DIRAC Resource Status System (RSS)

Federico Stagni

Overview

- What's the RSS
    - And why would you need it
- Who use it already
- Ontology and architecture
- How to use it

What's the RSS

DIRAC.ResourceStatusSystem

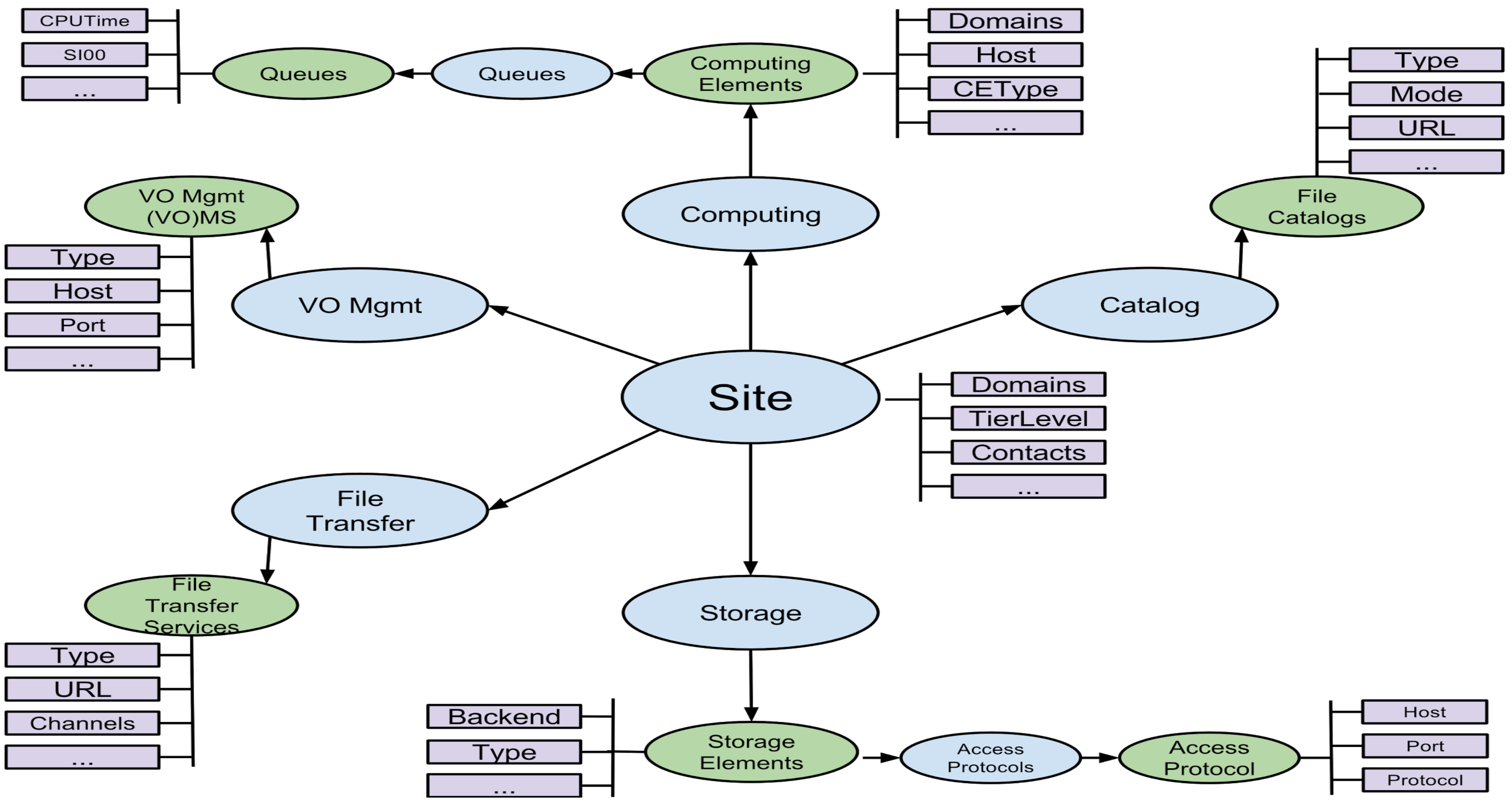
- For storing resource status in DIRAC
  - status information
- An advanced monitoring tool
  - Aggregating dispersed information
- An “autonomic computing” tool
  - The core is a generic policy system
  - Used for monitoring and management
  - Auto ban/un-ban, triggering tests, etc..

