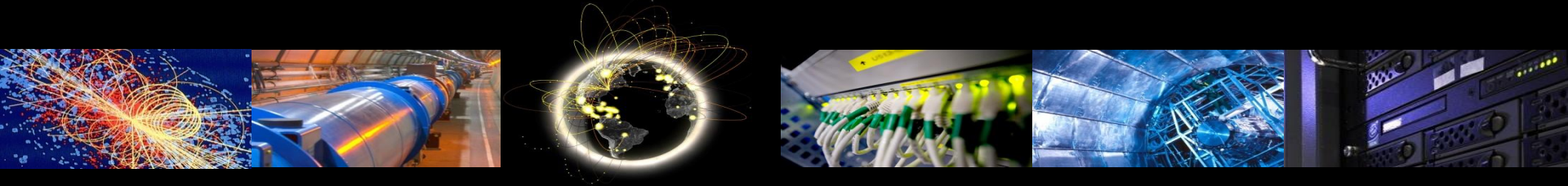


# Tokens in WLCG

TAG PMA Token-Based AuthN/Z Workshop

November 30th 2020



# Why Tokens?

- **Evolving Identity Landscape**
  - User-owned X.509 certificates -> federated identities (SAML & OpenID Connect)
- **Technology Readiness**
  - Increasing solutions for shielding users from the complexities of X.509 certificate management
  - Token-based authorization widely adopted in commercial services and increasingly by R&E Infrastructures
- **Data Protection**
  - Tightening of data protection (GDPR) requires fine-grained user level access control, certain provisioning practices may need to be adjusted

However, not all grid middleware supports token (OAuth2) based authorization (yet...!).

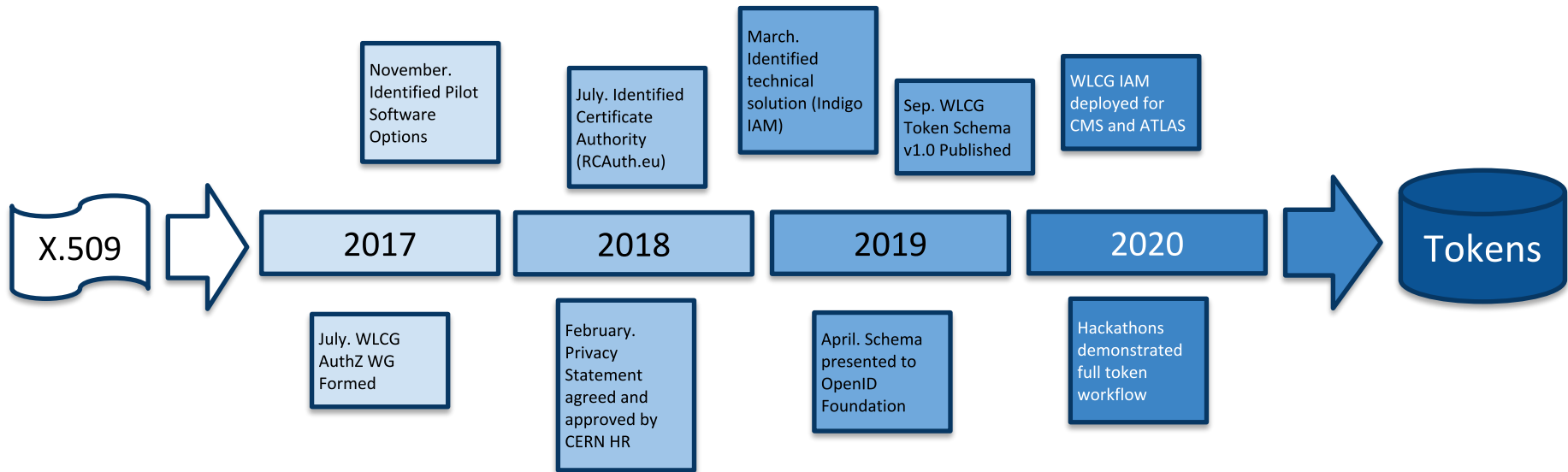
Objective: Understand & meet the requirements of a future-looking AuthZ service for WLCG experiments

# WLCG AuthZ WG

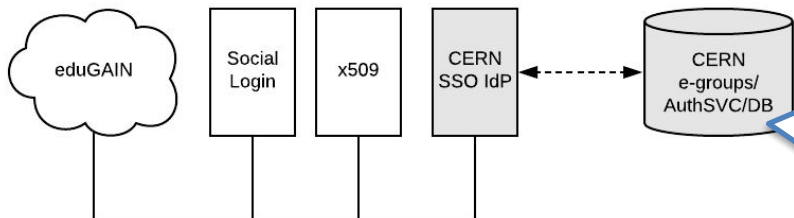
- Includes current major users of tokens in High Energy Physics
  - INDIGO IAM
  - EGI Check-in
  - SciTokens
  - dCache
  - ALICE
- Development work of pilot projects supported by:

The logos for AARC, EOSC-hub, and EOSCpilot are displayed. AARC is a stylized orange and blue logo. EOSC-hub is a blue circular logo with a white dot. EOSCpilot is a blue logo with a cluster of dots and the text "EOSCpilot" and "The European Open Science Cloud for Research Pilot Project".
- Priority to stick to industry and R&E standards wherever possible

