Analysis Facilities AAI

Brian Bockelman

AAI: Authentication and Authorization Infrastructure

What is the system or infrastructure that controls access to resources?

AAI: A Primer

- Authorization: Determining access rights or privileges to a given resource.
- While I may talk about this in terms of computing, authorization is familiar in our everyday life:
  - My ID card is used to access the elevators at the university.
  - My passport allows me to get into the country.
  - A baseball ticket gets me access into a baseball game.
  - A Zoom URL will get you into a boring meeting!

$$A_Z(\ldots) \rightarrow P_1, P_2, \ldots, P_N$$

Authentication, Authorization, Credentials, Oh My!

Often, the process of authorization involves two other items: authentication and credentials.

- Authentication: establishing an identity for a remote entity.
- Credentials: knowledge that establishes a fact (identity).
  - Not too far off from the 'credentials' the university provides: a diploma establishes the bearer has particular knowledge.

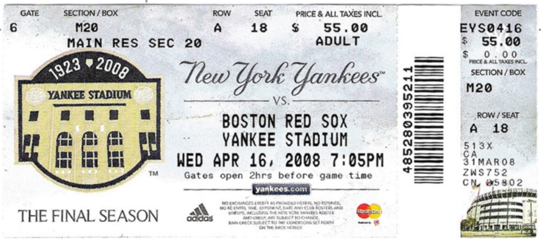
Potentially, username and password is the most familiar authorization scheme:

- A remote entity provides a username/password (credentials) to authenticate to an identity.
- This identity is mapped to a list of authorized access rights.

$$A_N(U, P) \text{ -> } I \qquad A_Z(I) \text{ -> } P_1, P_2, \ldots, P_N$$

Not every AAI requires authentication

AAI	Credentials	Authentication	Authorization
Gmail login	Password, 2FA	Username	Access to your inbox
Building access	ID card	Identity in HR database	Elevators
International Travel	Passport	Identity according to US Government	Enter Switzerland
Baseball Game	Ticket	NONE!	Sit in section 4, seat 34B
Webinar	Zoom URL	NONE!	Attend a wonderful seminar!



WLCG’s X.509-based AAI

Let’s think about how the WLCG’s X.509 AAI works:

- We establish a set of trusted entities (trust roots or, for X.509, CAs).
- Everyone picks their favorite CA, establishes an identity, and gets a credential.
    - **You** are responsible to managing the credential. Good luck!
- User then joins an organization (VO). The user’s X.509 identities are registered in a service that will issue signed assertions of membership (‘VOMS proxy’).
- Authorization at a resource provider is typically done based on the signed assertion (Alternate: resource provider periodically queries VO for membership lists).

WLCG AAI pain points

What pain points exist for users in this system?

- **Credential management**: Users are responsible for managing credentials.
- **Double jeopardy**: Identity vetting once at the CA and again to join the VO.
- **Obscure**: No one else uses this! Can’t rely on institutions for help.
- **Incompatible with browsers**: X.509 client certs can be used in browsers - but lose the VOMS aspects and delegation.
- **Narrow ecosystem**: If you want software to support GSI, then you’ll be writing GSI code. No broad ecosystem to share the burden with!

A token-based AAI for the WLCG

In 2017, the WLCG started defining an AAI based on capabilities and JSON Web Tokens (JWTs):

- Capabilities: statements of **what you can do** instead of **who you are**.
    - Capabilities are ideal for distributed systems; the WLCG tokens still carry identity for wwhere it's needed (think: home directories).
- This culminated in 2019 in the WLCG Common Token Profile. This was a starter's pistol for finalizing software and workflows across the software stack.
    - WLCG tokens are used as bearer tokens; whatever entity has the token (the 'bearer') is authorized!

