

dCache, towards Federated Identities and Anonymized Delegation

Paul Millar, on behalf of the dCache team

ISGC 2017 at Taipei, Taiwan

2017-03-09

<http://indico4.twgrid.org/indico/event/2/session/27/contribution/54>



Quick recap






Authn

Authz

Delegated Authentication

Sign-in using

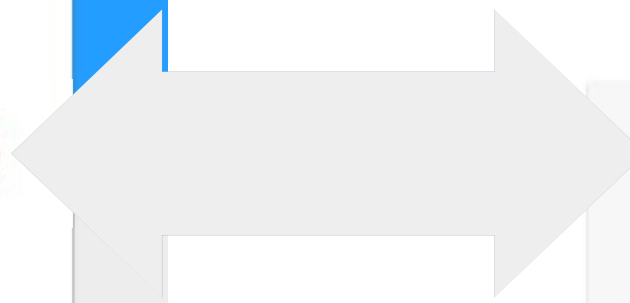
Facebook Twitter Google

or

Email or Username

Password


[Sign-in](#) or [create an account](#)



Google

One account. All of Google.

Sign in with your Google Account



Email

Password

[Sign In](#)

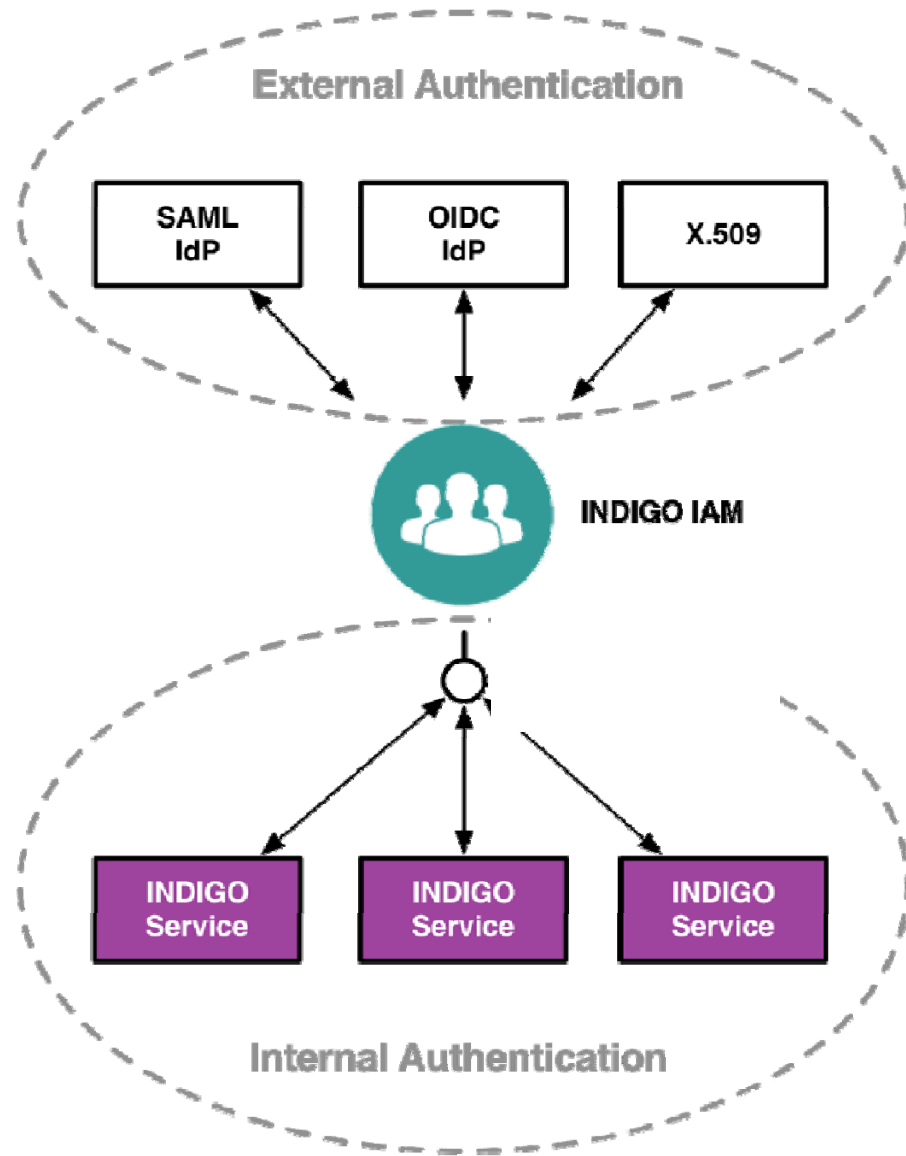
Stay signed in [Need help?](#)


