OIDC/OAuth2 based security framework

A.Tsaregorodtsev, A.Lytovchenko

CPPM-IN2P3-CNRS, France,

23d March 2023 ISGC'22, Taipei

DIRAC THE INTERWARE

Outline

- DIRAC Project
- Moving towards IODC/OAuth2 based security framework
  - Getting user credentials
  - Secure client/server protocol
  - Token Management
  - User Management
  - Pilot Factory
- Status and plans
- Conclusions

Disclaimer

The presented current status of the DIRAC OIDC/OAuth2 security framework is a work in progress in a rapidly evolving environment. Important changes are likely in the near future.

Installations and communities

Shared by multiple experiments/projects, both inside HEP, astronomy, and life science

<u>Experiment agnostic</u>

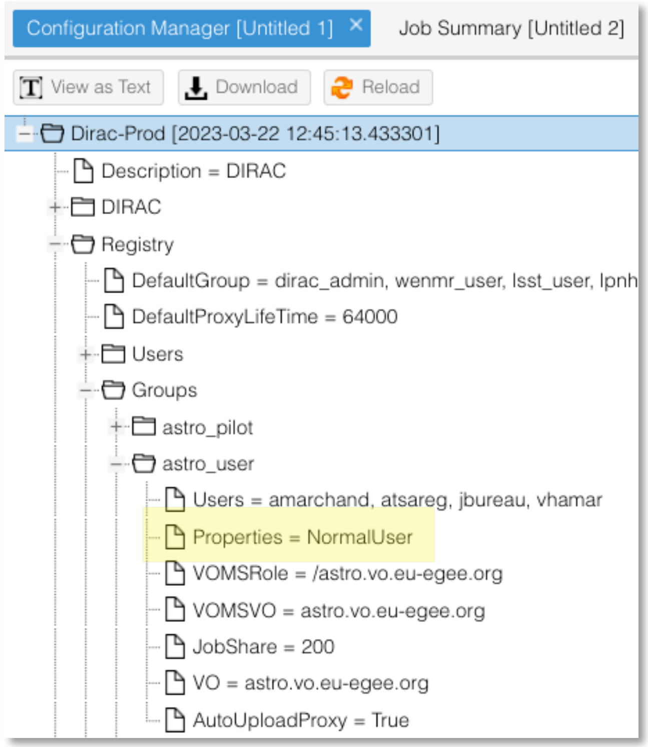
<u>Extensible</u>

<u>Flexible</u>

DIRAC THE INTERWARE

Current AAI system

- The DIRAC secure client/service protocol is based on X.509 certificates
  - **Users**: proxy certificates to access DIRAC services
  - **Services**: host certificates for inter-service communications
- Users are registered in the DIRAC Registry as members of some of the DIRAC groups
  - Automatic synchronization with VOMS
  - Users are identified by DN's of their certificates
- Group properties determine user rights with respect to the DIRAC services
  - Groups are embedded into the user proxies
- Long user certificate proxies are stored in the ProxyManager for asynchronous operations on the user's behalf



```
Dirac-Prod [2023-03-22 12:45:13.433301]
  Description = DIRAC
  DIRAC
  Registry
    DefaultGroup = dirac_admin, wenmr_user, lsst_user, lpnh
    DefaultProxyLifeTime = 64000
    Users
    Groups
      astro_pilot
      astro_user
        Users = amarchand, atsareg, jbureau, vhamar
        Properties = NormalUser
        VOMSRole = /astro.vo.eu-egee.org
        VOMSVO = astro.vo.eu-egee.org
        JobShare = 200
        VO = astro.vo.eu-egee.org
        AutoUploadProxy = True
```

New AAI technologies

- The new generation AAI frameworks are based on the OIDC/OAuth2 industry standards.
  - Multiple Identity Providers (IdP) offer services for user authentication and community management
- The DIRAC security framework is being updated to follow the new standards

