Argus: the new gLite AuthZ Framework

A. Ceccanti, A. Forti, V. Ciaschini
INFN-CNAF

www.eu-egee.org

Information Society and Media

Motivations

- **Different mw services use different authorization mechanisms**
  - Very hard for admins to understand what are the authorization rules at their site, very hard to ban/unban users
- **There is no central grid-wide banning list**
  - Urgent ban cannot be taken for granted
- **AuthZ is static!**
  - It's not straightforward to change authZ policies without reconfiguring services
- **It's not easy to monitor authorization decisions**

Argus main benefits

- **Consistent authorization decisions**
- **Makes banning/unbanning users very easy**
  - ban by subject, CA, VO
- **Enables composition of policies from distributed sources**
  - local policy + INFN policy + OCST policy + VO policy= effective policy
- **Enables dynamic policy management**
- **Has a flexible deployment model**
  - Has been designed with HA in mind

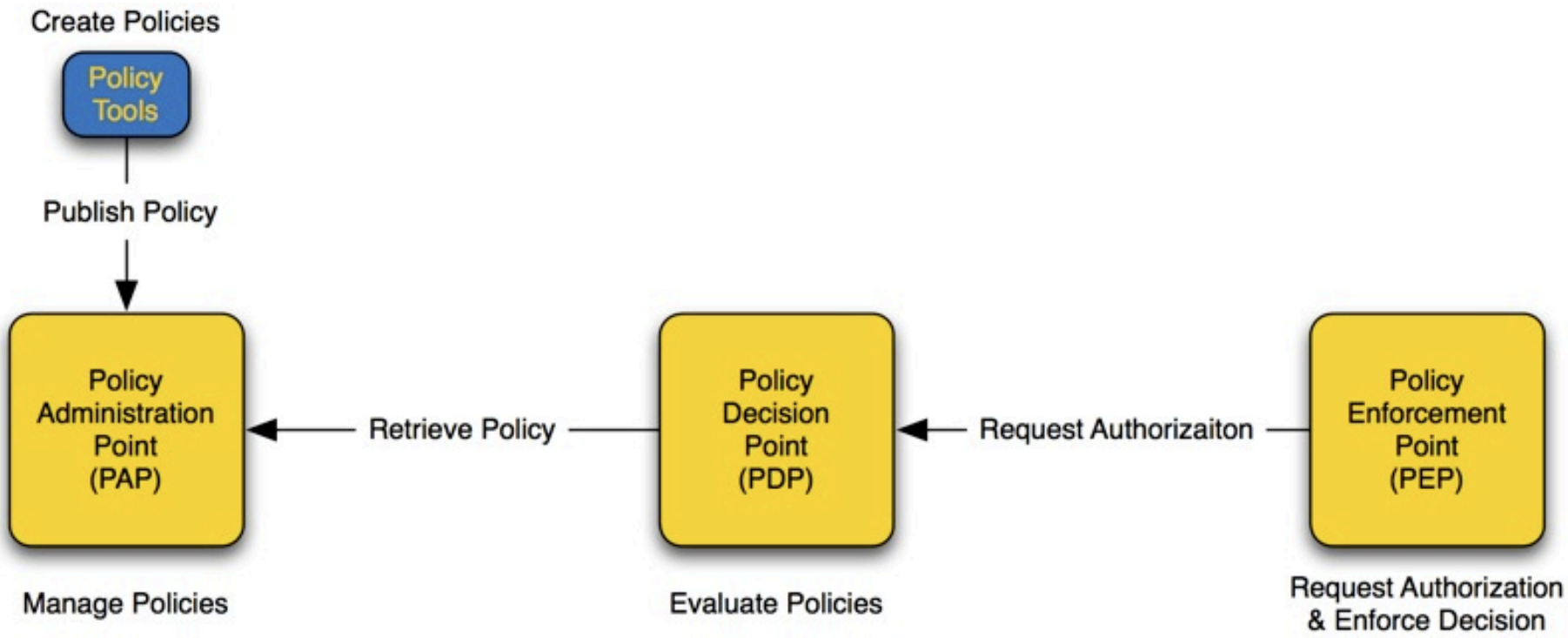
Institutions Involved

- **CNAF**
- **HIP**
- **NIKHEF**
- **SWITCH**

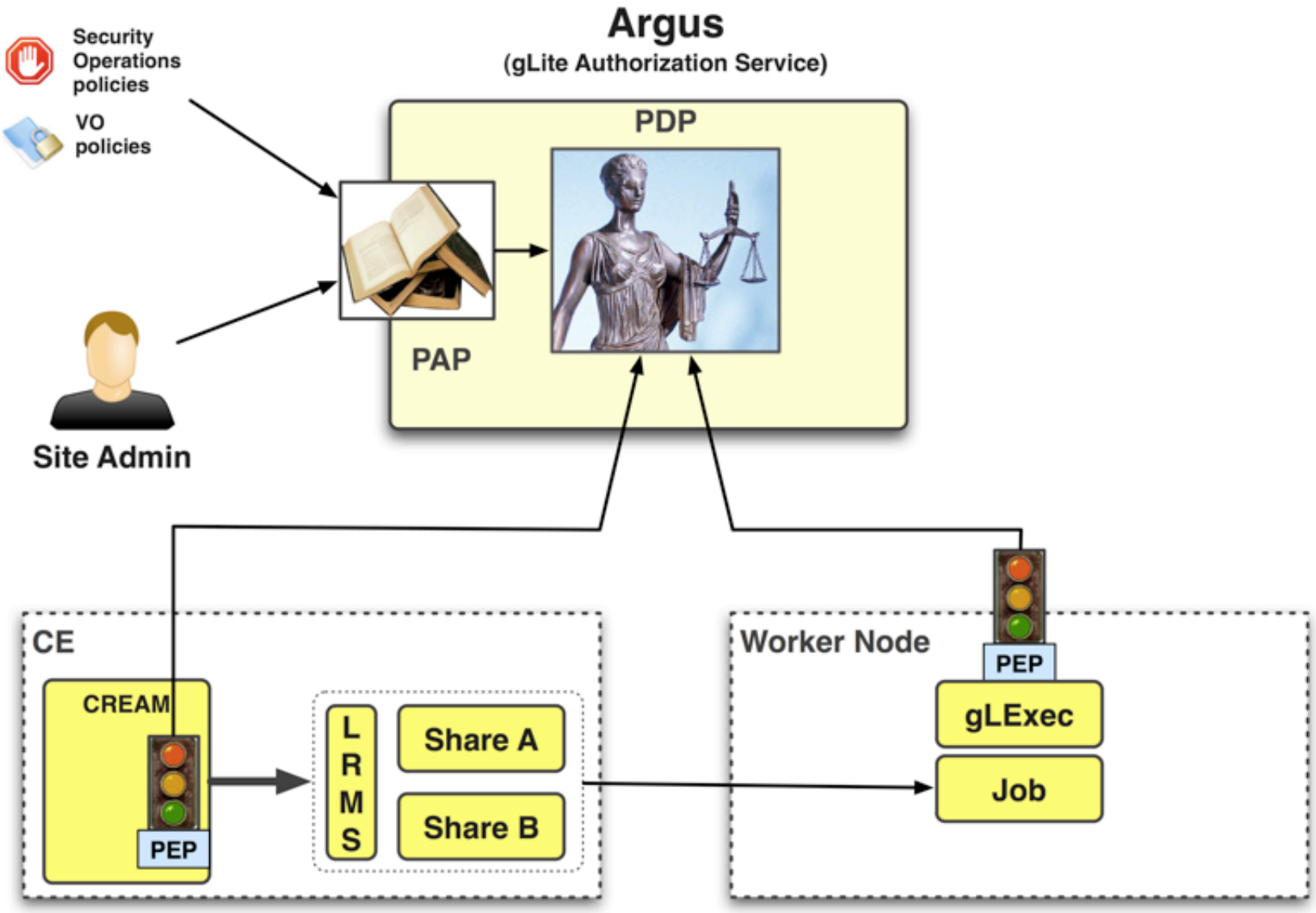
- **Deployment plan**
  - Devised together with SA1 / SA3
  - Reviewed and endorsed by TMB

