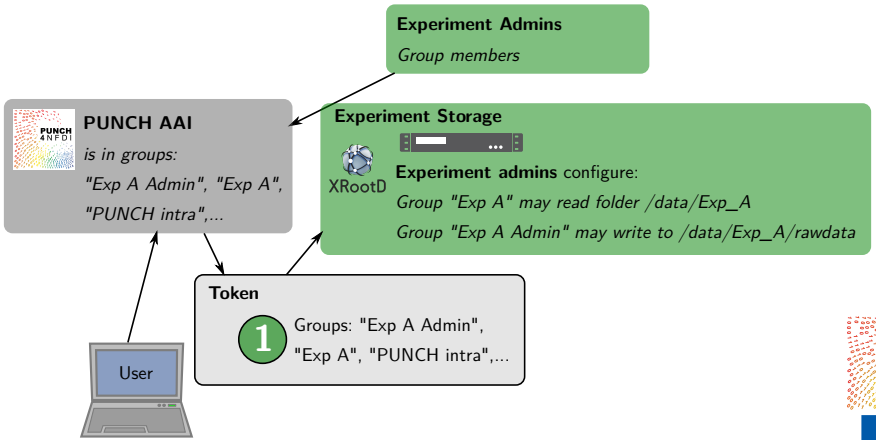
PUNCH AAI Developments

Oliver Freyermuth, Kilian Schwarz, TA6, Christoph Wissing

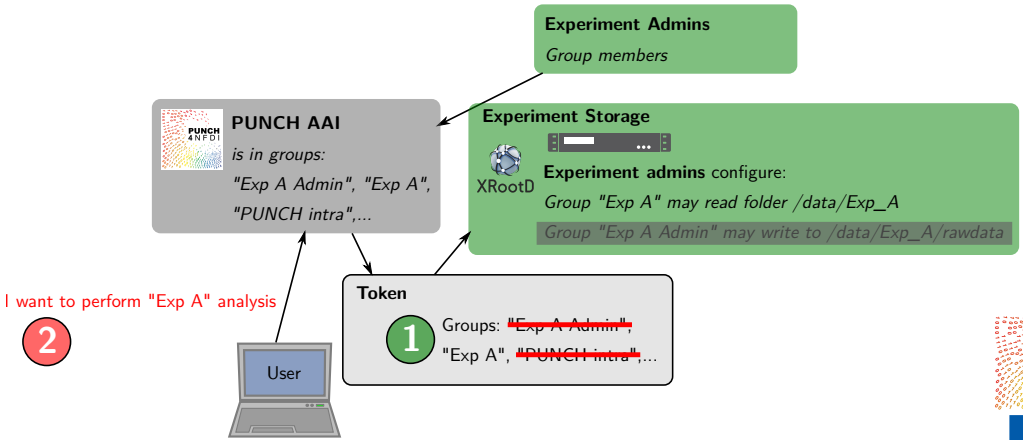
25th June, 2025

UNIVERSITÄT BONN

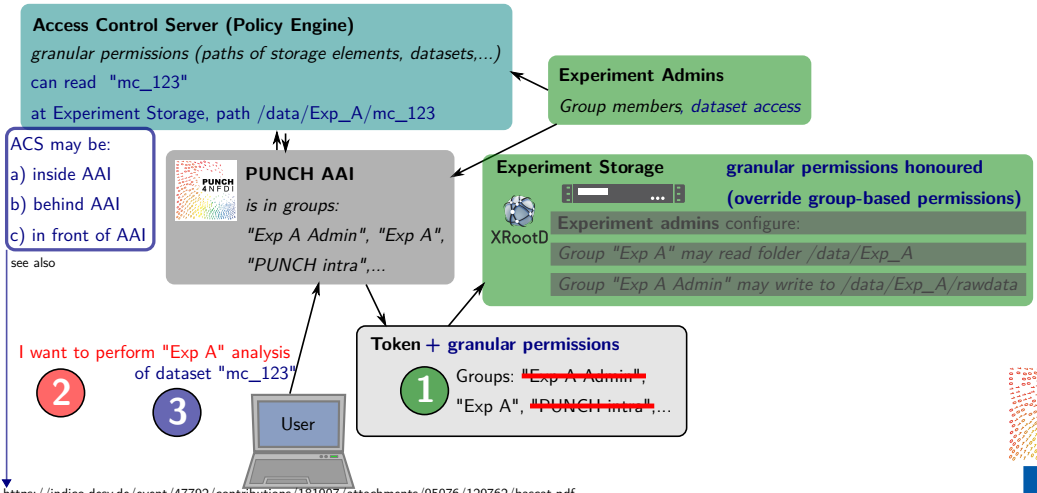
Visualization of Feature Requests: Group-based approach



Visualization of Feature Requests: Claims filtering



Visualization of Feature Requests: Granular permissions



Status of Requests

Request 1 'Group information in tokens'

✓ Implemented by Unity before we made an official request
⇒ Still needs to be tested in full within PUNCH
Note: (Delegatable) group management already offered by the AAI.