# Towards Tokens



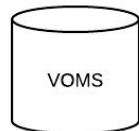
# AAI Design



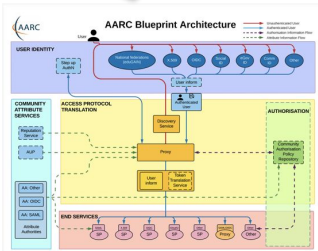
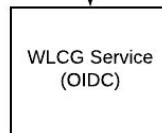
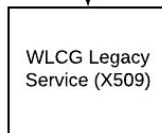
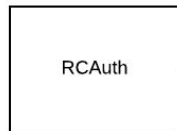
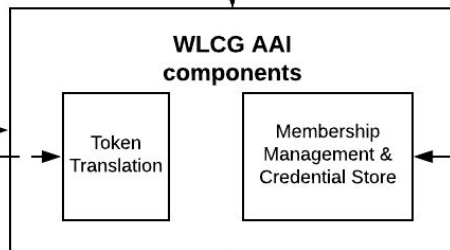
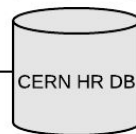
CERN SSO configured as sole Identity Provider, enables identity verification via HR DB (match CERN PersonID)

Follows the AARC Blueprint  
<https://aarc-community.org/architecture/>

VOMS Provisioning required for legacy services

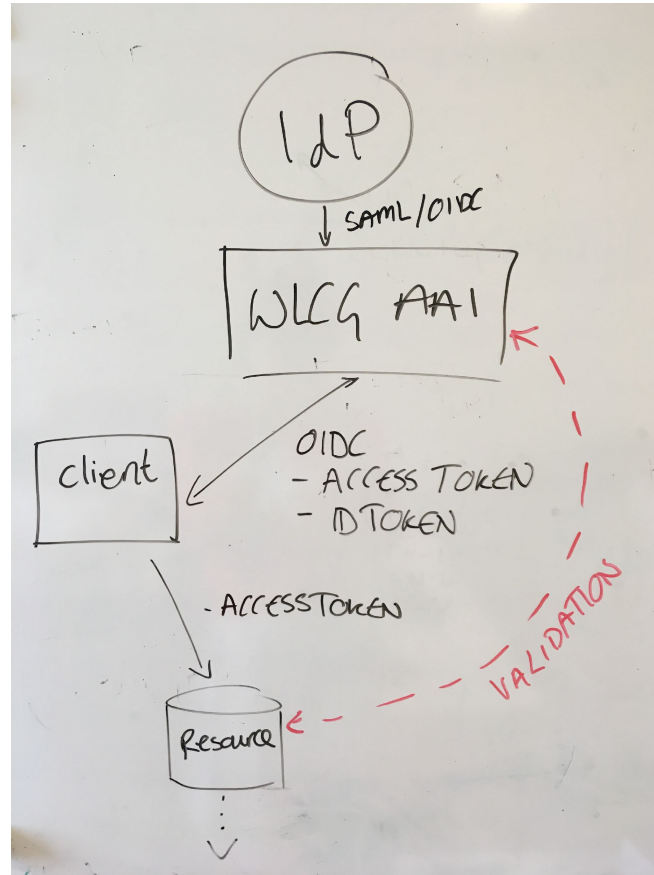


Integration with existing source of information (identity vetting)



WLCG AuthZ WG

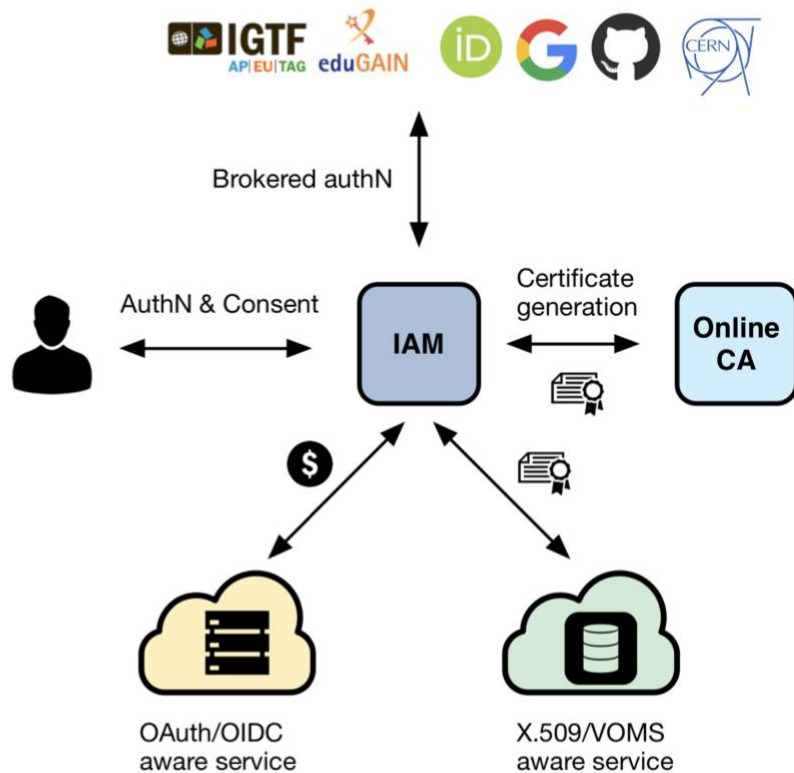
# Anticipated Token Flows



# INDIGO Identity and Access Management Service

A **VO-scoped** authentication and authorization service that

- supports **multiple authentication mechanisms**
- provides users with a **persistent, VO-scoped identifier**
- exposes **identity information, attributes** and **capabilities** to services via **JWT** tokens and standard **OAuth & OpenID Connect** protocols
- can integrate existing **VOMS**-aware services
- supports **Web** and **non-Web** access, **delegation** and **token renewal**



# Easy integration with relying services

**Standard OAuth/OpenID Connect** enables **easy integration** with off-the-shelf services and libraries.

IAM has been successfully integrated with

- Openstack, Atlassian JIRA & Confluence, Moodle, Rocketchat, Grafana, Kubernetes, JupyterHub, dCache, StoRM, XRootD (HTTP), FTS, RUCIO, HTCondor



# Deployments



**WLCG**  
Worldwide LHC Computing Grid

Welcome to **wlcg**

Sign in with your wlcg credentials

	<input type="text" value="Username"/>	
	<input type="password" value="Password"/>	

**Sign in**

[Forgot your password?](#)


Or sign in with

**CERN SSO**

Not a member?