- **Note abbreviation: authZ = authorization**

Argus High level overview

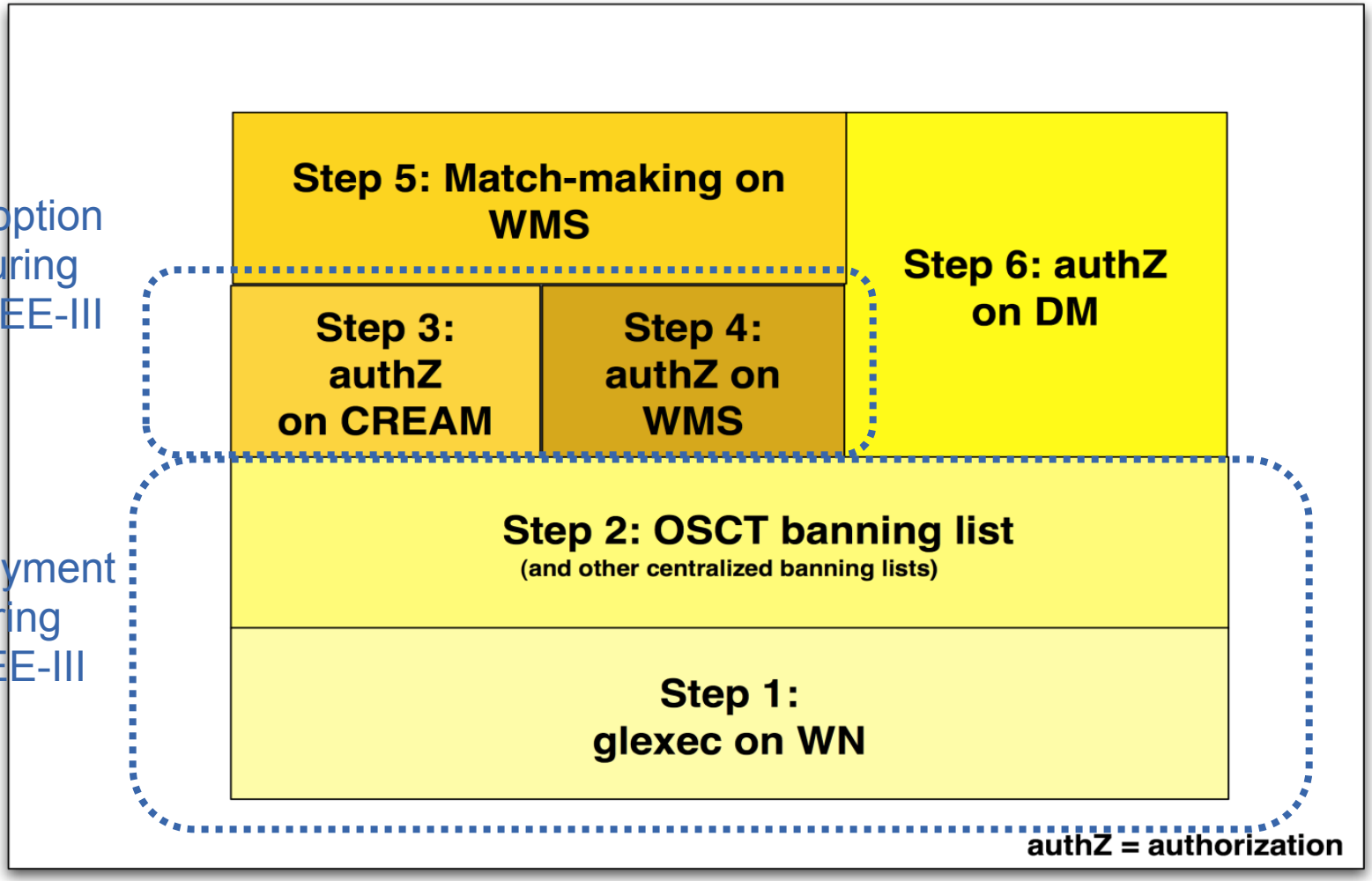


- PAP: author, store, manage and distribute policies
- PDP: evaluate requests against policies
- PEP: create request and enforce PDP decisions


Argus
(gLite Authorization Service)
Security Operations policies
VO policies
PDP
PAP
Site Admin
CE
CREAM
PEP
LRMS
Share A
Share B
Worker Node
PEP
gLExec
Job