[Create an account](#)

One Google Account for everything Google



INDIGO-DataCloud AAI framework





The INDIGO-DataCloud AAI

A. Ceccanti, M. Hardt, B. Wegh, P. Millar, S. Licehammer, M. Caberletti, E. Vianello

The INDIGO **Authentication and Authorization Infrastructure (AAI)** leverages and extends existing standards (OpenID Connect, OAuth2, SCIM) to enable secure composition of resources from multiple providers in support of scientific applications. The central **Identity and Access Management (IAM)** service provides tools to implement brokered user authentication, identity harmonization, account linking as well as Virtual Organization (VO) management. Identity information is provided to relying services via **OpenID Connect**, which allows simpler integration in off-the-shelf software. The **Token Translation Service (TTS)** integrates services that do not directly support OpenID Connect, by creating credentials on-demand from the identity information provided by the IAM. This poster introduces the main INDIGO AAI components and highlights the main architectural decisions that were taken during the first year of the INDIGO project.

1. Identity = OpenID Connect

The INDIGO [1] AAI identity layer leverages the **OpenID Connect** [2] standard to provide user authentication and identity information to INDIGO services. This approach provides several advantages:

- Standardized and widely adopted solution
- Accommodation of several authentication mechanisms (via identity brokering)
- Simplified integration in relying services
- Native support for delegation and offline access
- Native support for dynamic client registration

4. Identity provisioning

INDIGO IAM leverages the standard **System for Cross-Domain Identity Management (SCIM)** [8] version 2.0 to implement identity provisioning, de-provisioning and management.

The SCIM APIs provides means to **propagate identity and group information to relying services**, to implement, for instance, **dynamic account creation and other resource lifecycle management at various levels of the INDIGO infrastructure** depending on events related to user identity status.

2. Identity harmonization and brokering

The **INDIGO Identity and Access Management (IAM)** [3] service deals with user authentication and identity brokering, allowing users to authenticate with local username and password, SAML, X.509 certificates, or via a remote OpenID Connect provider (e.g., Google). The different identities are linked to a single INDIGO user profile, which provides each user with a **unique, persistent identifier**. This identifier is then used for auditing, accounting and authorization at all relying services. Other attributes, like group membership information, can be linked to the user profile, and then exposed to relying services via standard **OpenID Connect** interfaces.

5. Delegation and offline access

OAuth and OpenID Connect **support delegation and offline access natively**. To support **chained delegation across services**, in which a component can act both as a service and as a client for another downstream service, the INDIGO IAM provides a partial implementation of the **OAuth token exchange draft standard** [6]. The token exchange is used in particular to implement **controlled delegation of offline access rights across applications** (i.e., the ability to execute tasks on behalf of a user while the user is not connected).

3. Authorization = OAuth 2 + XACML

INDIGO services expose functionality through HTTP APIs protected by **OAuth 2** [4]; only agents presenting a valid and trusted **OAuth access token** are granted access to INDIGO services. This token is obtained by client applications from the **INDIGO IAM service** and provides access to identity information (e.g., group membership and other attributes) and other authorization information (e.g., OAuth scopes). Fine-grained, powerful and distributed attribute-based authorization is implemented by integrating the **Argus authorization service** [5] with the INDIGO AAI identity layer. This provides policy distribution and centralized XACML policy management.

6. Token translation and non-HTTP services

INDIGO AAI integrates services that rely on different authentication mechanisms (e.g., X.509 certificates, SSH keys, Amazon S3 keys) via the **INDIGO Token Translation Service (TTS)** [7]. The **TTS** translates an OpenID Connect authentication assertion, like the one issued by the IAM service, into one of the supported downstream service credentials. The TTS currently supports the generation of:

- SSH keys
- X.509 certificates
- OpenNebula username/password credentials

7. References

- The INDIGO-DataCloud project: <https://www.indigo-datacloud.eu/>
- OpenID Connect: <http://openid.net/connect/>
- INDIGO Identity and Access Management: <https://github.com/indigo-aa/i-am>
- OAuth 2.0: <https://tools.ietf.org/html/rfc6750>
- Argus Authorization Service: <https://github.com/indigo-aa/argus>
- OAuth token exchange draft standard: <https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-01>
- The INDIGO Token Translation Service: <https://github.com/indigo-aa/tts>
- SCIM: <http://www.oasis-open.org/committees/download.php?ctid=3232>

OpenID-Connect under the hood

```
paul@sparkplug:~$ curl -D- -s -o/dev/null -L https://prometheus.desy.de/Users/paul
HTTP/1.1 401 Unauthorized
paul@sparkplug:~$
```



Log into IAM.
Get access token

```
paul@sparkplug:~$ curl -L -H "Authorization: Bearer ey..QT7s" https://prometheus.d
This is a demo file containing some limited information to demonstrate permission h
```

This is a private file, only user paul can read it.

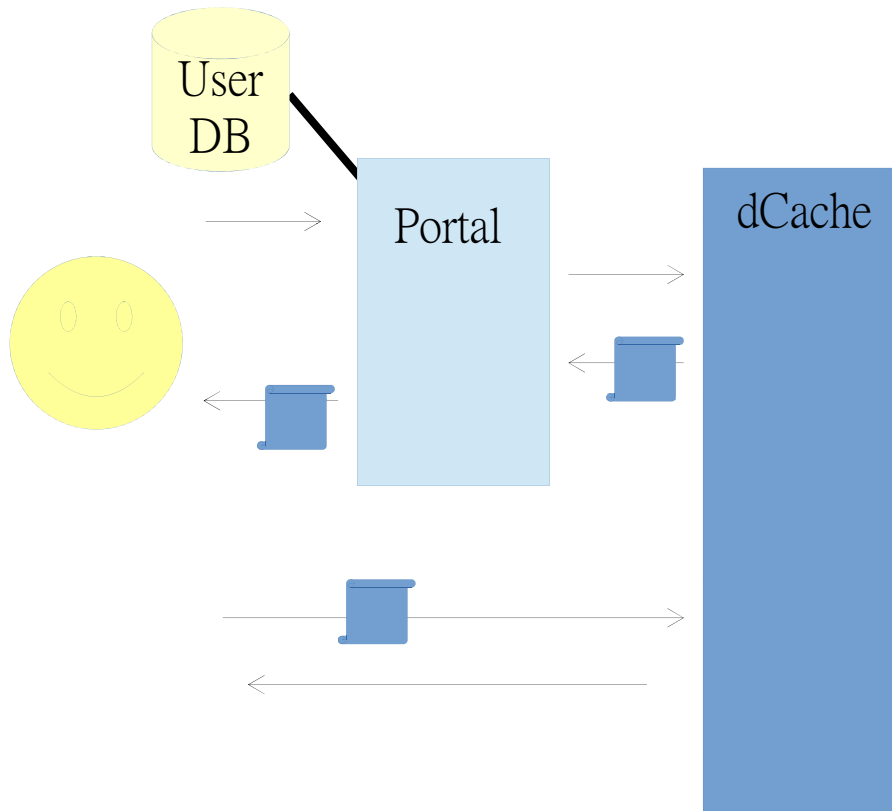
```
paul@sparkplug:~$
```

Authorisation without authentication?