**Apply for an account**

<https://wlcg.cloud.cnaif.infn.it>



**ATLAS**  
EXPERIMENT

Welcome to **atlas**

Sign in with


**Your X.509 certificate**

**CERN SSO**

Not a member?

**Apply for an account**

<https://atlas-auth.web.cern.ch>



**CMS**  
Compact Muon Solenoid

Welcome to **cms**

Sign in with

**CERN SSO**

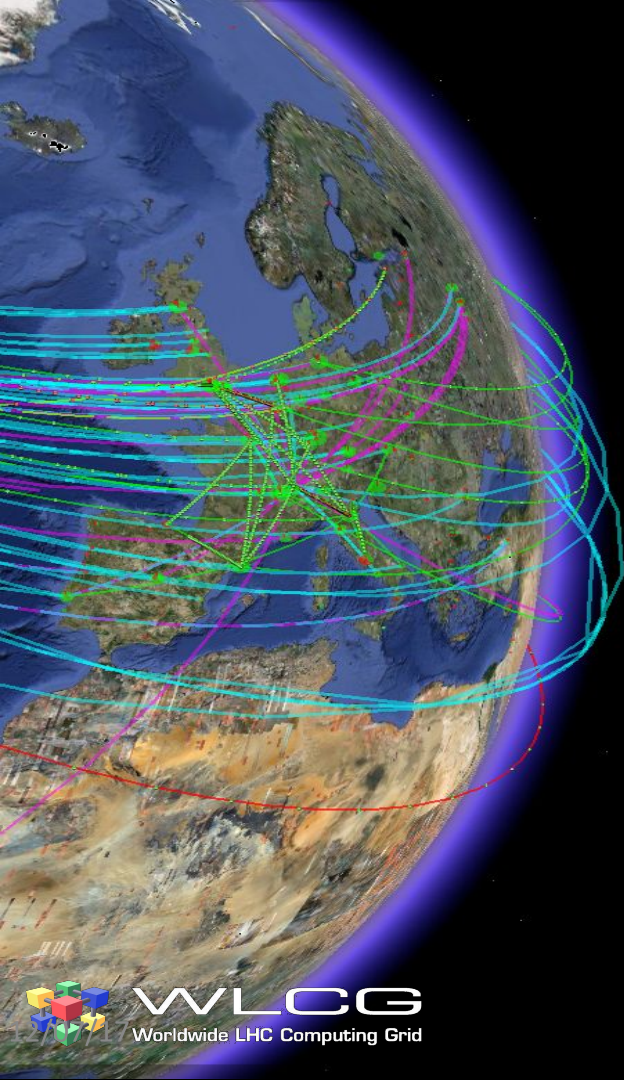
Not a member?

**Apply for an account**

<https://cms-auth.web.cern.ch>



# WLCG Token Schema



# Process

- Interviews with large VOs
- Surveyed existing schemas
- Compared with experience in Grid
- Gathered input from OpenID Foundation

*Priority to stick to standards wherever possible*

# Schema V1.0

- Published on Zenodo, September 25th 2019
- Allows middleware developers to enable token based authorization to an agreed schema
- Working document at <https://github.com/WLCG-AuthZ-WG/common-jwt-profile>

September 25, 2019

Technical note Open Access

## WLCG Common JWT Profiles

Altunay, Mine; Bockelman, Brian; Ceccanti, Andrea; Cornwall, Linda; Crawford, Matt; Crooks, David; Dack, Thomas; Dykstra, David; Groep, David; Igoumenos, Ioannis; Jouvin, Michel; Keeble, Oliver; Kelsy, David; Lassnig, Mario; Liampotis, Nicolas; Litmaath, Maarten; McNab, Andrew; Millar, Paul; Sallé, Mischa; Short, Hannah; Teheran, Jeny; Wartel, Romain

This document describes how WLCG users may use the available geographically distributed resources without X.509 credentials. In this model, clients are issued with bearer tokens; these tokens are subsequently used to interact with resources. The tokens may contain authorization groups and/or capabilities, according to the preference of the Virtual Organisation (VO), applications and relying parties.

Wherever possible, this document builds on existing standards when describing profiles to support current and anticipated WLCG usage. In particular, three major technologies are identified as providing the basis for this system: OAuth2 (RFC 6749 & RFC 6750), OpenID Connect and JSON Web Tokens (RFC 7519). Additionally, trust roots are established via OpenID Discovery or OAuth2 Authorization Server Metadata (RFC 8414). This document provides a profile for OAuth2 Access Tokens and OIDC ID Tokens.

Preview

Page: 1 of 35 Automatic Zoom

### WLCG Common JWT Profiles

Authored by the WLCG AuthZ Working Group

Version History:

Date	Version	Comment
------	---------	---------

[https://zenodo.org/record/3460258#.X8ED\\_i1Q1qs](https://zenodo.org/record/3460258#.X8ED_i1Q1qs)

Edit

New version

98

views

81

downloads

[See more details...](#)

Indexed in

OpenAIRE

Publication date:  
September 25, 2019

DOI:  
[DOI: 10.5281/zenodo.3460258](https://doi.org/10.5281/zenodo.3460258)

Keyword(s):  
[jwt](#), [OIDC](#), [OAuth2.0](#), [wlcg](#)

License (for files):  
[CC Creative Commons Attribution 4.0 International](#)

# Token Claims

## Common Claims

- sub
- exp
- iss
- acr
- aud
- iat
- nbf
- jti
- eduperson\_assurance (REFEDS)
- wlcg.ver (WLCG)
- wlcg.groups (WLCG)

**wlcg** prefix  
added to avoid  
collisions

## ID Tokens

- auth\_time
- general OIDC Claims

## Access Tokens

- scope (RFC8693)

