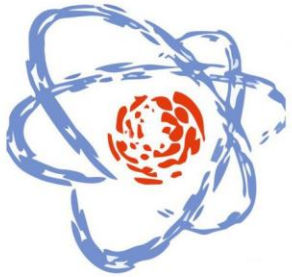


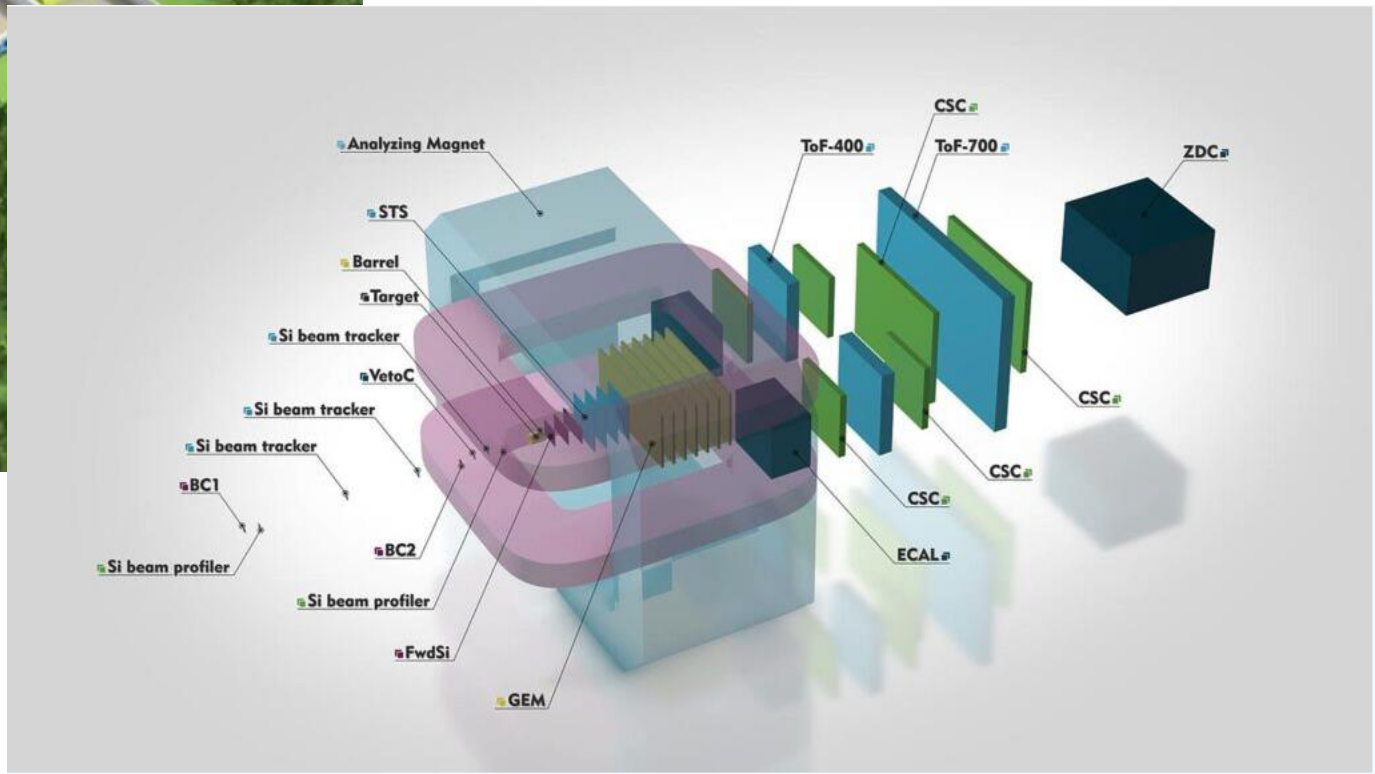
29th International Scientific Conference of Young Scientists and Specialists

Development Methodology of the Data Management System for the BM@N experiment

Igor Zhironkin,
Igor Pelevanyuk,
Konstantin Gertsenberger
JINR

BM@N


SPD
(Detector)
E-cooling
MPD
(Detector)
BM@N (Detector)
Extracted beam
Collider
Injection Complex
Nuclotron
Booster


