

On Authentication, Authorization and Single Sign On

Andrii Lytovchenko, CPPM

-
- **Introduction**
 - Prerequisites
 - Basic concepts
 - Main goal
 - **User management**
 - Registry service
 - DIRAC group
 - **Log in**
 - **Access to DIRAC services**
 - **Delegation**
 - **Pilot framework with tokens**
 - **Status and plans**
 - **Conclusion**



DIRAC is using only **X509 certificates** for user authentication, but using X509 certificates is complicated for the end-users:

- Complex issuing procedure, yearly renewal, installation in multiple places with a format conversion, loading in browsers, etc
- Users of many communities do not have access to Certification Authorities issuing X509 certificates

Single sign-on (**SSO**) is an authentication process that allows a user to access multiple applications with one set of login credentials.

The EGI Check-in service enables access to EGI services and resources using federated authentication mechanisms



► **NOTE:** still need the certificate uploaded to the ProxyManager, because it is still used in the DISET protocol

Identity Provider (IdP) is a service that creates, maintains and manages user identity information and provides it together with the user authentication.



Proxy Provider is a service that creates, maintains, and manages X509 certificate proxies and provides them together with the user authentication.



OAuth 2.0 is the industry-standard *delegation* protocol for conveying *authorization decisions* across a network of web-enabled applications and APIs, full specification - [RFC6749](#). More information [here](#).

Open ID Connect is an interoperable *authentication protocol* based on the OAuth 2.0 family of specifications. More information [here](#).

OAuth2 roles:

Resource owner is the **user** who is giving access to some portion of their account.

Resource server is the server that contains the user's information that is being accessed by the third-party application, capable of accepting and responding to protected resource requests using access tokens.

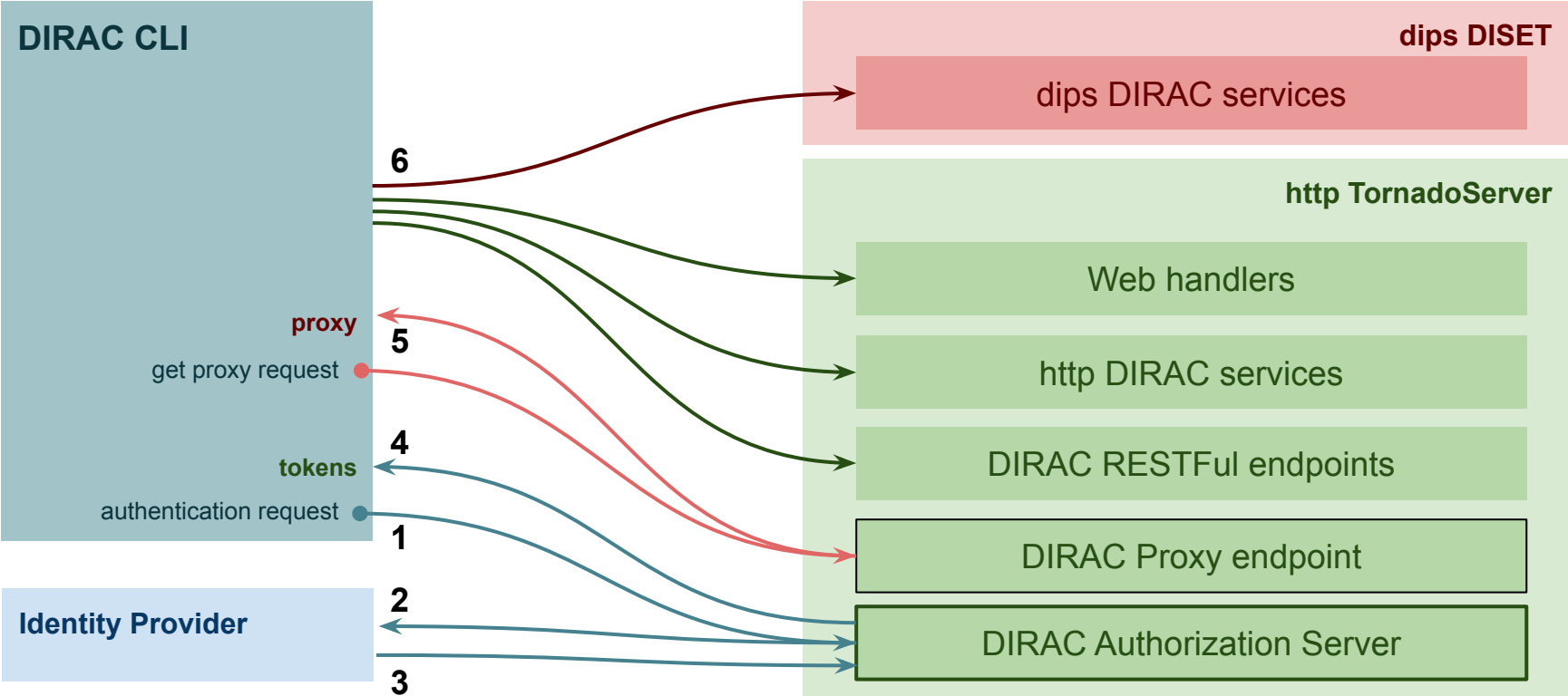
Authorization server is the server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

