FG-iRODS Documentation

FG-iRODS

The FG-iRODS service is proposed by the French France-Grilles Research Infrastructure. Access to the service is nominative and is available on request. Once access to the service has been validated by the France-Grilles team, you will be able to access the service with your certificate or with your password.

To access the service, you will be asked to:

- Sign the conditions of access to the service: http://www.france-grilles.fr/IMG/pdf/iRODS_Service_Policy.pdf;
- Provide a data management plan (it can be a simplified version of DMP).

Getting Started

Environment

First, create the *~/irods/irods_environment.json* file with the following content:

```
{
    "irods_host": "ccirodsfg.in2p3.fr",
    "irods_port": 5555,
    "irods_zone_name": "FranceGrillesZone",
    "irods_user_name": "<username>",
    "irods_default_resource": "<default_resource>",
    "irods_client_server_negotiation": "request_server_negotiation",
    "irods_client_server_policy": "CS_NEG_REQUIRE",
    "irods_default_hash_scheme": "SHA256",
    "irods_default_number_of_transfer_threads": 4,
    "irods_encryption_algorithm": "AES-256-CBC",
    "irods_encryption_key_size": 32,
    "irods_encryption_num_hash_rounds": 16,
    "irods_encryption_salt_size": 8,
    "irods_match_hash_policy": "compatible",
    "irods_maximum_size_for_single_buffer_in_megabytes": 32,
    "irods_ssl_verify_server": "cert"
}
```

The *<username>* and *<default_resource>* should be replaced by the values given to you during the registration procedure.

Client Installation

The client packages are made available for CentOS 7, Ubuntu 16.04 and Ubuntu 18.04.

Instructions for configuring your package manager to include the iRODS *APT* or *YUM*-based repository may be found at https://packages.irods.org.

Once the repository has been configured, install the *irods-icommands* package.

Check that your client installation is working with:

```
~$ iinit
```

The **iinit** permits to open a session to the iRODS instance. This command will ask you for the connection password. Once you have finished working with FG-iRODS, you can close the session with **iexit**.

The **ienv** command display your iRODS environment:

```
irods_version - 4.2.8
irods_client_server_negotiation - request_server_negotiation
irods_encryption_key_size - 32
irods_environment_file - /home/<user>/.irods/irods_environment.json
irods_default_hash_scheme - SHA256
irods_default_number_of_transfer_threads - 4
irods_host - ccirodsfg.in2p3.fr
irods_client_server_policy - CS_NEG_REQUIRE
irods_session_environment_file - /home/<user>/.irods/irods_environment.json.15934
irods_default_resource - <default_resource>
irods_encryption_algorithm - AES-256-CBC
irods_encryption_num_hash_rounds - 16
irods_encryption_salt_size - 8
irods_match_hash_policy - compatible
irods_ssl_verify_server - cert
irods_maximum_size_for_single_buffer_in_megabytes - 32
irods_port - 5555
irods_user_name - <username>
irods_zone_name - FranceGrillesZone
```

Using the iRODS icommands

ihelp permits to display the list of iRODS commands, as well as the help on a specific command:

```
~$ ihelp ils
Usage: ils [-ArlLv] dataObj|collection ...
Usage: ils --bundle [-r] dataObj|collection ...
Display data Objects and collections stored in irods.
Options are:
 -A  ACL (access control list) and inheritance format
 -l  long format
 -L  very long format
 -r  recursive - show subcollections
 -t  ticket - use a read (or write) ticket to access collection information
 -v  verbose
 -V  Very verbose
 -h  this help
 --bundle - list the subfiles in the bundle file (usually stored in the
     /myZone/bundle collection) created by iphybun command.

iRODS Version 4.2.8                ils
```

A full description of the icommands is available in the iRODS documentation.

The Working Directory

The **ils** command permits you to display the data in your iRODS-home directory.

```
~$ ils
/FranceGrillesZone/home/<username>:
```

- *FranceGrillesZone*: the name of the iRODS zone
- */home/<username>*: your default working directory

Uploading Data

In this section, some files will be uploaded to iRODS. First, create an example file, like `foo.txt`.

The file is uploaded to the iRODS server:

```
iput -K foo.txt
```

The *-K* option permits to verify the checksum. The file is now available on the iRODS server:

```
~$ ils
/FranceGrillesZone/home/<username>:
  foo.txt
```

Note: the commands to steer iRODS are very similar to bash commands and can easily be confused!

The file can be deleted with this command:

```
irm foo.txt
```

Logical and Physical Namespace

iRODS provides an abstraction from the physical location of the files, e.g. `/FranceGrillesZone/home/<username>/foo.txt` is the logical path which only iRODS knows. To get more details about the physical namespace, use:

```
~$ ils -L
/FranceGrillesZone/home/<username>:
  <username>         0 mcia;mcia-fgirods1          483 2020-11-20.09:30 & foo.txt
    sha2:veVzp+ApMzyVRzZN0BZIkDyFuqUp/4tM4sLVACp00B8=    generic    /vault1/resc/home/<u
```

The file `foo.txt` that bas been uploaded is known in iRODS as `/FranceGrillesZone/home/<username>/foo.txt`. It is owned by the user *<username>* and lies on the storage resource *mcia*. There is no other replica of that file in the iRODS system (0 in front of *mcia*). The size of the file is 483B. It is stored with a time stamp and a checksum. Actually, the checksum calculation was triggered by the option '-K' of the **iput** command.