DIRAC
THE INTERWARE

Transition requirements

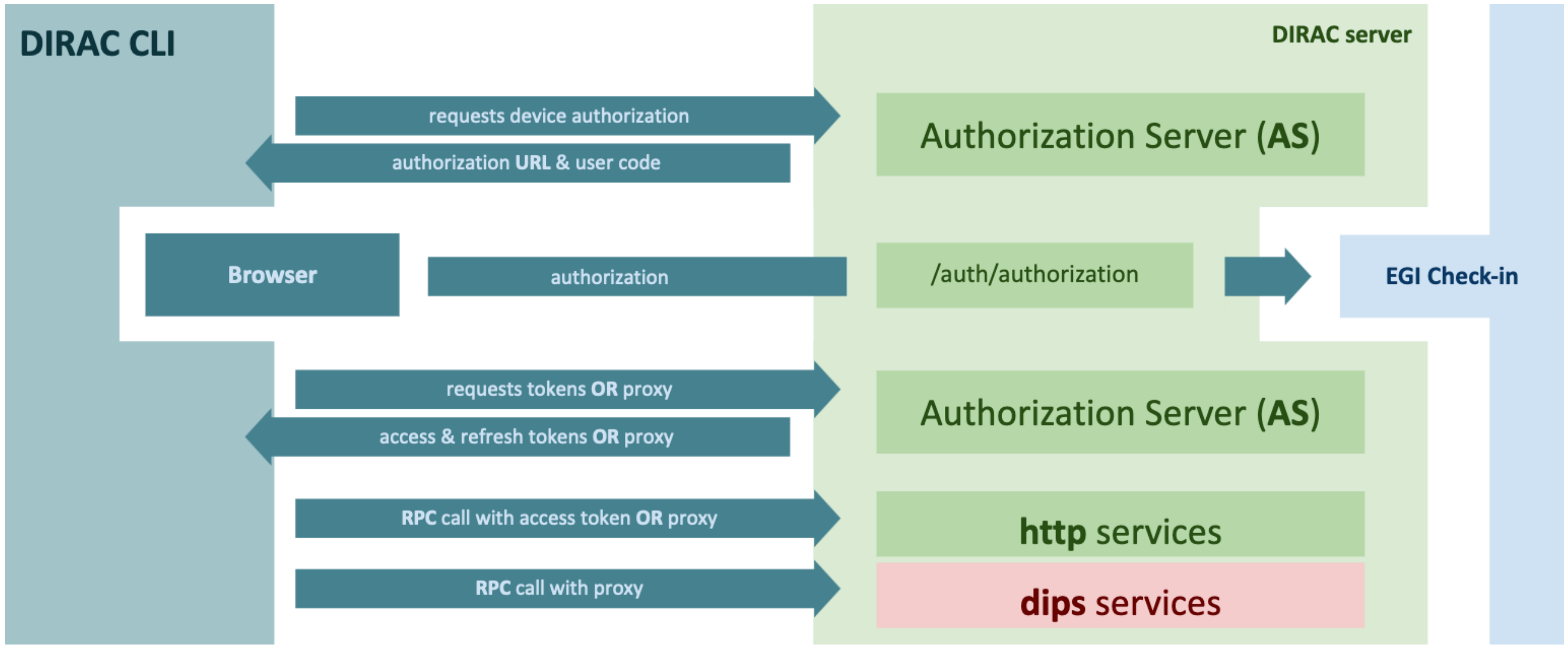
- Introduction of the new AAI framework should ensure continuity of multiple DIRAC services in production
  - In the transition period it should be compatible with both X.509 certificates and OAuth2 tokens to ensure smooth user's migration
  - Multi-VO DIRAC services should be compatible with different IdPs with different user profiles
  - DIRAC users should not see dramatic changes in their day-to-day work

DIRAC THE INTERWARE

AAI components

- The DIRAC client/service protocol based on tokens
  - Users accessing DIRAC services
  - Communications between DIRAC services and agents
- Token Management to provide valid tokens for asynchronous operations
- User Management based on information from IdPs
  - Reduce to minimum user management in DIRAC
- Connectors to external resources/services using tokens based AAI
  - Computing and storage services

User login protocol


DIRAC CLI
DIRAC server
requests device authorization
authorization URL & user code
Authorization Server (AS)
Browser
authorization
/auth/authorization
EGI Check-in
requests tokens OR proxy
access & refresh tokens OR proxy
Authorization Server (AS)
RPC call with access token OR proxy
http services
RPC call with proxy
dips services

- device flow (CLI) or authorization_code flow (Web) to the DIRAC Authorization Server redirected via authorization_code flow to a chosen Identity Provider

DIRAC Authorization Server

- DIRAC AS implements a standard Authorization Server interface but serves as an intermediate step towards backend Identity Provider (IdP)

  - It is a single entry point for the DIRAC users
    - No complicated local configuration required
  - Allows to interface multiple IdP's
    - Configured centrally allowing user to make a choice
  - Stores long refresh tokens in the TokenDB
    - For asynchronous user operations
    - No need to upload user tokens
  - Assists users and choosing the right DIRAC group
    - Group that user belongs to and which is managed by a chosen IdP
  - Returns a token with proper contents (claims) according to a chosen DIRAC group
  - Can return proxy certificate
    - if available in the ProxyManager service

User authentication:CLI

Example user login (Jupyter Notebook terminal)

1. Initialize the DIRAC client and start the authorization flow

```
jovyan@jupyter-916ace3e:~$ source /cvmfs/dirac.egi.eu/dirac/bashrc_egi
(base) jovyan@jupyter-916ace3e:~$ dirac-login --issuer=https://dirac.egi.eu/auth --token
Use the following link to continue
https://dirac.egi.eu/auth/device?user_code=RPMR-WTWW
Authorization pending.. (use CNTL + C to stop)
```