Claims in a WLCG token:


```
{
  "wlcg.ver": "1.0",
  "sub": "e608252d-e817-4071-b15c-1a72fe557b3f",
  "aud": "https://wlcg.cern.ch/jwt/v1/any",
  "nbf": 1683329405,
  "scope": "address phone openid profile offline_access storage.read:/
email wlcg wlcg.groups",
  "iss": "https://cms-auth.web.cern.ch/",
  "exp": 1683333005,
  "iat": 1683329405,
  "jti": "ba5125d2-46de-4bed-ada8-97bee554927c",
  "client_id": "5456c25d-b738-4db5-9ea2-4f946bfd2a8b",
  "wlcg.groups": [
    "/cms",
    "/cms/compute",
    "/cms/compute/scope",
    "/cms/country",
    "/cms/country/us",
    "/cms/integration",
    "/cms/uscms"
  ]
}
```

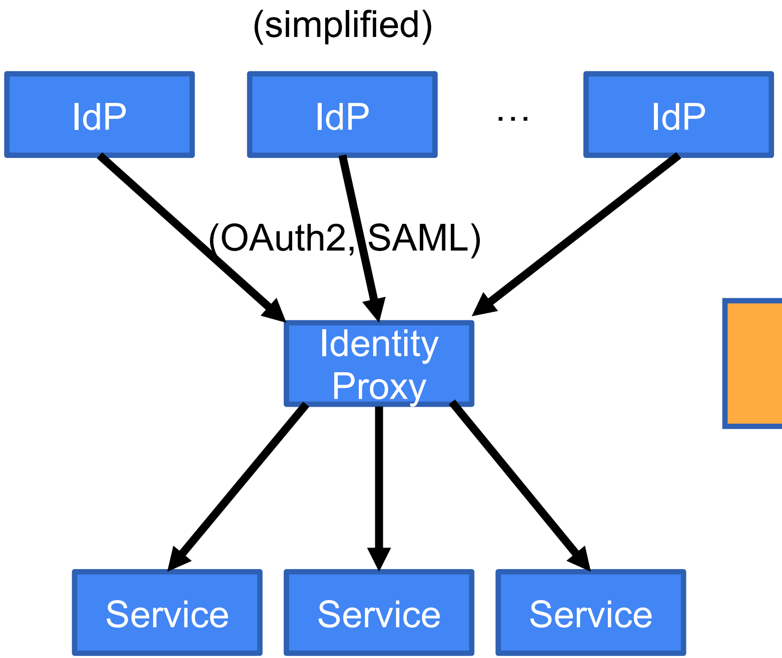

2022 milestone: all HTCondor-CE's switched to using WLCG tokens.

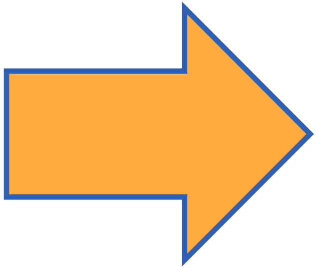
For a general progress report, see: WLCG transition from X.509 to tokens: Status and Plans (CHEP, Tuesday, Track 4).

We are making slow-but-steady progress toward migrating WLCG services to token-based auth

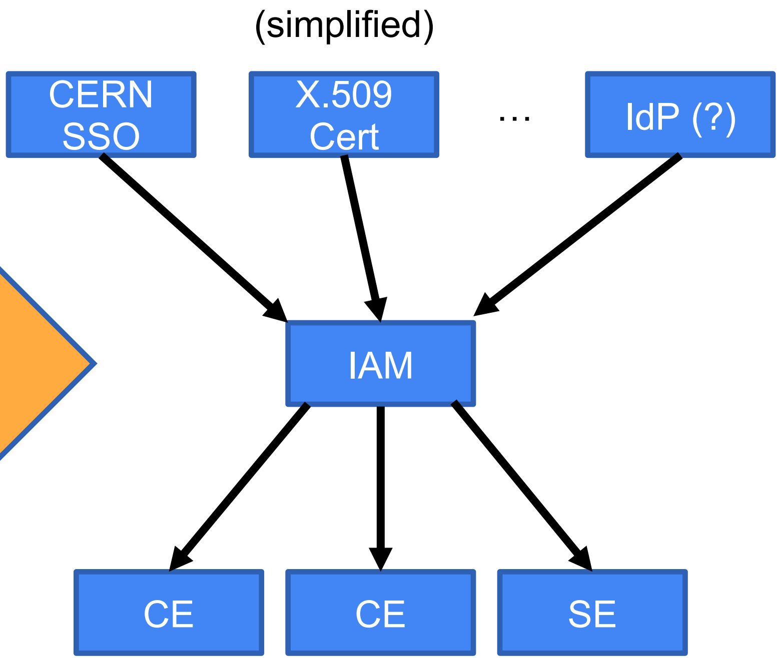
This spring, CMS kicked off a campaign to have all sites provide token-based authorization to their storage. 🎉