Argus deployment plan

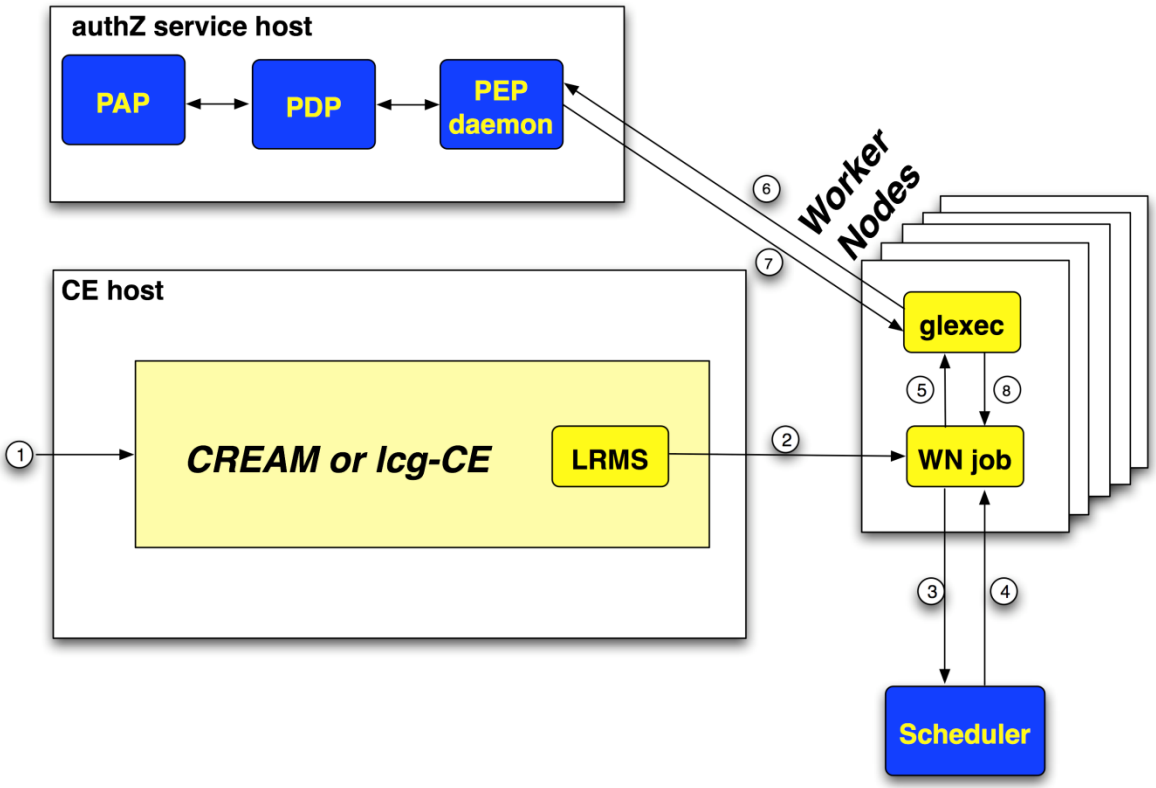
Guiding Principle: No big bang but gradually increasing use of authZ service through six *self-contained* steps



Proposed Deployment Plan (2/4)

1. gIExec on the WN:

- Only change on WN is new version of glexec / LCMAPS
- Use of authZ service is a configuration option
- Installation of authZ service on one host through YAIM
- ALL policies are local (i.e. no remote policies)
  - Only banning rules and enforcement of pilot job policy
- Note: No change to CREAM or lcg-CE (authZ policy only affects pilot jobs)

