Integration of the DIRAC File Catalogue for the BM@N experiment

Zhironkin Igor, JINR

Gertsenberger Konstantin, JINR

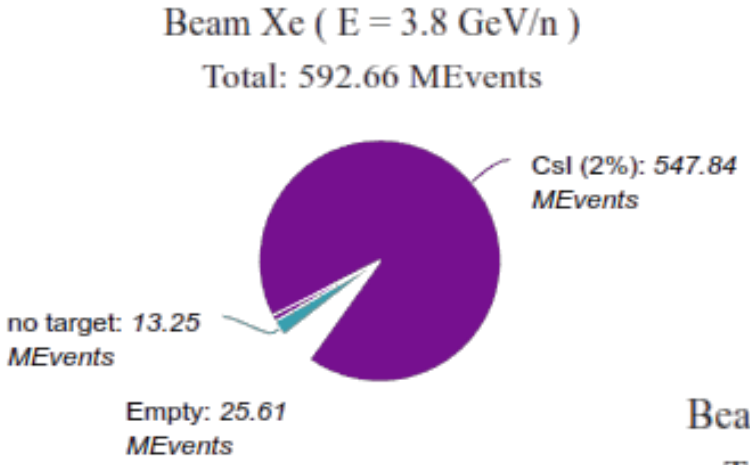
Amount of data

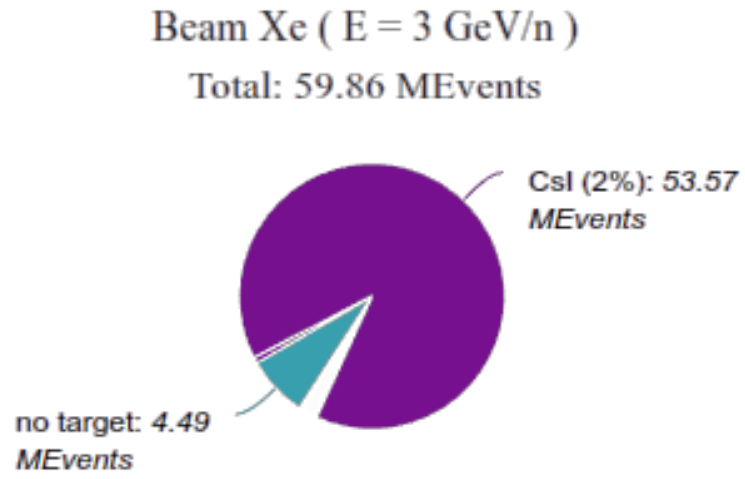
1st Physics BM@N Run

Two beam energy available for *Xe*-beam

CsI target is used as more similar to *Xe*

More than 600M events were collected





RAW → ***DIGIT*** → ***DSTexp*** → PhA

RAW: raw (binary) event data collected by the DAQ system after the Event Builder

DIGIT: detector readings (event digits) after the digitizer macro

DSTexp: reconstructed data of experimental events

Experimental data

645 x 10^6 events

(25 800 raw files)

1 raw file = 15 GB (25 000 events)
1 digit file ≈ 870 MB
1 dst file ≈ 2 000 MB

GEN → SIM → ***DSTsim*** → PhA

GEN: particle collisions description received by an event generator

DSTsim: reconstructed data of simulated events

Storage elements

NICA Cluster
ncx[101-106].jinr.ru
(LHEP, b.216)

EOS: 1 PB (*replicated*)
GlusterFS: **300 TB** (*for NICA*)
Sun Grid Engine: 300 cores/user

GRID Tier1&2 Centres
lxui.jinr.ru (CICC)
(MLIT, b.134)

EOS: 1 PB (*replicated*)
SLURM: 0 – 2500 cores
(*for NICA*)

HybriLIT platform (SC «Govorun»)
hydra.jinr.ru
(MLIT, b.134)

ZFS: 200 TB
Lustre (Hot Storage): **300 TB**$_{ssd}$ (*for NICA*)
SLURM: *bmn* – **192** cores

Current metadata

period_number	**-**	**INTEGER**
run_number	**-**	**INTEGER**
run_type	**-**	**SMALLINT**
start_datetime	**-**	**TIMESTAMP**
end_datetime	**-**	**TIMESTAMP**
beam_particle	**-**	**VARCHAR**
target_particle	**-**	**VARCHAR**
Energy	**-**	**FLOAT**
field_voltage	**-**	**FLOAT**
start_event	**-**	**INTEGER**
end_event	**-**	**INTEGER**
event_count	**-**	**INTEGER**
file_size	**-**	**LONG**

How to search for
the files we need…fast?