The AARC blueprint model





The WLCG model



Some Key Terminology

- **JWT**: JSON Web Token – a generic, standardized format for signed credentials.
- **WLCG Profile**: A common profile for the authorizations understood by the WLCG and a description of the token-based AAI.
- **Token Issuer**: The entity that created and signed the token.
- **OAuth2**: A framework describing how tokens can be moved around securely.

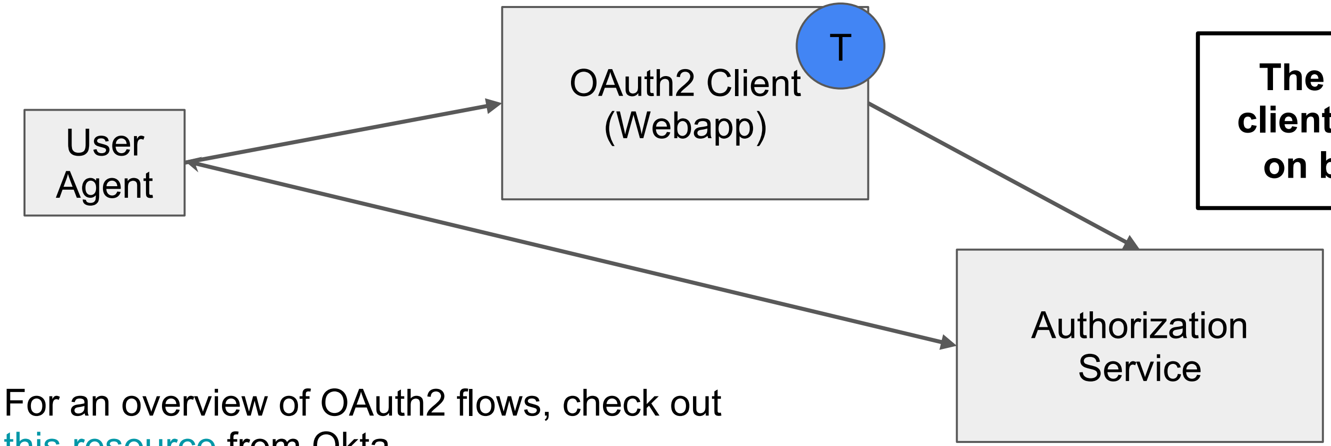
Encoded

eyJhbGciOiJSUzI1NiIsImtpZCI6ImtleS1
yczI1NiIsInR5cCI6IkpXVCJ9.eyJzY29wZ
SI6InJlYWQ6L3Byb3RlY3RlZCIsImF1ZCI6
Imh0dHBzOi8vZGVtby5zY2l0b2tlbnMub3J
nIiwidmVyIjoic2NpdG9rZW46Mi4wIiwiaX
NzIjoiaHR0cHM6Ly9kZW1vLnNjaXRva2Vuc
y5vcmciLCJleHAiOjE2ODI4OTQxMDIsImlh
dCI6MTY4Mjg5MzUwMiwibmJmIjoxNjgyODk
zNTAyLCJqdGkiOiI1NmJlZGZkOC01MmYxLT
Q0YTgtYTgzOS03NzlmN2RjYjkzNzUifQ.Zq
B14ELDoMrO5r9OrbZNu8N75YYGbsHZGJodO
4FdoSHdU0bFkr9LxXUdDDYMgZvVSodmyT95
XS2auwfFA6r1aENvtvkttOxqn6u80S46cvk
ZweiYr526CNmTpfwtTfjhUCf8CuxnrDokGc
v08W-

OAuth2 Authorization Code Grant Type

In a typical authorization code flow,

- The user (agent) contacts the OAuth2 **client** (the remote webapp),
- The user is redirected to the desired authorization service and authenticates,
- The authorization service returns an authorization code, redirects the user back to the client.
- The user gives the client the authorization code; the client can exchange this for a token.