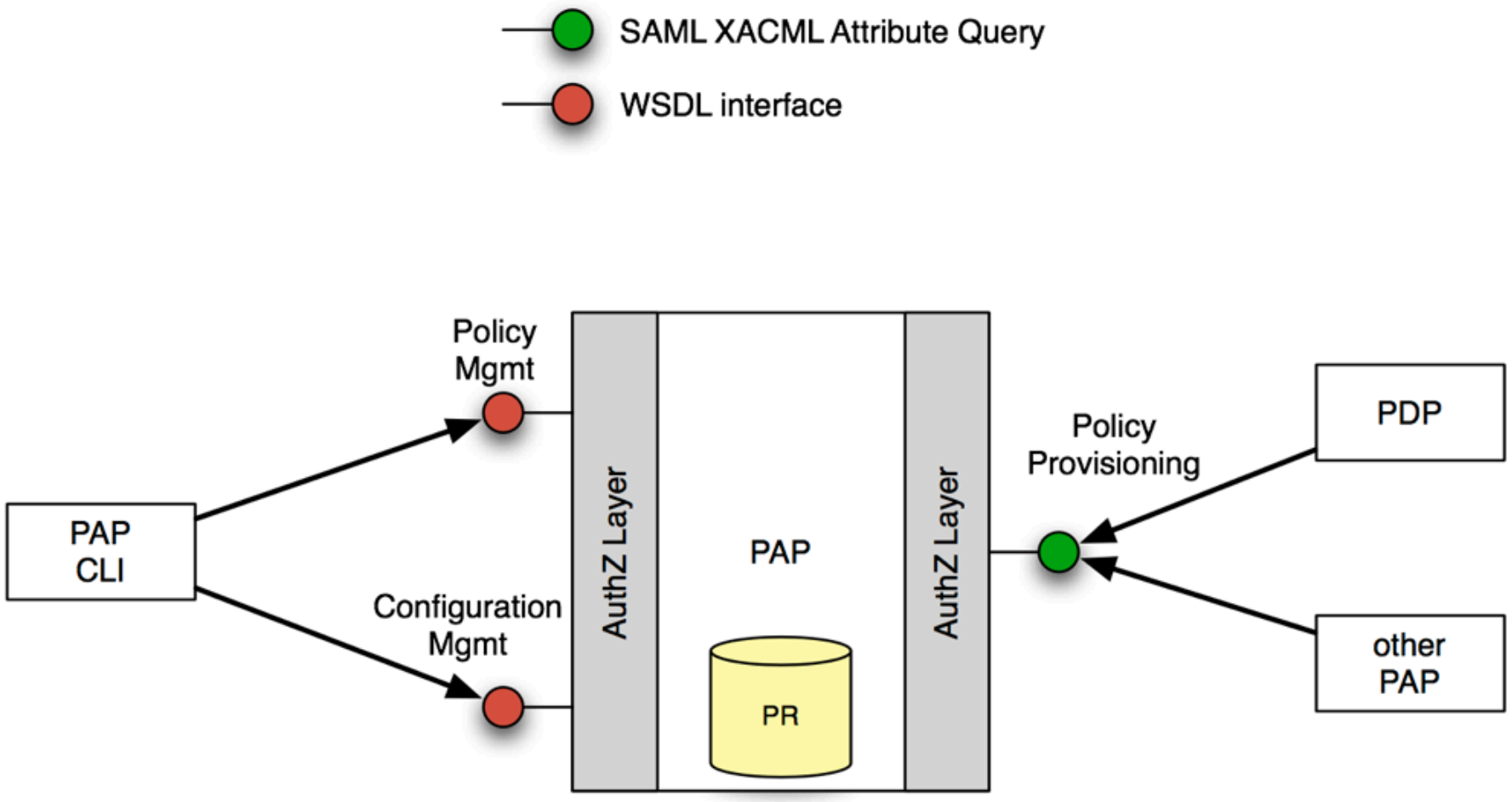


Short overview of the services

The Policy Administration Point

- **Developed at CNAF**
- **Provides:**
  - Tools for authoring, storing and managing policies used by the Authorization Service
    - Hides the complexity of the XACML from the users
  - A policy distribution mechanism
  - An authorization layer that defines “who can do what” on the PAP
    - who can write/manage policies, which other paps are trusted, etc...
- **Command Line Interface: “*pap-admin*”**
  - Provides scriptable access to all the PAP functionality
    - Policy management
    - Policy distribution
    - PAP Authorization and configuration

The PAP architecture



The Simplified Policy Language

Argus is designed to answer the question:

Can user X perform action Y on resource Z?

The SPL makes it easy to write policies that state which actions can be performed on which resources by which users.

```
resource ".*" {

    action ".*" {

        rule deny { vo = "lhcb" }

    }
}
```

The Simplified Policy Language

- **Hides XACML complexity but keeps much of its flexibility, e.g.:**
- **Ban non-italian users from CMS on my CE**