Client is the app that is attempting to act on the user's behalf or access the user's resources.





The main goal to introduce the AuthN/AuthZ mechanism to DIRAC based on Identity Provider services using OIDC(OAuth2) protocol and standard authorization code grant type.

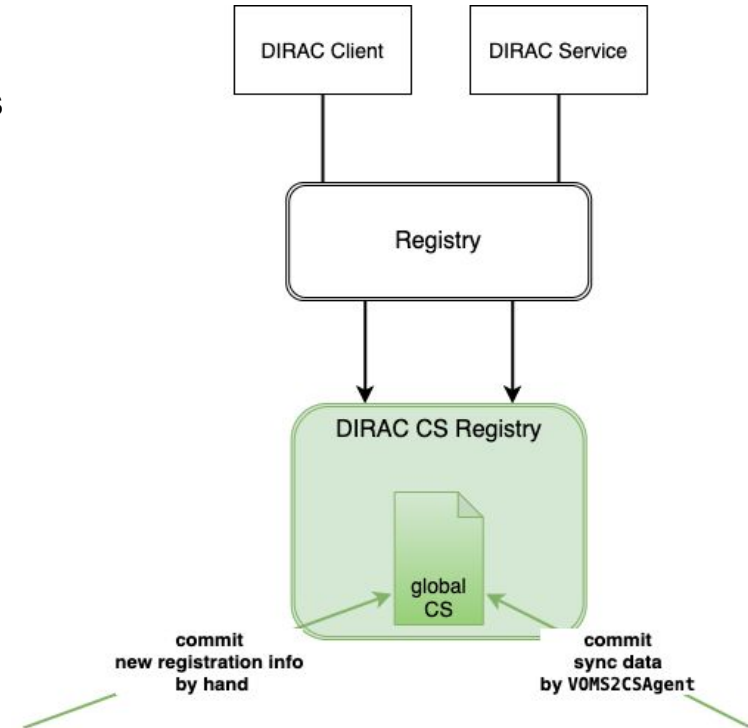


Five authorization steps shown on the previous slide:

1. The DIRAC Client sends a request to the DIRAC **Authorization Server (AS)**
2. DIRAC **AS** terminates the authorization session and redirects the user to the identity provider, where the user logs in
3. Identity provider returns to DIRAC **AS** user tokens, that DIRAC **AS** stored in the database
4. DIRAC **AS** returns access tokens to the DIRAC Client
5. The DIRAC Client makes a request to the DIRAC RESTful **Proxy endpoint** to get a proxy
6. The DIRAC Client can query DIRAC services using either tokens or proxy