Access tokens  
should include  
at least scope  
or group

*Note: Where unspecified, the origin is RFC7519 or OpenID Connect core*



# Group and Ver

**wlcg.groups:** group membership about an authenticated end-user. The claim value is an ordered JSON array of strings that contains the names of the groups a user is member of in the context of the VO that issued the Token. Group names **MUST** comply with the following grammar where group is defined recursively:

- `group ::= '/' groupname | group '/' groupname`
- `groupname ::= [a-zA-Z0-9][a-zA-Z0-9_.-]*`

*Examples: "/atlas/production", "/cms/itcms"*

AARC guideline for group syntax was deemed too verbose

**wlcg.ver:** the version of the WLCG token profile the relying party must understand to validate the token. ver names **MUST** comply with the following grammar:

- `vername ::= WLCG:[0-9]+\.[0-9]+`

*Example: "WLCG:1.0"*

# WLCG Capabilities (scope)

- Using standard scope claim, which may contain more than just capabilities
- Client must parse scope claim to extract capabilities

For a given storage resource, the defined authorizations include:

- **storage.read:** Read data. Only applies to “online” resources such as disk (as opposed to “nearline” such as tape where the **stage** authorization should be used in addition).
- **storage.create:** Upload data. This includes renaming files if the destination file does not already exist. This capability includes the creation of directories and subdirectories at the specified path, and the creation of any non-existent directories required to create the path itself (note the server implementation MUST NOT automatically create directories for a client). This authorization DOES NOT permit overwriting or deletion of stored data. The driving use case for a separate `storage.create` scope is to enable stage-out of data from jobs on a worker node.
- **storage.modify:** Change data. This includes renaming files, creating new files, and writing data. This permission includes overwriting or replacing stored data in addition to deleting or truncating data. This is a strict superset of `storage.create`.
- **storage.stage:** Read the data, potentially causing data to be staged from a nearline resource to an online resource. This is a superset of `storage.read`.

For a given computing resource, the defined authorization activities include:

- **compute.read:** “Read” or query information about job status and attributes.
- **compute.modify:** Modify or change the attributes of an existing job.
- **compute.create:** Create or submit a new job at the computing resource.
- **compute.cancel:** Delete a job from the computing resource, potentially terminating a running job.

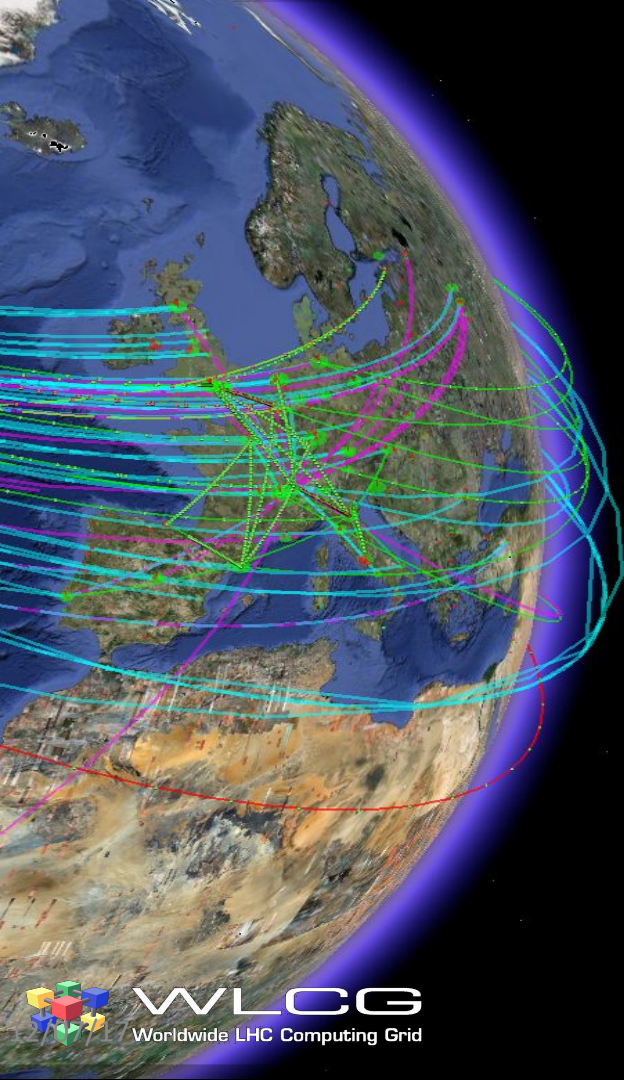
# Assurance

- We adopt the **eduperson\_assurance** multi-valued claim proposed by RAF to convey the assurance component values and profile.
- The **acr** claim is included in addition to the **eduperson\_assurance** claim to specifically convey the authentication assurance.

# Lifetimes

Token Type	Recommended Lifetime	Minimum Lifetime	Maximum Lifetime	Justification
Access Token & ID Token	20 minutes	5 minutes	6 hours	Access token lifetime should be short as there is no revocation mechanism. The granted lifetime has implications for the maximum allowable downtime of the Access Token server.
Refresh Token	10 days	1 day	30 days	Refresh token lifetimes should be kept bounded, but can be longer-lived as they are revocable. Meant to be long-lived enough to be on a “human timescale”.
Issuer Public Key Cache	6 hours	1 hour	1 day	The public key cache lifetime defines the minimum revocation time of the public key. The actual lifetime is the maximum allowable downtime of the public key server
Issuer Public Key	6 months	2 days	12 months	JWT has built-in mechanisms for key rotation; these do not need to live as long as CAs. This may evolve following operational experience, provision should be made for flexible lifetimes.

