OAuth based AuthN/ AuthZ in DIRAC

A.Lytovchenko,
CPPM-IN2P3-CNRS, Marseille,
19 Nov 2019, Lyon

Outline

- Current DIRAC security framework
- Enabling DIRAC to use OAuth AAI
- OAuthDIRAC extension
- OAuthManager service
- X509 Proxy Providers
- Username as unique id
- Statuses of groups
- Dinamic Registry
- Cookies for authetication
- Status and plans
- Conclusions

X509 based security

- DIRAC is using X509 certificates for user authentication
- Certificate proxy delegation protocol is used to pass the user credentials to remote components performing operations on behalf of the users
- User rights are determined by the group membership encoded in the DIRAC proxy extension
- The ProxyManager stores long-living user proxies in the ProxyDB and serves short (limited) proxies to the components operating on behalf of the user

DIRAC and VOMS

- DIRAC users are members of at least one VO managed by a VOMS service
- User rights defined in VOMS as groups and roles are translated into DIRAC group membership
  - VOMS synchronization with VOMS2CSAgent
- DIRAC proxies can be dressed with VOMS extensions to access external grid services
  - VOMS and DIRAC proxy extension coexist in the same proxy

X509 -> OIDC/OAuth

- 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, etc
  - Users of many communities do not have access to Certification Authorities issuing X509 certificates
- Need for a new non-X509 security infrastructure
  - Industry standard
  - Widely accepted
- OAuth2.0 + OIDC is the suitable choice
  - Although not a single one

OAuth/OIDC/SSO

- **OAuth 2.0** is the industry-standard *delegation* protocol for conveying *authorization decisions* across a network of web-enabled applications and APIs

- **O**pen **ID C**onnect – is an identity layer on top of the **OAuth 2.0**.
  - allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server

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

SSO solutions

- There are multiple examples of SSO solutions
- The EGI Check-in service enables access to EGI services and resources using federated authentication mechanisms
  - A hub between federated Identity Providers (IdPs) and Service Providers (SPs) that are part of EGI