The token goes to the client, which manages it on behalf of the user.

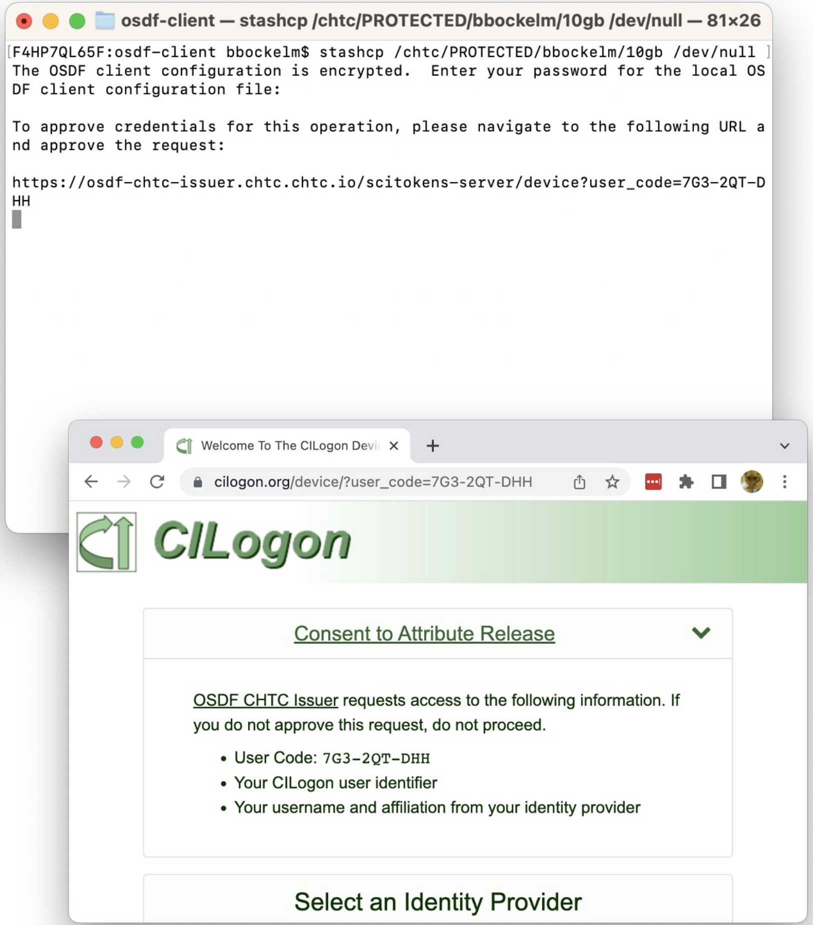
For an overview of OAuth2 flows, check out [this resource] from Okta.

OAuth2 Device Code Grant Type

The authorization code flow works wonderfully and is everywhere you look!

- … but only in a browser with a keyboard. What happens if the user agent is a TV? **A console**?

OAuth2 also defines a device code grant type, allowing the user to complete the flow from a browser separate from the user agent by copying a URL or 6-digit code from the original device.



Now, Let’s Go To the Facility!