# Authorization



# Authorization

- Two models
  - Groups e.g. /atlas/production
  - Capabilities e.g. storage.read/atlas
- Clearer guidance is required for Dual Mode, discussions ongoing

**"Access tokens may convey authorization information as both groups and capabilities. If both group membership and capabilities are asserted, then the resource server should grant the union of all authorizations for the groups and capabilities that it understands."** From the WLCG Token Schema

# Groups (in **wlcg.groups** claim)

- Authorization may be based on the **wlcg.groups** claim.
- The value of the **wlcg.groups** claim is an ordered JSON array of case-sensitive strings reflecting the VO groups of which the token subject is a member.
- **wlcg.groups** semantics are equivalent to existing VOMS groups. VOMS roles should be considered as optional (i.e. returned only if requested) **wlcg.groups**
- Requesting the **wlcg.groups** scope returns all default groups

# Capabilities (in scope claim)

- Authorization may be based on the scope claim.
- Format **\$AUTHZ:\$PATH** where **\$PATH** is mandatory (may be '/' for \*)
- Resources allow a **hierarchical** relationship to be expressed; an authorization for write:/baz implies a write authorization for the resources at /baz/qux (this is similar for read) authorizations.
- This authorization scheme **is not equivalent to POSIX semantics**. For example, if a token is issued with authorization read:/home, then an implementation may override normal POSIX access control and give the bearer access to all users' home directories.

# Scope Based Attribute Selection

- We propose to use scopes to implement an attribute selection mechanism for **both groups and capabilities** following the approach outlined in the OpenID Connect standard:
  - [https://openid.net/specs/openid-connect-core-1\\_0.html#ScopeClaims](https://openid.net/specs/openid-connect-core-1_0.html#ScopeClaims)
- Authorizations are requested using scopes and returned by the OIDC provider if the client and user are entitled

# Scope Based Attribute Selection

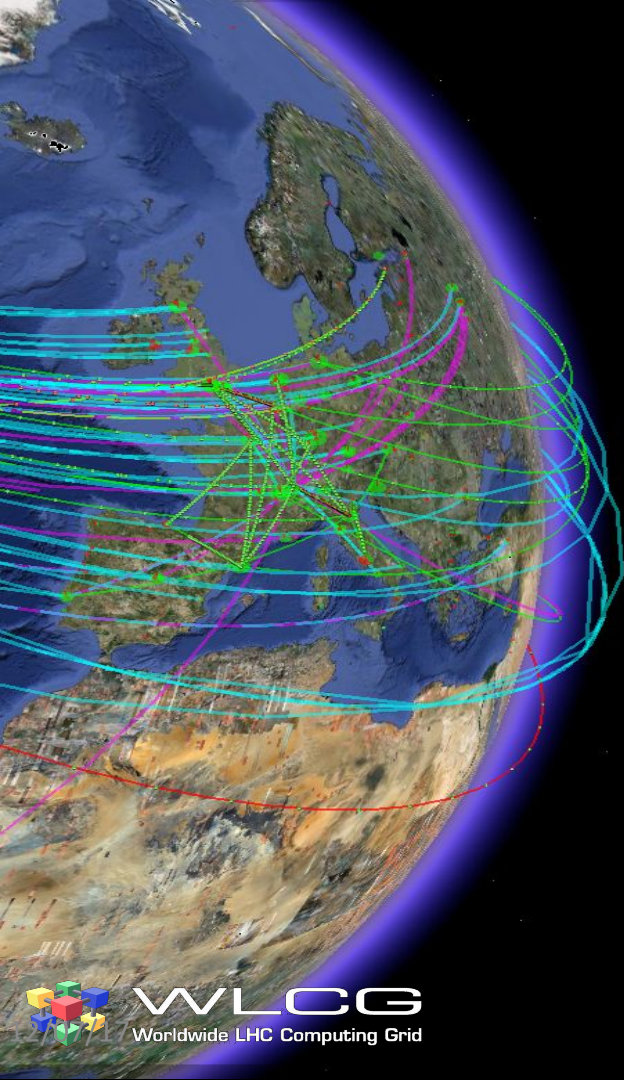
Capability requests are matched exactly

Scope Request	Claim Result
<code>scope=storage.read:/home/joe</code>	<code>"scope": "storage.read:/home/joe"</code>
<code>scope=storage.read:/home/joe storage.read:/home/bob</code>	<code>"scope": "storage.read:/home/joe storage.read:/home/bob"</code>
<code>scope=storage.create:/ storage.read:/home/bob</code>	<code>"scope": "storage.create:/ storage.read:/home/bob"</code>

/cms is a default group and always returned

Scope Request	Claim Result
<code>scope=wlcg.groups</code>	<code>"wlcg.groups": ["/cms"]</code>
<code>scope=wlcg.groups:/cms/uscms wlcg.groups:/cms/ALARM</code>	<code>"wlcg.groups": ["/cms/uscms", "/cms/ALARM", "/cms"]</code>
<code>scope=wlcg.groups:/cms/uscms wlcg.groups:/cms/ALARM wlcg.groups</code>	<code>"wlcg.groups": ["/cms/uscms", "/cms/ALARM", "/cms"]</code>
<code>scope=wlcg.groups wlcg.groups:/cms/uscms wlcg.groups:/cms/ALARM</code>	<code>"wlcg.groups": ["/cms", "/cms/uscms", "/cms/ALARM"]</code>
<code>scope=wlcg.groups:/cms wlcg.groups:/cms/uscms wlcg.groups:/cms/ALARM</code>	<code>"wlcg.groups": ["/cms", "/cms/uscms", "/cms/ALARM"]</code>

# Command Line Workflows



# Use Case

- Bulk of physicists' work is in the terminal
- Expectation to be able to submit jobs from the command line, authorised by Tokens
- Command line clients must be easy to use
  - *No client secret management by users*
  - *Minimal tool installation*
  - *Reasonable to expect occasional trips to the browser (ideally no more than weekly)*

# Client Evaluation

- Looked first at oidc-agent, but it didn't fit well for our use case
  - Refresh tokens stored in a separate daemon for each user
  - Passphrase needed when restarting daemon
  - No good way to auto-refresh access tokens sent with jobs
- Plan to store refresh tokens in Hashicorp Vault secrets manager
  - Vault plugins already existed to do most of what we needed
  - Wrote a relatively simple htgettoken script to take users through the flows of retrieving new access tokens via Vault
  - Bearer access tokens are stored in a local file
  - Job manager (HTCondor) can get new access tokens from Vault
  - **More details in Fermilab session tomorrow**

<https://github.com/WLCG-AuthZ-WG/client-tools/blob/master/client-tools.md>

# Bearer Token Discovery

- Once acquired, where should a token be stored locally?