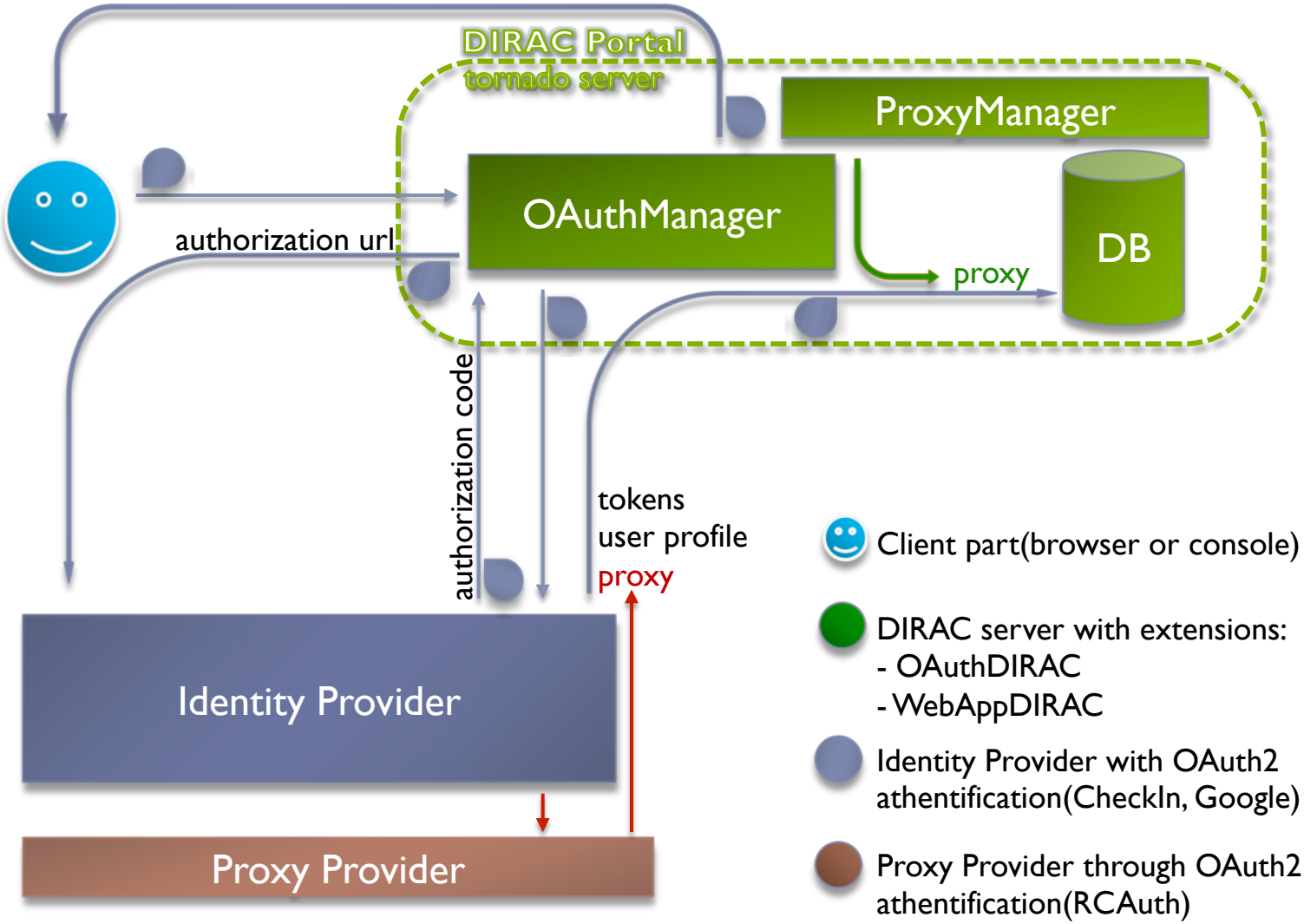
DIRAC support for SSO authentication

- Authenticate DIRAC users with the help of an external Authentication server
  - E.g. delegate it to EGI Check-In
- Get user profile information and eventually register users in DIRAC for supported VO's
  - Put users into DIRAC groups corresponding to the user profile
  - Similar to the procedure of synchronization with VOMS
- Ensure provisioning of X509 certificate proxies to be used for internal DIRAC client-server communications and for access to external services

New DIRAC "SSO" components

- OAuthDIRAC extension
  - OAuthManager service + OAuthManagerClient + OAuthDB
    - Generates authentication URL
    - Stores information on the user's OAuth session (session ID, AccessToken, RefreshToken)
  - AuthHandler in the WebApp framework
    - Providing OAuth callback http URL
    - REST interface for the command line authentication
- In WebAppDIRAC
  - Authentication based on interaction with OAuthManager service
  - User interface elements – login menu
- In DIRAC
  - Dynamic Registry

Authentication flow simplified



Setting up the OAuth components

- Install the DIRAC server with the extensions:
  - WebAppDIRAC
  - OAuthDIRAC
- Configure and start the OAuthManager service
- Configure and start the HTTP endpoints
  - AuthenticationHandler, AuthHandler, ConfigurationHandler in the WebApp/Tornado
- Register the client in OAuth2 authentication provider, e.g. Check-In or Google
  - Set authorization flow
  - Set redirect_uri(the OAuthManager HTTP endpoint in our case)
  - Set maximum of posible refresh token live time

configuration

- Set Identity Providers with some options in /Resources/IdProviders section:

```
{
 CheckIn                # Name of Identity Provider
  {
   Type = OAuth2        #This option to now that provider use OAuth2 authorization protocol
   issuer = https://aai-dev.egi.eu/oidc      #This option need to get oauth2 metadata from IdP
   client_id = 2C7823B4-wqenknsadljdas2-E5D06D955809 # ID and Secret of client that you registred
   client_secret = 732h9d0dn-3_CRcUf6paEMejjojAqQz5A  #   in Identity Provider
   Syntax       # In this section we set mechanism to parse incoming information from IdP
   {
   }
   proxy_provider = RCAut# Proxy Provider that able to generate proxy for user that
 }                         #   authorizated through this Identity Provider
}
```

configuration

- Set Proxy Providers with some options in /Resources/ProxyProviders section:

```
{
  RCAuth   # Name of Proxy Provider
  {
    ProxyProviderName = RCAuth
    ProxyProviderType = OAuth2#This option to now that provider use OAuth2 authorization protocol
    issuer = https://masterportal-pilot.aai.egi.eu/mp-oa2-serve# URL to get OAuth2 metadata
    client_id = myproxy:949241khasdkhkhk358d4981d# ID and Secret of client that you registred
    client_secret = lSh-Q32xh2pQc7rAlB_2qGVcQVNM#   in Identity Provider
    max_proxylifetime = 864000   # Maximum live time of proxy that Proxy Provider can create
    proxy_endpoint = https://masterportal-pilot.aai.egi.eu/mp-oa2-server/getproxy# URL that give access to
  }                                                                              #   get proxy in response
}
```

configuration

- Set OAuthManager http endpoint in /Framework/Production/Services section:

```
{
  URLs
  {
    OAuth = dips://ce-emi.bitp.kiev.ua:9244/Framework/OAuth      # URL of OAuthManager DIRAC service
    OAuthAPI = https://ce-emi.bitp.kiev.ua:9943/oauth2           # URL of OAuthManager http endpoint
  }
}
```

