DataLab Command Line Client

The DataLab Command Line Client (DCLC) is a Python-based package that provides an alternate way to interact with the various DataLab services. It is invoked via the **datalab** command.

The list of supported DataLab tasks can be obtained via **datalab -help:**

```
> datalab -help
Syntax: datalab [task] where [task] is one of:
  status - report on the user status
  addcapability - activate a capability on a VOSpace container
  exec - launch a remote task in the DataLab
  ln - link a file in DataLab
  mkdir - create a directory in DataLab
  mount - mount the default VOSpace
  get - get a file from DataLab
  listcapability - list the capabilities supported by this VOSpace
  launch - launch a plugin
  broadcast - broadcast a SAMP message
  mv - move a file in DataLab
  logout - logout of the DataLab
  ls - list a location in DataLab
  tag - tag a file in DataLab
  rm - delete a file in DataLab
  put - put a file into DataLab
  query - query a remote data service in the DataLab
  login - login to the DataLab
  cp - copy a file in DataLab
  rmdir - delete a directory in DataLab
```

All subcommands take the optional arguments:
debug - print debug log level messages [optional]
verbose - print verbose level log messages [optional]
warning - print warning level log messages [optional]

If a required argument is not specified on the command line, a prompt will be given for it. If you are specifying an argument on the command line then you need to put two dashes "--" in front of the argument name and an equals before the value of the argument: —argument=value.

Referencing files in DataLab

When you want to refer to a file in the DataLab (also called a remote file), you need to put a **vos://** prefix before it so that the DCLC knows it is a remote file you are referring to. If you want to be really precise, you can use the full identifier for the file (also known as the VOSpace identifier) which would be something like:

vos://datalab.noao.edu!vospace/nodes/sarah/data/table1.vot

However, you can also just use the location within your virtual storage area, in this case - vos://data/table1.vot - and the DCLC will translate this into the proper form for you. Note that if you need to identify a file within someone else's virtual storage, e.g., a data file that a collaborator is sharing with you, then you will need to use the full VOSpace identifier to refer to it.

Task reference

- **status**

This shows the status of the current user: whether they are logged in or not, the list of current jobs/queries?

```
> datalab status
User sarah is logged into the DataLab
```

- **addcapability**

This activates the specified capability in the specified directory by uploading the appropriate capability's configuration from $VOSPACE_CAPSDIR. It takes the following additional arguments:

fmt - the formats to accept [required]
dir - container name [required]
cap - capability name [required]
listcap - list available capabilities [optional]

```
> datalab addcapability —dir=vos://dbs --cap=tableingester --fmt=votable,fits,csv
```

- **exec**

This executes a remote processing job within the DataLab. It takes the following additional arguments:

cmd - name of remote task to run [required]
args - list of key-value arguments to submit to remote task [optional]

```
> datalab exec —cmd=cutout —args="pos=/tmp/vospace/sarah/ltg/ltg.csv, urls=/tmp/vospace/
sarah/ltg/img.vot, outdir=vos://datalab.noao.edu~vospace/sarah/ltg, nthreads=20"
```

- **ln**

This creates a (soft) link to the specified file at the given location. It takes the following additional arguments:

fro - location in DataLab of link [required]
to - location linked points to [required]

```
> datalab ln —fro=vos://dbs —to=http://some/data/file
```

- **mkdir**

This creates the specified directory. It takes the following additional arguments:

dir - directory in DataLab to create [required]

```
> datalab mkdir —dir=vos://test
```

- **mount**

This mounts the specified VOSpace (remote storage) via FUSE to appear as a local filesystem. It takes the following additional arguments:

vospace - VOSpace to mount [required]
mount - mount point for VOSpace [required]
foreground - mount the filesystem as a foreground operation [optional]
cache_limit - upper limit on local diskspace to use for file caching (in MB) [optional]
cache_dir - local directory to use for file caching [optional]
readonly - mount VOSpace readonly [optional]
cache_nodes - cache dataNode Properties [optional]
allow_other - allow all users access to this mountpoint [optional]
max_flush_threads - upper limit on number of flush (upload) threads [optional]
secure_get - use HTTPS instead of HTTP [optional]
nothreads - Only run in a single thread, causes some blocking. [optional]

```
> datalab mount —vospace=… —mount=/tmp/vospace
```

- **get**

This retrieves the specified remote file and optionally saves it to a local file. It takes the following additional arguments:
file - location in DataLab [required]
save - file to save to [optional]

```
> datalab get —file=vos://data/test.vot —save=test.vot
```

- **listcapability**

This lists the capabilities supported by the VOSpace service.

```
> datalab listcapability
The available capabilities are:
  tableingester
```