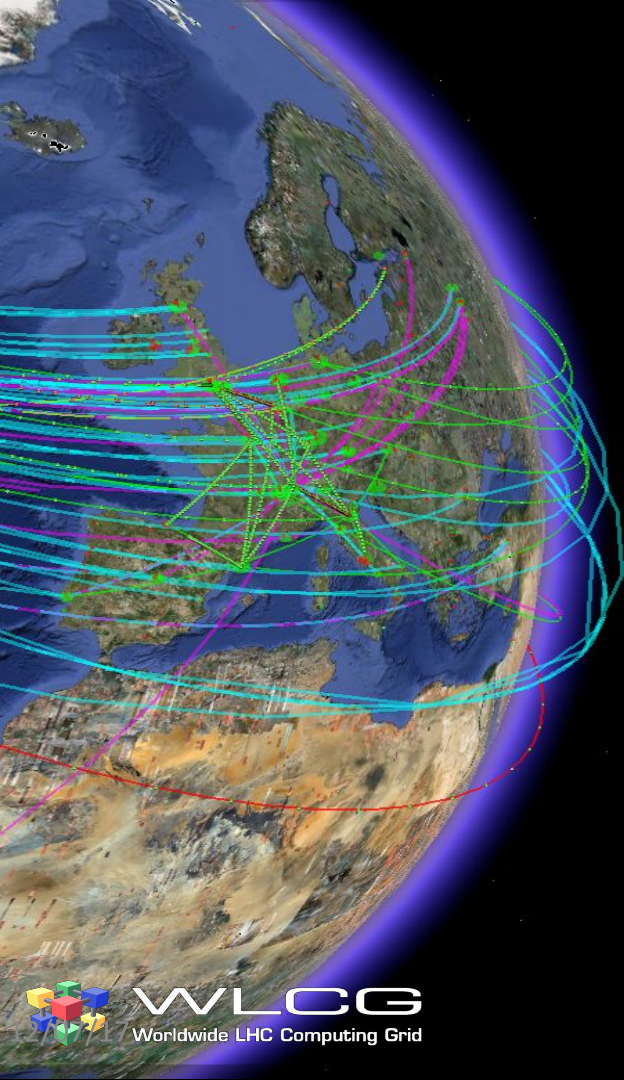
If a tool needs to authenticate with a token and does not have out-of-band WLCG Bearer Token Discovery knowledge on which token to use, the following steps to discover a token MUST be taken in sequence (where `$ID` below is taken as the process's effective user ID):

1. If the `BEARER_TOKEN` environment variable is set, then the value is taken to be the token contents.
2. If the `BEARER_TOKEN_FILE` environment variable is set, then its value is interpreted as a filename. The contents of the specified file are taken to be the token contents.
3. If the `XDGRUNTIME_DIR` environment variable is set\*, then take the token from the contents of `$XDGRUNTIME_DIR/bt_u$ID **`.
4. Otherwise, take the token from `/tmp/bt_u$ID`.

<https://github.com/WLCG-AuthZ-WG/bearer-token-discovery/blob/master/specification.md>

# Next Steps

- Additional work is needed to handle requests for
  - Audience
  - Scope
- Unclear where logic of which capabilities should be requested for a particular client should live (not known in IAM, not known in Vault)



# Upgrading Middleware

# Hackathons

- Drive adoption of token-based authN/Z in existing Grid middleware
- Two hackathons held, Jan and September 2020
  - <https://indico.cern.ch/event/870616/>
  - <https://indico.cern.ch/event/953075/>
- Significant progress made
  - demonstrated X.509-free third-party data transfers managed by FTS

# Token-based AuthN/Z for data management

All WLCG data management components now support\* token-based authN/Z based on the WLCG JWT profile

- RUCIO, FTS, XRootD, StoRM, dCache, DPM, EOS

Smoke test suite checks that HTTP Third-party copy work as expected and sends a daily report on the DOMA-TPC WG list

- <https://github.com/paulmillar/http-tpc-utils>
- recently extended to [support X509-free authN/Z](#)

\*with different levels of compliance with the profile

# JWT compliance test suite

- A JWT profile compliance test-suite to assess conformance with the WLCG profile
  - Check that issuer checks, signature checks, temporal validity, audience restrictions, path constraints, ... are honoured by the implementations
  - <https://github.com/indigo-iam/wlcg-jwt-compliance-tests>
- Run daily against selected endpoints to produce a compliance report
- Verifiable path towards interoperable token-based authN/Z support
  - currently focused on the data management use case

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
audience	28	14	14	00:00:17	<div><div></div></div>
basic-authz-checks	98	57	41	00:02:17	<div><div></div></div>
cern-eos	18	0	18	00:00:00	<div><div></div></div>
cnaf-amnesiac	18	18	0	00:00:15	<div><div></div></div>
infn-t1-xfer	18	18	0	00:00:12	<div><div></div></div>
manchester-dpm	18	6	12	00:00:26	<div><div></div></div>
nebraska-xrootd	18	10	8	00:00:45	<div><div></div></div>
prague-dpm	18	7	11	00:00:30	<div><div></div></div>
prometheus	18	12	6	00:00:25	<div><div></div></div>