Analyzing Magnet
STS
Barrel
Target
Si beam tracker
VetoC
Si beam tracker
Si beam tracker
BC1
Si beam profiler
BC2
Si beam profiler
FwdSi
GEM
ToF-400
CSC
ToF-700
ZDC
CSC
CSC
CSC
ECAL

DMS Tasks

- Provide dataset oriented, secure access to data
- Managing data:
  - Transfer data to/from/between sites
  - Delete data from sites
- Ensure data consistency at sites
- Workflow integration

Main points

- 1. Perform a subject area analysis and formulate requirements for the data management system being developed
- 2. Deploy the data storage infrastructure
- 3. Define a list of metadata and dataset selection scenarios based on the data selection criteria in the experiment
- 4. Select a management system platform based on the expected data volumes and the need for integration with computing
- 5. Implement the data management system in accordance with the specifics of the data flow of a specific experiment

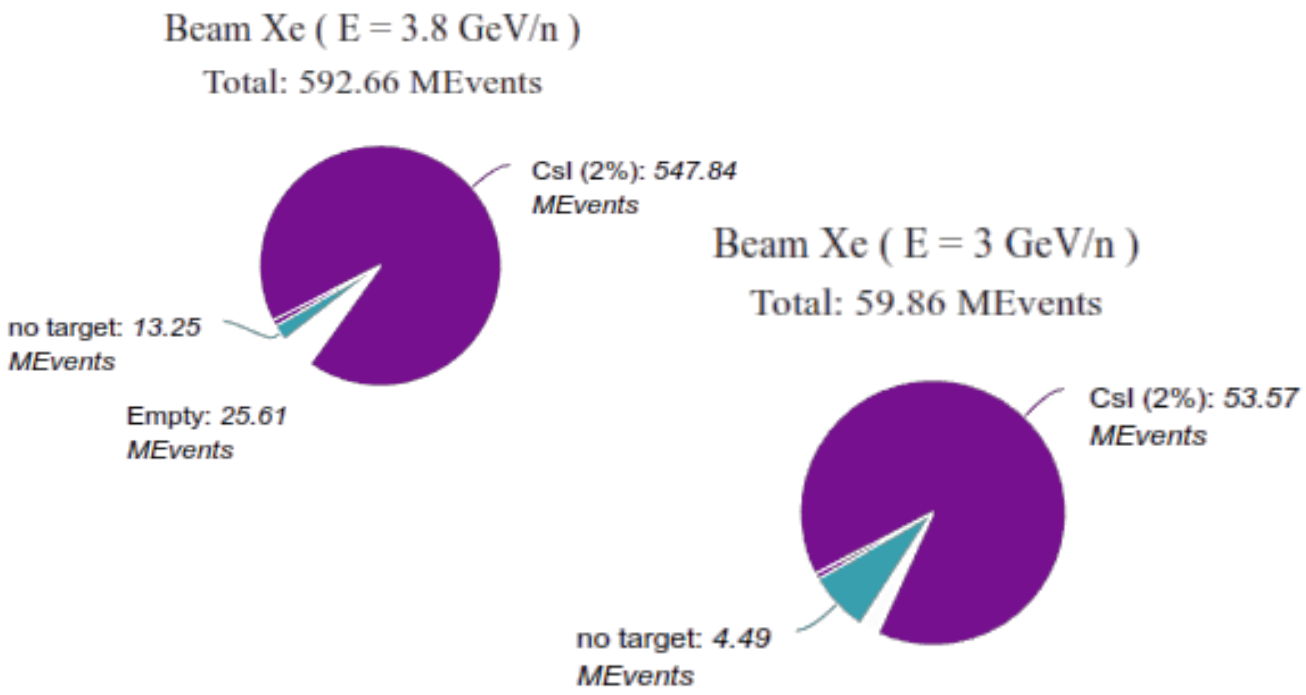
1.Perform a subject area analysis and formulate requirements for the DMS

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

2.Deploy the data storage infrastructure

2.1 Create a virtual organization on the authentication and authorization service

2.2 Register users of the virtual organization

2.3 Ensure separation of user rights

In grid computing, a **virtual organization** (VO) refers to a dynamic set of individuals or institutions defined around a set of resource-sharing rules and conditions.