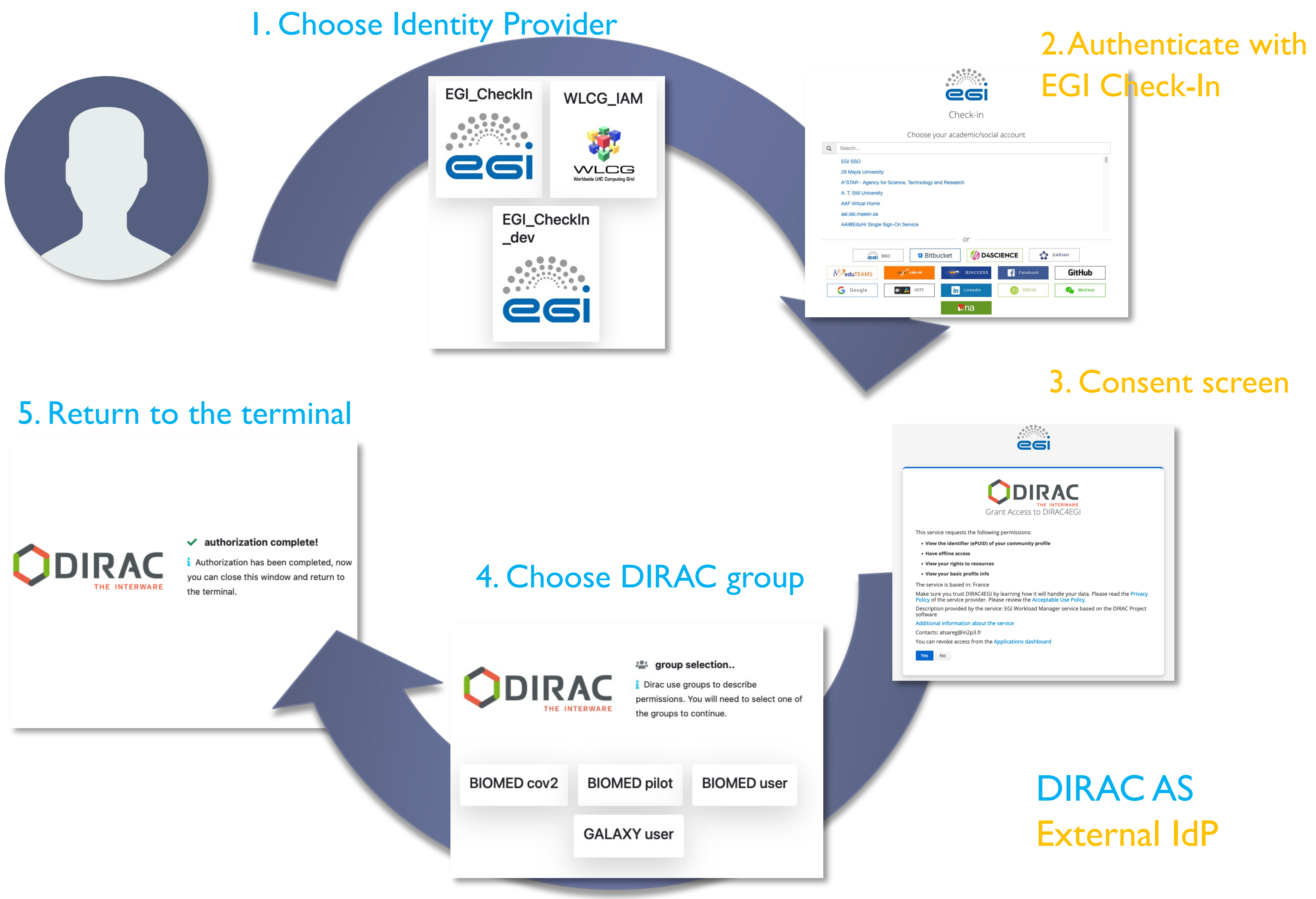
2. Authenticate via Check-In, choose DIRAC group (see next slide)

3. Store the received token

```
New token is saved to /tmp/bt_u1000.
subject       : 5cb92dfc-5dbe-403a-b7c8-ccdbeecd1a99
issuer        : https://wlcg.cloud.cnaf.infn.it/
timeleft      : 12:57:14
username      : atsareg
DIRAC group   : biomed_user
properties    : NormalUser
```

DIRAC THE INTERWARE

User authentication flow


1. Choose Identity Provider

EGI_CheckIn

WLCG_IAM

EGI_CheckIn_dev

2. Authenticate with EGI Check-In

Check-in

Choose your academic/social account

Search...

