DEEP UNDERGROUND NEUTRINO EXPERIMENT

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

Intro to Rucio concepts and operations

Doug Benjamin

12/14/21

@BrookhavenLab

Rucio – Scientific Data Management

https://rucio.cern.ch/

“Rucio is an open-source software framework that provides scientific collaborations with the functionality to organize, manage, and access their data at scale. The data can be distributed across heterogeneous data centers at widely distributed locations.”

Opening sentence from abstract on – published article on Rucio - https://link.springer.com/article/10.1007/s41781-019-0026-3 - citeas

“Rucio: Scientific Data management” - https://doi.org/10.1007/s41781-019-0026-3

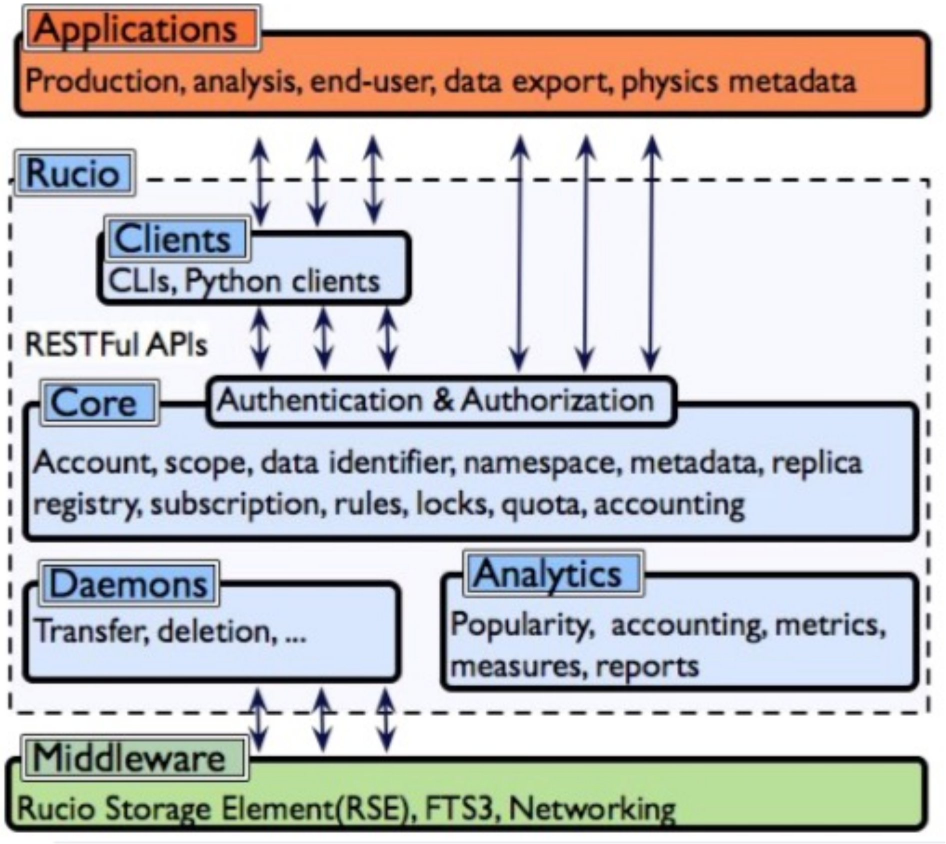
Rucio Components

https://rucio.cern.ch/documentation/Main-components-of-rucio

Clients – command line clients, Python clients, Javascript-based web UI

Server – authentication, provides a common python API

Note: Rucio relies on middleware for some activities (ie FTS3 for file transfer)


Applications
Production, analysis, end-user, data export, physics metadata
Rucio
Clients
CLIs, Python clients
RESTFul APIs
Core
Authentication & Authorization
Account, scope, data identifier, namespace, metadata, replica registry, subscription, rules, locks, quota, accounting
Daemons
Transfer, deletion, ...
Analytics
Popularity, accounting, metrics, measures, reports
Middleware
Rucio Storage Element(RSE), FTS3, Networking

Rucio Namespace – data identifier (DID)

