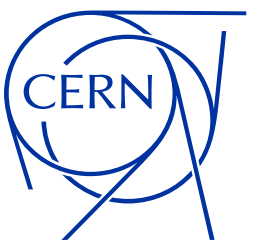
What’s Next for Tokens in Rucio

Dimitrios Christidis



Introduction

- Rucio offers coarse-grained token support since 2019
- New implementation for fine-grained tokens introduced in 2023, for third-party-copy transfer and central deletion
  - As a technology preview
  - Original implementation for other workflows still in place, but not actively maintained nor adequately supported
- A design document describes our understanding and goals

The workflows

- Central
  - Third-party-copy transfer
  - Deletion
- Client
  - Authentication
  - Download
  - Upload

Abbreviated anatomy of a token

```
{
  "sub": "12345678-9abc-def1-2345-6789abcdef12",
  "aud": "eosatlas.cern.ch",
  "scope": "storage.read:/atlasdatadisk offline_access",
  "exp": 123456789,
  …
}
```

Most of the decisions are related to the scope

Third-party-copy transfer: aftermath of DC24

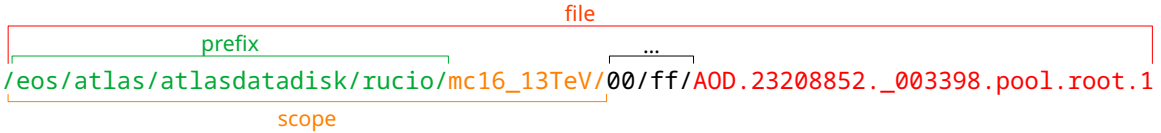
- DC24 was the first successful demonstration of the new token implementation at scale
- Performance and robustness:
  - The FTS token-refresh workflow was primitive; it has since been improved, with further improvements expected in the future
  - Capabilities of the token providers remain insufficiently explored
- Security:
  - RSE-wide `storage.modify` tokens are considered a security risk

FTS token-refresh workflow

- Idea: use long-lived tokens without `offline_access` scope
  - Seemingly less expensive for FTS and the token provider
  - If a token associated with an FTS transfer expires, then the transfer fails and it has to be resubmitted
  - Unlikely to become the default, but offers additional flexibility
- On its own, worse for security
  - Advantageous if paired with a more restrictive resource path

Resource path

- At least the RSE prefix, at most the path to the file
- Can be something in-between (e.g. scope), only if the LFN2PFN algorithm allows
  - Thus, only RSE-wide and file-specific can be offered by default



Risk assessment

- `storage.read`: data leak, service degradation
- `storage.stage`: service degradation
- `storage.create`: service degradation (including filling the storage with garbage), ‘mess’ (mass rename of existing files)
- `storage.modify`: *data loss*

Ongoing experimentation

- CMS:
  - Dataset-specific destination tokens
  - Refer to R. Chauhan's talk
- ATLAS:
  - File-specific destination tokens, without refresh
  - 2–5 Hz token-request rate, up to 10% of daily transfers

Upcoming experimentation

- Idea: use of `storage.create` on initial attempt, `storage.modify` on retries
  - Related to issue #7056
  - The `Recovery` activity and problematic storages may create unpredictable spikes in the token-request rate
- FTS won't be able to clean up remnants of failed transfers
  - But either they will be retried, or the Rucio Reaper will clean them

Tokens for tape RSEs

- For writing, tests to commence in the coming weeks
  - If overwriting is never allowed, then RSE-wide `storage.create` tokens might be always sufficient
- For reading (staging), a design discussion took place at the FTS & XRootD Workshop
  - Doesn't change anything for Rucio (use of `storage.stage` scope)
  - Awaiting development and deployment

Profile compliance is a challenge

- Paraphrasing P. Vokac: ‘everything is described in the standards, so the behaviour is implementation dependant’
- Two kinds of divergence:
  - Not authorising tokens that should be (annoying)
  - Authorising tokens that shouldn’t be (scary)
- This has to be a collective effort

Central deletion

- Workflow considered mature
- RSE-wide `storage.modify` is not considered a security risk
  - The audience restriction is sufficient
- Upcoming work is related to the Rucio protocol implementations (similar to the client workflows)

Client download: new REST API endpoint

- Clients must be able to request tokens
- A new REST API endpoint will be introduced whose parameters include the operation, DID, account, and RSE
  - Scope and resource path considerations similar to TPC transfer
- This endpoint will be clearly marked as experimental and no compatibility will be offered except for matching versions of the Rucio servers and clients

Client download: initial implementation

- Hidden behind a client configuration
- Won’t include any mechanism to make efficient reuse of tokens
- Will continue to be exclusive to WebDAV

Client download: expected challenges

- XRootD is more common than WebDAV for client workflows
- GFAL (and the XRootD client) do a certificate auto-discovery, which hinders the use of tokens
- Replacing the GFAL Rucio protocol implementation appears necessary, but comes with a lot of additional challenges
  - Decided to prioritise the API endpoint and the initial download implementation, but this remains an important prerequisite

Remaining client workflows

- Authentication: need to coordinate with other projects
  - OAuth user experience for CLI apps can be poor
  - Luckily, download and upload are independent
- Upload: begin development after the client download workflow is implemented and has somewhat matured

Near-term goals

- Continue the refinement of the third-party-copy transfer workflow until it can be considered production-ready
  - A reasonable and safe default configuration
- Introduce a plugin-based method to deviate from the default
  - Also consider other token providers and profiles
- Launch client download as a technology preview
- Publish a revision of the token design document

Questions?