EGI SSO
29 Mayis University
A*STAR - Agency for Science, Technology and Research
A. T. Still University
AAF Virtual Home
aai.lab.maeen.sa
AAI@EduHr Single Sign-On Service

or

3. Consent screen

Grant Access to DIRAC4EGI

This service requests the following permissions:

- **View the identifier (ePUID) of your community profile**
- **Have offline access**
- **View your rights to resources**
- **View your basic profile info**

The service is based in: France

Make sure you trust DIRAC4EGI by learning how it will handle your data. Please read the Privacy Policy of the service provider. Please review the Acceptable Use Policy.

Description provided by the service: EGI Workload Manager service based on the DIRAC Project software

Additional information about the service

Contacts: atsareg@in2p3.fr

You can revoke access from the Applications dashboard

Yes No

4. Choose DIRAC group

group selection..

Dirac use groups to describe permissions. You will need to select one of the groups to continue.

5. Return to the terminal

✓ **authorization complete!**

Authorization has been completed, now you can close this window and return to the terminal.

DIRAC AS External IdP

User authentication:CLI

Example user login with getting a proxy certificate

1. Initialize authorization flow to get proxy (no user certificate installed locally)

```
(base) jovyan@jupyter-916ace3e:~$ dirac-login --issuer=https://dirac.egi.eu/auth --proxy
Use the following link to continue
https://dirac.egi.eu/auth/device?user_code=QLST-JJQT
Authorization pending.. (use CNTL + C to stop)
```

2. Authenticate via Check-In, choose DIRAC group

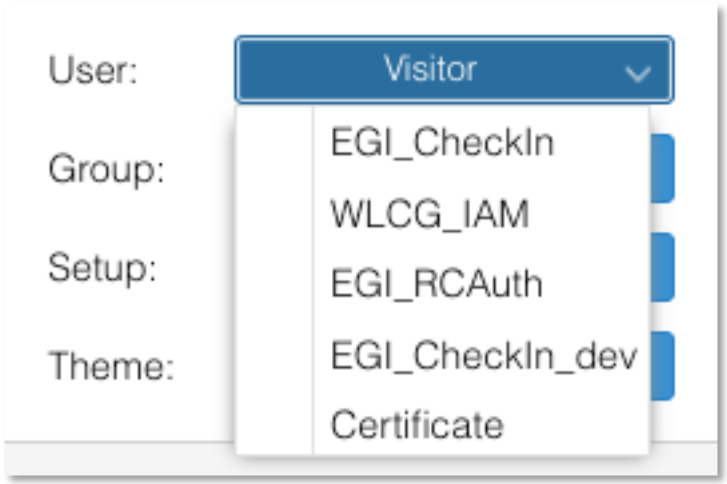
3. Store the received proxy

```
Proxy is saved to /tmp/x509up_u1000.
subject      : /O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev/CN=8471106824/…
issuer       : /O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev/CN=8471106824
identity     : /O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev
timeleft     : 06:29:58
DIRAC group  : biomed_user
path         : /tmp/x509up_u1000
username     : atsareg
properties   : Pilot, NormalUser, GenericPi
```

DIRAC
THE INTERWARE

User authentication: Web

- In the DIRAC Web Portal users are given a choice to login with a token or certificate
  - In the case of token, user follows the same flow as in the CLI case except for the IdP selection
  - The result is stored in the browser as secure cookie

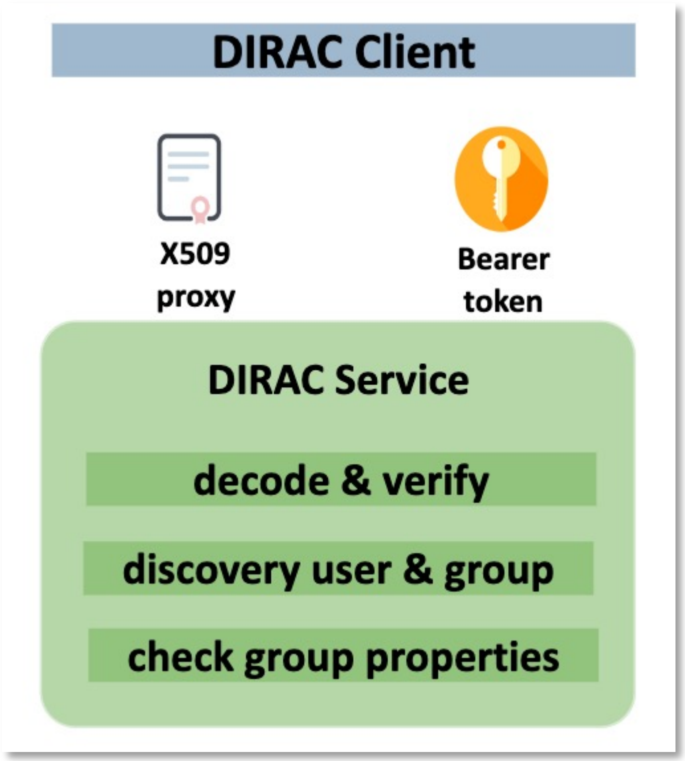


Client/service protocol

- DIRAC is migrating services to an HTTPS based protocol
  - Retiring custom DIPS protocol soon