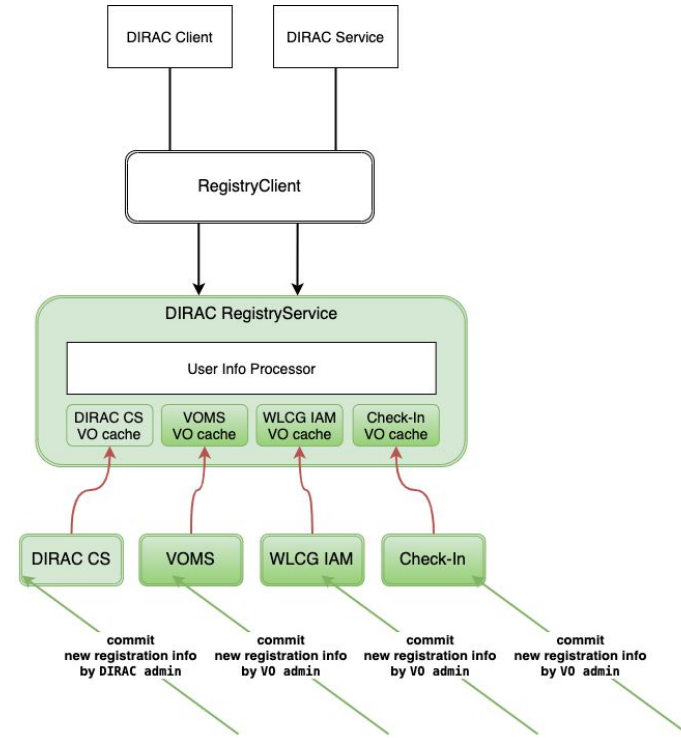
Currently, all users are described in the DIRAC configuration in the Registry section. The user information is synchronized with the VOMS servers using VOMS2CS agent.



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

User management is completely outside DIRAC, this is done by the VO administrators using appropriate web interface provided to VO upon registration.

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

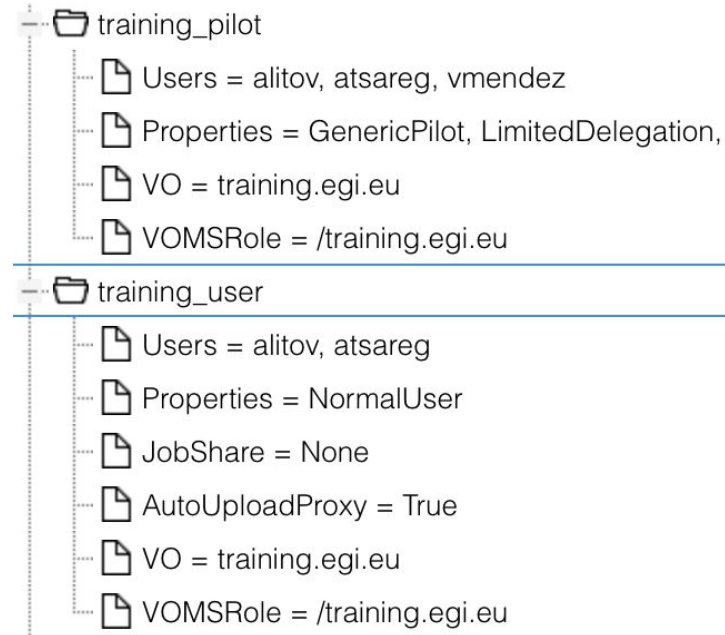


The main element of user management is the **DIRAC group** to manage permissions and separate activities.

DIRAC groups must be created for the registered VO, it must be associated with one of the roles defined by the VO.

At the moment, DIRAC groups are in "*many-to-one*" relation with VOMS role. For example, the user and pilot groups, which usually have the same VOMS role.

After the transition from VOMS VO to **IdPs**, it is necessary to ensure one-to-one correspondence between DIRAC groups and associated VO roles, for correct authorization using tokens.



Now, to start a working session with DIRAC, one needs to create a proxy certificate with the extension of the DIRAC group. This method remains intact. This is done with the command:

```
dirac-proxy-init -g my_group
```

But if a user wants to work without a certificate, user can do it with the command:

```
dirac-login -g my_group
```

This command uses the standard *OAuth2 DeviceCode flow* to obtain authorization from the **IdP**. As a result the user receives access tokens restricted to work with the requested group.

