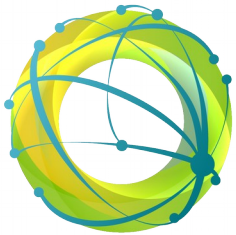

CERN



FTS

FTS & Community-managed QoS

Oliver Keeble on behalf of the FTS team

Exploiting QoS

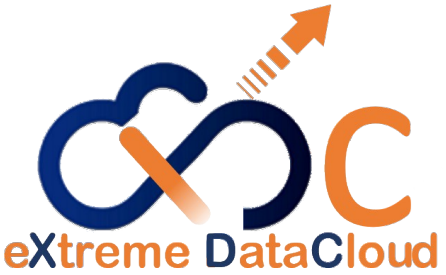
- Legitimise a greater variety of QoS on the infrastructure
  - Volatile, fast, durable, whatever
- Enable the storage systems to offer multiple QoS
  - “Software Defined Storage”
- Allow the user (community) to manage QoS transitions
  - Community-defined data lifecycle
  - The topic of this talk

- FTS has long supported our most common QoS transition
  - Bringonline
- To do so, it already has useful machinery
  - REST interface for submitting such requests in bulk
  - Scheduler with submission limiting
  - Queue management and state machine inc retries
  - Expectation of long running jobs (days)
  - Monitoring
  - Integration with many existing systems (inc Rucio)
- This made it an excellent candidate as a QoS manager

Data lifecycle

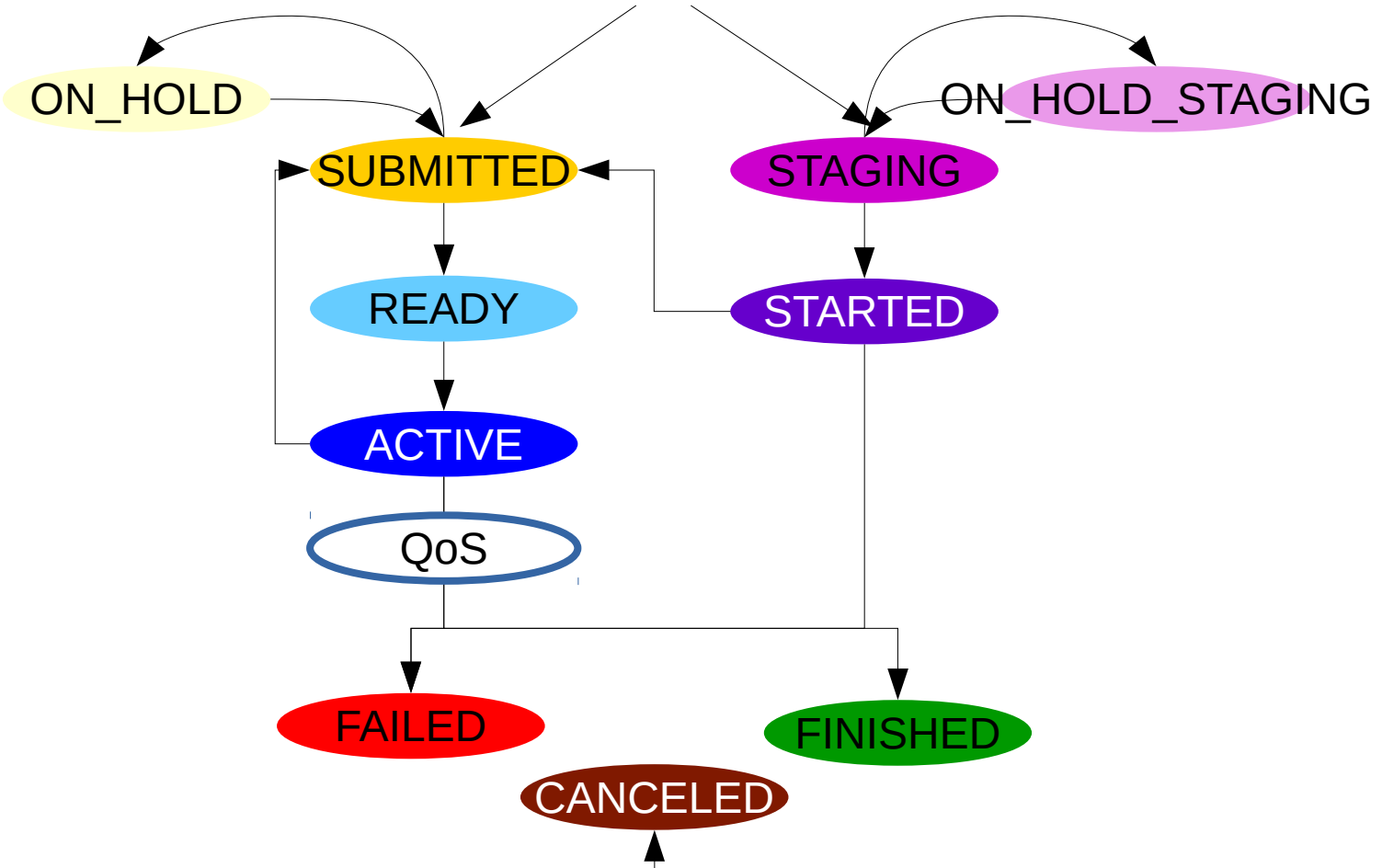
- Where's the value in enabling this?
- Data has a lifecycle
  - Different QoS are relevant at different times
  - Hot → Cold
  - More room for hot data if the less-used stuff can be stored more efficiently