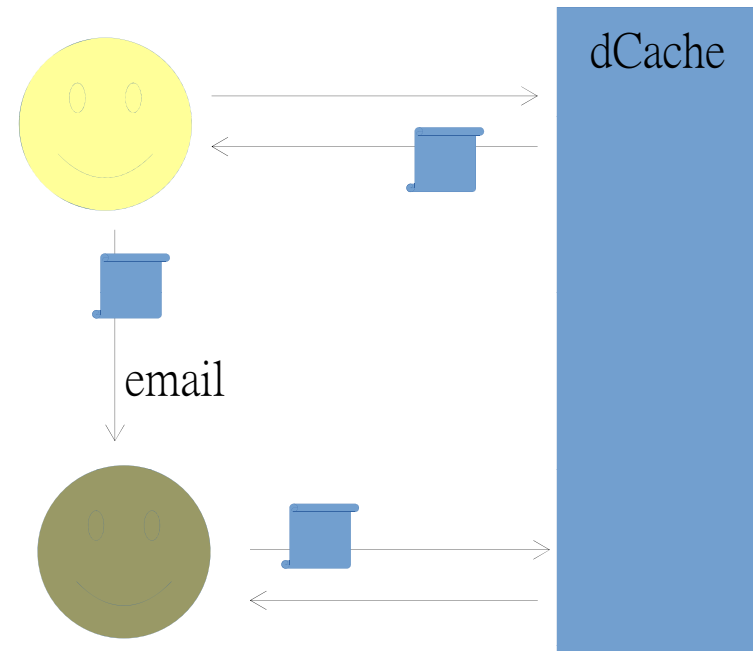


Photo by Alan Cleaver (CC-BY)

Authz token use-cases



Community Portals



Sharing

Constraints and solutions

- Redirection should work **without JavaScript**,
- Simple: **embed token** in redirection URL.

`http://webdav.example.org/path/to/file?authz= <TOKEN>`

- **Complete token** always sent with the request.
 - What can we do to stop someone **stealing** this token?
 - ... or make the token useless if they steal it.
-