The received access tokens are enough to access DIRAC services that work through the http protocol. But given that there are still DIRAC services that work via the DISET protocol, user will need also a proxy certificate.

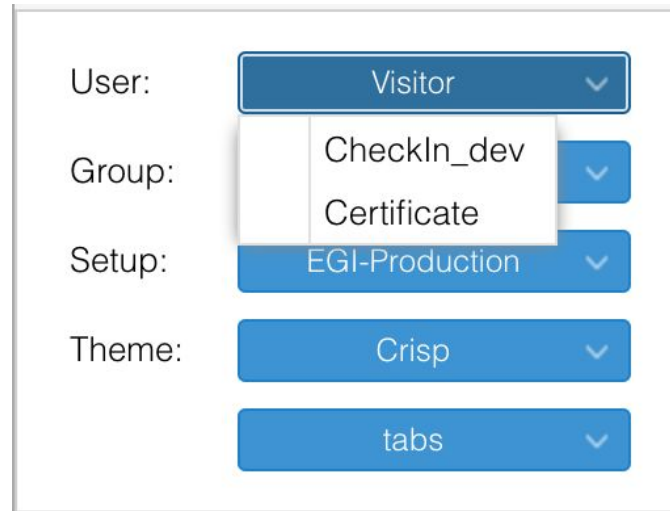
For the time when both access protocols, http and dips, will be in use at the same time, it will be possible to get the proxy certificate along with the tokens.

Therefore, adding the *--proxy* flag will result in users proxy from ProxyManager:

```
dirac-login -g my_group --proxy
```



WebApp portal users will be able to choose authorization methods by selecting a certificate or identity provider:



User: Visitor

Group: CheckIn_dev
Certificate

Setup: EGI-Production

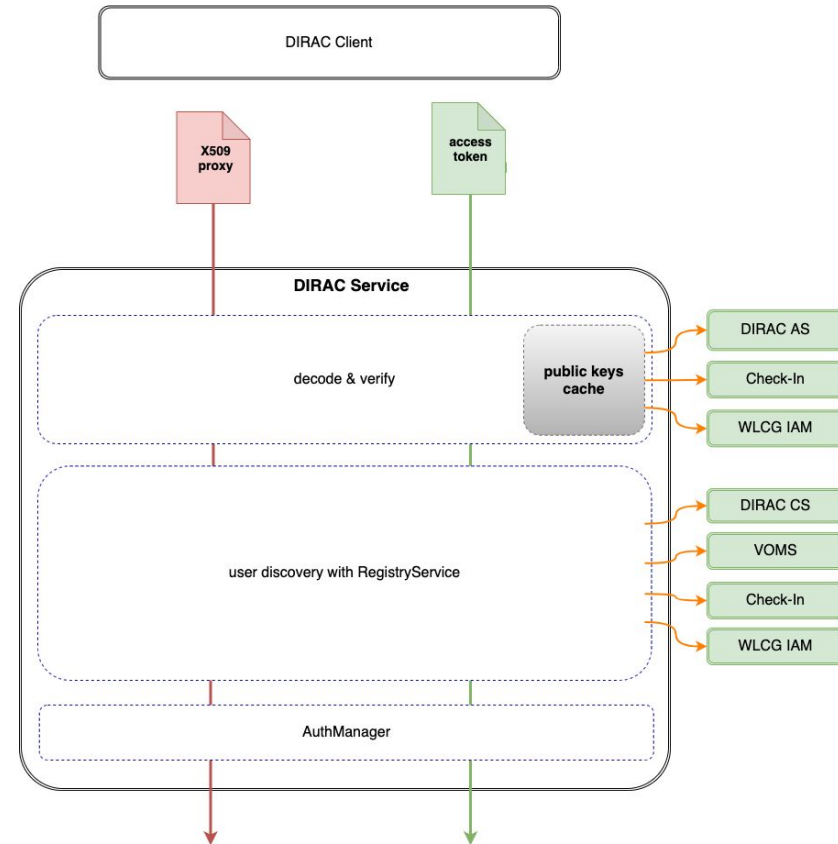
Theme: Crisp
tabs

Example of executing a command from DIRACCLI:

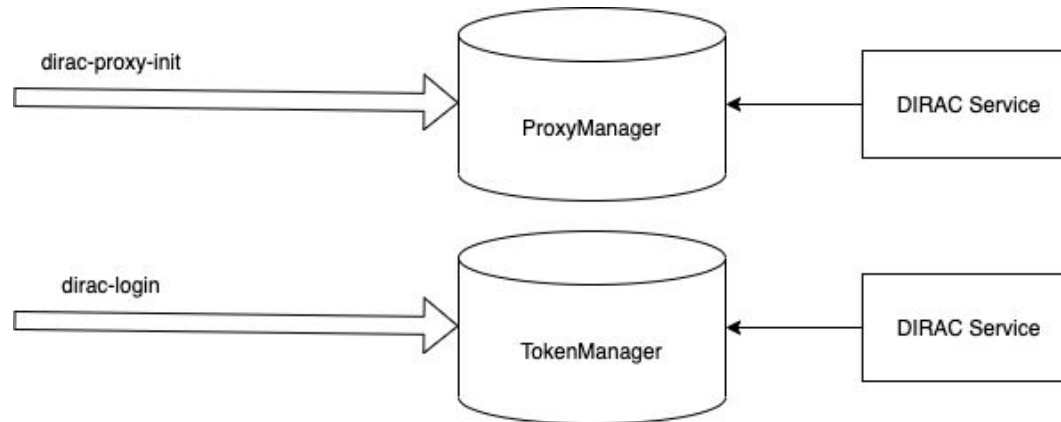
```
[alitov@mardirac DIRAC]$ dirac-login -g checkin-integration_user -P  
Use next link to continue, your user code is "PXXJ-ZNGF"  
https://marosvn32.in2p3.fr/DIRAC/auth/device  
Authorization pending.. (use CNTL + C to stop)
```

Consider the process of accessing the DIRAC service:

- the proxy or token received from the client with the request is first checked and validated. In the case of a token, the signature with the public key of the corresponding **IdP** is checked.
- Using the **Registry Service** and the information obtained from the proxy or token, the corresponding group is determined
- Finally, the **AuthManager** checks access rights of the received user group to the requested resource. The **AuthManager** implementation stays as it is.



Users delegate their access rights to DIRAC by uploading their long proxy certificates. In the case of OAuth2 access tokens, it will also have to be saved and updated, to be readily available without additional user interaction.



The existing method of delegation is to delegate a proxy by uploading it to **ProxyManager**. This happens transparently when executing the **dirac-proxy-init** command, or through the web interface.

Tokens are also deposited in the database in a similar service after successful authorization through the **IdP**, and an access token signed by DIRAC is returned to the user. Thus, DIRAC service being a registered client of the **IdP**, is able to maintain these tokens valid.



The most obvious use case of asynchronous user task execution is the pilot job framework. Pilots should be able to obtain valid tokens of the users owners of the payloads to be executed.

Discussions are currently underway on how the flow to getting user tokens by pilots will look like. The concept is being developed in DIRAC, but the work is still ongoing.



Several prototypes of using tokens were developed to provide user authentication and access to the DIRAC service as a technology preview. The elaborated architecture will be implemented and included in the release v7r3.

The implementation includes DIRAC original components. However, we consider the possible use of other relevant projects such as:

- MyToken: <https://mytoken-docs.data.kit.edu/>
- Vault: <https://www.vaultproject.io/>
- WLCG IAM: <https://indigo-iam.github.io/>



DIRAC must follow the tendency of moving the security infrastructures from X509 certificate to OAuth2/OIDC tokens

Several prototypes led to an architecture where the user and VO management will be done completely on the **Identity Provider** side.

Work is in progress to deliver the first fully functional implementation in the next major DIRAC release



Questions ?



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

