DiracX

Security Model

How DIRAC works currently

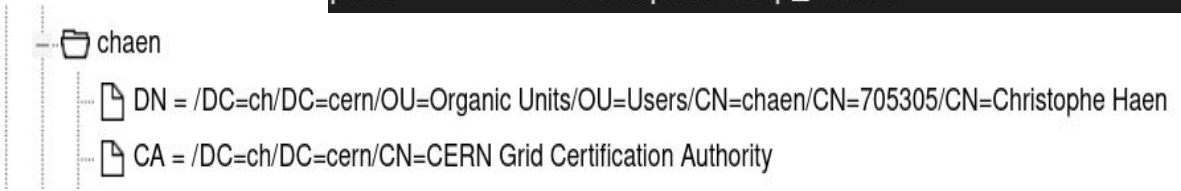
Identity based

User shows up with a proxy cert

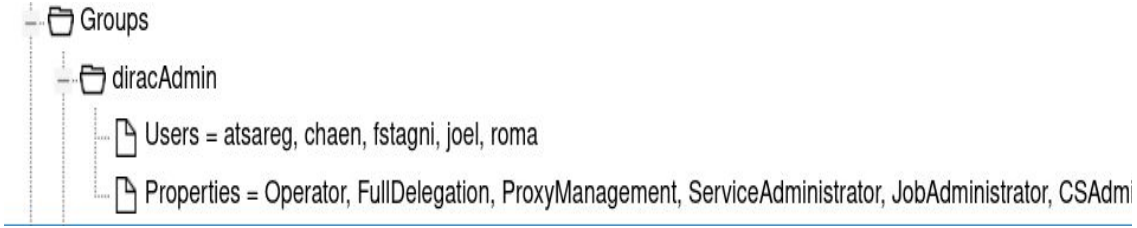
Contains: DN + Group

DN → CS → User ID

```
$ dirac-proxy-info
subject      : /DC=ch/DC=cern/OU=Organic Units/OU=U
issuer       : /DC=ch/DC=cern/OU=Organic Units/OU=U
identity     : /DC=ch/DC=cern/OU=Organic Units/OU=U
timeleft     : 23:45:58
DIRAC group  : diracAdmin
path         : /tmp/x509up_u1000
```