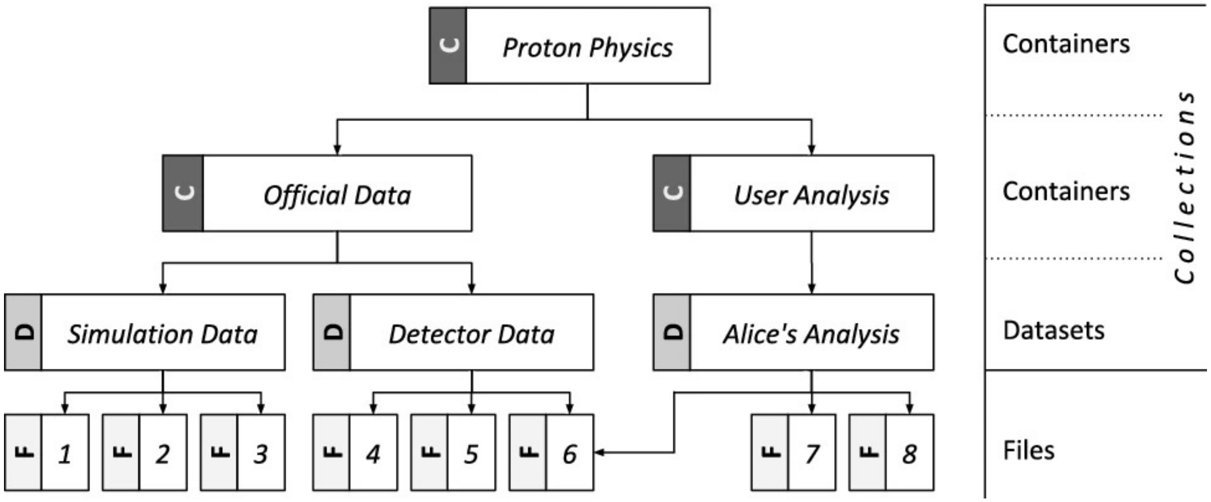


Figure #1 – Rucio paper

Files are the smallest operational unit of data.

Files can be grouped into datasets and datasets into containers and containers into other containers.

Data identifiers and scope:

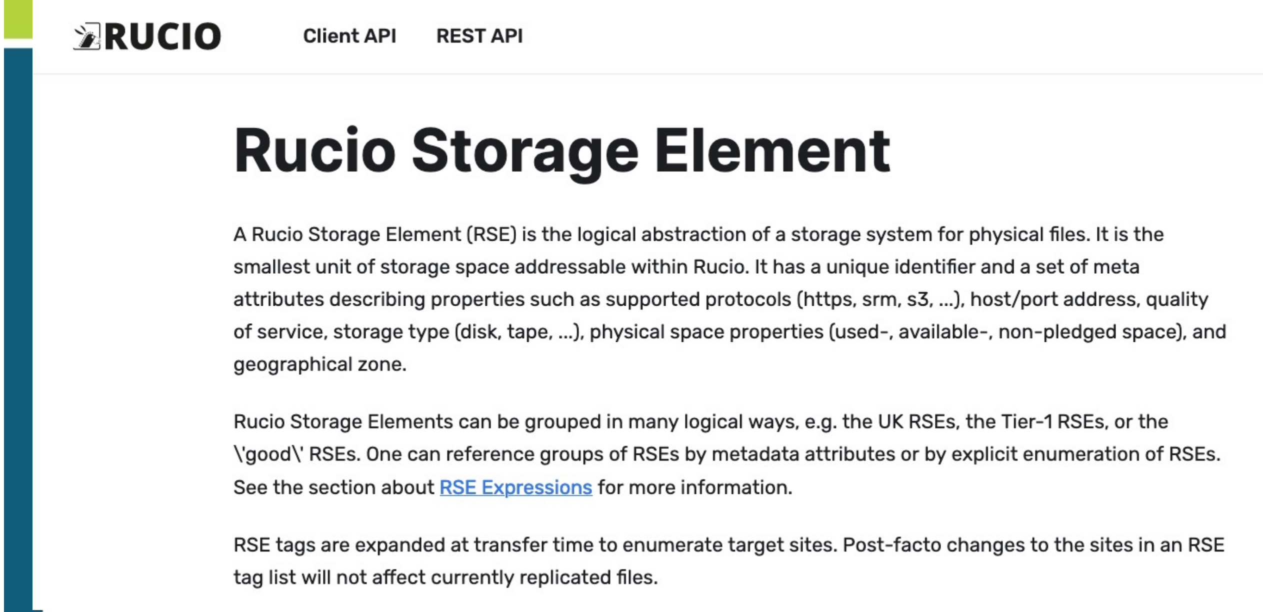
Naming scheme – two strings – scope and name – DID (scope:name)