FTS QoS workflow



- Receive transfer with QoS metadata
  - “target QoS”
- If the destination directory does not exist, create it with the appropriate QoS
- If destination dir exists, check if dir has requested QoS
  - If not, check required transition is permitted for a file in that dir
  - If not, fail
- Put the file
- Change QoS if necessary

New FTS state



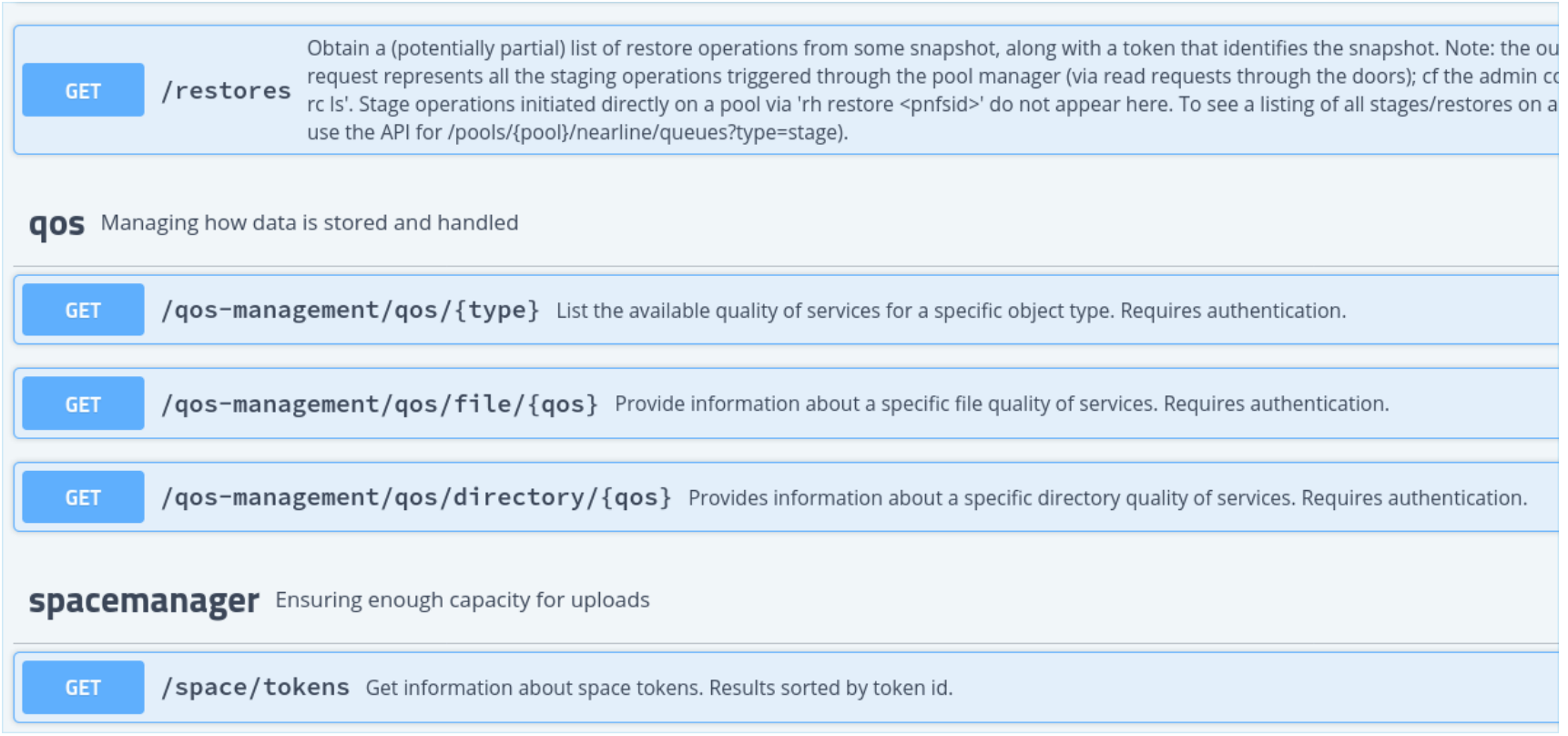
FTS Interface

- FTS submission interface and state machine have been updated
- CLI
  - fts-rest-transfer-submit –target-qos
- Job params in json submission
  - "params": { "target_qos": “3replicas” }

Storage interface

- Many storage systems already support multiple QoS
  - EOS : eos file convert <path> replica:2
  - dCache has incorporated QoS into its REST interface
- Administration is generally done by admins and not actively managed
  - EOS @ CERN typically has dual-replica
  - How should it be exposed to frameworks?

dCache REST



This interface has been (mostly) replicated in EOS as part of the XDC project

A common interface for frameworks?