- The HTTPS services support both X.509 and tokens
  - DIPS will not support tokens

- Access with both X.509 proxies and tokens is granted based on user identity and group membership
  - Proxies have DIRAC group embedded in an extension
  - Tokens must have enough information to map unambiguously onto DIRAC groups
  - After the DIRAC group is determined the same AuthManager tool is used for both protocols



Token to DIRAC group mapping

- Tokens are mapped onto DIRAC groups based on their claims.
  - Claims depend on user profiles used by IdP

- Example EGI Check-In:

  eduperson_entitlement:

  urn:mace:egi.eu:group:registry:biomed:role=member#aai.egi.eu

  -> biomed_user group
- Example WLCG IAM:

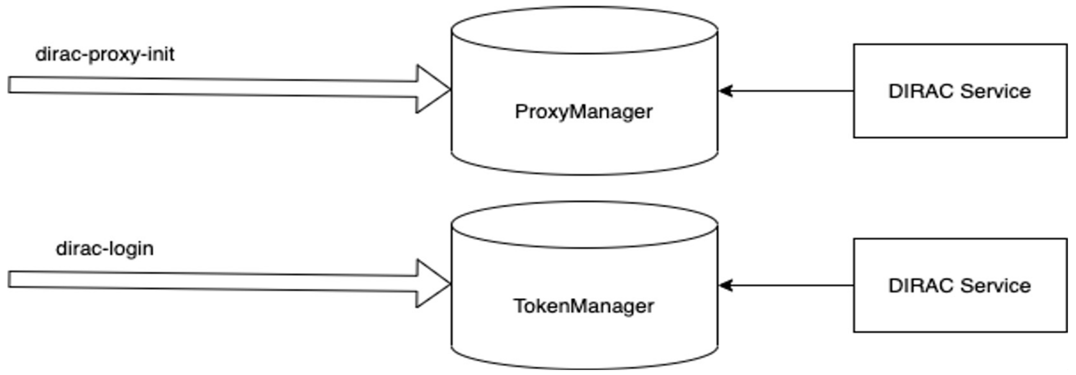
  wlcg.groups = /dirac/pilot -> dirac_pilot group

- The required claims must be present in the tokens
  - To avoid an expensive introspection of incoming tokens

- This is the work in progress
  - E.g. Check-In recent addition of parameterized scopes allowing to add **eduperson_entitlement** claims to a token

Token Management

- DIRAC AS is storing long Refresh Tokens of users in the TokenManager database
  - After successful user authentication
  - Similar to the ProxyManager database for X.509 proxies



- Refresh Tokens are used to request Access Tokens from corresponding IdPs with scopes corresponding to user's group requirement

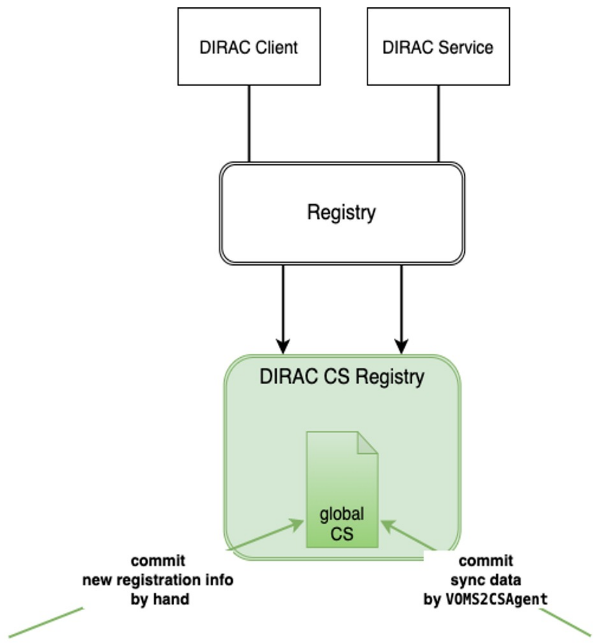
Token Management

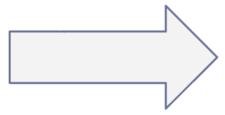
- TokenManager provides tokens to users while the login flow with appropriate claims and scopes
  - Both access and refresh tokens are provided
  - Users will be able to refresh their access tokens via a refresh flow involving TokenManager
    - This is a work in progress