Unique & Opportune

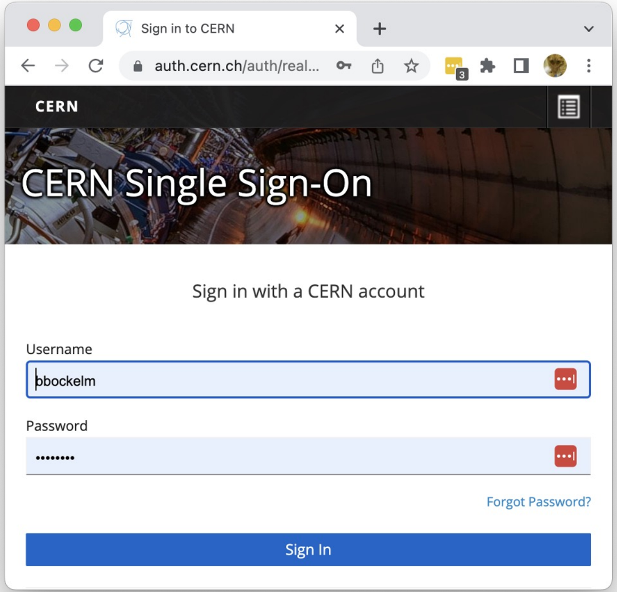
Analysis Facilities have unique requirements and present opportunities:

- **New Services, new decisions**: The rare opportunity to present a new user interface means we can make a fresh set of decisions.
  - The fact we’re ~50% through the transition to new credentials means we can attempt a break from tradition.
- Strong emphasis on **browser-based interfaces**: Use an infrastructure designed for the browser from Day 1.
- **Sustainability**: By utilizing industry-based tooling, we can
  - More quickly adopt other tools.
  - Have fewer software stacks we must maintain.

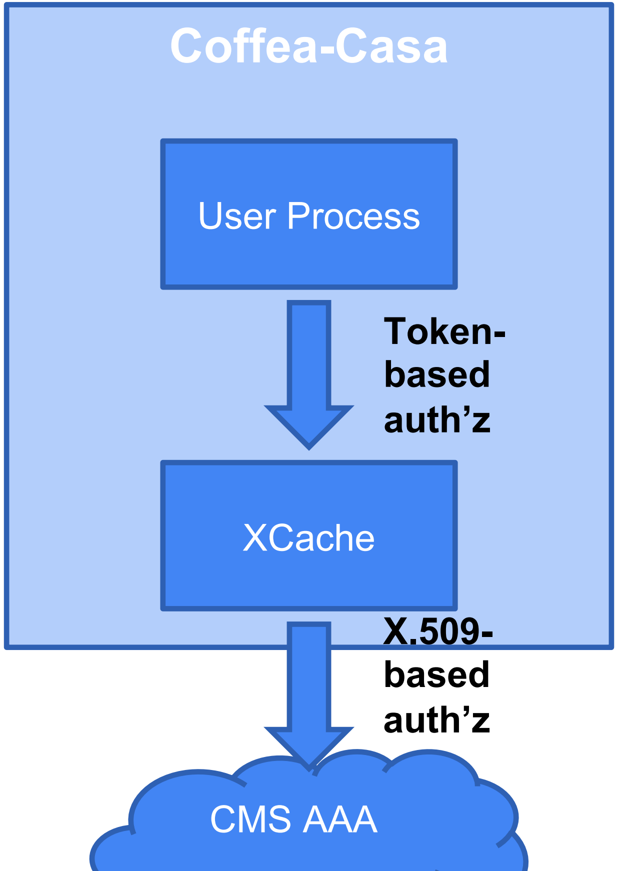
Ease of Use

Rare is the user who is unfamiliar with SSO.
A key concept in OAuth2 is the **client** manages the token, not the **user**. Have you ever seen your CERN SSO token?
Working against us:

- Not all services the user may access are token-enabled!
  - This is *slooooowly* coming into place though: CMS is starting a campaign to get all CMS storage token-enabled.
- **Terminals are a lousy client**: no built-in mechanism to auto-refresh tokens.
- **Tendency to over-complicate**: tokens can be incredibly fine-grained… but should they be?
  - Little need to have users juggle multiple tokens, esp. as the tooling improves.



Ease of Use: Credential Translation



Not all services support token-based auth'z!

To ease this transition, a facility can provide token-based credential translation. Example is XCache:

1. User's process contacts the local XRootD cache, authenticating with a token from CERN IAM.
2. XCache verifies the token, authorizes access to the cache.
3. On cache miss, XCache uses a service credential to download the files from a remote site.
    - If the remote site supports tokens, XCache can instead pass the user's token and skip X.509 authentication.