Downloading Data

The file stored in iRODS can be downloaded with:

```
~$ iget -K foo.txt foo-restore.txt
```

The `foo.txt` file has been downloaded and renamed to `foo-restore.txt`. With the *-K* option, the checksum of the local file is compared with the checksum of the file on the iRODS server.

Structuring Data

Creating Collections

On your computer, data are organised in folders. In iRODS, you will organising them the same way. However, folders are called *collections*.

To create an iRODS collection:

```
~$ imkdir mycollection
```

The `foo.txt` file can be moved to that collection with:

```
~$ imv foo.txt mycollection
~$ ils -L mycollection
/FranceGrillesZone/home/<username>/mycollection:
  <username>         0 mcia;mcia-fgirods1          483 2020-11-20.10:18 & foo.txt
    sha2:veVzp+ApMzyVRzZN0BZIkDyFuqUp/4tM4sLVACp00B8=    generic    /vault1/resc/home/<u
```

You see that the logical iRODS collection `/FranceGrillesZone/home/<username>/mycollection` has the physical counterpart `/vault1/resc/home/<username>/mycollection`. So data does not end up on the iRODS server randomly but follows the structure.

Data can also be put directly into an iRODS collection:

```
~$ iput -K -r bar.txt mycollection
~$ ils  /FranceGrillesZone/home/<username>/mycollection
/FranceGrillesZone/home/<username>/mycollection:
  bar.txt
  foo.txt
```

The *-r* flag can be used for recursive upload.

Navigating through Collections

To get your current iRODS working directory, use:

```
~$ ipwd
/FranceGrillesZone/home/<username>
```

If you do not specify a full path, but only a partial path like `mycollection/<file>`, iRDS automatically uses the current working directory as a prefix. This directory can be modified with:

```
~$ icd mycollection
```

Managing Metadata

To access the full potential of iRODS, it is required to use metadata.

Creating Metadata

Each file can be annoted with *Attribute*, *Value*, *Unit* triples (AVU). These triples are added to the iRODS database (iCAT) and are searchable. Metadata can be added to a file with:

```
~$ imeta add -d foo.txt 'length' '20' 'words'
```

The Unit field can be empty:

```
~$ imeta add -d foo.txt 'project' 'example'
```

Metadata can also be added to a collection:

```
~$ imeta add -C mycollection 'author' 'John Smith'
```

Listing Metadata

To list metadata on data objects (files), do:

```
~$ imeta ls -d foo.txt
AVUs defined for dataObj /FranceGrillesZone/home/<username>/mycollection/foo.txt:
attribute: length
value: 20
units: words
```

and the following on collections:

```
~$ imeta ls -C mycollection
AVUs defined for collection /FranceGrillesZone/home/<username>/mycollection:
attribute: author
value: John Smith
units:
```

Querying Metadata

To query the iCAT metadata catalogue, the **iquest** is used:

```
~$ iquest "select COLL_NAME, META_COLL_ATTR_VALUE where META_COLL_ATTR_NAME like 'author
COLL_NAME = /FranceGrillesZone/home/<username>/mycollection
META_COLL_ATTR_VALUE = John Smith
----------------------------------------------------------
```

If you are looking for a data object rather than a collection, replace the *META_COLL_ATTR_NAME* attribute with *META_DATA_ATTR_NAME*. There are a lot of predefined attributes that can be used in your searches:

```
~$ iquest attrs
```

The output can be filtered for a specific attribute value:

```
~$ iquest "select COLL_NAME, META_COLL_ATTR_VALUE where META_COLL_ATTR_NAME like 'author
and META_COLL_ATTR_VALUE like 'John%'"
COLL_NAME = /FranceGrillesZone/home/<username>/mycollection
META_COLL_ATTR_VALUE = John Smith
----------------------------------------------------------
```

NOTE: the '%' is a wildcard.

Access Control

iRODS has similar Access Control Lists (ACL) as a unix file system, with read, write and own rights. The current access rights of your data can be checked with:

```
~$ ils -r -A
/FranceGrillesZone/home/<username>/mycollection:
        ACL - jpansanel#FranceGrillesZone:own
        Inheritance - Disabled
  bar.txt
        ACL - <username>#FranceGrillesZone:own
  foo.txt
        ACL - <username>#FranceGrillesZone:own
```

After the *ACL* keyword, the rights are specified. In this case, <*username*> owns all files listed. None else has access rights.

Collections have a *Inheritance* flag. If this flag is set to true, all content of the folder will inherit the accession rights from the folder. The inheritance applies only to newly uploaded files.

To add accession rights to a colleague:

```
~$ ichmod read <colleague> foo.txt
```

The user <*colleague*> can now access the `foo.txt` file.