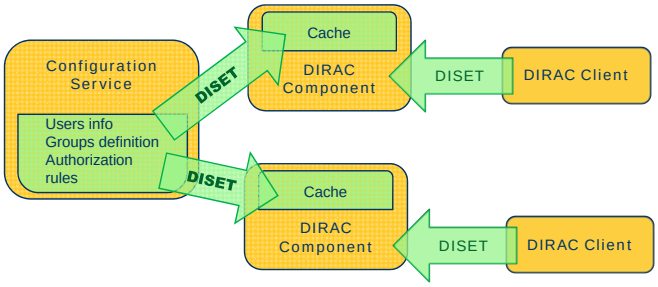
DIRAC framework



DIRAC Security Framework

All DIRAC components rely on a low level framework that provides the necessary basic functionality. This framework contains:

- **DISET: DIRAC's secure communication protocol for RPC and file transfer**
- **Configuration System: Providing redundant distributed mechanism for configuration and service discovery.**

All DIRAC connections are handled by DISET. DISET uses OpenSSL through a custom python binding (derived from pyOpenSSL). This provides grid authentication and encryption, using X509 certificates and grid proxies.

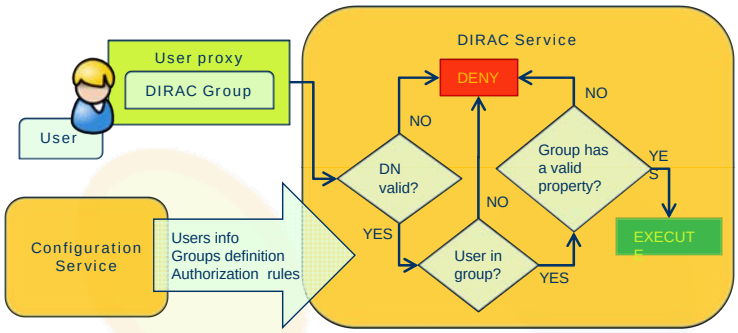
Permissions and proxies

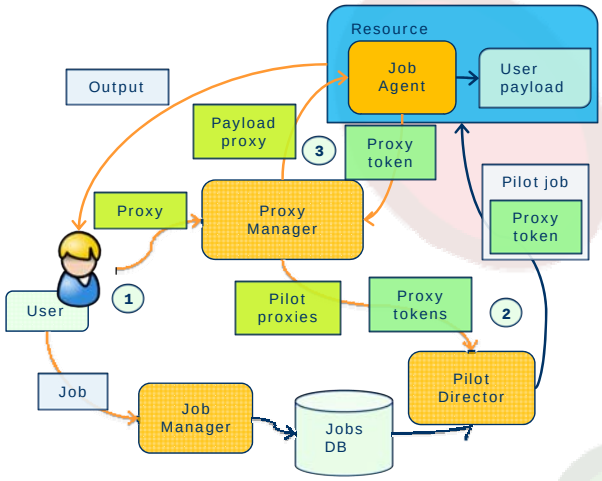
Not all users are allowed to perform all actions.

DIRAC implements a authorization schema to decide if a given entity can execute an action. All actions have a set of valid properties. The requesting entities have to present at least one allowed property for that action.

All users in DIRAC are assigned to a set of groups depending on their privileges, and each group has a set of properties. At any time a given user can only act using one of his/her groups.

Users define under which group they want to act by embedding the group in their proxy. DIRAC provides this functionality when creating a new proxy. Thus having the group signed directly by the user, the *user group cannot be changed (or added if it's not there)* after the proxy has been created.





Payload and proxies

DIRAC has to make sure the user payload runs under the user credential. DIRAC stores the user proxy and takes it to the resource where the user payload will run. All the proxy managing is done under a very strict security schema to minimize the damage that a stolen proxy may cause. The way DIRAC does it is the following:

1. A user uploads a proxy with the group embedded to the Proxy Manager and then submits a job to DIRAC.
2. In order to submit pilot jobs to the resources, the Pilot Director downloads a non-limited proxy using its own credential, and requests a proxy token. This token will have to be presented by the pilot job to be able to retrieve the real user proxy.
3. When the Job Agent matches a job to run in the resource, it downloads the payload proxy using its own credentials and token. Before the payload starts to run the payload environment is changed, so the payload automatically only sees the user proxy.

Pilot proxies have a special DIRAC group embedded. They can only belong to a very restricted set of users.

See [108] by R. Graciani et al. for more information about pilot jobs.

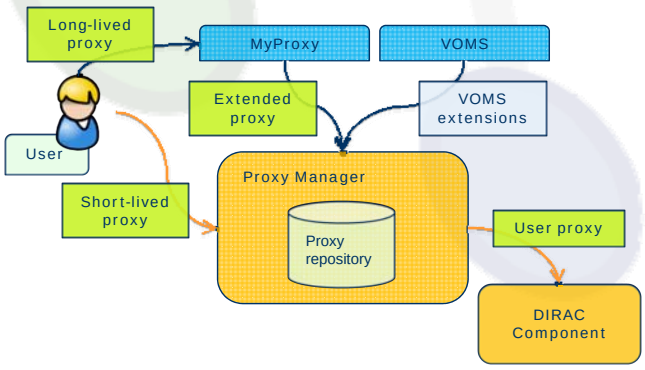
Proxy management

DIRAC has its own component for managing proxies. The Proxy Manager is a repository where users can upload their proxies. It will be used later on by all DIRAC components that require a user proxy. All proxy movements through the network are done through delegation.

The Proxy Manager can use other grid middleware proxy management components to enhance its functionality. For instance it can use VOMS to add the required attributes to a proxy.

DIRAC only keeps a short-lived user proxy in the system. Typically user's proxy life time is shorter that the time a user job stays in the system, and DIRAC needs to keep the proxy alive while the job is in the system. That requires DIRAC to extend the proxies in the system that are about to expire.

The DIRAC Proxy Management system talks to the MyProxy service and request new proxies for those about to expire when needed.



A.Casajus and R.Graciani (Universitat de Barcelona) on behalf of the LHCb DIRAC team