8th DUW

# DIRAC Resource Status System (RSS)

Federico Stagni

# What's the RSS

## DIRAC.ResourceStatusSystem

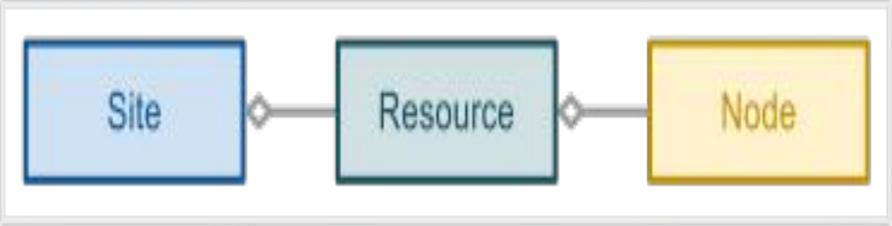
docs!

1) For storing resource status in DIRAC
   - status information
2) An advanced monitoring tool
   - Aggregating dispersed information
3) An “autonomic computing” tool (a small AI - no learning nor training)
   - The core is a generic policy system
   - Used for monitoring and management
   - Auto ban/un-ban, triggering tests, etc..

as of today, RSS is not capable of handling multi-VO resources status

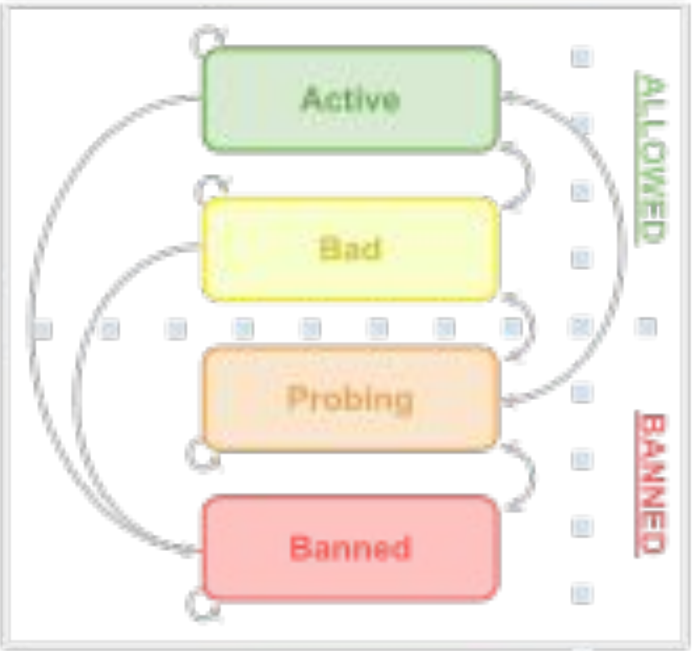
# Which “resources”?

- Everything under /Resources in CS
  - ...with caveats :)
- So the distributed computing resources:
  - Computing Elements
    - which (may) expose queues
  - Storage Elements
    - in DIRAC sense
  - FTS servers
  - Catalogs
  - Sites
    - which in fact “own” the resources above



# And what’s a “status”?

- A status is either
  - assigned (by an admin)
  - determined (by RSS)
- Each resource can have 4 states
  - Active: all OK
  - Bad: it’s not all OK but we can keep use (~Warning)
  - Banned: not OK, use is suppressed
  - Probing: Test state (normal use is suppressed)



# OK, but what can RSS do in practice?

**Stuff like:**

- Keeping/exposing a status of all your resources
  - trivial…
  - but that status will be used for deciding if to use that resource!
    - e.g. uploading file to a certain SE or not?
    - submitting pilots to a CE or not?
- Site_X / SRM_endpoint_Y is declared in downtime in GOC DB
  - then RSS can ban all the DIRAC SEs behind SRM_endpoint_Y
- Many pilots are failing to be submitted at CE_Z
  - Then I'll change its status
- ...