- Add /WebApp/TypeAuths section in the DIRAC portal configuration file to describe auth types that will be shown on the portal taskbar:

```
TypeAuths
{
 CheckIn        # Name of Identity Provider that need to enable in a portal
 { }
 Google
 { }
}
```

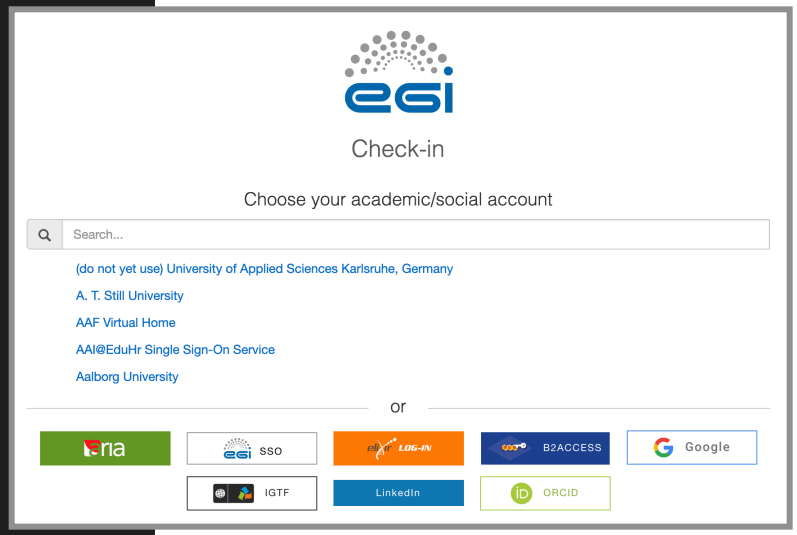
Web Portal authentication

Command line authentication

```
[[dirac@ce-emi pro]$ python DIRAC/FrameworkSystem/scripts/dirac-proxy-init.py -O CheckIn -g training_user -q
OAuth authentification from CheckIn.
Use link to authentication..
https://ce-emi.bitp.kiev.ua:9943/oauth2/oauth?getlink=MZ7XnO4iyMYTx9Vw2wkpBbHrm3Gz8f
```

```
[ Waiting 3.0 minutes when you authenticated..     ..*       $ [3~

Proxy generated:
subject      : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko/CN=3461819742
issuer       : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko
identity     : /DC=org/DC=ugrid/O=people/O=BITP/CN=Andrey Litovchenko
timeleft     : 23:59:59
DIRAC group  : training_user
rfc          : True
path         : /tmp/x509up_u3310
username     : alitov
```



Proxy Providers

- ProxyProvider is a new Resource type for services generating X509 certificate proxies on demand

