FTS & DataChallenge24

Mihai Patrascoiu for the FTS team
Data Challenge Workshop
(9 - 10th November 2023)

FTS & DC'24

FTS commitment for the DataChallenge '24

- Full SciTags Support (Packet Marking)

- Functional OAuth2 Tokens Support (WLCG)

- Monitoring info for SciTags & Token adoption

SciTags

FTS & DC'24 − SciTags

FTS must propagate SciTags (passed by the client) to the involved TPC storages

- Requires work in full FTS stack:

    FTS submission > FTS server > Transfer Agent > Gfal2

- FTS-1829: FTS3 SciTag work → (fts3 3.12.11)

- DMC-1344: Gfal2 Scitag work → (gfal2 2.22.0)

SciTags: Development

Development in accordance with

Flow and Packet Marking Technical Specification (link)

Summary:

- FTS support only for HTTP protocol
    - XRood must use opaque info: `?scitag=<scitag>`
- Header sent always to **active** party
- Validation: $2^6 < scitag < 2^{16}$

```
HTTP COPY /<path>
    ....
    Source: <source_url>
    SciTag: <scitag>
    ....
```

SciTags: Submission

Example:

```
$ gfal-copy --scitag <scitag> <src> <dst>
```

```
$ fts-rest-transfer-submit -s https://fts3-pilot.cern.ch:8446/
          --scitag <scitag> <src> <dst>
```

JSON
submission
```
{
  "files": [
    { ...
      "scitag": <scitag>,
      ...
    }
  ]
}
```

SciTags: Status

- Full support for SciTags available

- Released in FTS v3.12.11 (RPMs available)

- Deployed on FTS3-Pilot

- *Note: Upgrade requires schema change*

- Scitag field available already in Monit

  (`monit_prod_fts_enr_complete_*` index)

The Token Saga

FTS & DC'24 – Tokens

FTS must provide token-based / proxy-less disk-to-disk transfers

during the DataChallenge'24

- Token workflow & development plan presented at <u>DOMA-BDT</u> (21st June 2023)

- FTS will support the standard OAuth2 flow for DC24 (token-exchange + refresh)

- Working prototype already provided on FTS3-Pilot (target goal: DC'24 Workshop)

- Authentication method field sent to Monit
  (monit_prod_fts_enr_complete* / data.auth_method: <certificate|oauth2>)

Tokens Approach

The FTS team wants to test the standard OAuth2 as much as possible

during DC24! (general use-case for most communities; IAM for scalability)

- Each transfer submission requires: 1 token for source, 1 token for destination
- The client has to ensure FTS can perform overwrite / clean-up on destination

DC'24 Timelines

Weeks leading to DC'24 Workshop (9-10th November 2023)

- Finish SciTags and the new Token submission interface

- Deploy to FTS3-Pilot instance

Weeks leading to DC'24

- Implementation continues: full token lifecycle management

Post DC'24

- Review of DC'24 performance

- Move discussion towards tape + tokens

Tokens: At a glance

FTS interaction will require a token, called: **<fts-token>**

- Used by FTS to identify and authorize the client request (sub, VO, [wlcg.groups])

- **May require** `aud` (FTS service) and FTS-specific `scopes` (names not decided yet)

FTS transfers will require tokens for src/dst: **<src-token>** & **<dst-token>**

- Every transfer (submission) will have at least two tokens

Tokens: Client submission

```
$ fts-rest-whoami --fts-access-token <fts-token>
                  -s https://fts3-pilot.cern.ch:8446/
```

```
$ fts-rest-transfer-submit --fts-access-token <fts-token>
              -s https://fts3-pilot.cern.ch:8446/
              --src-access-token <src-token>
              --dst-access-token <dst-token>
                      <src> <dst>
```

```
$ curl -s -X POST -H "Authorization: Bearer <fts-token>"
        --data @submission.json
                https://fts3-pilot.cern.ch:8446/jobs
```