Ontology /1

- This RFC defines how the /Resources section of CS should be, and the resources ontology at the base of RSS
- Key concepts:
  - Community (VO)
  - Site (access point → locality!)
  - Domain (WLCG, Gisela, EGI...)
  - Resource Type (Computing, Storage, Catalog, FileTransfer, Database, CommunityManagement)

/Resources/Sites/[SiteName]/[ResourceType]/[Name Of Service]/[TypeOfAccessPoint]/[NameOf AccessPoint]

/Resources/Domains/[Domain Name]


CPUTime
SI00
...
Queues
Queues
Computing Elements
Domains
Host
CEType
...
Type
Mode
URL
...
VO Mgmt (VO)MS
Computing
File Catalogs
Type
Host
Port
...
VO Mgmt
Catalog
Site
Domains
TierLevel
Contacts
...
File Transfer
File Transfer Services
Storage
Type
URL
Channels
...
Backend
Type
...
Storage Elements
Access Protocols
Access Protocol
Host
Port
Protocol

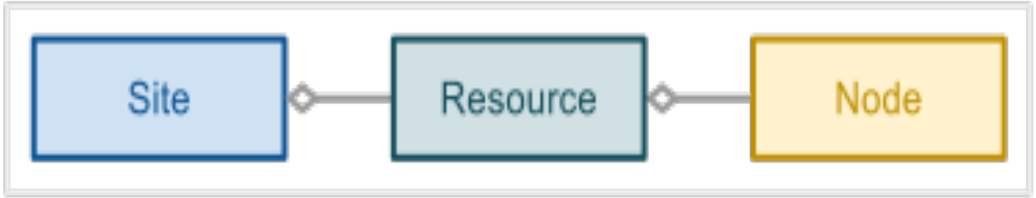
In the RSS

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

→ Sites

→ Resource

→ 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: Status Summary page (all “resources” combined)

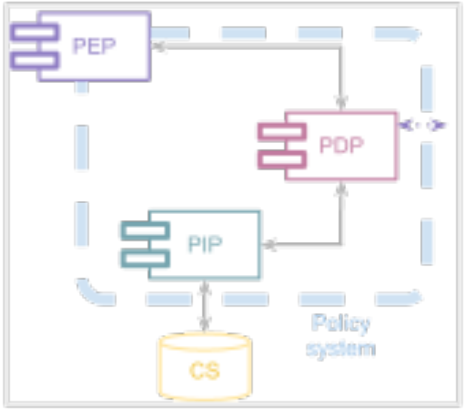
RSS for advanced monitoring

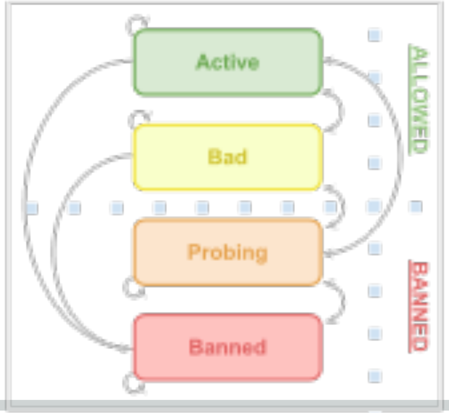
- DB: ResourceManagementDB
- Service: ResourceManagementHandler (mostly exposes the cached monitoring information)
- Agents: CacheFeederAgent: populates a cache of (useful, configurable, VO-specific) monitoring information
  - e.g.: downtimes, failure rates, external monitoring results …
    - Use “commands”: implementation of the Command pattern → not yet clients!
      - Downtimes, accounting, jobs, transfers, space token occupancy...