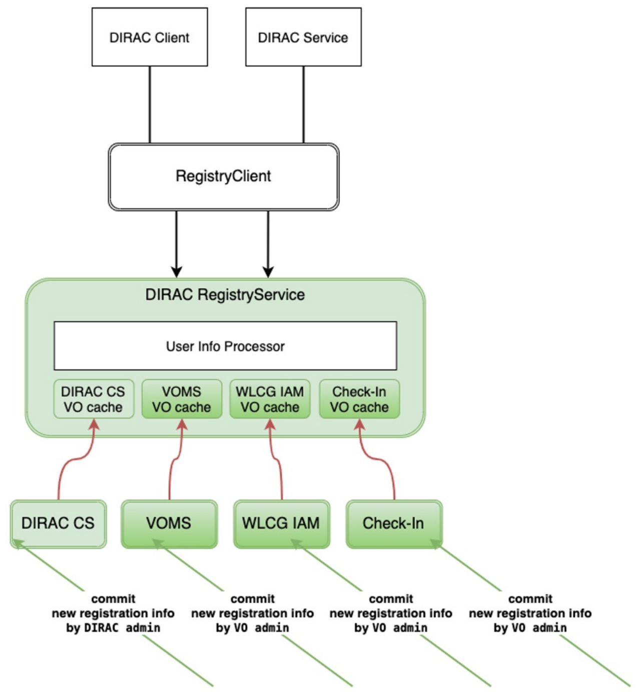
- TokenManager provides tokens to other DIRAC components to perform operations on the user's behalf
  - E.g. pilot user token for the DIRAC SiteDirector (aka Pilot Factory) submitting pilot jobs

User Management

- Currently, all users are described in the DIRAC Registry statically
  - Added by hand or by periodic syncing with VOMS
  - Admins are notified of non-registered users connection attempts with tokens
- Moving towards complete user management by IdP services
  - One-to-one correspondence of DIRAC groups and IdP claims/scopes
- Static Registry will be replaced by a dynamic Registry Service


DIRAC Client
DIRAC Service
Registry
DIRAC CS Registry
global CS
commit new registration info by hand
commit sync data by VOMS2CSAgent





Dynamic User Registry

- **Registry service** receives information about VO users from **IdP's** or VOMS services. There is no need to store this information in the DIRAC configuration

- User management is completely outside DIRAC, this is done by VO administrators using IdP web interfaces designed for this function

- DIRAC trusts and relies entirely on the information received from the relevant **IdP**.

- This is the work in progress

Pilot framework

- Pilot Factories are using tokens
  - Either user tokens via a refresh flow provided by the TokenManager
  - Or client tokens via client_credential flow
- The preferred method will be determined in real operations
- Client-credentials
  - A la robot certificates or service credentials
  - OK for single-VO installations
  - For multi-VO installations (e.g. EGI) different scenarios are possible
    - Issuer per VO – Check-In is reluctant to do that
    - Client (subject) per VO – more operations overhead for the WMS
    - Making VO groups available in tokens and accessible by the CE services via plugins
- Access to Computing Elements is demonstrated to work with both types of tokens
  - WLCG IAM tokens
  - HTCondorCE using SCITOKENS protocol
  - ARC CE using the REST interface
  - Using Check-In tokens for submitting pilots is the work in progress
    - Recently added **compute** scopes to the Check-In profile

DIRAC THE INTERWARE

Pilot framework

- Cloud Sites are accessed with the Openstack Application Credentials
  - created after authentication with tokens
  - Operational for EGI Federated Cloud sites with the Check-In tokens
- The Pilot framework will continue to use proxies for the pilot/service communications at the first stage
  - Using tokens or other secure protocols will be considered

DIRAC THE INTERWARE

Status and plans

- Token based authentication is being enabled for the EGI users
  - VOs: dteam, biomed
  - First users using tokens in the dirac.egi.eu Web Portal, EGI Jupyter Notebooks
- Several sites (HTCondorCE) are running pilots submitted with tokens in production
  - WLCG IAM tokens
- Next steps in the EGI DIRAC service:
  - Setting up pilot VOs in EGI Check-In to be compatible with their DIRAC description
    - biomed
  - Enabling HTTPS services will allow users to access the DIRAC WMS with tokens (job submission, monitoring, retrieving results)

Status and plans (cont'd)

- Development plans – long way ahead. Several points to mention:
  - Dynamic Registry Service
  - Refreshing user tokens in CLI
  - Integration with the RCAuth service
  - Connection to storage services
    - Enabling ACLs stored in the DIRAC File Catalog for accessing physical storages
  - Follow the evolution of IdProvider services

DIRAC
THE INTERWARE

Conclusions

- DIRAC is undergoing a serious upgrade including a new client/service protocol based on HTTPS and the new security framework based on the OIDC/OAuth2 protocols.

- The new security framework was developed that allowed integration of Identity Provider services (Check-In and WLCG IAM) for user management, provisioning of user tokens necessary to communicate with the DIRAC services and performing asynchronous user operations.

- Accessing Computing Element services (HTCondorCE, ARC) by the DIRAC Pilot Factory using tokens was demonstrated

- The development of the security framework continues to include access to new third party services (storage, RCAuth), delegation of user management to Identity Providers as well as the overall optimization of its performance.