Tokens: Client submission

A **token for each transfer** must be provided!

```
{
  "files": {
    "sources": [URL1, URL2, ...],
    "destinations": [URL3, URL4, ...],
    "checksum": <xsum>,
    "filesize": <size>,
    "metadata": <metadata>
  },
  "params": { ... }
}
```
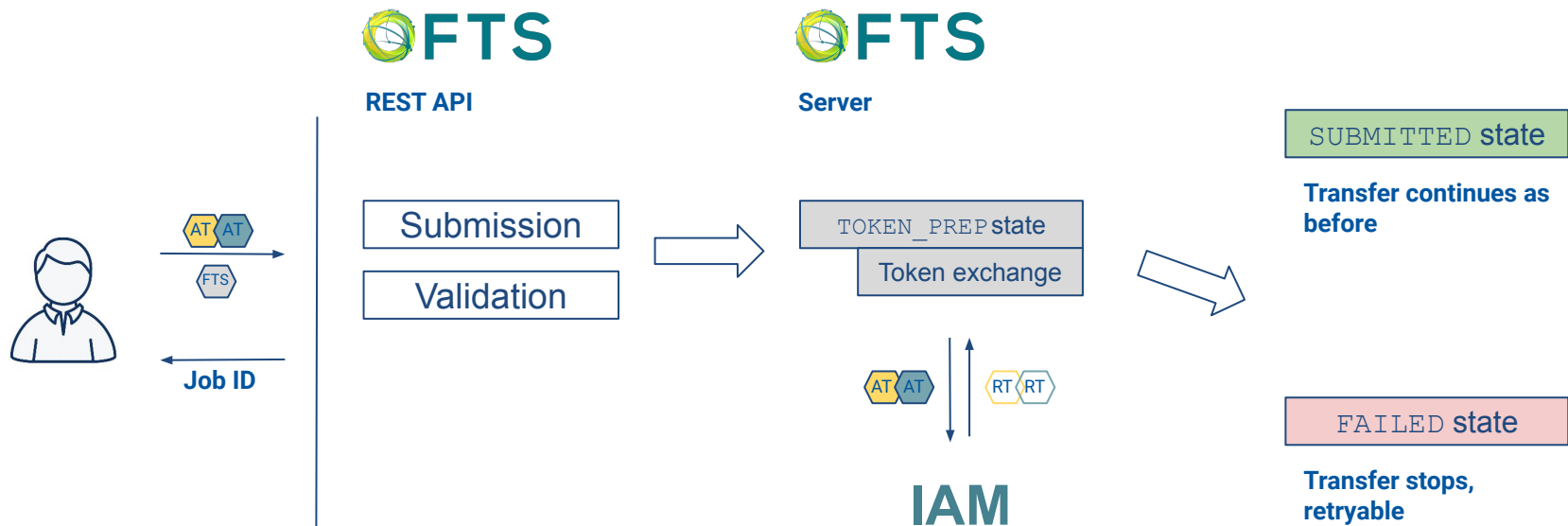
```
{
  "files": {
    "sources": [URL1, URL2, ...],
    "destinations": [URL3, URL4, ...],
    "source_tokens": [AT1, AT2, ...],
    "destination_tokens": [AT3, AT4, ...],
    "checksum": <xsum>,
    "filesize": <size>,
    "metadata": <metadata>
  },
  "params": { ... }
}
```
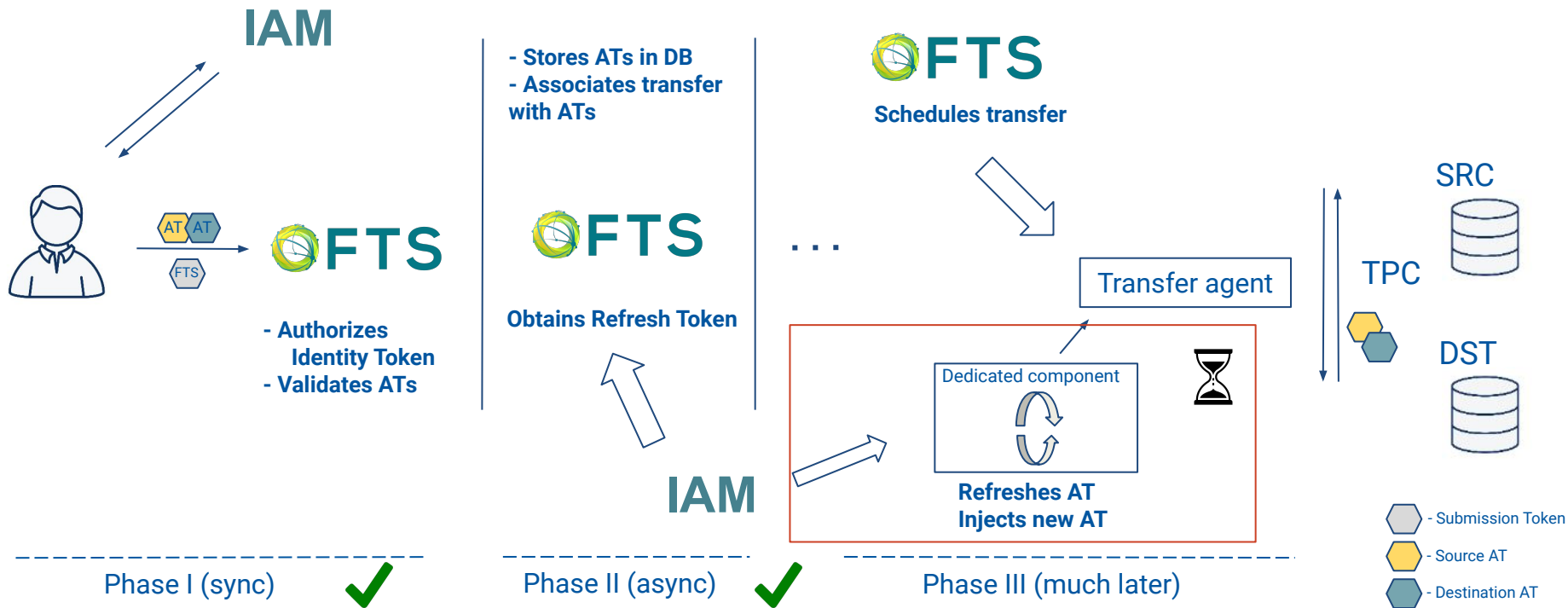
Tokens: A new `TOKEN_PREP` state