Files, datasets, containers uniquely identified for all time - no DID reuse.

https://rucio.cern.ch/documentation/File_dataset_container

Rucio Storage Element

https://rucio.cern.ch/documentation/Rucio_storage_element



Rucio Storage Element

A Rucio Storage Element (RSE) is the logical abstraction of a storage system for physical files. It is the smallest unit of storage space addressable within Rucio. It has a unique identifier and a set of meta attributes describing properties such as supported protocols (https, srm, s3, ...), host/port address, quality of service, storage type (disk, tape, ...), physical space properties (used-, available-, non-pledged space), and geographical zone.

Rucio Storage Elements can be grouped in many logical ways, e.g. the UK RSEs, the Tier-1 RSEs, or the \'good\' RSEs. One can reference groups of RSEs by metadata attributes or by explicit enumeration of RSEs. See the section about RSE Expressions for more information.

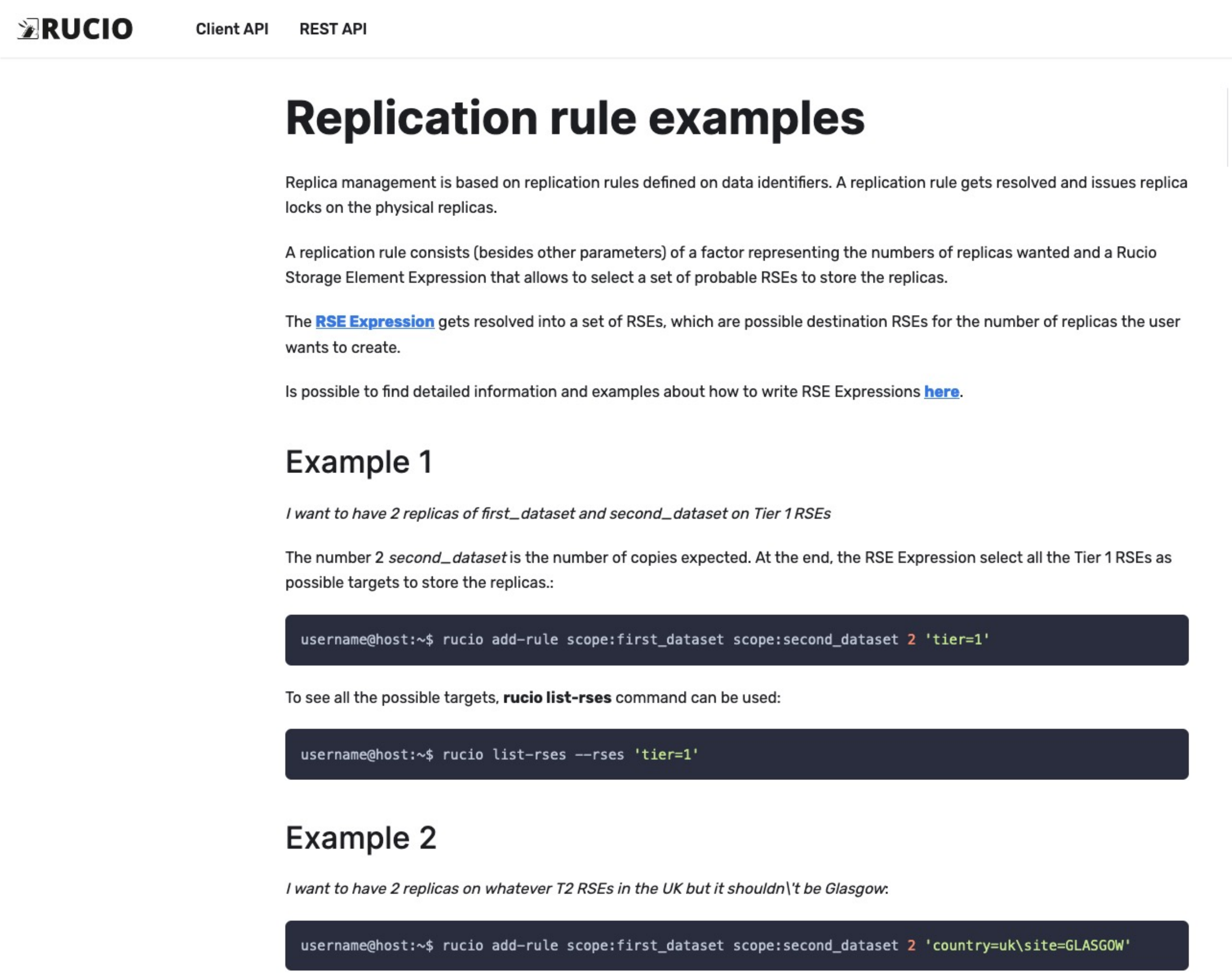
RSE tags are expanded at transfer time to enumerate target sites. Post-facto changes to the sites in an RSE tag list will not affect currently replicated files.