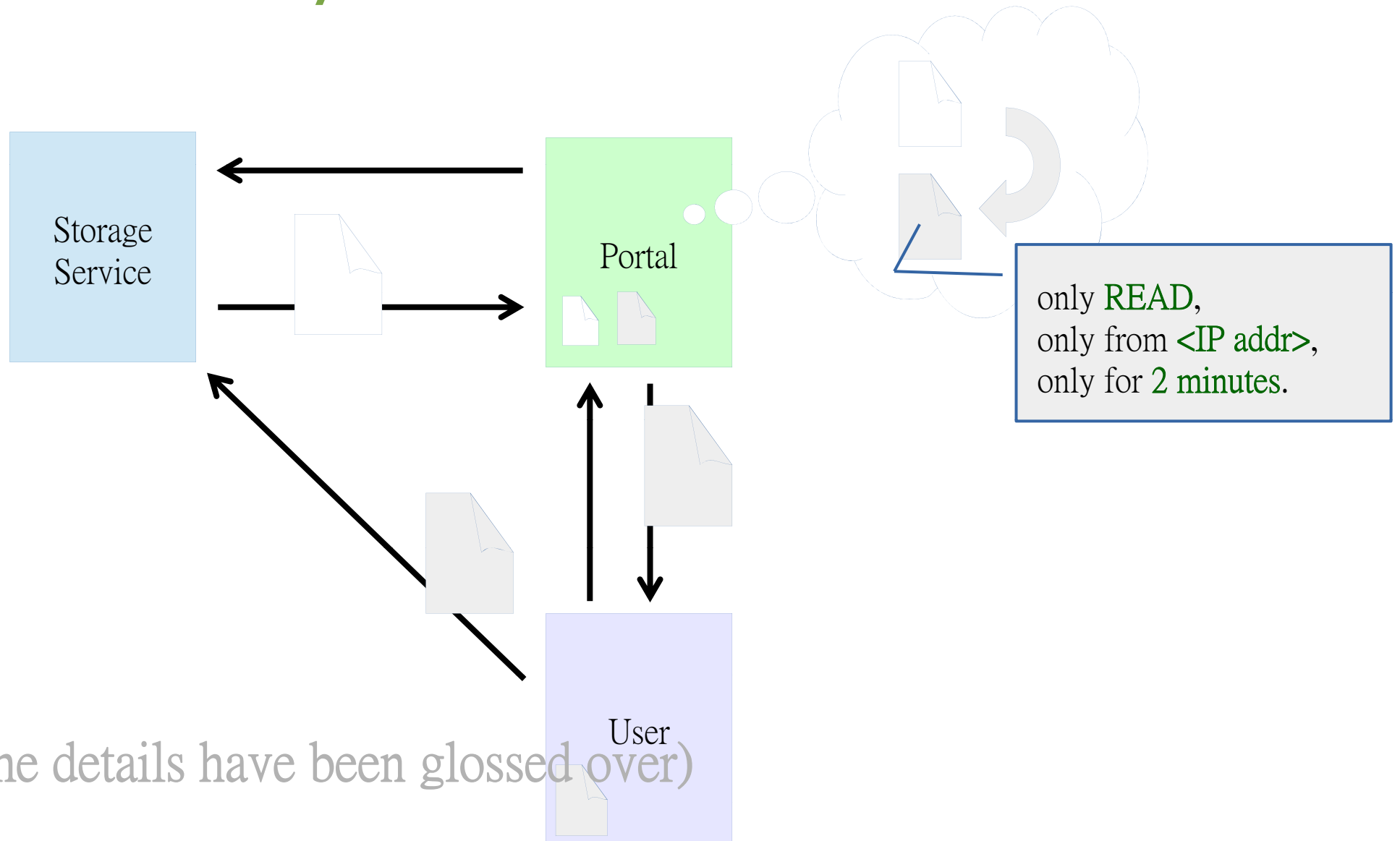
Introducing Macaroons



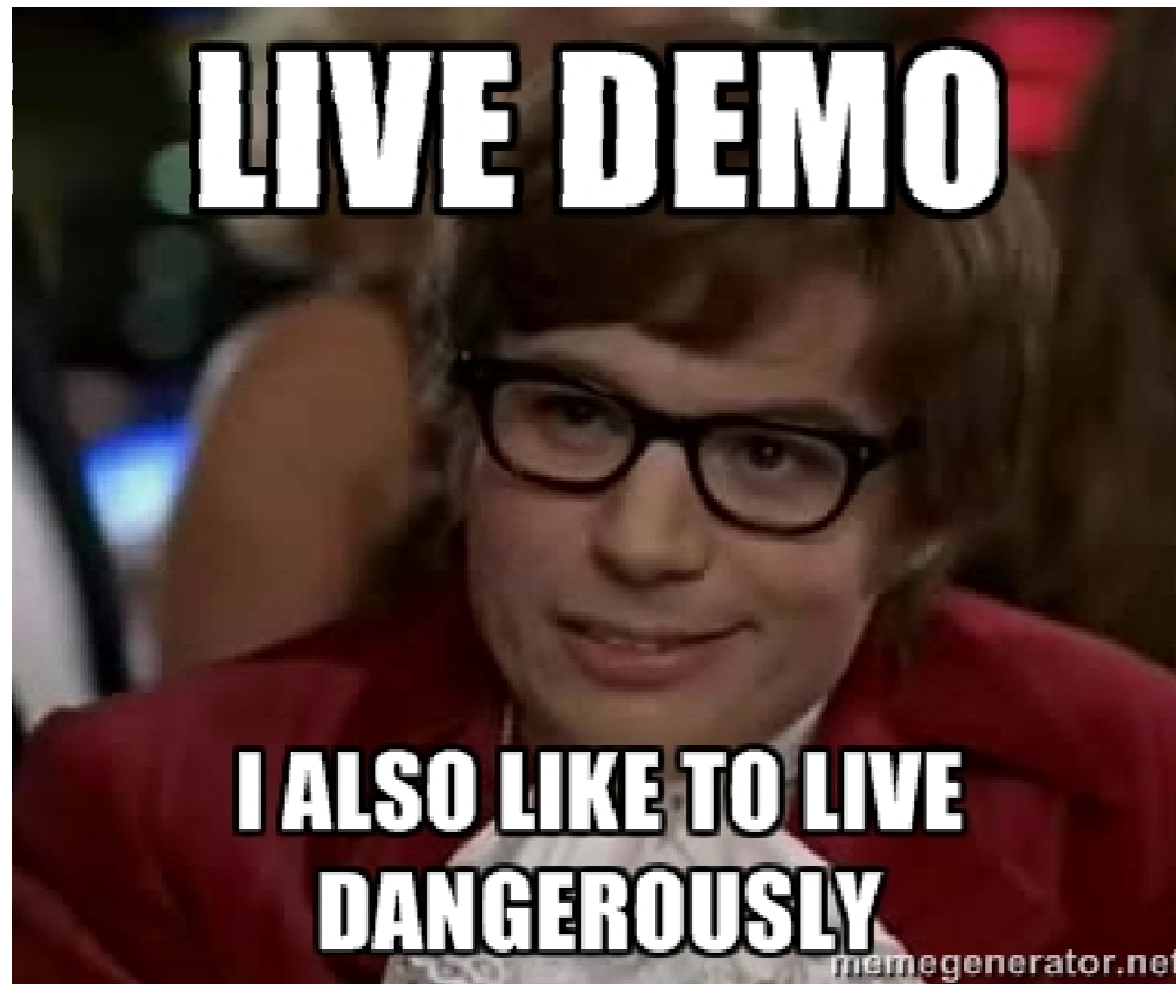
Macaroons 101

- Macaroon is a **bearer token**.
- Macaroon contains zero or more **caveats**.
- Each caveat **limits** something about the macaroon:
 - who** can use it,
 - when** they can use it, or
 - what** they do with it.
- Anyone can **add** a caveat to a macaroon:
 - Create a new macaroon that is more limited.
- The caveats in a macaroon cannot be

