

Resources in DIRAC

Federico Stagni (on behalf of
Mario Ubeda)



- ▶ What's this
- ▶ /Resources RFC
- ▶ New RSS
 - ▶ Ontology and code
- ▶ ToDo



What's this

DIRAC.ResourceStatusSystem

- ▶ For storing resource status in DIRAC
 - ▶ status information
- ▶ An advanced monitoring tool
 - ▶ Aggregating dispersed information
- ▶ An “autonomic computing” tool
 - ▶ The core is a generic policy system
 - ▶ Used for monitoring and management
 - ▶ Auto ban/un-ban, triggering tests, etc..



Before the code

RFC (click!)

- ▶ Defines how the /Resources section of CS should be → not an /Operations section!
- ▶ Result of VERY long discussions
 - ▶ Completed ~6 months ago
 - ▶ Not in production yet
 - ▶ Code had to be written (done, not integrated yet)

Before the code /2

- ▶ Key concepts:

- ▶ Community (VO)
- ▶ Site (access point → locality!)
- ▶ Domain (WLCG, Gisela, EGI...)
- ▶ Resource Type (Computing, Storage, Catalog, FileTransfer, Database, CommunityManagement)

`/Resources/Sites/[Site Name]/[Resource Type]/
[Name Of Service]/[Type Of Access Point]/[Name
Of Access Point]`

`/Resources/Domains/[Domain Name]`



RFC → RSS (click!)

The CS structure is mapped in a 3 level hierarchy, each entry with a status:

- ▶ → Sites
- ▶ → Resources
- ▶ → Nodes



RSS for status information

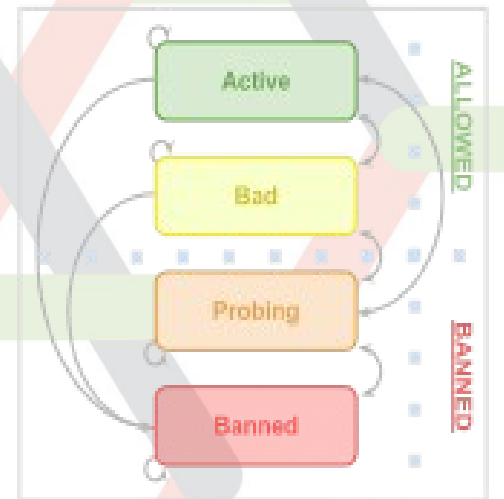
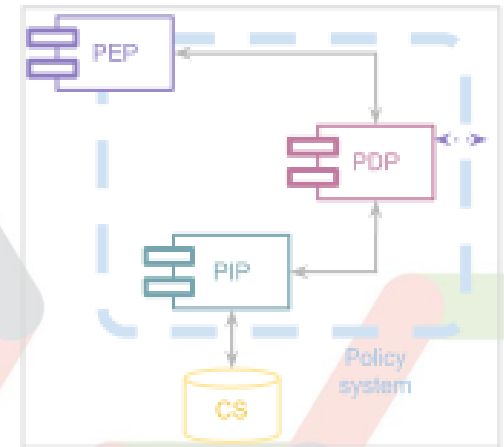
- ▶ DB:
 - ▶ ResourceStatusDB: tables for: Status, Log, History
 - ▶ Status: 3 families of identical tables: Site, Resource, Node
 - ▶ Log: mostly for debugging purposes
 - ▶ History: keeps historical changes of status
- ▶ Service
 - ▶ ResourceStatusHandler (expose ResourceStatusDB)
- ▶ Client
 - ▶ ResourceStatusClient: for interacting with the ResourceStatusDB
 - ▶ ResourceStatus: object that keeps the connectivity with the DB/Service - refreshing DictCache of Storage Element status
- ▶ Web (in dev within LHCbWebDIRAC)
 - ▶ Status Summary page (all “resources” combined)

RSS for advanced monitoring

- ▶ DB:
 - ▶ ResourceManagementDB
- ▶ Service
 - ▶ ResourceManagementHandler (mostly exposes the cached monitoring information)
- ▶ Agents:
 - ▶ CacheFeederAgent
 - ▶ Populates a cache of (useful, configurable, VO-specific) monitoring information
 - e.g.: downtimes, failure rates, external monitoring results ...
 - ▶ Use “commands”
- ▶ Commands (implementation of the Command pattern) → not yet clients!
 - ▶ Downtimes, accounting, jobs, transfers, space token occupancy...
- ▶ Web (in dev within LHCbWebDIRAC):
 - ▶ Detailed information for each of the Sites (book-markable pages)
 - ▶ Downtimes overview