- CDMI is a RESTful, open data management interface, standardised by SNIA.
  - https://www.snia.org/cdmi
- The Indigo Datacloud project added QoS elements as an extension to this interface
  - https://www.rd-alliance.org/groups/storage-service-definitions-wg
  - The XDC project inherited this work
  - This is the interface that FTS uses
    - It does not use the rest of CDMI!
- Indigo Datacloud provided an implementation with plugins for different backends (dCache in particular)
  - The XDC project adapted the dCache plugin for use with EOS
  - Result : a CDMI interface exists for EOS

CDMI : BringOnline

REQUEST (get current state)

```
GET
/user123/container1/myDataobject?
capabilitiesURI;metadata
HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.1
```

RESPONSE

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.1
Content-Type: application/cdmi-object

{ "capabilitiesURI" :
"/cdmi_capabilities/dataobject/profile2",
"metadata" : {
"cdmi_size" : "37",
"cdmi_data_redundancy_provided": "3",
"cdmi_geographic_placement_provided":
["DE", "FR"],
"cdmi_latency_provided": "3600000",
} }
```

CDMI : BringOnline

REQUEST (current capabilities)

```
GET
/cdmi_capabilities/container/
profile2 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-
capability
X-CDMI-Specification-
Version: 1.1
```

RESPONSE

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.1
Content-Type: application/cdmi-capability

{ "objectType": "application/cdmi-capability",
 ...
"capabilities":
{
"cdmi_capabilities_templates": "true",
 ...
"cdmi_capabilities_allowed": "/cdmi_capabilities/container/profile1"
},
"metadata":
{
"cdmi_data_redundancy": "3",
"cdmi_geographic_placement": ["DE", "FR"],
"cdmi_latency": "3600000" }}
```

CDMI : BringOnline

REQUEST (other capabilities)

```
GET /cdmi_capabilities/container/profile1 HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-capability
X-CDMI-Specification-Version: 1.1
```

RESPONSE

```
HTTP/1.1 200 OK
X-CDMI-Specification-Version: 1.1
Content-Type: application/cdmi-capability
{ "objectType": "application/cdmi-capability",
 ...
"capabilities":
{
"cdmi_capabilities_templates": "true",
 ...
"cdmi_capabilities_allowed":
"/cdmi_capabilities/container/profile2"
},
"metadata":
{
"cdmi_data_redundancy": "4",
"cdmi_geographic_placement": ["DE", "FR"],
"cdmi_latency": "100" }}
```

CDMI : BringOnline

REQUEST (trigger transition)

```
PUT /user123/container1/myDataobject
HTTP/1.1
Host: cloud.example.com
Content-Type: application/cdmi-object
X-CDMI-Specification-Version: 1.1
{
"capabilitiesURI":
"/cdmi_capabilities/dataobject/profile1"
}
```

REQUEST (poll)

```
GET
/user123/container1/myDataobject?
capabilitiesURI;metadata
HTTP/1.1
Host: cloud.example.com
Accept: application/cdmi-object
X-CDMI-Specification-Version: 1.1
```

gfal2

- FTS accesses storage through the gfal2 library.
- CDMI support has therefore been added to gfal2
- The following CDMI QoS operations have been implemented by the HTTP plugin
  - file : current QoS (*gfal2_check_file_qos()*) ; requested QoS ; available transitions ; request transition
  - dir : default QoS ; available transitions
- Exposed by the C++ and Python APIs

Archive monitoring

- Related development
- FTS will be able to fail a job if the destination file does not end up on tape within a certain time
- New ARCHIVING state similar to the QoS one described above
- User sets “archive_timeout” during job submission
- Will work with xrootd and SRM
  - On CTA, will check the “m-bit”
- https://its.cern.ch/jira/browse/FTS-1423

Status of FTS and gfal2 developments

- XDC QoS stuff
  - QoS interface, management, CDMI support
  - Developments currently on “XDC branches”
    - REST interface stable
    - gfal2 stable
    - Implementation of “QoS daemon” not yet complete
  - Available on the fts3-xdc.cern.ch endpoint
  - All developments will ultimately be merged into mainline releases
- Archive monitoring
  - Required by CMS
  - Development priority for this year

Conclusions

- FTS's staging machinery has been adapted for arbitrary QoS transitions
- Uses an updated gfal2 library to interact with storage via CDMI
  - Supported by dCache & EOS
- This allows large managed transitions initiated by the experiment
  - → Data Lifecycle
- Work funded by the XDC project