Request 2 'Claim filtering' (i. e. not all groups in tokens)

✓ Implemented and tested (still need to test workflows)

Request 3 'Granular permissions'

⧗ Description finalized, got quotations for different possibilities,
now finalizing description of the policy engine API

Granular Authorization

Storage

Scopes: `storage.read`, `storage.create`, `storage.modify`, `storage.write`

- Usually scoped for a specific path, i. e. `storage.read:/someexp/somefile.dat`
- Inspired by WLCG Common JWT Profiles which are inspired by SciTokens
- Tested also in ILDG with Indigo IAM
- Used in the tools in Storage4PUNCH: dCache, XRootD
- May also be used inside Rucio / FTS (now or in the future)

Compute

Scopes: `compute.read`, `compute.modify`, `compute.create`, `compute.cancel`

- May be scoped to a specific compute resource
- Expected by Compute4PUNCH batch system (HTCondor)
- Inspired by WLCG Common JWT Profiles which are inspired by SciTokens

Technical Solution 'A': External Policy Service 'behind' AAI

- When a specific (custom) scope pattern is requested by the user (e. g. `storage.read:*`), an external service (with REST API) is queried by the AAI.
- The API of that external service is currently discussed. A potential specification could be:
  - For `http_status==200`, output MUST be a single JSON object.
  - 'Permission denied' is also possible.

Example (REST-like API)

Input Request with information (user, audience etc.) sent to the external service for triggering scope (`storage.read:/punch4nfdi/subdir/file`).

Output One single JSON object, the result of which is added as a sub-json to the JWT-AT which is generated for the user, Example:
`{["storage.read:/punch4nfdi/subdir/file"]}`

Technical Solution ‘A’: External Policy Service ‘behind’ AAI

Example JWT-AT generated by the AAI:


```
{
        "typ": "at+jwt",
        "alg": "RS256"
}
{
        "sub": "6c611e2a-2c1c-487f-9948-c058a36c8f0e",
        "aud": "public-oidc-agent",
        "scope": "openid offline_access storage.read:/punch4nfdi/subdir/file",
        "iss": "https://login.helmholtz.de/oauth2",
        "exp": 1683731886,
        "iat": 1683727886,
        "jti": "108ed829-9871-4f23-922b-be977be48476",
        "client_id": "public-oidc-agent",
        "storage": {
            [
                "storage.read:/punch4nfdi/subdir/file"
            ]
        }
}
```

Technical Solution 'B': Token Exchange with Trusted Client

Enabling full support for token-exchange endpoints (see https://indigo-iam.github.io/v/v1.7.2/docs/reference/api/oauth-token-exchange/ and https://www.rfc-editor.org/rfc/rfc8693 allowing community-specific services, which act as registered and trusted clients of the AAI, to request access tokens with fine-grained additional scopes, e.g. scopes not present in the Bearer token and implementing capability-based authorization.
Users can then request access tokens through such a client, which decides (based on community-defined access policies) whether to perform the token exchange and to return the desired AT to the user.

concept by Hubert Simma

Technical Solution: How to proceed

Extended AAI Meeting June 4th, fruitful discussion and decisions:

1. Go for Policy Engine behind AAI with an API queried by the AAI ('solution A')
2. Keep path open to Token Exchange as alternative possibility (i. e. 'solution B')
3. Add VO name (`punch4nfdi` in our case) as path component

Note we stay in a close loop with Helmholtz / HIFIS and the Unity developer team.

Feedback by Unity Developers

Token Exchange 'behind the AAI' may lead to scaling issues and extra validation steps.
$\Rightarrow$ Go for 'solution A' and use a REST-like API,
if needed, develop 'solution B' later.

Open Tasks

Reminder

General goal: Try to stay close to WLCG workflows (to reuse tools).
Small changes to dCache & XRootD needed for different token profile (fine with both).