VO – bmn.nica.jinr
user – Igor Zhironkin
role – pilot

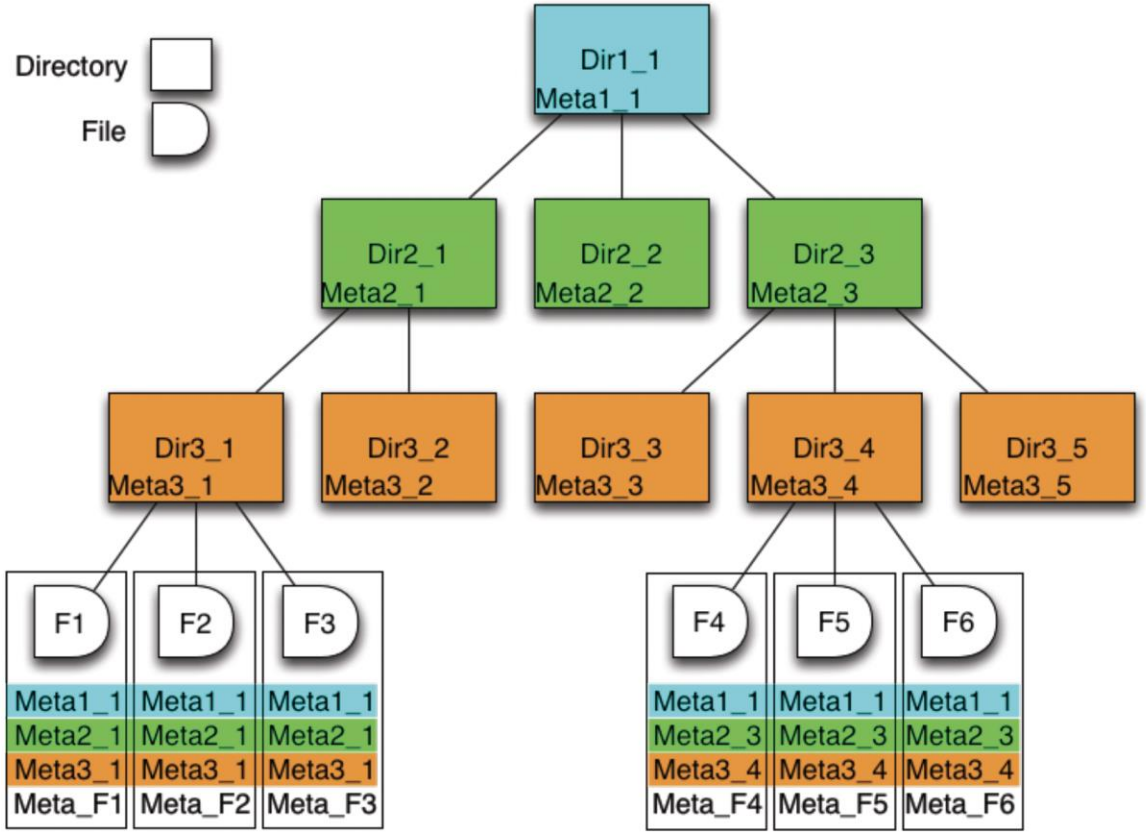
3. Define a list of metadata and dataset selection scenarios