DUNE has several RSE's

```
$ rucio list-rses

CERN_PDUNE_CASTOR
CERN_PDUNE_EOS
DUNE_FR_CCIN2P3
DUNE_FR_CCIN2P3_DISK
DUNE_FR_CCIN2P3_XROOTD
DUNE_US_BNL_SDCC
EDINBURGH
FNAL_DCACHE
FNAL_DCACHE_PERSISTENT
FNAL_DCACHE_STAGING
FNAL_DCACHE_TEST IMPERIAL LANCASTER
LIVERPOOL
MANCHESTER
NIKHEF
PRAGUE
QMUL
RAL_ECHO
RAL-PP
SCRATCH_DCACHE
T3_US_NERSC
```

Replication Rules used to move files between RSE's



Replication rule examples

Replica management is based on replication rules defined on data identifiers. A replication rule gets resolved and issues replica locks on the physical replicas.

A replication rule consists (besides other parameters) of a factor representing the numbers of replicas wanted and a Rucio Storage Element Expression that allows to select a set of probable RSEs to store the replicas.

The **RSE Expression** gets resolved into a set of RSEs, which are possible destination RSEs for the number of replicas the user wants to create.

Is possible to find detailed information and examples about how to write RSE Expressions **here**.

Example 1

I want to have 2 replicas of first_dataset and second_dataset on Tier 1 RSEs

The number 2 *second_dataset* is the number of copies expected. At the end, the RSE Expression select all the Tier 1 RSEs as possible targets to store the replicas.:

```
username@host:~$ rucio add-rule scope:first_dataset scope:second_dataset 2 'tier=1'
```

To see all the possible targets, **rucio list-rses** command can be used:

```
username@host:~$ rucio list-rses --rses 'tier=1'
```

Example 2

I want to have 2 replicas on whatever T2 RSEs in the UK but it shouldn\'t be Glasgow:

```
username@host:~$ rucio add-rule scope:first_dataset scope:second_dataset 2 'country=uk\site=GLASGOW'
```

https://rucio.cern.ch/documentation/replication_rules_examples

All Rucio Transfers managed by FTS3 (or soon also Globus)

Random thoughts:

Rucio at DUNE under active development –

- It remains to be seen what parts of Rucio will be used by DUNE (it will be different from LHC experiments – and that’s OK)

Will users need to access rucio directly?

- Not likely, end users will use DUNE standard interface
- Developers should develop to the evolving DUNE standard interface

Developers (ie you all) might use the Rucio python API in your work.

Useful Rucio links

Main website – http://rucio.cern.ch/

Rucio Documentation – https://rucio.cern.ch/documentation/

- Client API documentation - https://rucio.cern.ch/documentation/rucio_client_api
- REST API documentation - https://rucio.cern.ch/documentation/rucio_rest_api

Rucio Github Repo - https://github.com/rucio/rucio

CERN File Transfer Service (v3) FTS3 –

https://fts3-docs.web.cern.ch/fts3-docs/index.html

Discussion Time