- Web (cached info are displayed)

RSS for autonomic management /1

- A policy system runs the policies: PolicyEnforcement/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 (based on the state machine)





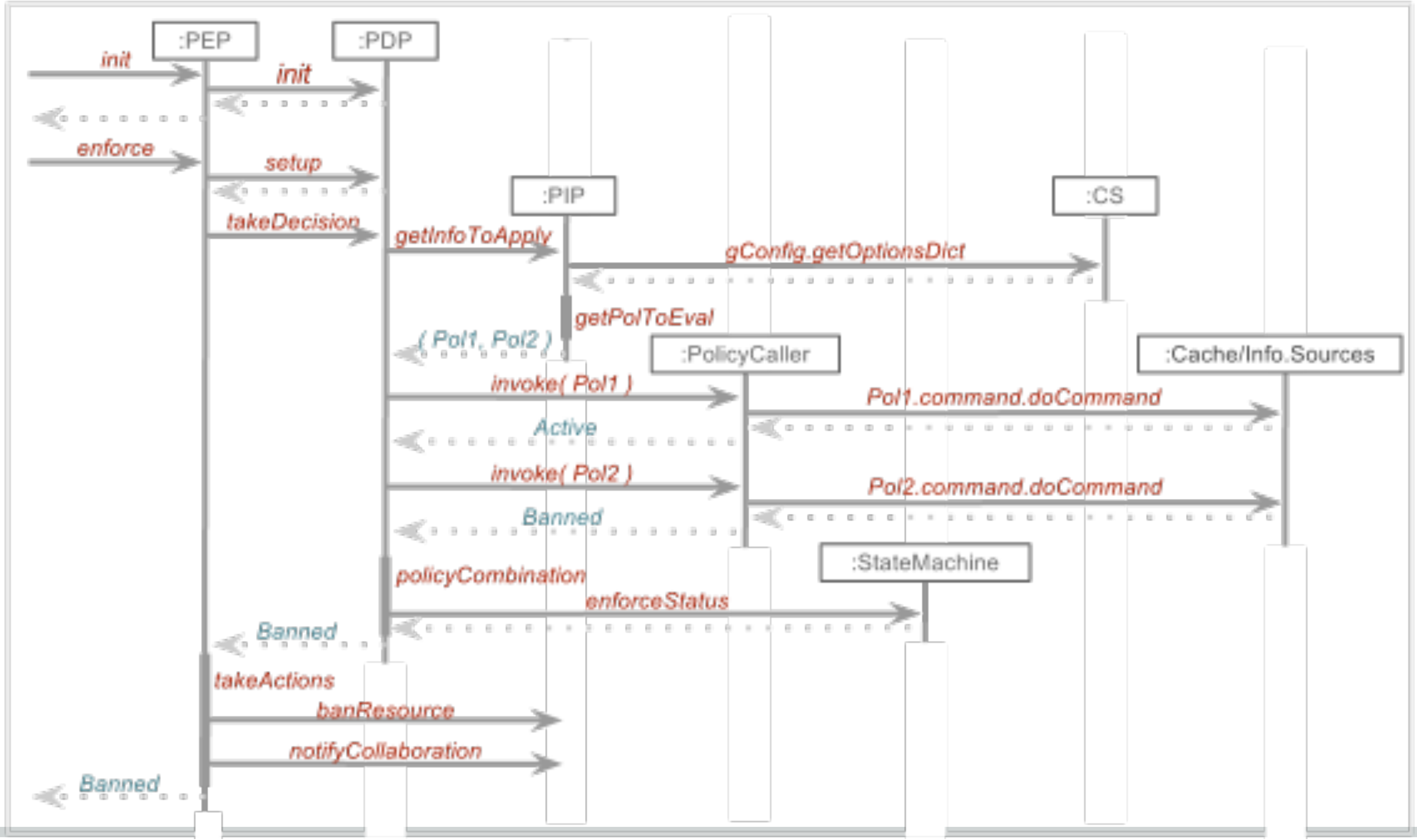
DIRAC
THE INTERWARE