The types of resources that DIRAC can handle include:

- *Computing* Resources, including Grids, Clouds, HPCs and Batch systems
- *Storage* Resources
- *Catalog* Resources

Many communities use DIRAC, the oldest and most experienced being the LHCb collaboration. Other communities include, but are not limited to, Belle2, ILC, and CTA

File catalog

- File Catalog is a service to keep track of all the physical file replicas in all the SE's
  - Stores also file properties:
    - Size, creation/modification time stamps, ownership, checksums
    - User ACLs
- DIRAC relies on a central File Catalog
  - Defines a single logical name space for all the managed data
  - Organizes files hierarchically like in common file systems
- VO's can ask for dedicated File Catalog services
  - No interference with other users, catalog is chosen based on the user identity
  - Customized behavior
    - Example: Eiscat 3D File Catalog in the EGI DIRAC Service
      - 139M files
      - Custom access policies

DFC: metadata

DFC is Replica and Metadata Catalog

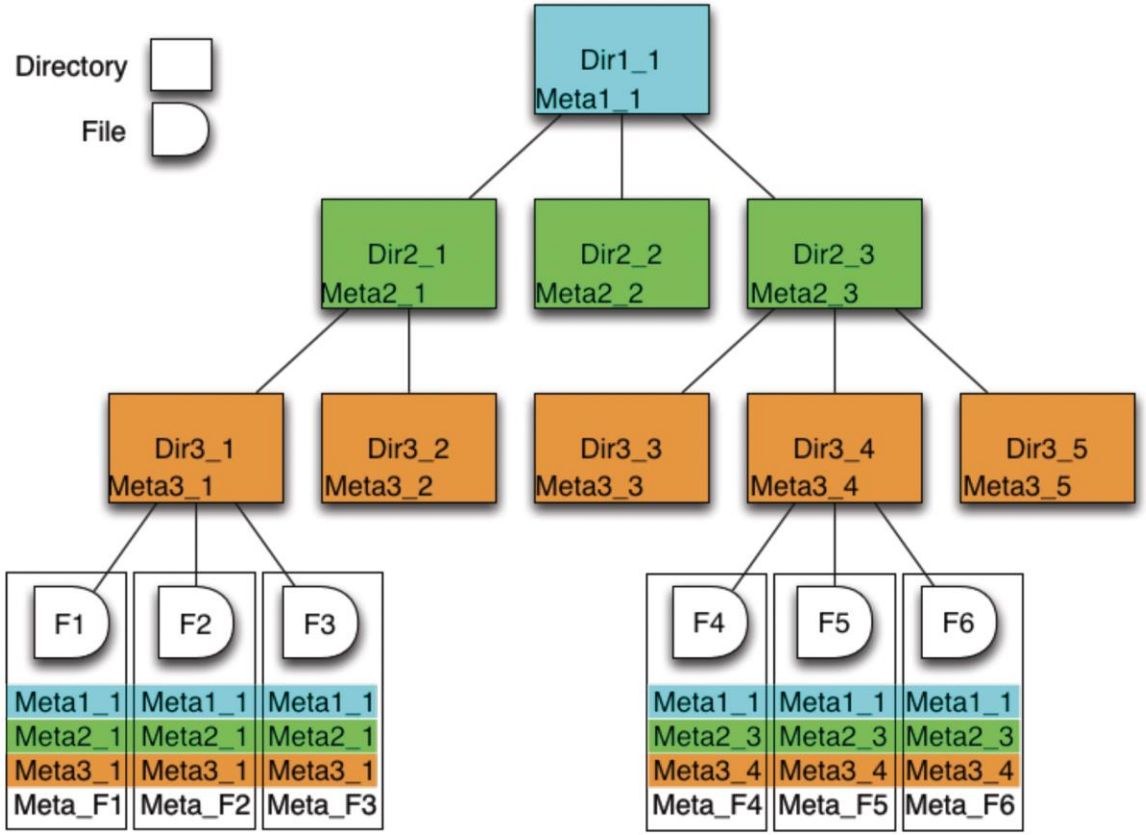
- User defined metadata
- The same hierarchy for metadata as for the logical name space
- Metadata associated with files and directories
- Allow for efficient searches