DIRAC
THE INTERWARE

Acknowledgements

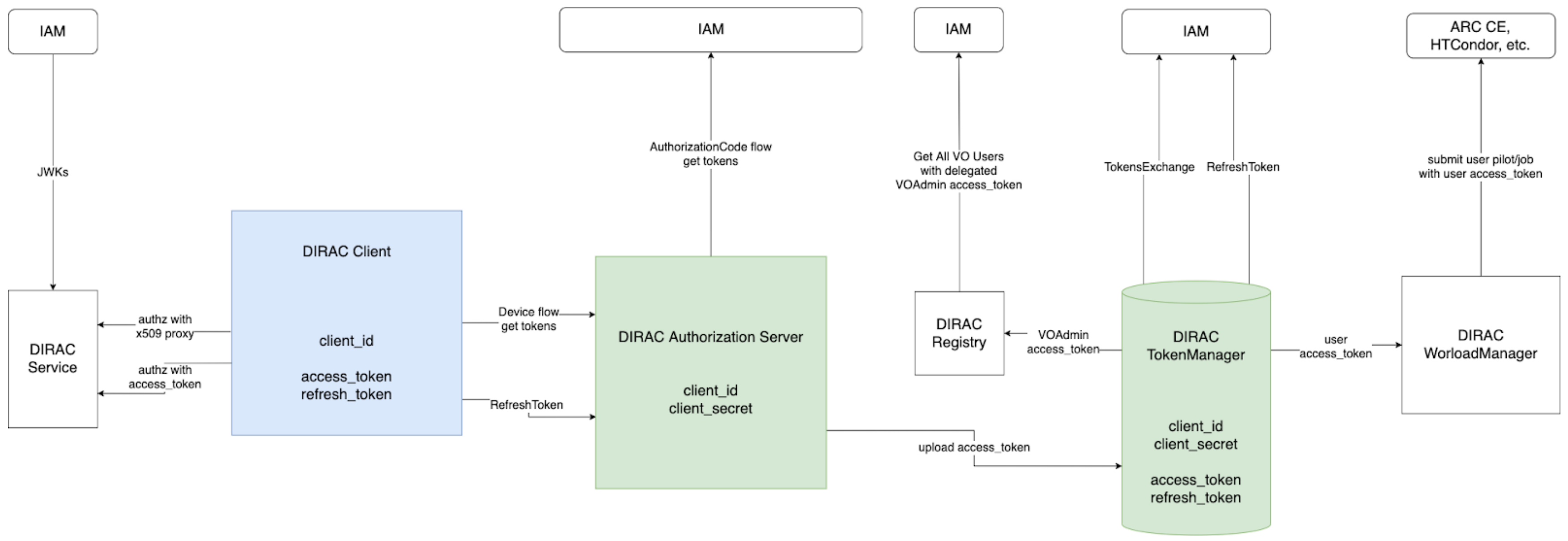
- This work is co-funded by the EOSC-hub project (Horizon 2020) under Grant number 777536
- EGI-ACE receives funding from the European Union's Horizon 2020 research and Innovation programme under grant agreement no. 101017567