RSS for autonomic management

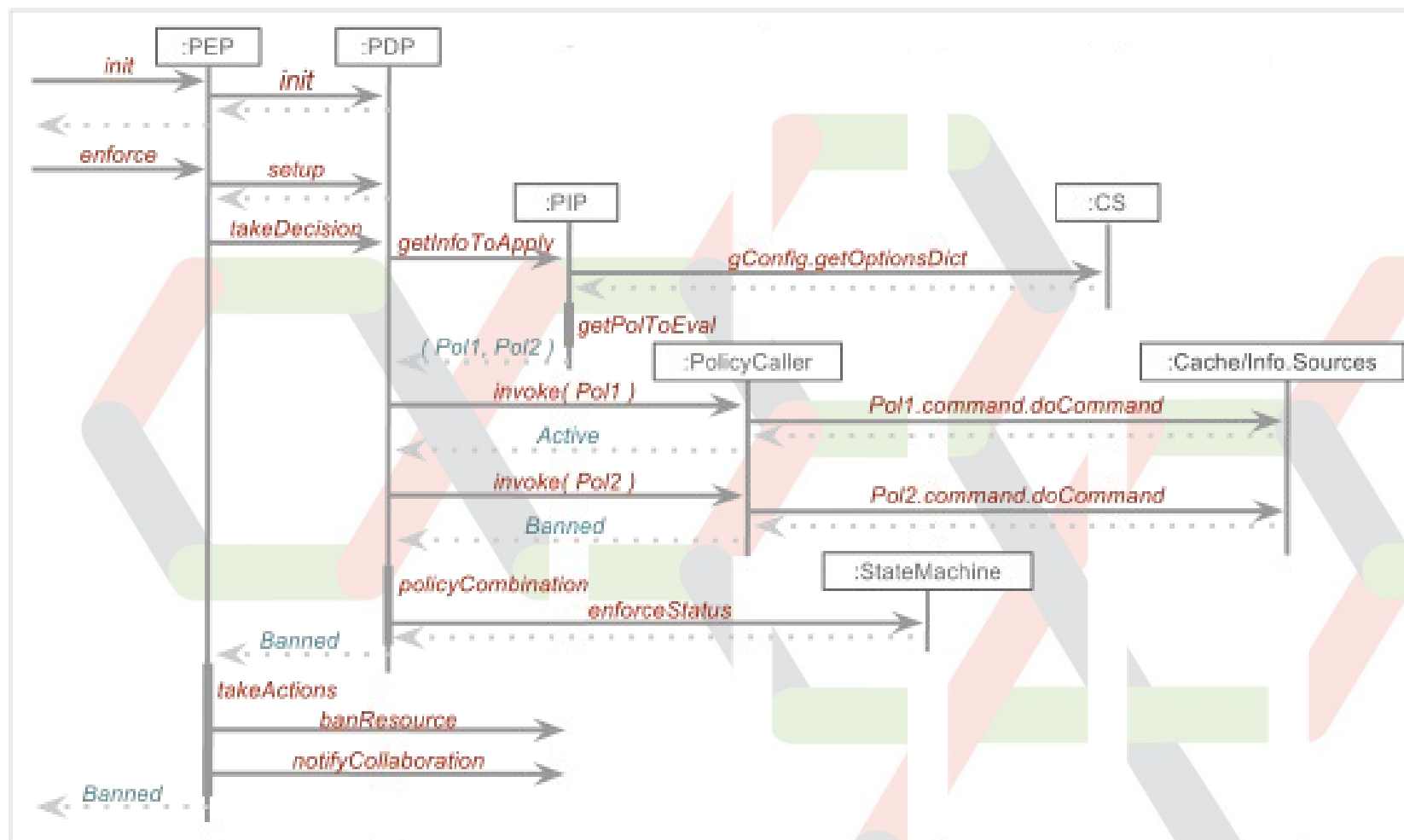
- ▶ A policy system runs the policies: Policy Enforcement/Decision/Information Points
- ▶ A policy is an implementation of a logic rule
- ▶ A policy uses an (aggregated) monitoring information to assess the status of a resource (state machine implemented!)