Efficient Storage Usage reports

- Suitable for user quota management

Stored ancestor/successor file relations

- Simple provenance catalog



DFC through: command line

dirac-dms-add-file

Upload a file to the grid storage and register it in the File Catalog

Usage:

```
dirac-dms-add-file [options] ... LFN Path SE [GUID]
```

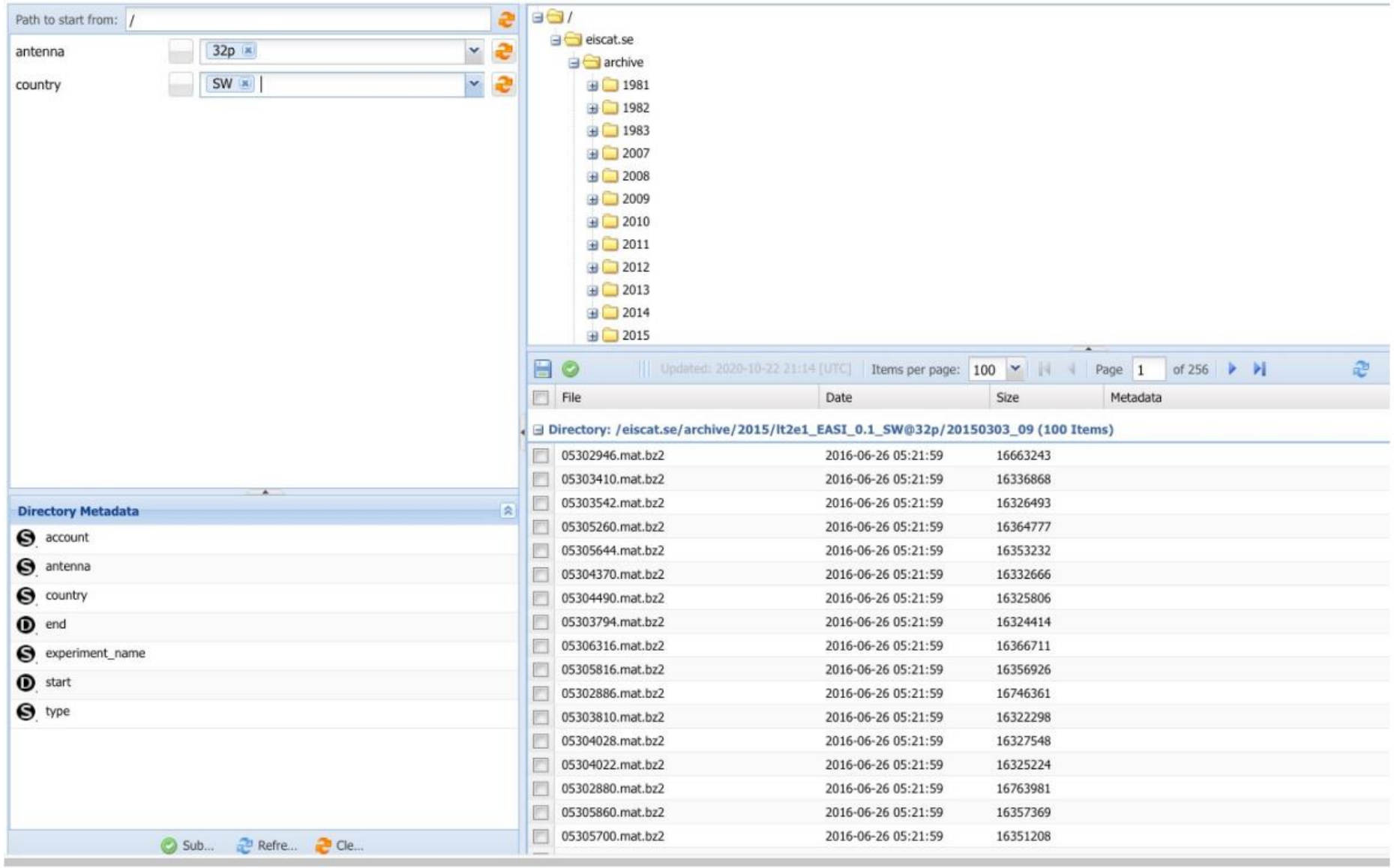
dirac-dms-catalog-metadata

Get metadata for the given file specified by its Logical File Name or for a list of files contained in the specifed file