# Next steps

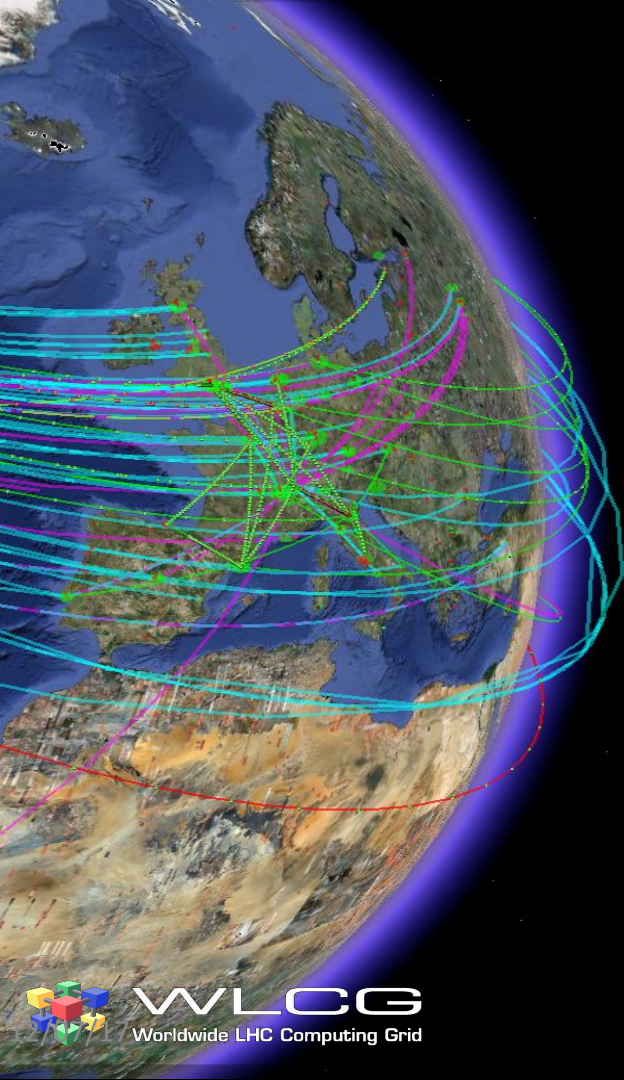
Improve WLCG JWT profile compliance in SEs, RUCIO, FTS

Improve support for group-based authorization

- required to support access to home directories and other use cases where identity is relevant for authorization

Move focus from data management to integration with compute

- to get the full chain working without X.509/VOMS

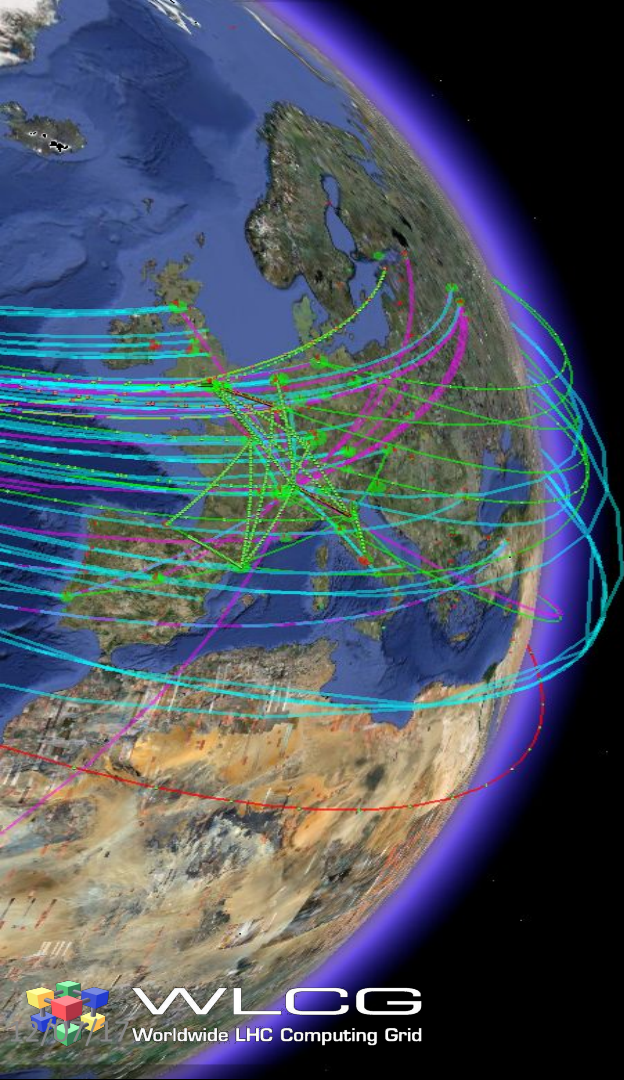


# Phasing out VOMS and X.509

# Overview

- X.509 will continue to play a vital role in our infrastructure as host certificates, but users will no longer need to manage the certificate lifecycle themselves
- It is likely that power users, e.g. admins, may need to manage certificates for much longer
- This is a very early plan and significant changes are expected

	2019	2020	2021	2022	2023
Integrate RCAuth.eu for on-demand IOTA X.509					
Migrate VOs to IAM, retire VOMS Admin					
Add Token support to Middleware					
Dual mode (IAM issues X.509/VOMS and Tokens)					
Privilege Tokens, analyse remaining X.509 use					
Begin removing X.509 User Certificate Support					



# Use Case: Tokens at OSG

# OSG and Tokens

OSG is aiming to retire the Grid Community Toolkit (fork of the Globus Toolkit) by January 2022.