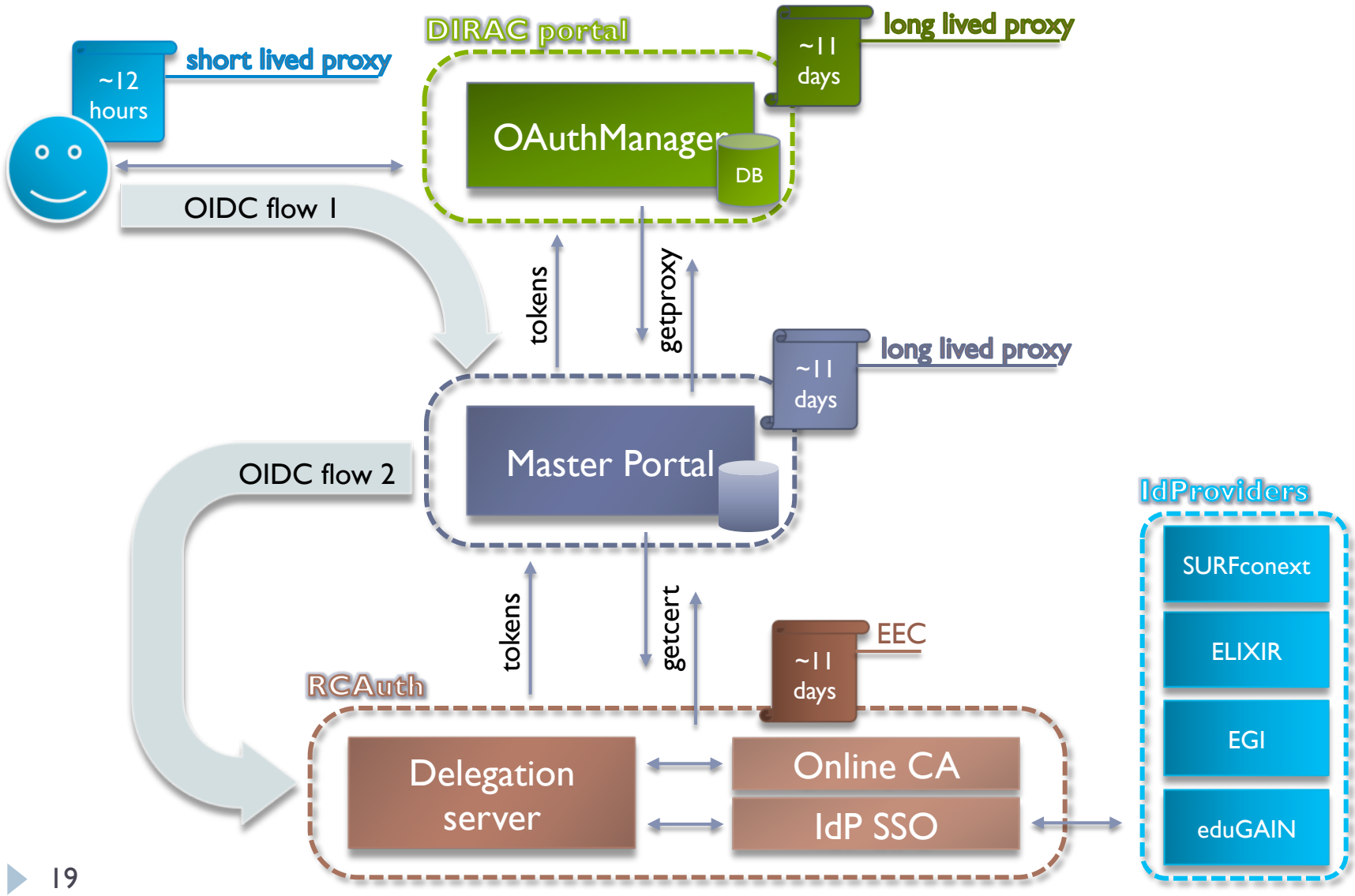
- Current implementations
  - DIRAC CA proxy provider – generates user proxy from a certificate signed by the DIRAC CA
  - RCAuth proxy provider

RCAuth

- RCauth.eu is a Research and Collaboration Authentication CA Service for Europe

- To obtain proxy certificates from the RCauth.eu online CA do not directly contact the RCauth CA, but use an intermediate service, a so-called Master Portal where you must register your client. Master Portal is an OpenID Connect Provider, with an integrated protected endpoint for obtaining proxy certificates.

RCAuth


DIRAC portal
~12 hours
short lived proxy
~11 days
long lived proxy
OAuthManager
DB
OIDC flow 1
tokens
getproxy
~11 days
long lived proxy
Master Portal
OIDC flow 2
IdProviders
SURFconext
ELIXIR
EGI
eduGAIN
tokens
getcert
~11 days
EEC
RCAuth
Delegation server
Online CA
IdP SSO

Proxy renewal

- We have to have valid proxy in the ProxyManager to perform operations on behalf of the user
- With X509 certificates stored proxies are renewed once per year by the users
- Renewal of proxies provided by the DIRAC CA
  - Just ask for the new proxy
- Renewal of RCAuth proxy is another complex flow using the OAuth AccessToken (and most likely RefreshToken) stored in the OAuthDB
  - To be done

Proxies in the ProxyDB

- Proxies are stored in DIRAC now with embedded DIRAC group extension

Proxy Manager

Selectors

User: atsareg

Group:

Delete | Items per page: 25 | Page 1 of 2 | Updated: 2019-05-15 05:10 [UTC]