Tasks

- Development of Policy Engine service with API needed (can reuse existing tools such as Open Policy Agent).
- AAI is part of IAM4NFDI community AAIs and hence used by different NFDI consortia, so we need to embed the VO in our scopes, plan:
  - `storage.read:/punch4nfdi/` etc. for storage
    ⇒ No changes to tools needed, i. e. works for XRootD and dCache without chan
  - `compute.read:punch4nfdi` etc. for compute
    ⇒ Still need to investigate whether HTCondor or underlying token library need modification.

Note: VO part would not be needed with dedicated instance.

UNIVERSITÄT BONN

Timeline & Outlook

1. Finalize description of interface between AAI and policy engine and send request to developers.
2. After development request is with Unity developers, implementation is fast (about a month).
3. Have a simple service emulating the interface of a policy engine (e. g. always with a constant reply) in PUNCH 1.0.
4. Actual implementation of full Policy Engine $\Rightarrow$ PUNCH 2.0.
5. Keep the possibility to develop 'variant B' (Token Exchange interface) later on.

Note: Local POSIX File Systems

Local POSIX file systems can not be integrated directly with any AAI.
One approach: Use them through an AAI-enabled storage service.

Note: Indico integrated with AAI via plugin by `unconventional.dev`.

UNIVERSITÄT BONN

Thank you

for your attention!

Embed group information in tokens

For now, special procedure with `oidc-agent` required (see documentation, simplification foreseen).

Access Token


```
{
 [...],
 "preferred_username": "o.freyermuth",
 "scope": "openid offline_access profile",
 "eduperson_entitlement": [
  "urn:mace:dir:entitlement:common-lib-terms",
  "urn:geant:dfn.de:nfdi.de:punch:group:PUNCH4NFDI:punch_intra#login.helmholtz.de",
  "urn:geant:h-df.de:group:HDF#login.helmholtz.de",
  "urn:geant:dfn.de:nfdi.de:punch:group:PUNCH4NFDI#login.helmholtz.de",
  "urn:geant:dfn.de:nfdi.de:punch:group:PUNCH4NFDI:elsa_one#login.helmholtz.de"
 ]
}
```

Reduced permissions in tokens

- Do not expose all groups to used service
- Work with reduced / minimal privileges

Token Request

```
scope=eduperson_entitlement:...:PUNCH4NFDI:elsa_one#login.helmholtz.de
```

Access Token

```
{
 [...],
 "preferred_username": "o.freyermuth",
 "scope": "openid offline_access profile
 ↪  eduperson_entitlement:...:PUNCH4NFDI:elsa_one#login.helmholtz.de",
 "eduperson_entitlement": [
  "urn:geant:dfn.de:nfdi.de:punch:group:PUNCH4NFDI:elsa_one#login.helmholtz.de"
 ]
}
```

Query PUNCH service for granular permissions

- Granular file access permissions require a scope policy system
- Extension in Unity not likely ⇒ external Access Control Server

Token Request

```
scope=storage.read:/example/subdir/file
```

Access Token

```
{
 [...],
 "preferred_username": "o.freyermuth",
 "scope": "openid offline_access profile storage.read:/example/subdir/file",
 "storage": {
          [
   "storage.read:/example/subdir/file"
  ]
 }
}
```

Technical Solution ‘B’: Token Exchange with Trusted Client

Enabling full support for token-exchange endpoints (see https://indigo-iam.github.io/v/v1.7.2/docs/reference/api/oauth-token-exchange/ and https://www.rfc-editor.org/rfc/rfc8693 allowing community-specific services, which act as registered and trusted clients of the AAI, to request access tokens with fine-grained additional scopes, e.g. scopes not present in the Bearer token and implementing capability-based authorization.
Users can then request access tokens through such a client, which decides (based on community-defined access policies) whether to perform the token exchange and to return the desired AT to the user.

Some ‘Pros’ and ‘Cons’

Variant A: Policy Engine ‘behind’ AAI

- 👍 Acts more like the AAI used in WLCG (Indigo), i. e. as if policy engine was embedded
- 👍 All clients can directly talk to the AAI (e. g. `oidc-agent`)
- 👍 Service needs only to be reachable from AAI
- 👎 Not a ‘standardized’ approach

Variant B: Trusted Client with Policy Engine in front of AAI

- 👍 Standardized approach
- 👎 Clients need to go via the trusted client service
- 👎 PUNCH needs to operate a world-wide reachable service exchanging tokens for users