3.1. Select metadata and determine their types based on the specifics of the experiment

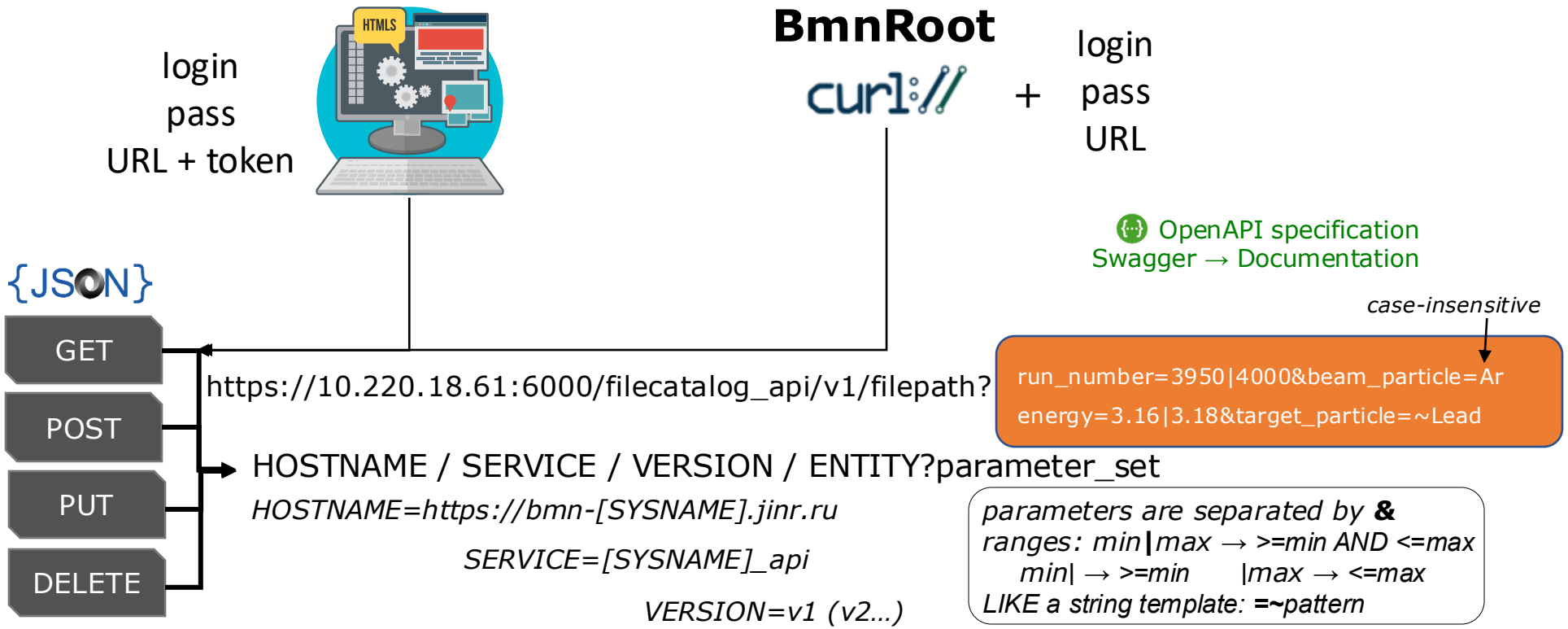
period_number	**-**	**INTEGER**
run_number	**-**	**INTEGER**
run_type	**-**	**SMALLINT**
start_datetime	**-**	**TIMESTAMP**
end_datetime	**-**	**TIMESTAMP**
beam_particle	**-**	**VARCHAR**
target_particle	**-**	**VARCHAR**
energy	**-**	**FLOAT**
field_voltage	**-**	**FLOAT**
start_event	**-**	**INTEGER**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

3.2. Determine the metadata storage structure

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



3.3. Determine the metadata queries and representation



BM@N File Catalog, SYSNAME = **filecatalog** (prototype)

4. Select a data management system platform

Rucio was originally developed to meet the requirements of the high-energy physics experiment ATLAS

Rucio now is continuously extended to support the LHC experiments and other diverse scientific communities.

- Highly scalable
- Policy driven
- Good for big amount of data
- Automated Data Rebalancing

An open source middleware for distributed computing

- Started as an LHCb project.
- Experiment-agnostic since 2009.
- Developed by communities, for communities.
- Workload management system integrated
- Publicly documented, active assistance forum,

yearly users workshops, open developers meetings and hackathons, already in JINR

5. Implement the data management system

5.1. Add logical data storage elements (SE) and configure interaction with the File Catalog (FC) system

- Via CLI or cfg file.
- Host, port, available plugins, etc. for SE
- Database install, security management, globalReadAccess, LFNPFNConvention, uniqueGUID etc. for FC

5.2. Add the necessary metadata fields to the File Catalog service In accordance with previously mentioned metadata list

5.3. Implement integrity and consistency checks

Consistency check -- basically goes through storages and compare existing files with registered ones in the File Catalog.

