B2SAFE Hands-on

- Recap on writing rules
- Recap on replicating data between iRODS grids
- Inspect the EUDAT rulebases for handling PIDs and replication
- Integration between data replicas in iRODS and the Handle system

Recap: iRODS simple rules

Example of a simple rule and its execution:

```
HelloWorld{
    if(*name=="<CompareName>"){
        writeLine("stdout", "Hello *name!");
        }
    else { printHello; }
}
INPUT *name="<YourName>"
OUTPUT ruleExecOut, *name
```

B2SAFE rule for PID registration

B2SAFE is a set of iRODS rules that extend the standard iRODS rule set. You can call these rules in your own rules.

You will find two folders in your home directory:

- *rules*: contains some example rules
- *rulebase*: a copy of the EUDAT B2SAFE rulebase with which this iRODS instance is enabled (changing the code in here will NOT change the iRODS server's behaviour, it is merely a reference for you.)

Have a look at the test rule *rules/eudatCreatePid.r*. This rule:

- Creates a a file *test_data.txt*
- What is the content of this file?
- Uses the rule *EUDATCreatePID* to create a PID for that p[articular file. The rule is defined in *rulebase/pid-service.re*
- It then removes the PID and subsequently the file

Let us inspect the rulebase for PIDs *rulesbase/pid-service.re*:

- List of all rules
- First four rules are general functions implementing workflows to create search and update PIDs.
- These rules are followed by functions for very specific tasks, i.e. creating, updating and deleting information in the iCAT catalogue and in the Handle system
- Functions with EUDATi* do something in the iCAT catalogue
- Functions with EUDATe* handle data in the external Handle system

Exercise: PIDs for collections

Write a rule that takes as argument the path to a collection and creates PIDs for the collection, all subcollections and all files. Hint: Have a look at the rules in *rulebase/pid-service.re*.

Solution:

```
createPidColl{
    writeLine("stdout", "Create PIDs for *collPath recursively");
    EUDATPidsForColl(*collPath)
}
INPUT *collPath='/aliceZone/home/<user>/<your collection>'
OUTPUT ruleExecOut
```

Exercise: Retrieve PIDs

Write a rule that retrieves all PIDs given the collection path. Hint: Have a closer look at the function *EUDATiFieldVALUEretrieve*.

Have a look at the iCAT metadata entries and in the Handle system for these files and collections.

- What information can you find in the iCAT catalogue?
- Which informtion can you find in the Handle record at http://hdl.handle.net/?
- Have a closer look at the URL field.
- We will find out about the LOC field later.

Solution

```
fetchPIDs{

    #Get PIDs for files
    foreach (*Row in SELECT DATA_NAME,COLL_NAME WHERE COLL_NAME like '*collPath') {
        *objPath = *Row.COLL_NAME ++ '/' ++ *Row.DATA_NAME;
        EUDATiFieldVALUEretrieve("*objPath", "PID", *value);

        writeLine("stdout", "*Row");
        writeLine("stdout", "*objPath has PID *value");
    }
    #Get PIDs for subcollections
    foreach (*Row in SELECT COLL_NAME WHERE COLL_NAME like '*collPath/%') {
        *subCollPath = *Row.COLL_NAME
        EUDATiFieldVALUEretrieve("*subCollPath", "PID", *value);

        writeLine("stdout", "*Row");
        writeLine("stdout", "*subCollPath has PID *value");
    }

}
INPUT *collPath='/aliceZone/home/<user>/<your collection>'
OUTPUT ruleExecOut
```

Note, that you can only retrieve PIDs in that way when you have access to the iRODS instance. How would you retrieve the PIDs when this is not the case?

Recap: Data transfers between iRODS zones

Exercise

Transfer a data file from */aliceZone/home/* to */bobZone/home/#aliceZone* and introduce a link in the iCAT metadata catalogue so that you can retrieve the copy easily.

You can do that by executing the single commands, or you can write a rule to do the task.

Solution

```
irsync -r i:/aliceZone/home/<user>/<collection> \
    i:/bobZone/home/<user>#aliceZone/<new collection>
ils /bobZone/home/<user>#aliceZone/<new collection> -r

imeta add -C <collection> REPLICA /bobZone/home/<user>#aliceZone/<new collection>
imeta ls -C <collection>
```

B2SAFE rule for replication

Have a look at the B2SAFE rule for replicating data and automatically assigning it a PID. Open the test rule *rules/eudatRepl_coll.r*. This rule

- Creates a local test collection with a data file
- Creates a PID for the original collection
- Assigns the destination for the replica
- Replicates the data
- Assigns PIDs for the original data file, replica collection and replica data file
- Removes replicas, their PIDs and the original data and their PIDs

Example: Copy a file from *aliceZone* to *bobZone*

```
replicateFile{
    *res = EUDATReplication(*sourcefile, *destfile, "true", "false", *response);
}
INPUT *sourcefile='/aliceZone/home/<user>/<file>',
*destfile='/bobZone/home/<user>#aliceZone/<file>'
OUTPUT ruleExecOut
```

The two files are linked by PIDs. Check the iCAT AVUs for the file on *aliceZone*:

```
imeta ls -d testfile1.txt

AVUs defined for dataObj testfile1.txt:
attribute: eudat_dpm_checksum_date:demoResc
value: 01483802842
units:
----
attribute: EUDAT/ROR
value: 21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff
units:
----
attribute: PID
value: 21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff
units:
```

- There is the date of the latest checksum check
- EUDAT/ROR: The PID of the original file
- PID: The PID

Since this is the original file the PID and the ROR are the same. Now let us have a look at the rpelica at

bobZone:

```
imeta ls -d /bobZone/home/alice#aliceZone/testfile1.txt

AVUs defined for dataObj /bobZone/home/alice#aliceZone/testfile1.txt:
attribute: EUDAT/PPID
value: 21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff
units:
----
attribute: eudat_dpm_checksum_date:demoResc
value: 01484127792
units:
----
attribute: EUDAT/ROR
value: 21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff
units:
----
attribute: PID
value: 21.T12996/5ebd89b2-d7e2-11e6-a7ae-fa163ed79f1d
units:
```

Again we see the field for the latest checksum check. The file has a PID, which is different from the ROR. And we see the entry PPID. The PPID is the PID to the direct parent of the replica, whereas the ROR always denotes the very first element of the replication chain. In this example of a replication chain of length 2, the ROR and the PPID are the same.

You see that *bobZone* uses a different Handle prefix, since it is a different administratory domain.

Inspect the Handle records at http://hdl.handle.net/.

Exercise

Write an iRODS rule that replicates a data collection from *aliceZone* to *bobZone*.

Solution

```
replicateFile{
    *res = EUDATReplication(*sourcefile, *destfile, "true", "true", *response);
}
INPUT *sourcefile='/aliceZone/home/<user>/<collection>',
*destfile='/bobZone/home/<user>#aliceZone/<collection>'
OUTPUT ruleExecOut
```

Exercises

1. Build a chain of three or more replicas. You can also use B2SAFE to replicate to a different iRODS path. What information can you get from the Handle registry and what is stored in the iCAT metadata catalogue?
2. Write an iRODS rule to check whether all replicas are still correct.
3. Given only the PID to the original file, retrieve all PIDs of the replicas. Use the B2HANDLE python library or the HANDLE REST API.
4. GridFTP and B2HANDLE: Write a script that up and downloads data to the gridFTP server on eudat-training3 (this gridFTP server is NOT coupled to an iRODS instance). Create PIDs upon upload. The downloading script should use these PIDs. (Example on https://github.com/chStaiger/ELIXIR-gridftp-PID)