User	DN	Group	Expiration date (UTC)	Persistent
User: atsareg				
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	beapps_pilot	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	beapps_user	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_admin	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_cloud	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_pilot	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_test	2019-08-20 08:58:46	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_tutorial	2019-08-20 08:58:46	True
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	dirac_user	2019-08-20 08:58:45	False
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	eiscat_common	2019-08-20 08:58:46	True
atsareg	/O=GRID-FR/C=FR/O=CNRS/OU=CPPM/CN=Andrei Tsaregorodtsev	eiscat_owner	2019-08-20 08:58:45	False

- Proxies returned by external proxy providers does not contain this extension
- Switching to storing only proxies without DIRAC extension
  - The extension will be added on the fly whenever the proxy delegation will be requested

username as unique id

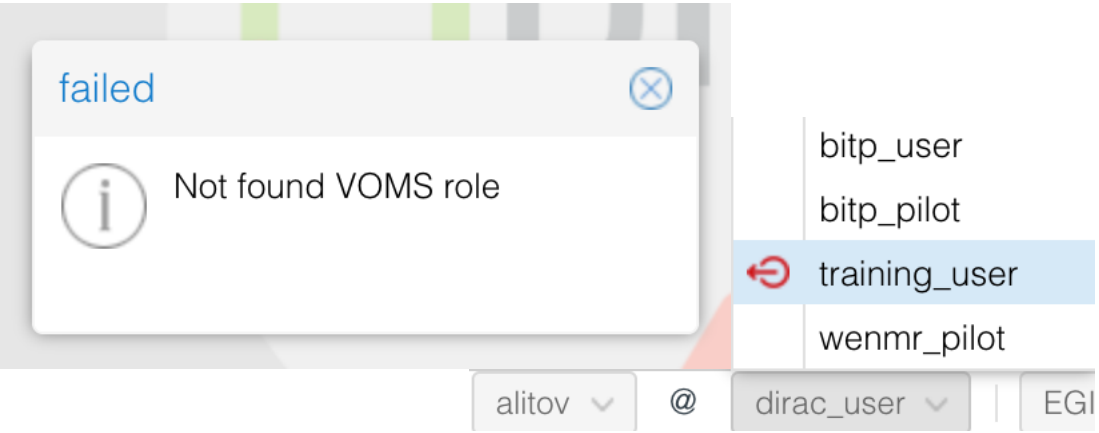
- To do any action in DIRAC you must be username@group, this is sufficient for initialization. But in some cases, userDN is used instead of the username as a unique identifier(ProductionSystem, TransformationSystem for example).

```
CREATE TABLE Productions(
  ProductionID INTEGER NOT NULL AUTO_INCREMENT,
  ProductionName VARCHAR(255) NOT NULL,
  Description LONGBLOB,
  CreationDate DATETIME,
  LastUpdate DATETIME,
  AuthorDN VARCHAR(255) NOT NULL,
  AuthorGroup VARCHAR(255) NOT NULL,
  Status CHAR(32) DEFAULT 'New’,
  PRIMARY KEY(ProductionID),
  INDEX(ProductionName)
) ENGINE = InnoDB DEFAULT CHARSET = utf8;
```

- If user have a few or none(like dirac_user group users) DNs, DIRAC take first from list or deny accsess.
- So, using username as unique identificator everywhere will make to use any number or absence of DNs, and simplify sessions/tokens(or some else access) implementation process instead certificates.
- DN can be found for username@group by using Registry.getDNForUsernameInGroup method.