The output is two counters and two vectors of paths to files that doesn’t match

Missing from EOS 991567:
/bmn.nica.jinr/exp/dst/run8/24.12.0/mpd_run_Top_7327_ev1_p7.root
/bmn.nica.jinr/exp/dst/run8/24.12.0/mpd_run_Top_7327_ev1_p8.root
/bmn.nica.jinr/exp/dst/run8/24.12.0/mpd_run_Top_7327_ev1_p9.root
/bmn.nica.jinr/exp/dst/run8/24.12.0/mpd_run_Top_7328_ev0_p0.root
. . .
Missing from FC 4:
/eos/nica/bmn/exp/digi/run8/25.04.0/mpd_run_Top_7797_ev1_p2.root
/eos/nica/bmn/exp/digi/run8/25.04.0/mpd_run_Top_7797_ev0_p5.root
/eos/nica/bmn/exp/digi/run8/25.04.0/mpd_run_Top_8106_ev0_p70.root
/eos/nica/bmn/exp/digi/run8/25.04.0/mpd_run_Top_7444_ev1_p14.root

By integrity checking we mean checking the hash sums of files, making sure that they contain what we expect to see.

5.4. Implement application programming interfaces

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

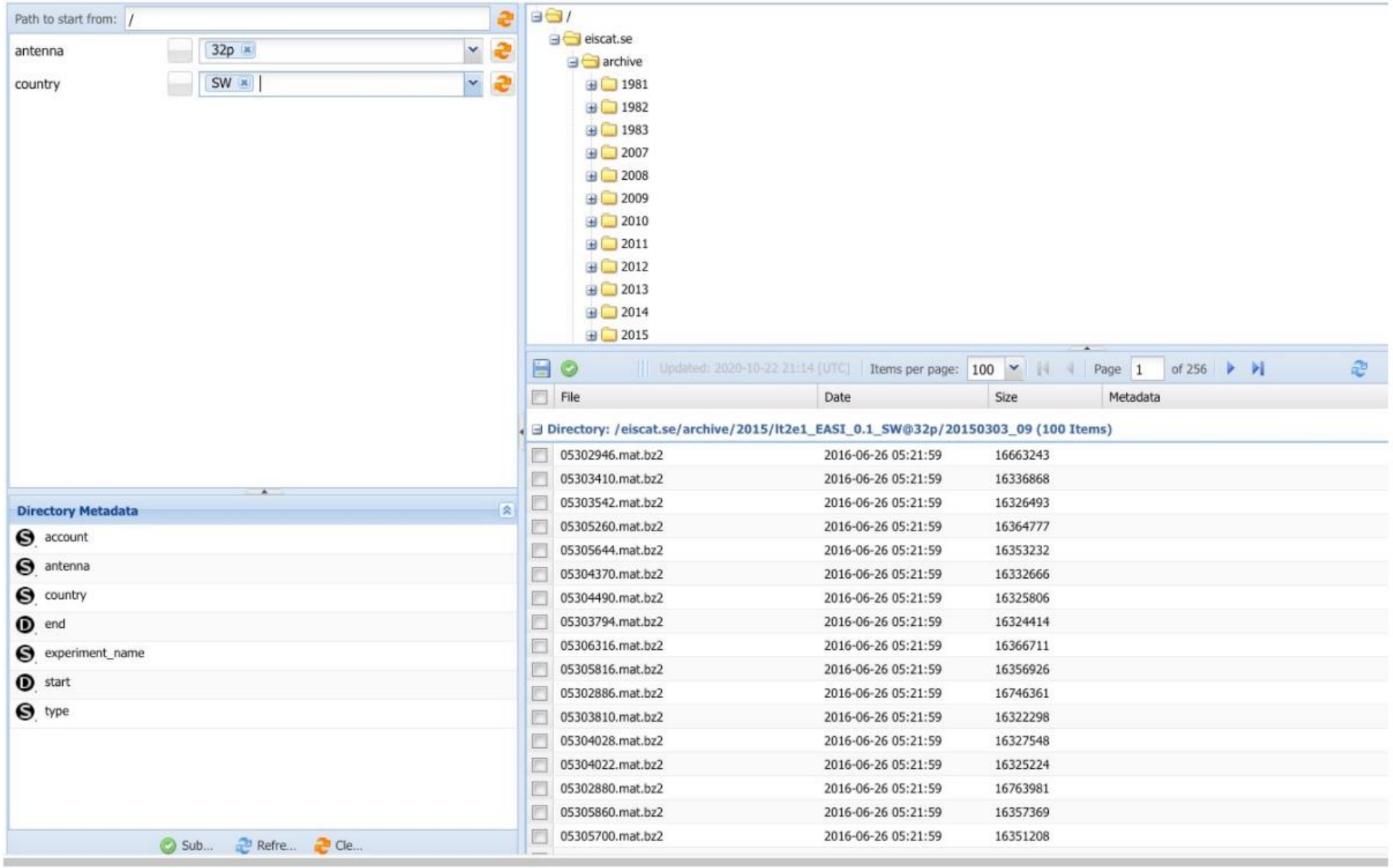
Get metadata for the given file specified by its Logical File Name or for a list of files contained in the specifed file