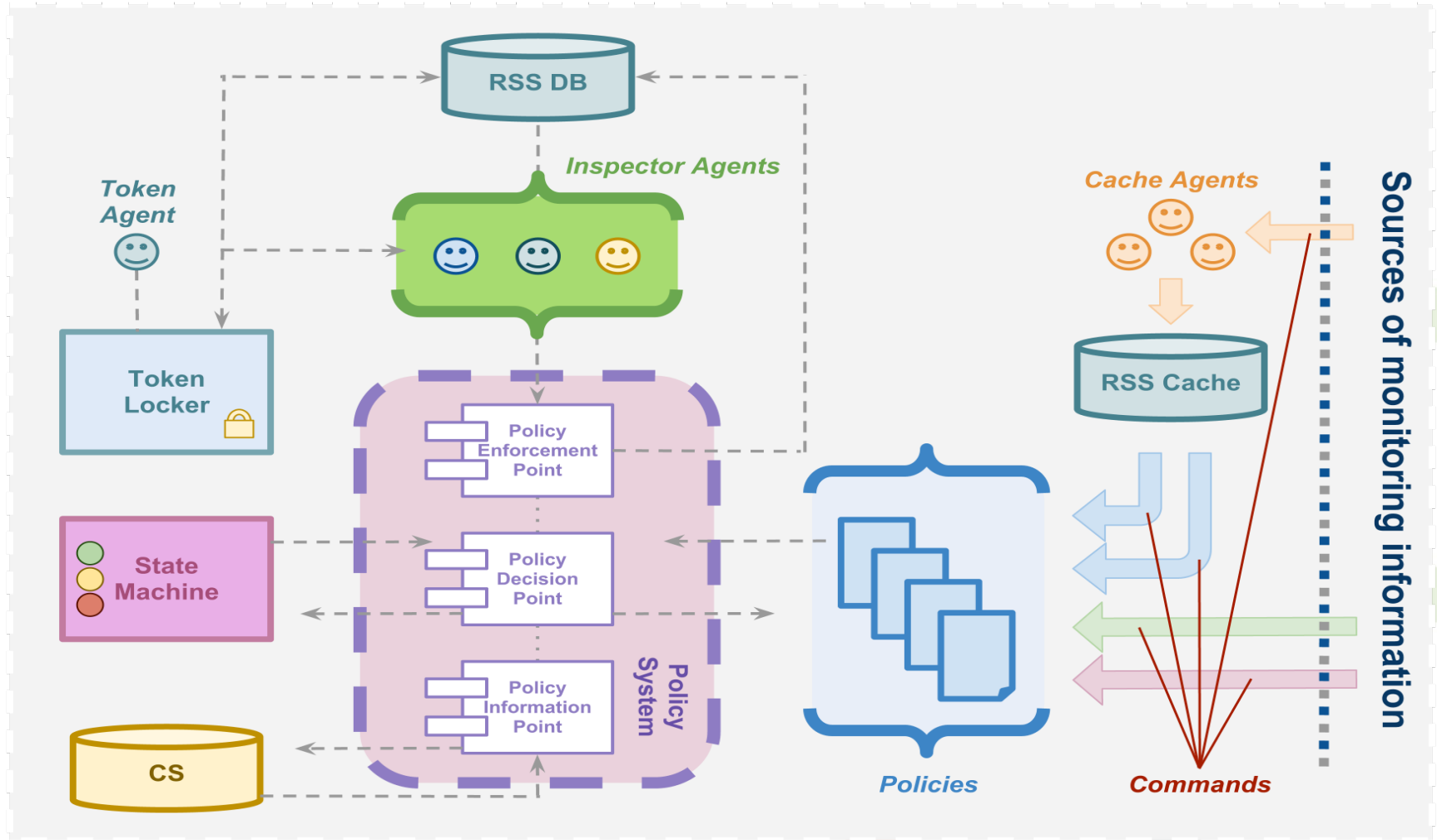
RSS for autonomic management /2

- ▶ Agents
 - ▶ ElementInspectorAgent
 - ▶ TokenAgent
- ▶ And you need the policies:
 - ▶ Most of them will be VO-dependent
 - ▶ Configurable via CS