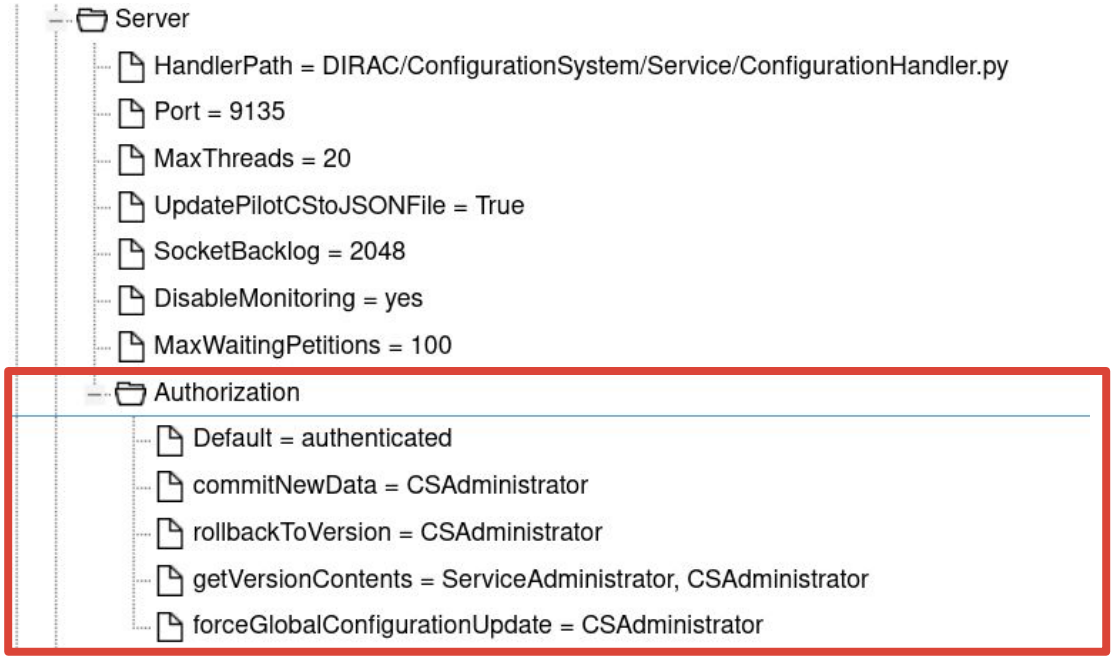


Group → CS → Properties



How DIRAC works currently: properties

String identifier. Matched to RPC calls (ish)




```
class SecurityProperty(str, Enum):
    #: A host property. This property is used::
    #: * For a host to forward credentials in an RPC call
    TRUSTED_HOST = "TrustedHost"
    #: Normal user operations
    NORMAL_USER = "NormalUser"
    #: CS Administrator - possibility to edit the Configur
    CS_ADMINISTRATOR = "CSAdministrator"
    #: Job sharing among members of a group
    JOB_SHARING = "JobSharing"
    #: DIRAC Service Administrator
    SERVICE_ADMINISTRATOR = "ServiceAdministrator"
    #: Job Administrator can manipulate everybody's jobs
    JOB_ADMINISTRATOR = "JobAdministrator"
    #: Job Monitor - can get job monitoring information
    JOB_MONITOR = "JobMonitor"
```

What does token auth look like?

```
$ curl -H "Authorization: Bearer ${myAccessToken}" $DIRACX_URL/api/auth/userinfo
{
  "sub": "gridpp:df101c3f-0285-58b3-7de2-9a3c9141675a",
  "vo": "gridpp",
  "dirac_group": "gridpp_user",
  "properties": [
    "NormalUser"
  ],
  "preferred_username": "chaen"
}
```

- Everything is inside the token → no need to do lookups
- Access and identity are considered separate matters

DiracX Token contents

No longer use the group for assuming the properties

(but it can provide a default set during issuance)

What do we keep the same?

- The properties themselves (for now)
- We still need group/identity
  (quotas, etc)
- Users only interact with groups
  (groups are an alias to a set of properties)

```
$ dirac whoami
{
  "sub": "gridpp:df101c3f-0285-58
  "vo": "gridpp",
  "dirac_group": "gridpp_user",
  "properties": [
    "NormalUser"
  ],
  "preferred_username": "chaen"
}
```

```
$ dirac login gridpp --group gridpp_user
Logging in with scopes: ['vo:gridpp', 'group:gridpp_user']
Now go to: https://diracx-cert.app.cern.ch/api/auth/device
.....Saved credentials to /home/chaen/.cache/diracx/creden

Login successful!
```

What do we want to change?

- Multi VO
  - Info in the token
  - As a design principle
- Separate properties
  - Allow to request specific properties (task)
  - Many use cases: CI, external tools, etc

Policies

- DIRAC has a quite flexible system for WMS: ***JobPolicy***
- Allows to define authorization rules for the various job operations based on group membership
- Make something more generic which can be applied to other systems (Transformation, CS, DFC).
  - e.g. expose only part of the namespace

Admin VO vs VO Admin

- VO Admin:
  - Can do everything within a given VO

- A special Admin pseudo-VO:
  - can see the state of everything
  - can’t do some actions (e.g. submit job)
  - can do others (e.g. kill job)

How do we get a token?

Standard ways:

- Web: Redirect to IdP
- CLI: “Device flow” (like you saw from Chris)
- Others: Extension could support others (e.g. kerberos)

For compatibility we can also:

- use a proxy to get a token
- use a token to get a proxy

Pilots

- OAuth was designed for TECH_COMPANY_1 to talk to OTHER_BIG_TECH_COMPANY as MEATBAG_123456789
- We have a lot more VOs (and a lot fewer users)
- Pilots are special, tech companies don’t really send something into the wild
- Need to do something custom 🤮

The three pilot credentials

- Credential A:
  - Is submitted together with the pilot
  - Long lived, very limited scope
- Credential B:
  - Is given in exchange of Credential A (could: limit IPs, single use, …)
  - Pilot lifetime, can only request N jobs + report pilot info
- Credential C:
  - Retrieved with Credential B
  - Job lifetime, permissions depend on the job itself

The impersonator

- DIRAC servers need to talk to DiracX as **MEATBAG_123456789**
  - Return a token for a proxy
  - Sandbox management
  - etc

- **TheImpersonator** allows just that :-)
  - Based on shared secrets

```
with TheImpersonator(credDict) as client:
    res = client.jobs.initiate_sandbox_upload(sandbox_info)
```

Conclusions

- We intend to make it the transition from X509 in DIRAC to Token in DiracX
  - *Transparent* to users
  - More flexible for experts
  - **But mostly: transparent to Daniela**

- Security model is available for review in
  - Has already been reviewed by an expert
  - See:

DIRACGrid/diracx#136

Questions?