# Activation

- By default, RSS is “InActive”
  - how to activate (CS flag)
- You can activate only PARTIALLY the RSS
  - … so only 1), or 1) and 2) of slide 2...
  - If you want RSS to keep at least the status of the resources (point 1) you just need “State = Active”
  - For all the rest, there’s a lot of configuration that can be applied

# RSS for status information

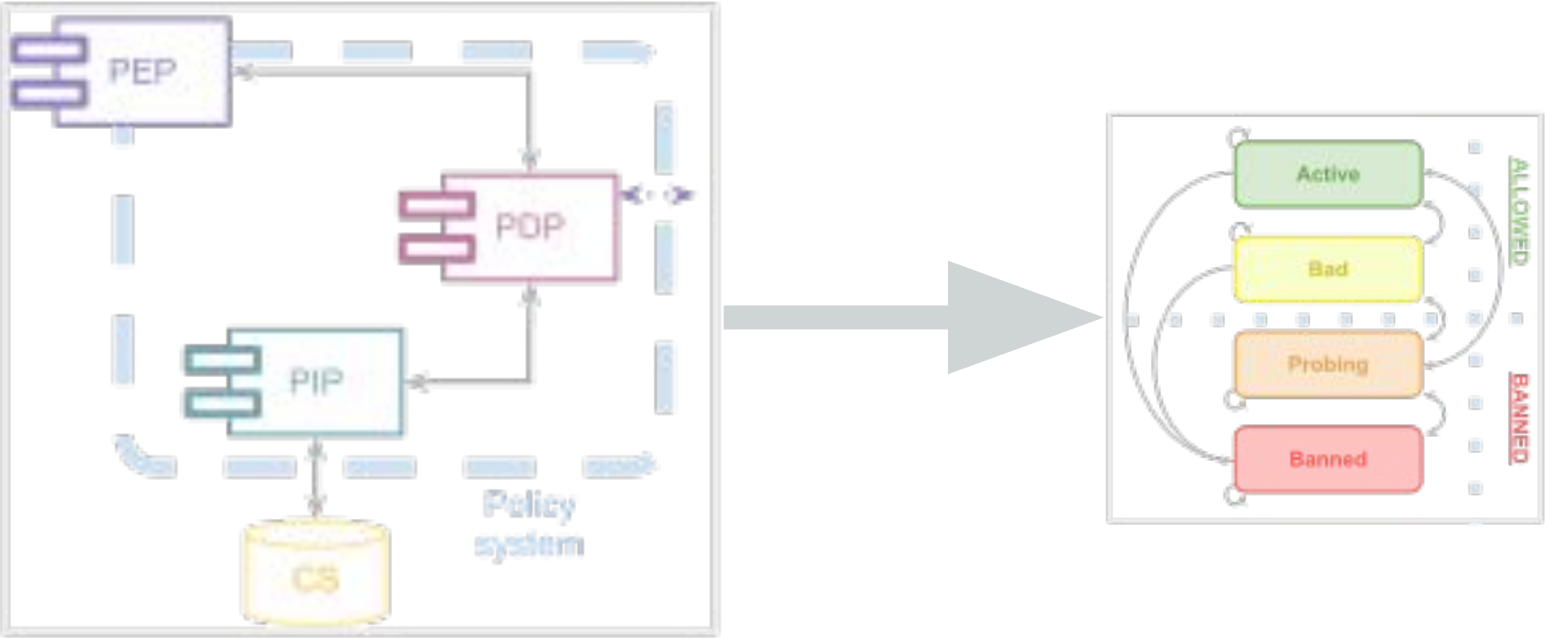
- DB:
  - ResourceStatusDB: tables for: Status, Log, History
    - Status: 3 families of identical tables: Site, Resource, Node
    - Log: mostly for debugging purposes
    - History: keeps historical changes of status
- Service
  - ResourceStatusHandler (expose ResourceStatusDB)
- Client
  - ResourceStatusClient: for interacting with the ResourceStatusDB
  - ResourceStatus: object that keeps the connectivity with the DB/Service – refreshing DictCache of SEs/CEs status
  - SiteStatus: object that keeps the connectivity with the DB/Service – refreshing DictCache of Sites 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