statuses of groups

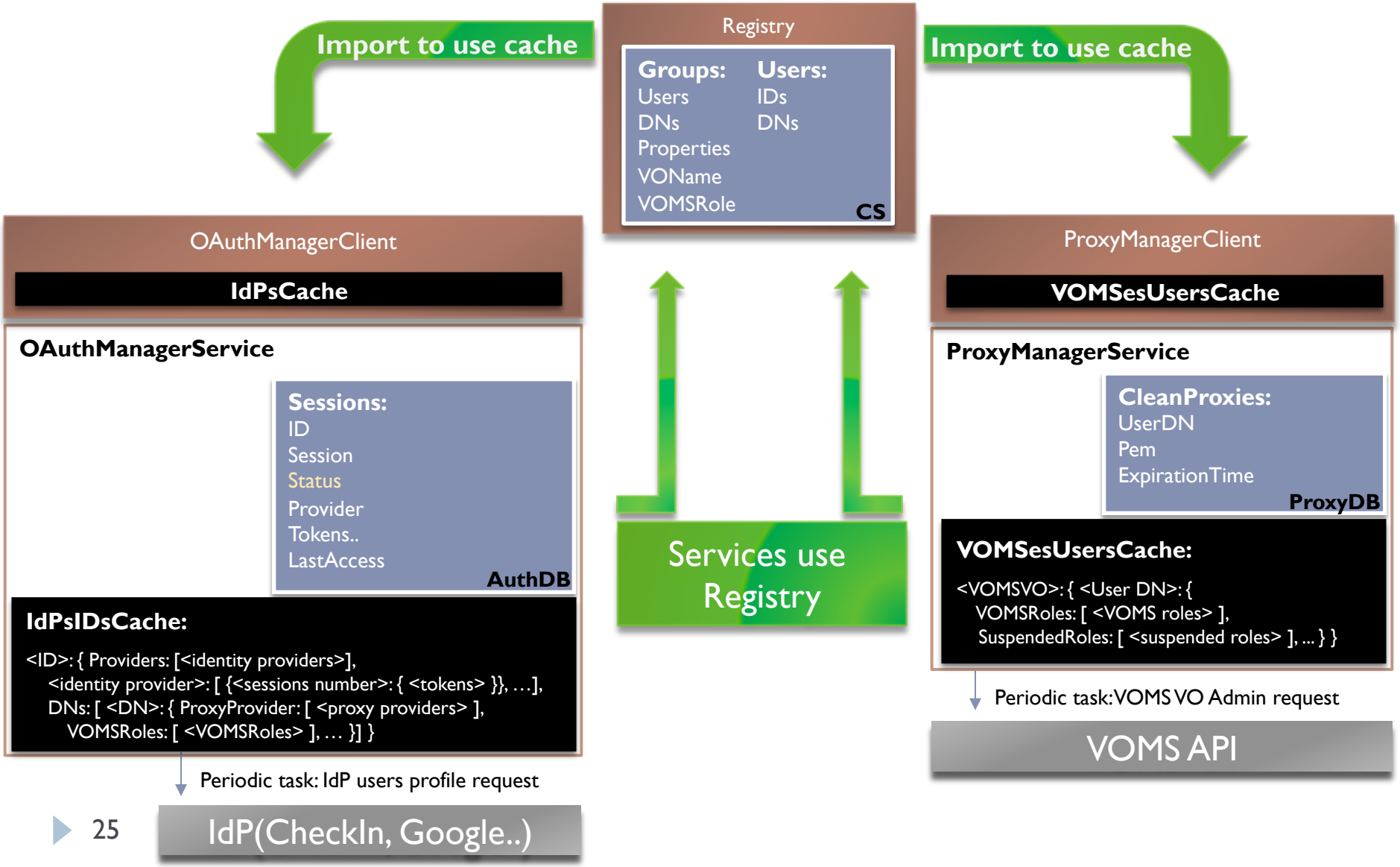
- In DIRAC portal, menu "groups" show all avaliable groups, but without status details. If you suspended in some VOMS VO that has been used by some group in list, you must know about it.
- So, when opened list groups, status of this groups avalible also, like as 'You suspended' or 'Need upload certificate to work with this group'.



Dynamic Registry

- Registry as a helper client of the ConfigurationService, contains dynamic information such as users in a group, suspended status, etc.
- With the implementation of ID providers, user profile information has emerged, which can be remotely changed(dynamic information).
  - To update user information one needs to look for changes, modify the CS(mark which ID provider made modification) and update the CS for all clients accessing the new information. There is a risk of configuration version confusion.
- So, Registry dynamic information can be stored as cache.
- User information that returned from Identity Providers, stored in OAuthManagerClient as CacheDict and if this class is not implemented Registry will not use it.
- VOMS VOs users information are stored in ProxyManagerClient as CacheDict and replicated to a temporary file. All VO admin proxies are stored here, it helps to update information by VOMS API.

dynamic Registry



Cookies for authentication

- There exist several methods for authentication such as CheckIn or Certificate. When user chooses authentication method, DIRAC stores cookie (in console case it will be stored in /tmp/cache_u<UID>):
  - Current authentication type as {'AuthType': <IdP name>}
  - If authentication is success, session information will added as {<IdP>: <session id>}

- Cookies will be used for authentication through https – future DIRAC client/server protocol

Conclusions

- There are many good reasons to replace the X509 based security framework by the one using OAuth/OIDC/SSO technologies
- The support of the OAuth/OIDC/SSO in DIRAC is implemented and demonstrated to work with the DIRAC4EGI service – Web Portal and command line client
- On demand X509 proxy generation is enabled with various proxy providers including the RCAuth service