Download / Share with macarons



(some details have been glossed over)

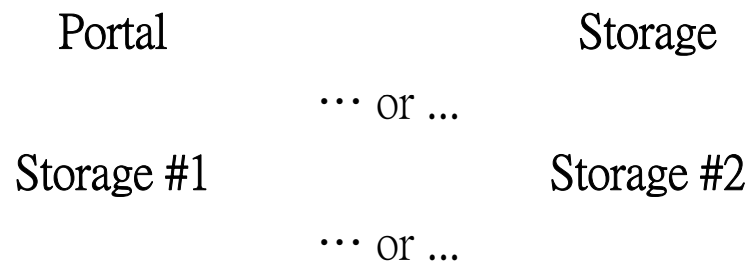


Coming to dCache with v3.1

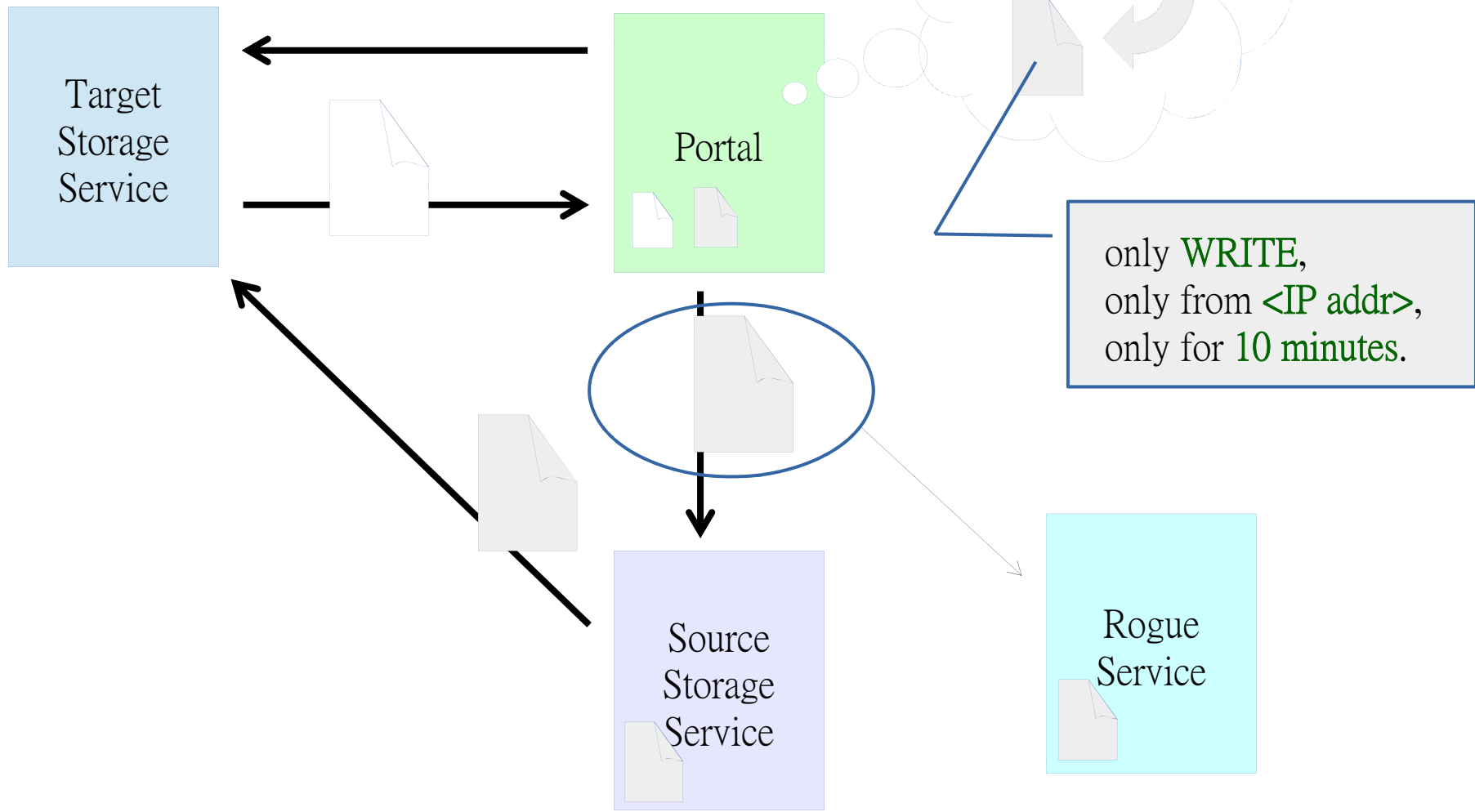
- Supporting **caveats** with IP address, paths, activity*, expiry time.
 - * list, download, upload, manage, delete, read-metadata, update-metadata.
 - Simple **RESTful API** to acquire a macaroon.
 - Initially targeting **HTTP/WebDAV**.
 - Anticipate support for GridFTP coming next
-