Usage:

```
dirac-dms-catalog-metadata [options] ... <LocalFile|LFN> Catalog [Catalog]
```

DFC through: web interface



DFC through: python API

putAndRegister(*lfn, fileName, diracSE, guid=None, path=None, checksum=None, overwrite=False*)

Put a local file to a Storage Element and register in the File Catalogues

'lfn' is the file LFN 'file' is the full path to the local file 'diracSE' is the Storage Element to which to put the file 'guid' is the guid with which the file is to be registered (if not provided will be generated) 'path' is the path on the storage where the file will be put (if not provided the LFN will be used) 'overwrite' removes file from the file catalogue and SE before attempting upload

getReplicaMetadata(*lfn, storageElementName*)

get the file metadata for lfns at the supplied StorageElement

Parameters

- **self** – self reference
- **lfn** (*mixed*) – LFN string, list if LFNs or dict with LFNs as keys
- **storageElementName** (*str*) – DIRAC SE name
- **singleFile** (*bool*) – execute for the first LFN only

setMetaQuery(*queryList, metaTypeDict=None*)

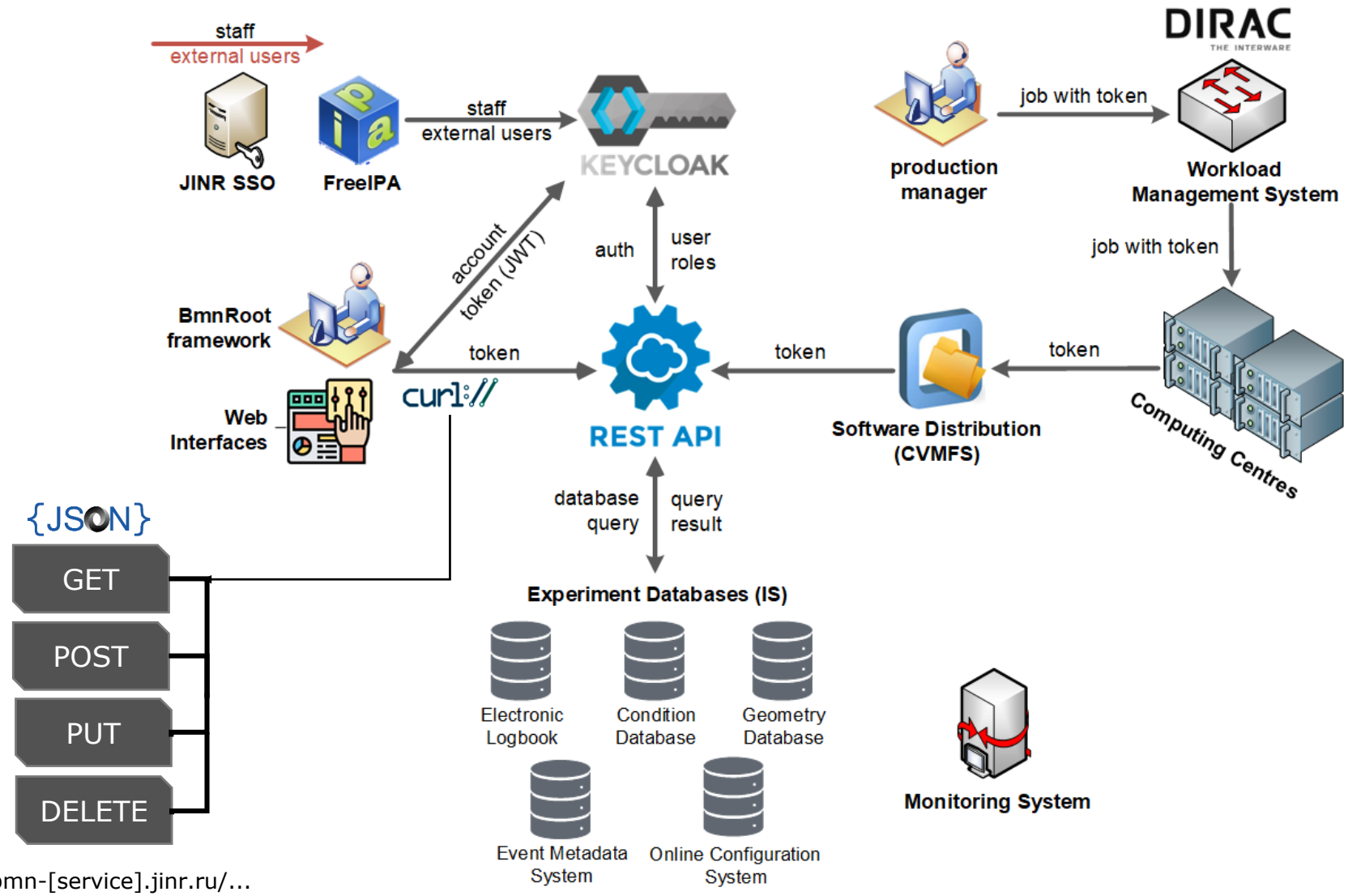
Create the metadata query out of the command line arguments