Ease of Use: Going back to the terminal

How do we make terminal-based usage easier? With device-code flow, we can get a token into the environment. <u>How do we update it?</u>

- No traditional way to do this in Unix… but systemd in RHEL9 provides 'user-level units' that can periodically fire.
- User can launch **oidc-agent** which runs a-la ssh-agent… kinda. Requires the user to still request a new token.
- Develop some WLCG-common "bearer token request" rules which tell tools that need a token how to create one. Security concerns abound, however.
- **My favorite approach now**: integrate with VO-level tools, have them know when and what token to request.
    - Example: htgettoken work done at FNAL.

Sidebar:
There's a similar challenge for moving tokens from the interactive environment into the task environment.

Sidebar:
Does anyone want to write a token acquisition and renewal service for SSH login hosts? All the tools should be there in RHEL9; "should be easy".

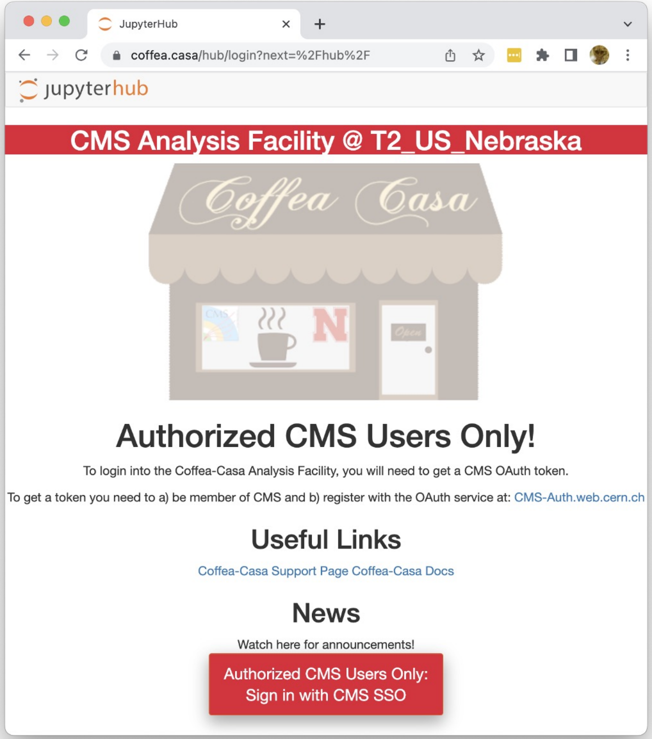
Example: Coffea-Casa

At the Coffea-Casa AF in Nebraska, we utilize OAuth2 with identities from the CMS IAM instance.

- Authentication is tied to CERN SSO.
- Membership is tied to experiment information in the CERN HR database.

Everyone in CMS is invited to use Coffea-Casa.

No local account needed – everything is based on the CERN-based authorization.

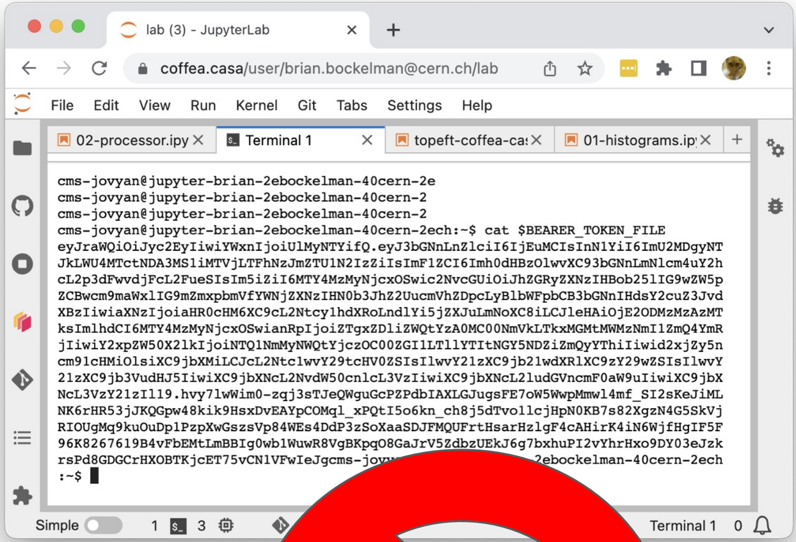


Example: Coffea-Casa

As a side-effect of the sign-in process, JupyterHub receives an access token and a refresh token.

