FTS v3.10
Development Report

Mihai Patrascoiu
on behalf of the FTS team
Open Source software for reliable and large-scale data transfers within WLCG

Developed at CERN

Carles Garcia Cabot
Python Developer

Edward Karavakis
Project Leader

Mihai Patrascoiu
C++ Developer
- Latest Numbers

8 WLCG instances
BNL, CERN (4), FNAL, RAL, MIT

16 non-WLCG instances
CERN (DAQ, Public), RAL, KEK(2), Imperial (also used by CMS), PIC, MWT2, CESNET (WebFTS + RCAuth), JINR, CNAF, SARA, SLAC, IHEP, Fermilab (containers), FENIX Research Infrastructure (Human Brain Project)

~37 Virtual Organizations
ATLAS, CMS, LHCb, AMS, NA62, Compass, ILC, Magic, Belle, Mice, Xenon, Snoplus, Gridpp, Dune, LZ, Solidexperiment.org, SKA, Ligo, Icecube, Elixir, NP02(part of Dune), CAST, ESCAPE, Eiscat.se, Virgo, Pierre Auger Observatory, BES III, JUNO, CEPC, FENIX-RI, CTA, T2K, Project8, ICARUS, FASER, Folding@Home

Transferred in total in 2018: 848 PBs and 986M files
Transferred in total in 2019: 1.27 EBs and 1.08B files
Transferred in total in 2020: 897 PBs and 806M files

~65% by CERN FTS instances (so far)
- **Core features**

**Intuitive**
- Simple interaction to submitting transfers via REST API
- UI-based WebFTS portal

**Robust**
- Checksums and retries are provided per transfer
- Deploy and run “zero config”

**Flexible**
- Multiprotocol support (HTTP, GridFTP, XRoot, SRM, S3, ...)
- Real-time monitoring and runtime config (via Web Admin)

**Adaptive**
- Runtime optimisation (maximise throughput without burning the storages)
- Priorities / Activities support for transfer classification
FTS ecosystem
Updated Components

- **FTS Rest Flask**
  - New deployment of FTS Rest on Python Flask
  - Move the code base from from Pylons to Flask framework
  - Code migrated to Python3
  - Backwards compatible API

- **Data Transfer Orchestrator (DTO)**
  - Former CDR (Central Data Repository): software used by NA62 for DAQ to move data to CASTOR
  - Generalised the functionality to support any VO for Data Acquisition
  - Code parallelisation, writing to CTA

*Attributions to Marco Borreto and Cristina Voineag*
New Features

- OIDC Token support
  - FTS transfer workflow w/o x509 certificates
  - Multiple IAM support (XDC, WLCG)

- QoS daemon
  - Generalize the mechanism for transitions and polling
  - Supports CDMI QoS transitions

- Archive Monitoring
  - Follow a transfer until successful arrival (or not) on tape storage
OIDC Token Support

- A new authentication mode supported: OpenID Connect
- Token validation done offline or via introspection (based on PyOIC)
- Allows for Audience and Scope claims validation
- Refresh token + access token stored in database (to be used during transfer)
- Enabled by sending `<access_token>` to FTS-REST server

```bash
fts-rest-transfer-submit --access-token=<access_token>
-s https://fts3-devel.cern.ch:8446/ <src> <dst>
```
OIDC Token Support

1. Submit job with token
2. Exchange token
3. Send new access token and refresh token
4. Submit third-party transfer with
5. Data transfer with

IAM

FTS

SE1

SE2
QoS daemon

- Triggers and monitors *long-standing* storage transitions
  - Tape → Disk, Disk → Tape, QoS transition
- Replaces the FTS-bringonline daemon
- Workload is split via database partitioning
- Interaction with storage endpoints done via Gfal2
- QoS transitions are implemented according to CDMI-QoS specification
  - Relies on Gfal2 QoS API (v2.18.0)
QoS daemon

Staging

- **Active operation**
  - Initiated by prepare request

- **Polling**
  - Check file online

SRM: XAttr user.status
CTA: Xrd query prepare

Archiving

- **Passive operation**
  - Initiated by transfer to disk buffer

- **Polling**
  - Check file nearline

SRM: XAttr user.status
CTA: Xrd query prepare

QoS Transition

- **Active operation**
  - Initiated by CDMI QoS Transition Request

CDMI QoS Polling
Check target QoS achieved

EGI Conference 2020
DataTransfer Workshop
FTS Archive Monitoring

- **FTS Server**
  - executes file transfer as normal
  - If file transfer is completed and archive-timeout > 0 → ARCHIVING

- **FTS QoS daemon**
  - Fetches ARCHIVING files from the database
  - Polls ARCHIVING files until:
    - on-tape
    - timeout reached (timeout propagated to client)
    - error encountered (propagated to client)
  - Relies on Gfal2 for the polling query
    - Supports SRM & XRootD protocols
Transfer to Tape - Archive Monitoring

- Transfer considered successful when data arrives **on tape**
- On-Tape status checked by FTS
- Enabled by `--archive-timeout <seconds>` job argument

```
-archive-timeout <#>
```

![Diagram of data transfer process]

Source → Tape disk buffer → FTS

Report final state:
- FINISHED
- FAILED
FTS v3.10.0 (+ Gfal2 v2.18.3)

- QoS daemon to replace bringonline daemon
- Support for Archiving & (CDMI) QoS transitions
- Full support for TPC with token authorization
- Improved CLI clients

Stress tested with CTA
Available on FTS3-devel instance
Planned release in November 2020
What’s next?

- Active participation in DOMA Working Groups
  - Upcoming SRM → SRM transfers using HTTP-TPC
  - More granular authorization for JWT token submissions
  - Revise the QoS language

- Explore different backends for the QoS daemon

Deploy the new FTS-Flask REST server (Python 3)
  → Running on FTS3-devel instance

- CentOS 8 releases for FTS & Gfal2
Thank you!

Contributions:
Edward Karavakis
Carles Garcia Cabot
*Andrea Manzi

cern.ch/fts
fts-devel@cern.ch
https://gitlab.cern.ch/fts/fts3
https://gitlab.cern.ch/dmc/gfal2
[Backup] Archiving Job

• **New ARCHIVING state**
  - File goes to ARCHIVING when transfer finished & archive_timeout > 0
  - Job goes to ARCHIVING when all files are in ARCHIVING
  - ARCHIVING transitions to FAILED or FINISHED (CANCELED support to be added)
  - Multihop jobs: only last hop is eligible for ARCHIVING

• Initiate by submitting `archive_timeout=<seconds>` to FTS-REST

```bash
fts-rest-transfer-submit --archive-timeout=<seconds> 
  -s https://fts3-devel.cern.ch:8446/ <src> <tape_dst>
```
FTS implements QoS transitions via the CDMI QoS specification.

The transition is initiated by a CDMI QoS request (request `target_qos`) and considered finished when the `current_qos = target_qos`.

The transition is monitored by FTS via CDMI QoS polling.

```
fts-rest-transfer-submit --target-qos=<qos_value> 
-s https://fts3-devel.cern.ch:8446/ <src> <dst>
```

[1] CDMI: Open data management interface, standardised by SNIA
[2] CDMI-QoS: Extensions to the interface brought by the IndigoDatacloud