findFilesByMetadata(*metaDict, path='/', timeout=120*)

Find files given the meta data query and the path

```
def metaGet(meta):
    mq = MetaQuery()
    metaTD = {'period_number': "integer",
              'run_number': "integer",
              'run_type': "integer",
              'start_datetime': "date",
                  'end_datetime': "date",
                  'beam_particle': "string",
                  'target_particle': "string",
                  'energy': "float",
                  'field__voltage': "float",
                  'start_event': "integer",
                  'end_event': "integer",
                  'event_count': "integer",
                  'file_size': "integer" }
    metaD = mq.setMetaQuery(stringToList(meta), metaTD)
    fc = FileCatalogClient()
    files = fc.findFilesByMetadata(metaD['Value'])
    return files['Value']
```

REST API at BM@N



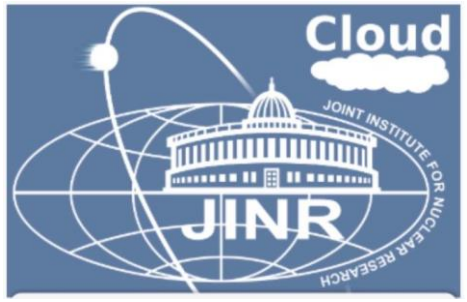
https://bmn-[service].jinr.ru/...

Flask – web framework

Flask decorators

```python
@app.route('/filecatalog_api/v1/<path:filepath>', methods=['PUT'])
def metaPut(filepath):
    if not request.json:
        return dropError('JSON file with metadata and credentials needed')
    filepath = "/" + filepath
    fc = FileCatalogClient()
    res = fc.setMetadata(filepath, request.json)
    if not res['OK']:
        return dropError("eh, can't set metadata for some reason")
    else:
        return "good"
```

```python
@app.route('/filecatalog_api/v1/<path:filepath>', methods=['DELETE'])
def metaDelete(filepath):
    filepath = "/" + filepath # hello
    fc = FileCatalogClient()
    res = fc.getFileUserMetadata(filepath)
    meta = res['Value']
    for field in meta:
        if type(meta[field]) == int or type(meta[field]) == float:
            meta[field] = 0
```



VM at 10.220.16.16

. . .

CURL request examples

GET -- curl
"10.220.16.16:6000/filecatalog_api/v1/event_count=25000|25001&run_number=8426&energy=|3“


```
["/bmn.nica.jinr/vo/raw/run8/mpd_run_Top_8426_ev1_p2.data","/bmn.nica.jinr/vo/raw/run8/mpd_run_Top_84
26_ev1_p3.data","/bmn.nica.jinr/vo/raw/run8/mpd_run_Top_8426_ev0_p0.data","/bmn.nica.jinr/vo/raw/run8
/mpd_run_Top_8426_ev0_p2.data","/bmn.nica.jinr/vo/raw/run8/mpd_run_Top_8426_ev1_p0.data","/bmn.nica.j
inr/vo/raw/run8/mpd_run_Top_8426_ev0_p1.data","/bmn.nica.jinr/vo/raw/run8/mpd_run_Top_8426_ev1_p1.dat
a"]
```


PUT -- curl -X PUT -H "Content-Type: application/json" -d '{"run_number“:333, "energy“:17}'
10.220.16.16:6000/filecatalog_api/v1/bmn.nica.jinr/test/metadataTest/mpd_run_Top_8427_ev1_p10.data

DELETE -- curl -X DELETE
10.220.16.16:6000/filecatalog_api/v1/bmn.nica.jinr/test/metadataTest/mpd_run_Top_8427_ev1_p10.data

Allowed for everyone, right?