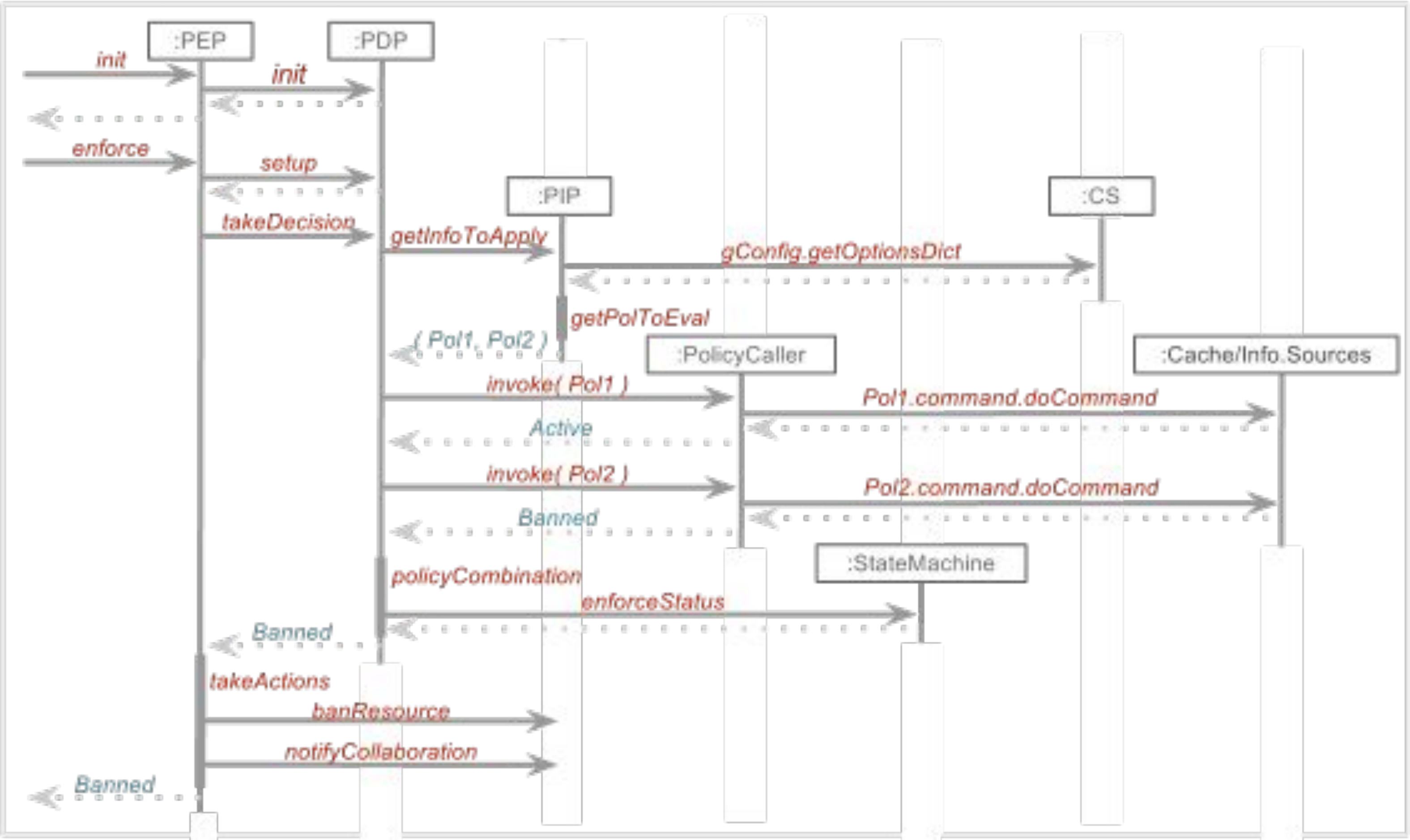
- A policy system runs the policies: PolicyEnforcement/Decision/Information Points
- A policy is an implementation of a logic rule
- A policy uses an (aggregated) monitoring information to assess the status of a resource (based on the state machine)