Policy System



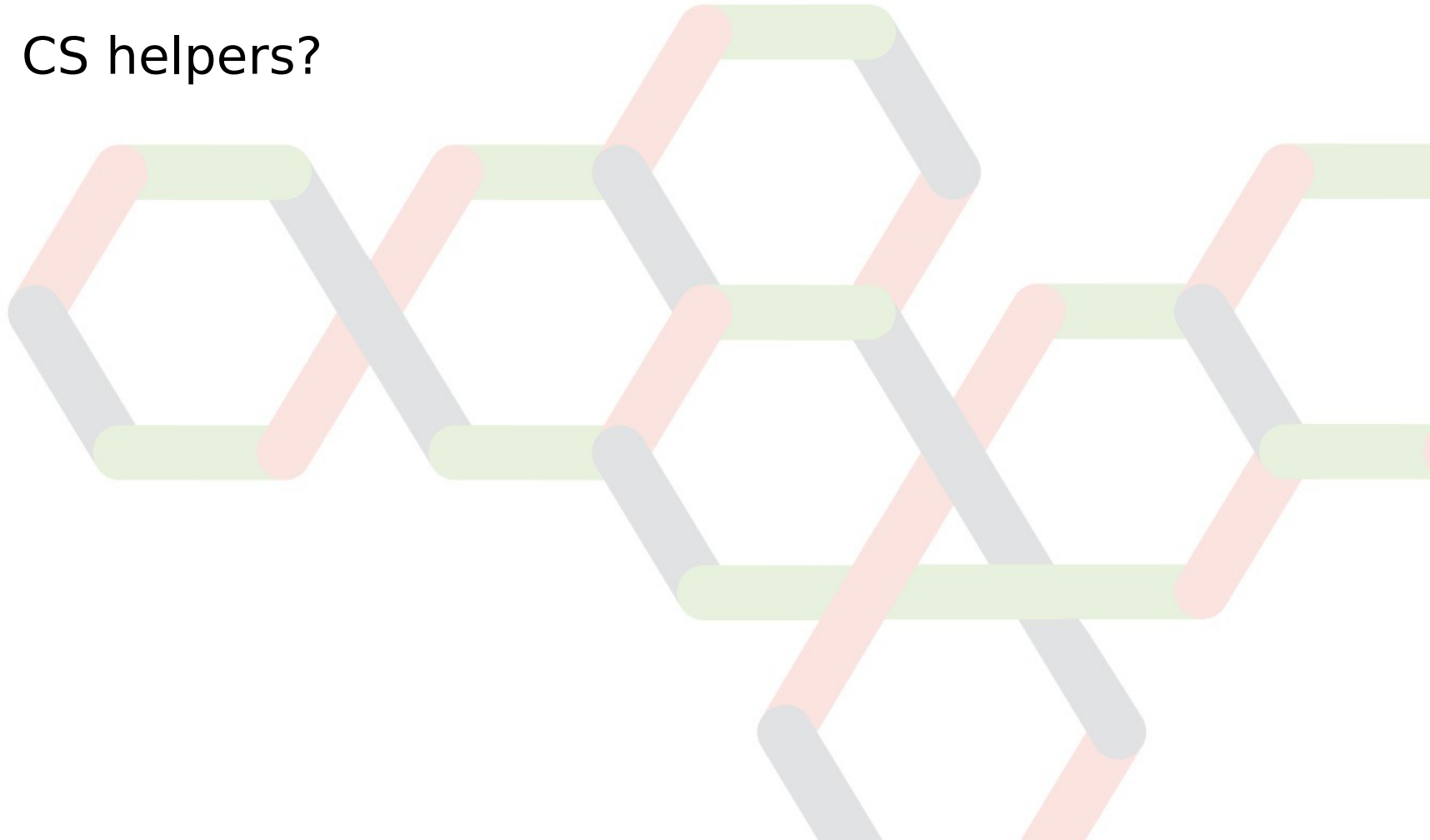
Complete ontology



Nice but...

- ▶ This is not integrated:
 - ▶ <https://github.com/DIRACGrid/DIRAC/pull/929>
 - ▶ DIRAC v6r6 or v6r7?
- ▶ Right now there is code running code with the old schema
 - ▶ Mostly for testing
 - ▶ Not really scalable

- ▶ Migration from old CS schema to new one
 - ▶ CS helpers?



Questions



Further references

- ▶ <http://cdsweb.cern.ch/record/1452204>
- ▶ <https://indico.cern.ch/contributionDisplay.py?contribId=144&sessionId=8&confId=149557>