- Periodically, JupyterHub will use the refresh token to get a new access token in the background.
- The token is persisted on disk and always available in the user environment.

Token-based authorization occurs as a side-effect of the login session.

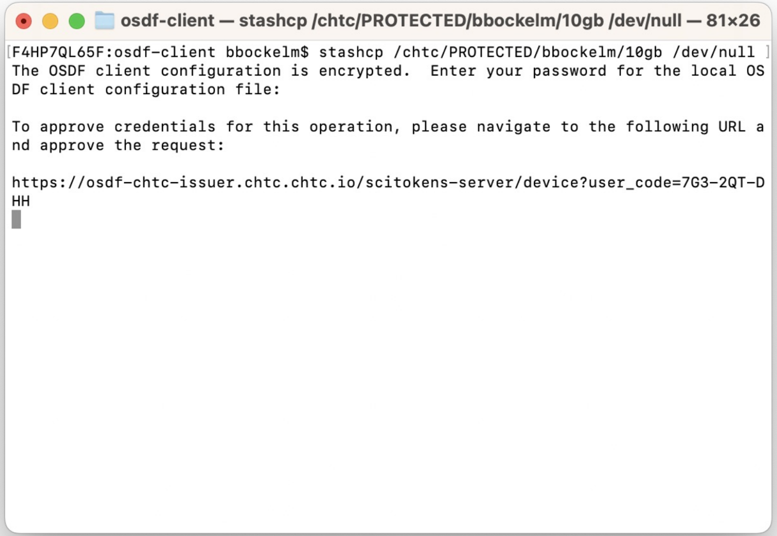


Users never need to know the token exists or manage the token.

Example: OSDF Client

The Open Science Data Federation (OSDF) allows users to effectively access objects in a unified namespace from a variety of origins.

- The clients are terminal-based or integrated with HTCondor.
- Authorization is done with tokens.
- The terminal-based client (stashcp):
  - Knows what object is to be accessed.
  - Determines the corresponding token issuer.
  - Dynamically registers an OAuth2 client (if one isn't present)
  - Requests an appropriate token.
  - Saves the refresh token into an encrypted config file. Password is saved into the session's kernel keyring – a logged-in user only needs to type their password once.



What's the point?
A little bit of higher-level knowledge built-in can make for a pleasant user experience, even in the terminal!

So, what’s the future of Analysis Facility AAI?

So, what's the future of Analysis Facility AAI?

There's no single definition of an analysis facility; there will be no single definition of an Analysis Facility AAI.

- No analysis facility in Run 1 used the same AAI as grid resources.
- No analysis facility in Run 2 used the same AAI as grid resources.
- No analysis facility in Run 3 used the same AAI as grid resources.
- My prediction:

No analysis facility in HL-LHC will use the same AAI as distributed resources.

Instead: What will be common approaches, important properties, and shared tools?

1. The AAI must be compatible with the distributed services.
   - This implies analysis facilities must be ready for token-based auth.
   - While we're in the awkward transition between X.509 and tokens, decisions must be made: **do both** or **invest in token translation services**?
2. The AAI should use federated ID where possible; the AARC blueprint is a great place to start.
   - Services like CILogon are your best friend.
3. Agents should manage credentials, not users.
4. The future – **tokens** – shines in the browser. Still many opportunities to make it shine in the terminal.
   - Challenge: This is one place where we may get ahead of industry!

Let's collaborate on common tools and learn from each other!

Thanks!

This material is based upon work supported by the National Science Foundation under Grant No. 1836650, 2030508, and 2114989. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.