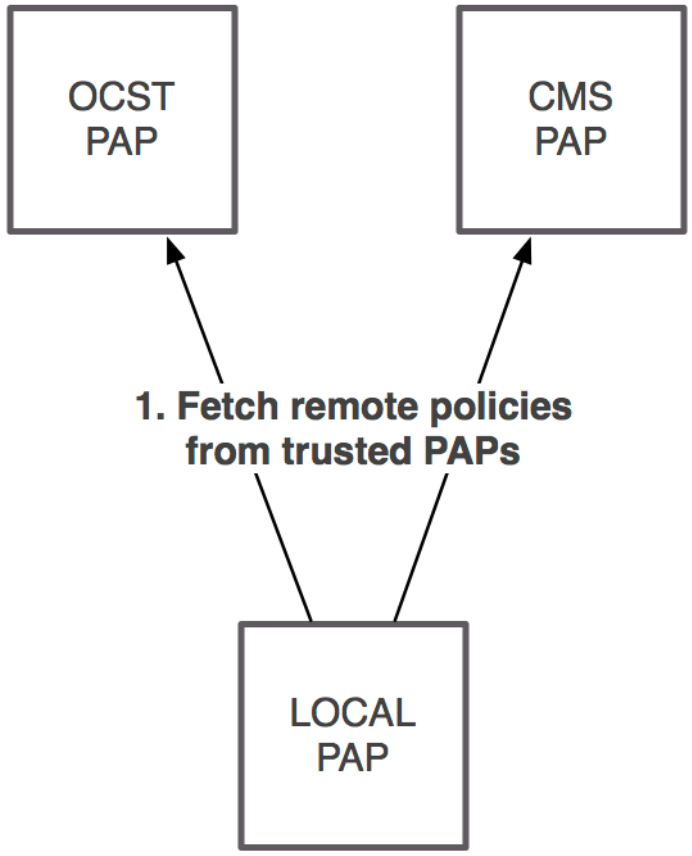
```
resource “my-ce”{
    action “.*” {

        rule permit {
            vo = “cms”
            ca = “CN=INFN CA,O=INFN,C=IT”
        }

        rule deny { vo = “cms” }
    }
}
```

Policy distribution

- PAPs can fetch policies from other trusted PAPs
- Policies can then be ordered so that it's possible to define which policies takes precedence
- The default configuration always gives precedence to local policies
- This mechanism enables centralized Grid wide banning



PAP command line tools

- List currently active policies:

```
pap-admin list-policies
```

- Import policies expressed in the SPL from a file:

```
pap-admin add-policies-from-file my-policies.txt
```

- easily ban/un-ban users, vos:

```
pap-admin ban vo test_vo
pap-admin un-ban vo test_vo
```

- add a generic policy:

```
pap-admin add-policy --resource 'ce_1' --action '.*' \
   permit pfqan='/test_vo/Role=pilot'
```

Argus Server components

- **Policy Administration Point**
  - described before...
- **Policy Decision Point**
  - the XACML engine that evaluates policies against incoming requests coming from the PEPs
  - can be replicated for scalability/fault tolerance
- **Policy Enforcement Point Daemon:**
  - Converts lightweight protocol used by the thin clients (PEPs) in SAML/XACML and forwards request to the PDP
  - can be replicated for scalability/fault tolerance
  - implements user mapping

Argus client components

- **C/JAVA PEP libraries**
  - these libraries are integrated in the services that will talk to Argus
- **gLexec/LCMAPS plugin**
  - callout to Argus on the WN (leverages the C pep library)
- **PEPCLI test/debugging tool**
  - makes it easy to test authZ policies on the service

Current status

- **Argus v. 1.0 (patch 3076)**
  - Certified
  - Pilot deployment started at several sites
    - CNAF, SWITCH, FZK, CESNET
    - VOs could test it **now**!
- **Argus v. 1.1 (patch 3536)**
  - Fixes some bugs/vulnerabilities found 1.0
    - Client authN now available on all services
    - Small usability fixes
  - Defines Argus WN AuthZ profile
    - set of attributes and data types on which authz decisions are based
  - Ready for certification this week

Argus Evolution: Integration

- **Integrate Argus in existing middleware:**
  - CREAM (in the next release)
  - WMS
    - at the AuthZ level (when?)
    - inside matchmaking (more complex, when?)
  - STORM
    - ??

- **New attribute profiles (could/will) be defined for each integration**

PAP evolution

- **PAP web application**
  - Search and management of policies
  - Monitoring and configuration
- **RDBMS Policy persistence**
  - Enables High Availability
- **RESTful interface for Policy Management**
  - Improves client application performance
  - Cleaner API towards other services
- **Evolution of the Argus Simplified Policy Language to match user requirements**