN). If the target local path is not given, then the file will be put into the local directory. It can download collections recursively using the `-r` flag.

chelonia getFile [-r] <source LN> [<target filename>]

Example output of the method:

```
$ chelonia get /newcoll/thing newlocalname
'/newcoll/thing' (6 bytes) downloaded as 'newlocalname'.
```

1.1.8 delFile

With the `delFile` or `rm` method it is possible to delete one or more files from the system.

chelonia delFile <LN> [<LN> ...]

Example output of the method:

```
$ chelonia rm /newcoll/othername
/newcoll/othername: deleted
```

1.1.9 modify

With the `modify` or `mod` method it is possible to modify some metadata of an entry.

chelonia modify <LN> <changeType> <section> <property> <value>

The possible values of 'changeType' are 'set' (sets the property to value within the given section), 'unset' (removes the property from the given section - the 'value' does not matter) and 'add' (sets the property to value within the given section only if it does not exist yet).

To change the number of needed replicas for a file:

```
chelonia modify <LN> set states neededReplicas <number of needed replicas>
```

To close a collection:

```
chelonia modify <LN> set states closed yes
```

A closed collection should not be modified later. If it gets modified its state becomes 'broken'.

To change metadata key-value pairs:

```
chelonia modify <LN> set|unset|add metadata <key> <value>
```

1.1.10 policy

With the `policy` or `pol` method it is possible to modify the policy of the entry

chelonia policy <LN> <changeType> <identity> <action list>

The possible values of 'changeType' are 'set' (sets the action list to the given user overwriting the old one), 'change' (modify the current action list with adding and removing actions) and 'clear' (clear the action list of the given user).

The 'identity' could be currently three things: the DN of a user; the name of a VO (with the syntax: 'VOMS:<VO name>'); or 'ALL' for all users.

The 'action list' is a list of actions prefixed with '+' or '-', e.g. '+read +addEntry -delete'.

These are the actions which can be used for access control:

- *read*: user can get the list of entries in the collection; user can download the file
- *addEntry*: user can add a new entry to the collection;
- *removeEntry*: user can remove any entry from the collection
- *delete*: user can delete the collection if it is empty; user can delete a file
- *modifyPolicy*: user can modify the policy of the file/collection
- *modifyStates*: user can modify some special metadata of the file/collection (close the collection, change the number of needed replica of the file)
- *modifyMetadata*: user can modify the arbitrary metadata section of the file/collection (these are property-value pairs)

There is an implicit default policy: the owner always has all the rights. Checking the 'stat' of new collections:

```
$ chelonia stat /newcoll
'/newcoll': found
  states
    closed: no
  entry
```

```
    owner: /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=Nagy Zsombor
    GUID: 41CBD461-09BE-46FD-8A1B-767C7D427AF9
    type: collection
  parents
    0/newcoll: parent
  timestamps
    created: 1257435820.26
  entries
    thing: A63658B4-2C6E-46A3-8238-7D291F8F81C2
```

shows no policies, but it shows the owner. This collection has no additional policies just the default one: the owner can do anything, noone else can do anything.

Let's set it in a way that all users can read the contents of this collection:

```
$ chelonia policy /newcoll change ALL +read
Setting action list of '/newcoll' for user ALL to +read: set.
$ chelonia stat /newcoll
'/newcoll': found
  [...]
  policy
    ALL: +read
  [...]
```

Then we can set that all the members of the knowarc VO would be able to add entries to this collection:

```
$ chelonia policy /newcoll change VOMS:knowarc +addEntry
Setting action list of '/newcoll' for user VOMS:knowarc to +addEntry: set.
$ chelonia stat /newcoll
'/newcoll': found
  [...]
  policy
    ALL: +read
    VOMS:knowarc: +addEntry
  [...]
```

And for example we can set a specific user to be able to remove entries from this collections:

```
$ chelonia policy /newcoll change \
    "/C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=TestUser" +removeEntry
Setting action list of '/newcoll'
    for user /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=TestUser to +removeEntry: set.
$ chelonia stat /newcoll'/newcoll': found
  [...]
  policy
    /C=HU/O=NIIF CA/OU=GRID/OU=NIIF/CN=TestUser: +removeEntry
    ALL: +read
    VOMS:knowarc: +addEntry
  [...]
```

1.1.11 unlink

With the `unlink` method it is possible to remove a file or collection just from its parent collection without removing the file or collection itself. This means that the file or collection wouldn't be part of its former parent collection anymore. It would be still possible to access it with its GUID, or if it was linked from another collection too. **NOTE**: If we don't know the GUID or the logical name of any other link, then we cannot access the file or collection anymore.

chelonia unlink <LN>

If there is a file called '/newcoll/thing', it is in the listing of the '/newcoll' collection:

```
$ chelonia list /newcoll
'/newcoll': collection
  thing <file>
```

The file is in the entries of the collection:

```
$ chelonia stat /newcoll
'/newcoll': found
  entries
    thing: A63658B4-2C6E-46A3-8238-7D291F8F81C2
  [...]
```

It is possible the 'stat' the file with the Logical Name '/newcoll/thing':

```
jim:~ zsombor$ chelonia stat /newcoll/thing
'/newcoll/thing': found
  states
    checksumType: md5
    neededReplicas: 3
    size: 6
    checksum: a0186a90393bd4a639a1ce35d8ef85f6
  [...]
```

Now with the 'unlink' method it is possible to remove the file from the '/newcoll' collection, but not from the system:

```
$ chelonia unlink /newcoll/thing
Unlinking '/newcoll/thing': unset
```

Now the file is not in the collection anymore:

```
$ chelonia list /newcoll
'/newcoll': collection
    empty.
$ chelonia stat /newcoll/thing
'/newcoll/thing': not found
```

But with the GUID of the file, it can still be accessed:

```
$ chelonia stat A63658B4-2C6E-46A3-8238-7D291F8F81C2
'A63658B4-2C6E-46A3-8238-7D291F8F81C2': found
  states
    checksumType: md5
    neededReplicas: 3
    size: 6
    checksum: a0186a90393bd4a639a1ce35d8ef85f6
  [...]
```

1.1.12 credentialDelegation

With the `credentialDelegation` or `cre` method it is possible to delegate credentials to the Bartender.

chelonia credentialDelegation

1.1.13 removeCredentials

With the `removeCredentials` or `rem` method it is possible to remove the previously delegated credentials.

chelonia removeCredentials

1.1.14 makeMountPoint

With the `makeMountPoint` or `makemount` method it is possible to create a mount point within the namespace of Chelonia which points to a GridFTP server.

chelonia makeMountPoint <LN> <URL>

The 'LN' is the requested Logical Name for the mount point, the 'URL' points to the GridFTP server.

1.1.15 unmakeMountPoint

With the `unmakeMountPoint` or `unmount` method it is possible to remove a previously created mount point.

chelonia unmakeMountPoint <LN>

The 'LN' is the Logical Name of the mount point.