- **launch - THIS COMMAND IS NOT YET AVAILABLE**

It takes the following additional arguments:

```
> datalab launch
```

- **broadcast**

This sends a SAMP message of the specified type and with the given parameters. It takes the following additional arguments:

type - SAMP message type [required]
pars - Message parameters [required]

```
> datalab broadcast —type=… —pars=…
```

- **mv**

This moves the specified remote file/directory between the two locations. It takes the following additional arguments:

from - location in DataLab to move from [required]
to - location in DataLab to move to [required]

```
> datalab mv —from=vos://data/test.vot —to=vos://work/test.vot
```

- **logout**

This logs out the user from a DataLab session and optionally unmounts their remote storage space. It takes the following additional arguments:

unmount - mountpoint of remote VOSpace [optional]

```
> datalab logout —unmount=/tmp/vospace
Unmounting remote space
You are now logged out of the DataLab
```

- **ls**

This lists a remote directory. It takes the following additional arguments:

from - location in DataLab to list [required]
format - format for listing [required]

```
> datalab ls —from=…
```

- **tag**

This tags a remote file with a user-specified label. It takes the following additional arguments:

file - file in DataLab to tag [required]
tag - tag to add to file [required]

```
> datalab tag —file=vos://dbs/votable.vot —tag="A crucial data file"
```

- **login**

This logs a user into a DataLab session and optionally mounts their remote storage space (see **mount**). It takes the following additional arguments:

user - username of account in DataLab [required]
password - password for account in DataLab [required]
mount - mountpoint of remove VOSpace [optional]

```
> datalab login —user=sarah —password=herr1ng —mount=/tmp/vospace
Welcome to the DataLab, sarah
Initializing mount
```

- **rm**

This deletes a remote file. It takes the following additional arguments:

file - file in DataLab to delete [required]

```
> datalab rm —file=vos://dbs/test.vot
```

- **put**

This uploads a local file to the remote storage space. It takes the following additional arguments:

file - file to put into DataLab [required]
target - location in DataLab [required]

```
> datalab put —file=/home/sarah/simulations/run5.txt —target=vos://dbs/simul1.dat
```

- **query**

This runs a query against either the db directly (synchronous) or via the TAP service (asynchronous). It takes the following additional arguments:

uri - database URI [required]
ofmt - requested output format [optional] - 'csv', 'ascii', 'votable'
out - output filename [required]
sql - input SQL filename [optional]
in - input filename [optional]
async - asynchronous query [optional]
addArgs - additional arguments to pass to the query service [optional]

Note that tables within your MyDB need to be identified with a **mydb://** prefix in either the query or the output argument.

```
> datalab query —adql="select id, ra_j2000, dec_j2000, g, g_i, i_z from lsdr2.stars" —ofmt="csv"
—out="vos://lsdr2.csv"
> datalab query —adql="select id, ra_j2000, dec_j2000, g, g_i, i_z from lsdr2.stars"
—out="mydb://table1"
> datalab query —adql="select id, g_i, i_z from lsdr2.stars l, mydb://table1 m where l.id = m.id"
—ofmt="csv"
> datalab query —sql=complex.sql —async=true —ofmt='csv'
```

- **cp**

This copies a remote file between the two specified locations. It takes the following additional arguments:

from - location in DataLab to copy from [required]
to - location in DataLab to copy to [required]

```
> datalab cp —from=vos://dbs/test.vot —to=vos://results/test.vot
```

- **rmdir**

This deletes a remote directory. It takes the following additional arguments:

dir - directory in DataLab to delete [required]

```
> datalab rmdir —dir=vos://dbs
```

NOT YET IMPLEMENTED IN DATALAB:

- **mktable**

This creates the specified table in a user's MyDB. It takes the following additional arguments:

table - name of the table to be created [required]
schema - SQL schema (file or string) for table [required]

```
> datalab mktable —table="table1" —schema="id bigint, ra double, decl double, zred float, mag float"
```

- **cptable**

This copies a table in the user's MyDB. It takes the following additional arguments:

from - name of table to be copied from [required]
to - name of the table to be copied to [required]

```
> datalab cptable —from="table1" —to="table2"
```

- **mvtable**

This renames a table in the user's MyDB. It takes the following additional arguments:

from - old name of the table [required]
to - new name of the table [required]

```
> datalab mvtable —from="table1" —to="table2"
```

- **rmtable**

This deletes a table in the user's MyDB. It takes the following additional arguments:

table - name of the table to delete [required]

```
> datalab rmtable —table="table2"
```

- **lstable**

This lists the tables in the user's MyDB. If a table is specified then this returns the schema for that table. It takes the following additional arguments:

table - name of the table to list [optional]

```
> datalab lstable
```