RSS for autonomic management /2

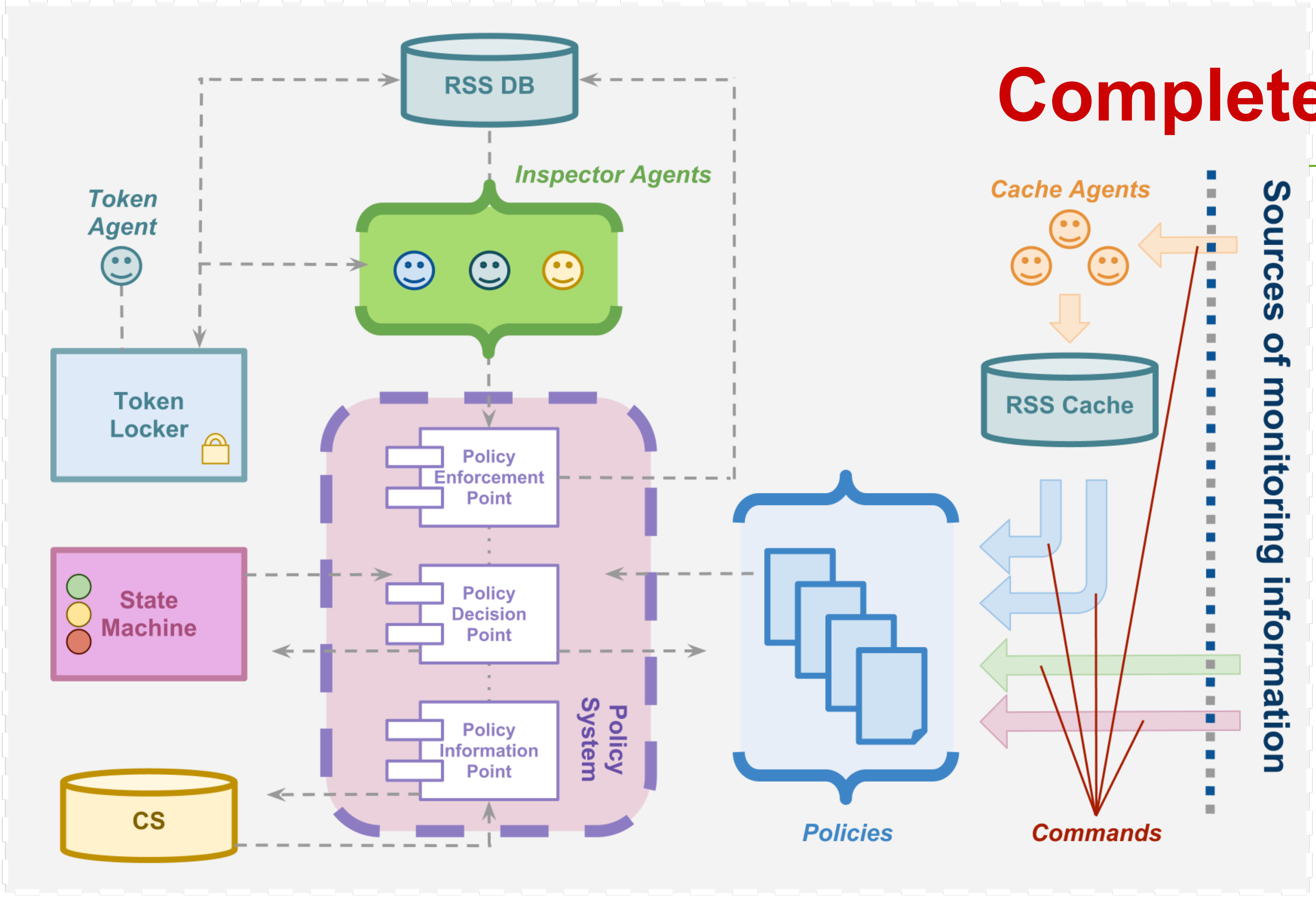
- Agents
  - ElementInspectorAgent
  - TokenAgent
- And you need the policies:
  - Most of them will be VO-dependent
  - Configurable via CS

DIRAC
THE INTERWARE

Policy System



Complete ontology



Questions/comments

?