http://diracgrid.org

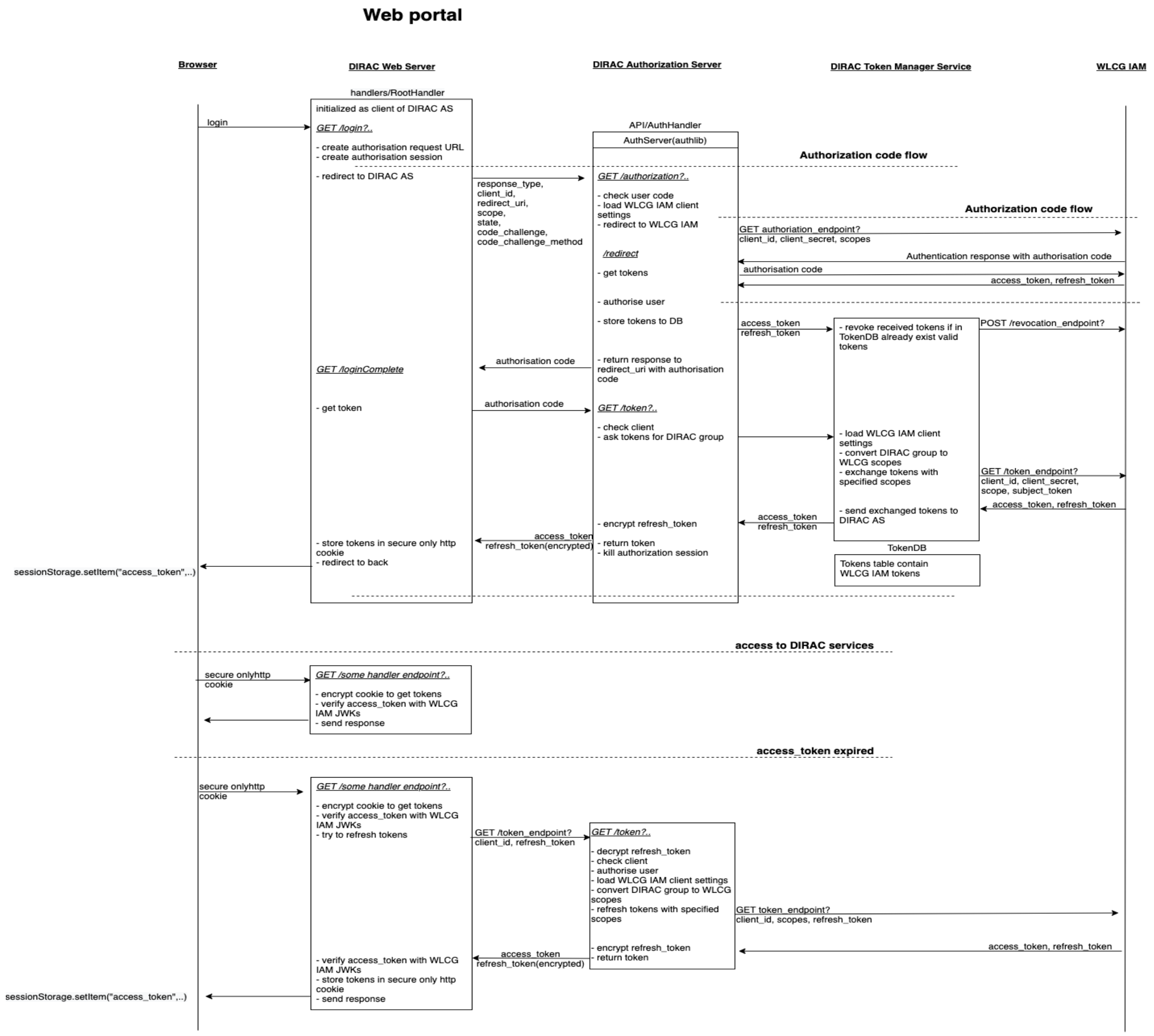
DIRAC
THE INTERWARE

Backup slides

Tokens generation, storage and usage to access DIRAC and third party services



Web portal authentication sequence diagram



Web portal
Browser
DIRAC Web Server
DIRAC Authorization Server
DIRAC Token Manager Service
WLCG IAM
handlers/RootHandler
initialized as client of DIRAC AS
login
GET /login?..
- create authorisation request URL
- create authorisation session
- redirect to DIRAC AS
API/AuthHandler
AuthServer(authlib)
Authorization code flow
response_type, client_id, redirect_uri, scope, state, code_challenge, code_challenge_method
GET /authorization?..
- check user code
- load WLCG IAM client settings
- redirect to WLCG IAM
Authorization code flow
GET authoriation_endpoint? client_id, client_secret, scopes
Authentication response with authorisation code
/redirect
- get tokens
authorisation code
access_token, refresh_token
- authorise user
- store tokens to DB
access_token refresh_token
- revoke received tokens if in TokenDB already exist valid tokens
POST /revocation_endpoint?
authorisation code
- return response to redirect_uri with authorisation code
GET /loginComplete
- get token
authorisation code
GET /token?..
- check client
- ask tokens for DIRAC group
- load WLCG IAM client settings
- convert DIRAC group to WLCG scopes
- exchange tokens with specified scopes
GET /token_endpoint? client_id, client_secret, scope, subject_token
access_token, refresh_token
- send exchanged tokens to DIRAC AS
access_token refresh_token
- encrypt refresh_token
access_token refresh_token(encrypted)
- return token
- kill authorization session
- store tokens in secure only http cookie
- redirect to back
sessionStorage.setItem("access_token",..)
TokenDB
Tokens table contain WLCG IAM tokens
access to DIRAC services
secure onlyhttp cookie
GET /some handler endpoint?..
- encrypt cookie to get tokens
- verify access_token with WLCG IAM JWKs
- send response
access_token expired
secure onlyhttp cookie
GET /some handler endpoint?..
- encrypt cookie to get tokens
- verify access_token with WLCG IAM JWKs
- try to refresh tokens
GET /token_endpoint? client_id, refresh_token
GET /token?..
- decrypt refresh_token
- check client
- authorise user
- load WLCG IAM client settings
- convert DIRAC group to WLCG scopes
- refresh tokens with specified scopes
GET token_endpoint? client_id, scopes, refresh_token
access_token, refresh_token
- encrypt refresh_token
- return token
access_token refresh_token(encrypted)
- verify access_token with WLCG IAM JWKs
- store tokens in secure only http cookie
- send response
sessionStorage.setItem("access_token",..)