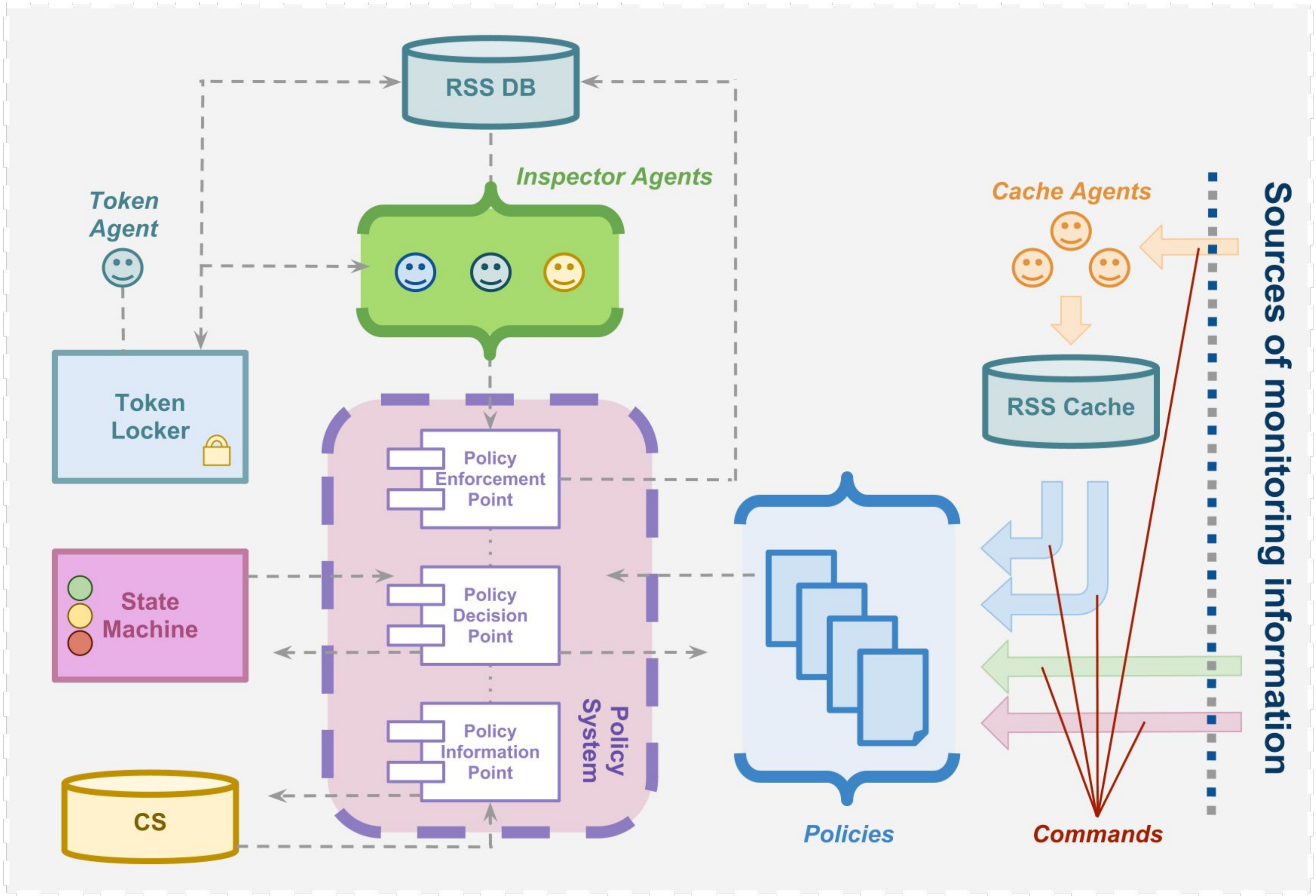
# RSS for autonomic management /2

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

# Policy System



# Complete ontology



# RSS in v6r19 and before

- Several RSS developments have been included in DIRAC v6r19
    - Ability to store status of Computing Elements
    - Ability to store status of FTS and Catalogs
    - Ability to store status of Sites
        - Replacing table SiteMask from JobDB
            - Again, IFF RSS is “Active”

# Next developments for RSS

- Multi-VO support
  - Needs a developer :)
    - won’t come from LHCb
  - And a testing/certification infrastructure
- Support for resources hierarchy
  - Banning a Site bans also all the related Resources
  - Banning a Computing Element bans all the related Queues, etc
  - Reactivation of the Site is restoring the Resources status at the moment of banning