Keycloak

Open Source Identity and Access Management

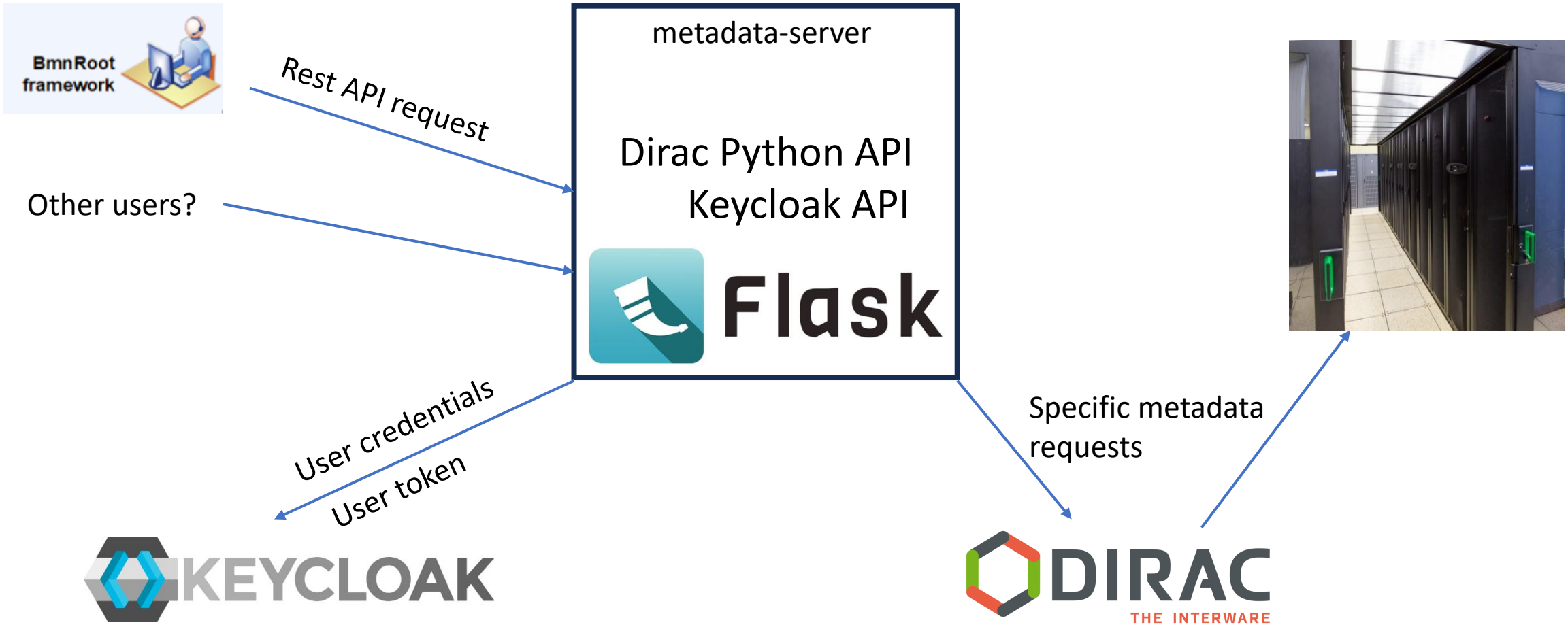
```
keycloak_openid = KeycloakOpenID(server_url="http://localhost:8080/auth",
                                 client_id="jinr-metadata-server",
                                 realm_name="myrealm",
                                 client_secret_key="------------------------")
```

```
def authByJson(jsn):
    if 'username' in jsn and 'password' in jsn:
        try:
            token = keycloak_openid.token(request.json["username"], request.json["password"], grant_type="password")
        except:
            return "user or password is incorrect"
        userinfo = keycloak_openid.userinfo(token['access_token'])
        return userinfo['preferred_username']
    else:
        return "no credentials finded"
```

Request with credentials will be like:

curl -X GET -H "Content-Type: application/json" -d '{"username":"reader", "password":"diracreader"}' "10.220.16.16:6000/filecatalog_api/v1/target_particle=csi*"

Everything together



What’s next?

- Run 8 files already in the file catalogue and with proper metadata.
- We got a service to search (GET) for the files with specific metadata
  UPDATE and DELETE if we need to

Todo:

- Add metadata simultaneously with file upload to file catalogue
- Developing a REST API for root software in order to work with this service.

Thank you!

Igor Zhironkin
FLNP, JINR
jironkin@jinr.ru