Usage:

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

5.4. Implement application programming interfaces

DFC through: web interface



5.4. Implement application programming interfaces

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

```python
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']
```

5.4. Implement application programming interfaces

REST API Request types

POST - add new metadata field
- run consistency check. Return check ID, status can be verified through get
- upload and register file from NCX to CICC

GET - get file list matching specific metadata
- get specific file metadata
- get all metadata fields
- is file exist
- get file catalog stats. (Number of files/directories/replicas etc.)
- get status of the last consistency check
- get result of the last consistency check

PUT - update file metadata

DELETE - delete specific file metadata
- remove specific metadata field
- remove file from CICC and FC

5.4. Implement application programming interfaces

C++ API for BmnRoot

- Choose one of the **FileCatalog** class functions that best suits your needs. Like: *…GetFileList(…), GetFileInfo(…), UpdateFileInfo(…), DeleteMetadataField(…)*

Depending on use case:

- Prepare **FileInfo** object containing metadata info (*int RunNumber, string BeamParticle* etc.), or declare one if you need it as a result
- Define **FileCondition** (*less, greater, null, greaterOrEqual* etc.) to specify a metadata range for the selection request

5.5. Implement statistics and monitoring services

Monitoring

Continuously observing system metrics to identify real-time issues and anomalies, triggering alerts

- CPU and memory usage
- Cumulative transfer size
- File transfer speed
- Success/failure transfer amount
- Requests duration

Telegraf/InfluxDB/Grafana stack as a solution

5.5. Implement statistics and monitoring services

Logging

Logging involves recording detailed, timestamped events (like errors or user actions), primarily for historical analysis and troubleshooting

- Authentication failures, file transfer errors, connection errors, file not found or already exists, consistency check results, etc.

OpenTelemetry is one of the option to choose. OpenTelemetry is a collection of APIs, SDKs, and tools to instrument, generate, collect, and export telemetry data. Open source, as well as vendor- and tool-agnostic, meaning that it can be used with a broad variety of observability backends

5.6. Implement a graphical interface

GUI via Web portal. Basically, for a visual representation of the results of the work of monitoring and logging systems. Graphs, tables, anything visually pleasing. As well as another convenient interface for access to the DMS.

5.7. Integrate the File Transfer Service

The File Transfer Service (FTS) is a bulk data mover responsible for **queuing**, **scheduling**, **dispatching** and **retrying** file transfer requests.

It will receive the files to transfer, as well as the list of destinations. The files will then be grouped together and submitted as jobs to the fts servers. These jobs will be monitored, retried if needed, the new replicas will be registered, and the status of the files will be reported

Conclusion

1. Explore data flows, data volumes, data request scenarios of your experiment
2. Choose DMS basis (RUCIO, DIRAC, etc.)
3. File catalog and metadata catalog is the main feature, configure them!
4. Implement integrity and consistency checks
5. Implement API’s, authorization/logging/monitoring/FTS services if needed

Thank you!