# Questions/comments

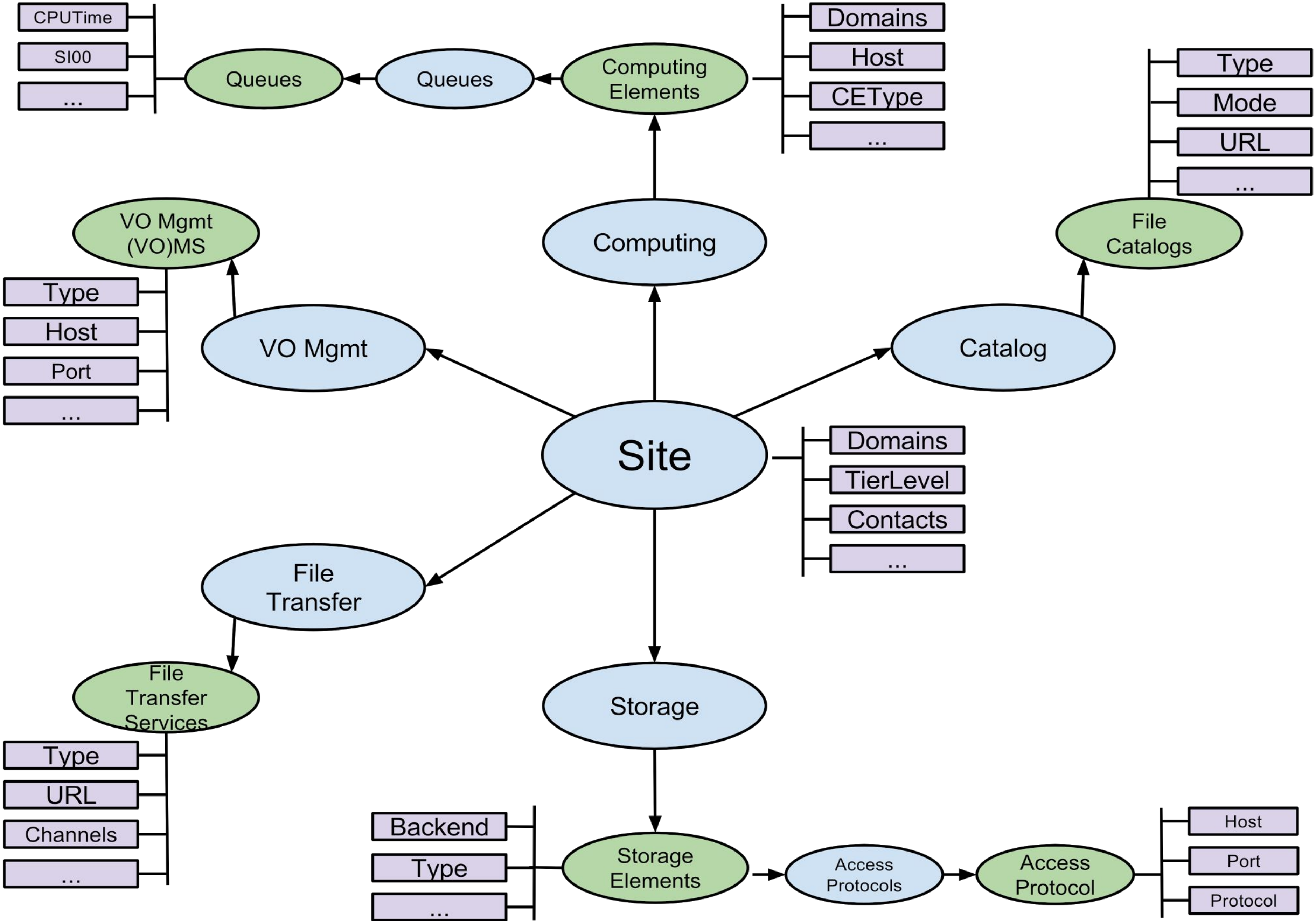
?

# 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
...
File Catalogs
VO Mgmt (VO)MS
Computing
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