Transfers submitted with tokens go through a new, initial state: `TOKEN_PREP`

- First state in the transfer state diagram

- Tokens in this state are awaiting for a refresh token to be obtained

  - Successful token-exchange → `SUBMITTED` state

  - Failed token-exchange → `FAILED` state

- If refresh token pre-exists → `SUBMITTED` state

Tokens: `TOKEN_PREP` diagram

REST API

Server

Submission

Validation

Job ID

`TOKEN_PREP` state

Token exchange

IAM

`SUBMITTED` state

Transfer continues as before

`FAILED` state

Transfer stops, retryable

Tokens: lifecycle management

IAM

FTS

- Stores ATs in DB
- Associates transfer with ATs

Schedules transfer

AT AT FTS

FTS

FTS

. . .

Transfer agent

Obtains Refresh Token

- Authorizes Identity Token
- Validates ATs

SRC

TPC

DST

IAM

Dedicated component

Refreshes AT
Injects new AT

- - - - - - - - - - - - Phase I (sync) ✔

Phase II (async) ✔

Phase III (much later)

- Submission Token
- Source AT
- Destination AT

Ending remarks

- Close development between FTS and IAM teams

    - Many thanks to CNAF team and other IAM actors involved

- Prototype up-and-running on FTS3-Pilot

    - Looking for early adopters: clients, different token providers, SEs, etc…

- Looking forward to validating Rucio ↔ FTS interaction

    - …and discovering new errors (from clients and SEs alike)

- FTS development well underway to meet DataChallenge'24

Bonus: Brian's story

Hands-on afternoon with Brian led to first successful CMS IAM FTS transfer!

```
$ export FTS_TOKEN=$(oidc-token CMS --scope "offline_access")

$ export SRC_TOKEN=$(oidc-token WLCG
    --scope "offline_access storage.read:/")

$ export DST_TOKEN=$(oidc-token CMS
    --scope "offline_access
            storage.read:/store/temp/sitetest
            storage.create:/store/temp/sitetest
            storage.modify:/store/temp/sitetest")
```

Bonus: Brian's story

```
$ fts-rest-transfer-submit --fts-access-token ${FTS_TOKEN}
                           --src-access-token ${SRC_TOKEN}
                           --dst-access-token ${DST_TOKEN}
          -s https://fts3-pilot.cern.ch:8446/

    https://eospublic.cern.ch/eos/public/opstest/dteam/hello.txt

  https://eoscms.cern.ch/eos/cms/store/temp/sitetest/bbockelm/hello_world_eospublic.txt

< f8f02394-7d82-11ee-b122-fa163ece561c (log link)
```

```
INFO    Tue, 07 Nov 2023 16:33:25 +0100; Davix: < Content-Type: application/xml; charset=utf-8
INFO    Tue, 07 Nov 2023 16:33:25 +0100; Davix: < Date: Tue, 07 Nov 2023 15:33:25 GMT
INFO    Tue, 07 Nov 2023 16:33:25 +0100; DESTINATION Source and destination file size matching
INFO    Tue, 07 Nov 2023 16:33:25 +0100; Transfer finished successfully
```

Thank you!

Backup

FTS token lifecycle management

Token lifecycle management steps:

1. FTS receives AT from the submitting client

2. Token is validated and associated with transfer in FTS database

3. A refresh token is obtained for the AT as soon as possible (async from submission)

4. Transfer is scheduled with token loaded from the database

   a. Dedicated component refreshes ATs past expiration threshold via **token-exchange**

   b. Dedicated component can re-inject token to transfer when needed

FTS token lifecycle management

New developments needed:

1. Handling of identity token
2. New Submission API (two ATs per transfer)

   Client facing (first to arrive)

3. (Async) "Token-exchange" component
4. (Just-in-Time) "Token-refresh" component

   Impacts IAM

5. Injecting token credentials to Transfer Agent
6. Gfal2 loading tokens from credential file (build on cern-fts/gfal2#13)

Anatomy of the transfer process

Transfer Agent

Check source exists
`gfal2.stat(src)`

Check destination exists
`gfal2.stat(dst)`

Overwrite
`gfal2.rm(dst)`

storage.modify needed

Perform copy
`gfal2.copy(`
`    src, dst)`

storage.modify needed

1. Copy file
2. XSum source
3. XSum dest
4. **ON ERROR: Clean dest file**

Check destination size
`gfal2.stat(dst)`

- Source AT (storage.read)

- Destination AT (storage.create minimum) (storage.modify maximum)