Backup slides

OpenID Connect delegation



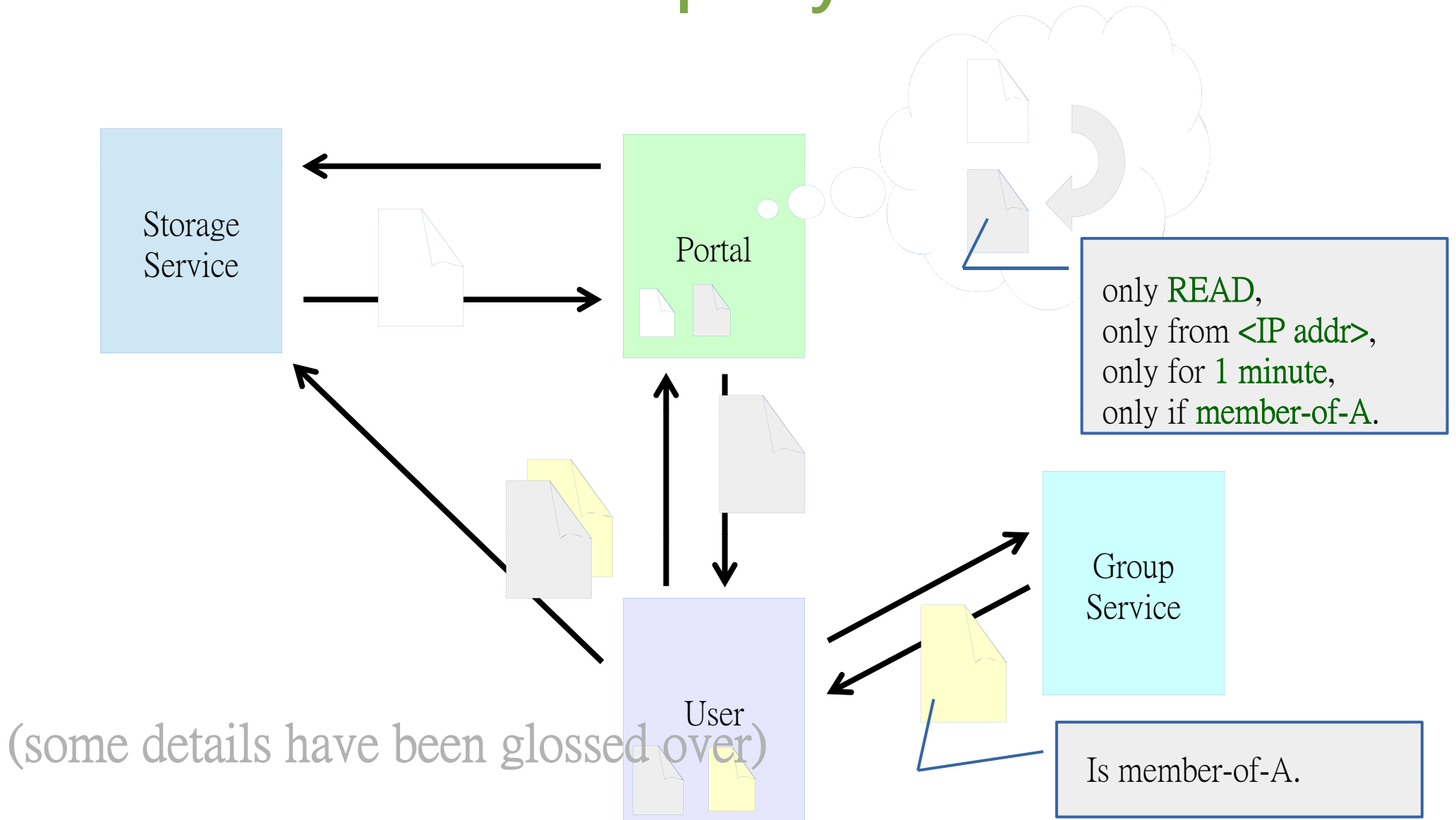
Example: 3rd-party copy



3rd party caveats – extra cool!

- A 1st party caveat can be satisfied by the client.
 - A 3rd party caveat requires proof from some other service; e.g.
 - only **fred@facebook**,
 - only members of **VO ATLAS**,
 - only if not part of a **denial-of-service attack**.
 - The proof is another macaroon: a **discharge macaroon**.
-

Download with 3rd-party caveat



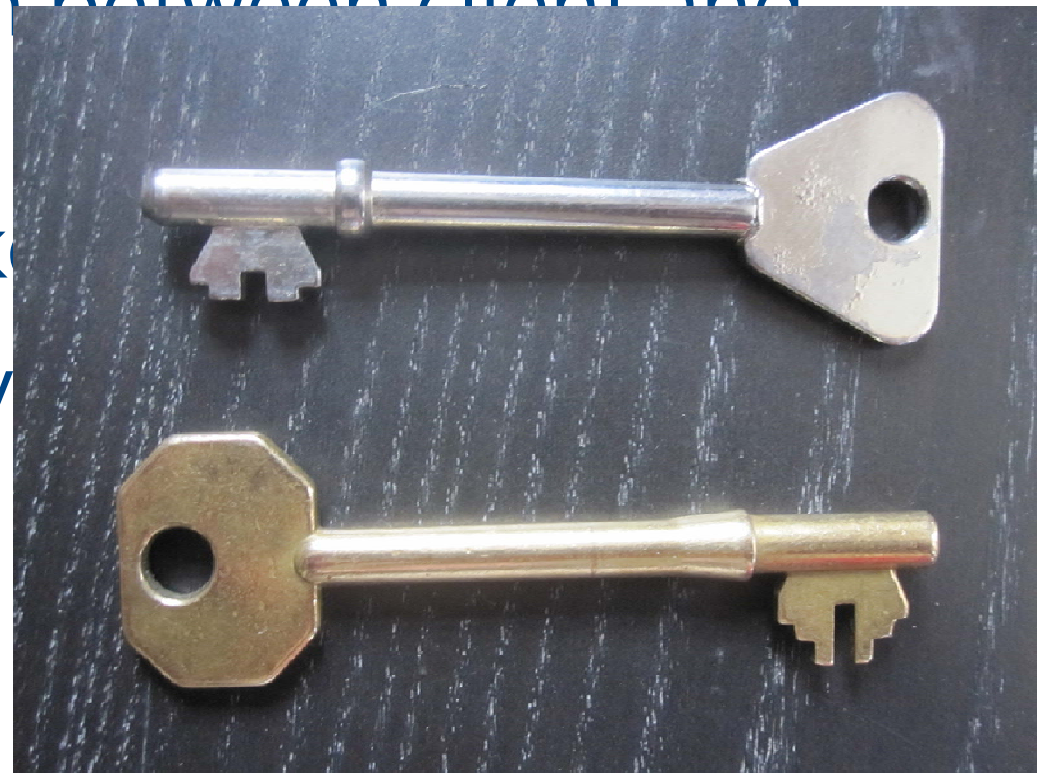
What are bearer tokens?

Bearer token is something the user presents with a request so the server will authorise it. There's no interaction between client and server.

Examples of bearer tokens:
HTTP BASIC authn, any
stored as a cookies.

Counter-examples:

- X.509 credential,
- SAML,



Group membership, too

- An OIDC provider can assert the user is a member of various groups
- Group membership may require higher level of LoA:

For example, if the group is “loose collaboration” a site might require higher LoA; if the group is “commercial entity” a site might require lower LoA

One solution: a bearer token