- Adoption of tokens plays a major role in this plan.
- Bearer tokens provide the ability to describe what the bearer is authorized to do, not who the bearer is.
  - Avoids the issue of identity mapping.
  - We believe this is the right way to handle authorization in a distributed infrastructure.
  - Identity is the wrong thing for authorization: but still necessary for traceability.

We are adapting our services -- primarily, the HTCondor-CE and XRootD (using the HTTP/1.1 protocol) -- to be able to accept SciTokens and WLCG tokens.

- The same library ([scitokens-cpp](#)) provides compatibility with both profiles.

# Tokens - used since 2018

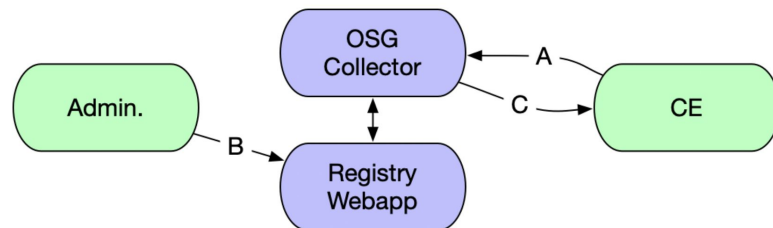
We have been using tokens internally since ~2018.

- HTCondor will generate a token for each job run by the “OSG VO”.
  - This token permits the user to read and write from their home directory.
  - The token is sent along with the payload job, permitting it to make HTTP GET/PUT calls to storage run by the OSG VO.
- Done automatically for users - OAuth2 not used to acquire the token.

# Beyond WLCG Tokens

Before a CE is allowed to get pilot jobs from the OSG, the administrator must be authorized and register the CE with the OSG.

- However, the identity used to register the CE (usually, something in the InCommon federation ID) isn't the same identity used to authenticate the CE (a 'grid certificate').
- In fact, when we submit jobs to a service, all we get is an organizational validation (OV) for IGTF-based credentials.



**Figure 4. The CE credential acquisition workflow.** First, the CE requests an IDTOKEN from the OSG collector over a secure, anonymous connection (A). The CE administrator then authenticates with the registry web application and approves the corresponding request (B). Finally, the collector sends the IDTOKEN back to the CE (C).

We have developed a webapp which issues tokens to a CE based upon the approval of an individual's CILogon identity.

The token acquisition process mimics the OAuth2 “device flow” - intended for use cases where the device (here, the CE) does not have a functioning web browser.

- Not actually OAuth2 though - the HTCondor-CE uses the CEDAR protocol, not HTTP.

# CE Approval Flow

Once the CE has a token, it can create an authenticated session with the OSG.

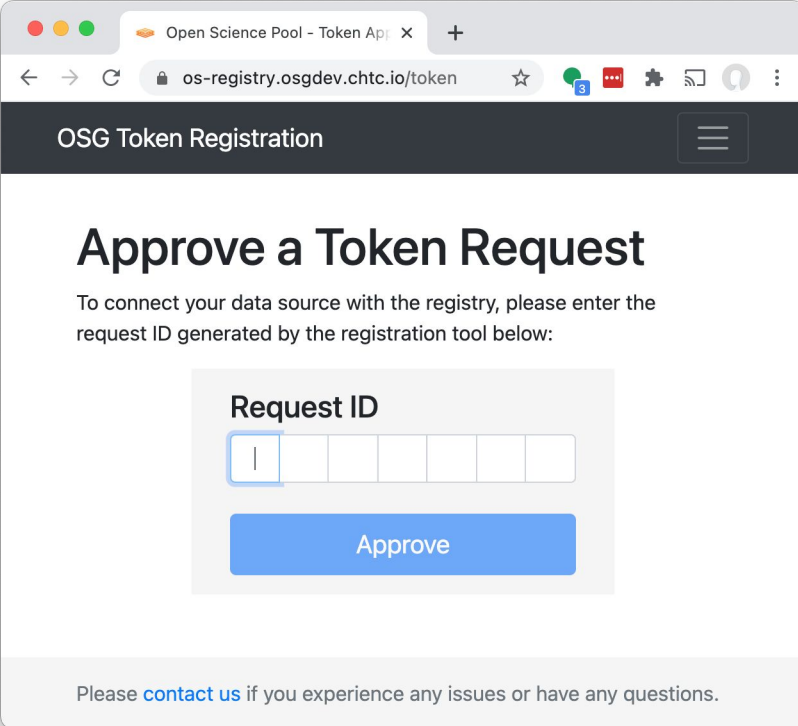
Any client wanting to submit to the CE can exchange a SciToken (WLCG token) for a token issued by the CE.

- The token is exchanged through the session established with the OSG.

The client can use this token to authenticate with the CE.

- It can use HTCondor's connection-reversing protocol to avoid the server needing a public IP address.
- Token is signed by the CE's symmetric key, allowing mutual authentication.

The result? Clients can submit pilot jobs to the CE without the CE needing a host certificate or a public IP address.



The screenshot shows a web browser window with the address bar displaying "os-registry.osgdev.chtc.io/token". The page title is "OSG Token Registration". The main heading is "Approve a Token Request". Below the heading, a message states: "To connect your data source with the registry, please enter the request ID generated by the registration tool below:". There is a form with a label "Request ID" and a text input field containing a single character "I". Below the input field is a blue button labeled "Approve". At the bottom of the page, a footer message says: "Please [contact us](#) if you experience any issues or have any questions."

# OSG and Tokens

OSG is using JWTs (both the WLCG and SciToken profiles) as bearer tokens to authenticate & authorize clients.

- These tokens are issued by the VO and we eventually plan to replace the use of X.509 proxies for client authentication.
- The tokens allow us to use capability-based authorization and avoid identity mapping.

Beyond traditional use of tokens to authenticate clients and X.509 to authenticate servers, we are experimenting with trust models that allow us to have tokens used by the service as well.



# Questions?