Architectural Enhancements to the ONAP Policy Framework for Casablanca

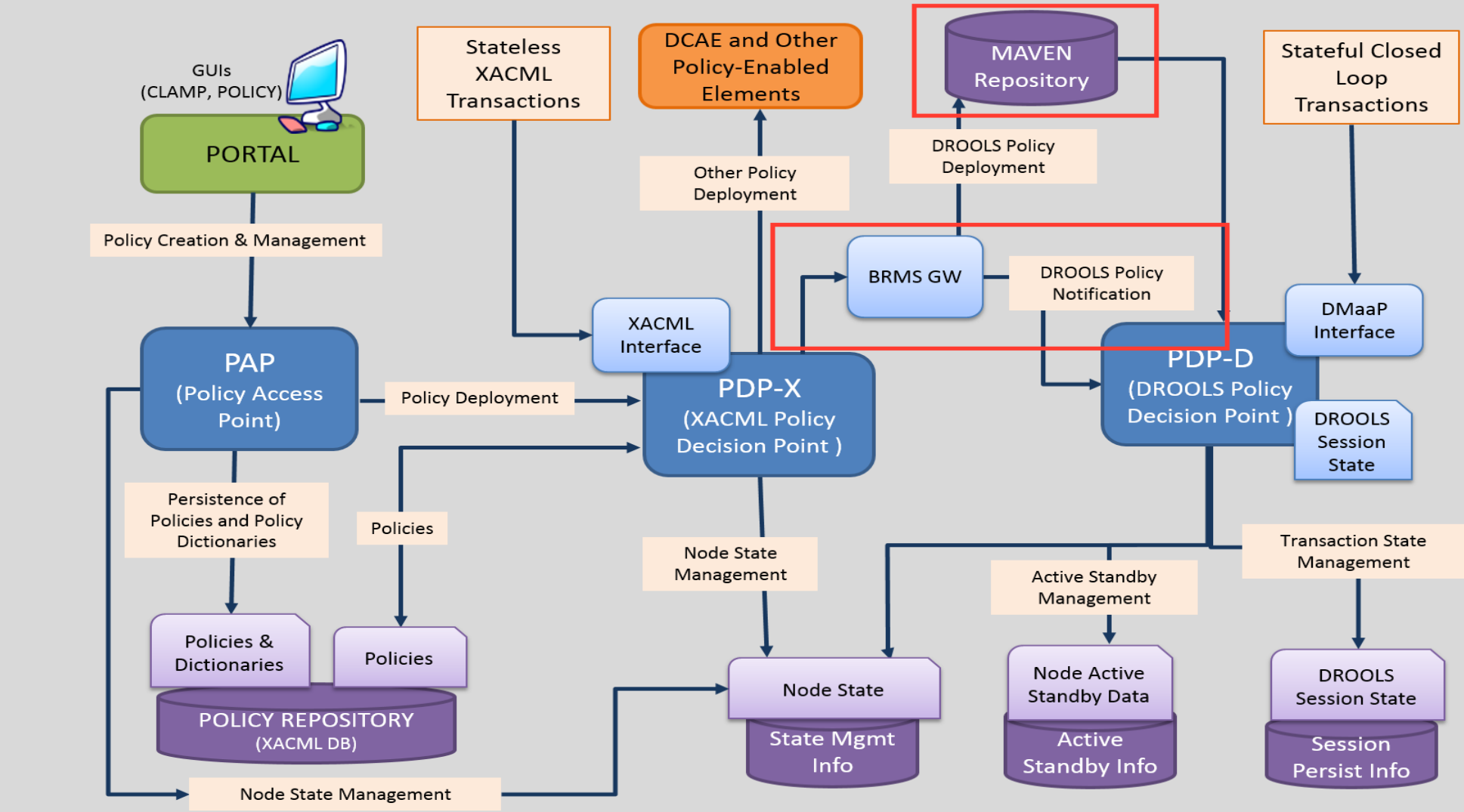
Pamela Dragosh
Liam Fallon

5th June, 2018

Highlights

- Simplification of the structure of the ONAP PF
  - More straightforward to develop policies
  - Microservice oriented, PDP is the main unit of scalability
- Generic model driven support for policies of all types
  - Arbitrary PDPs are supported
  - Policy design and deployment is the same for all policies
- Policy Lifecycle Management
  - Passive/Test/Safe/Active modes supported
  - Policy Upgrade and Rollback
- Unified and Controlled Policy execution
  - All policy execution performed in PDPs
  - Synchronous and asynchronous execution supported
  - Lay a stronger foundation for conflict detection, mitigation, and prevention

Current Architecture



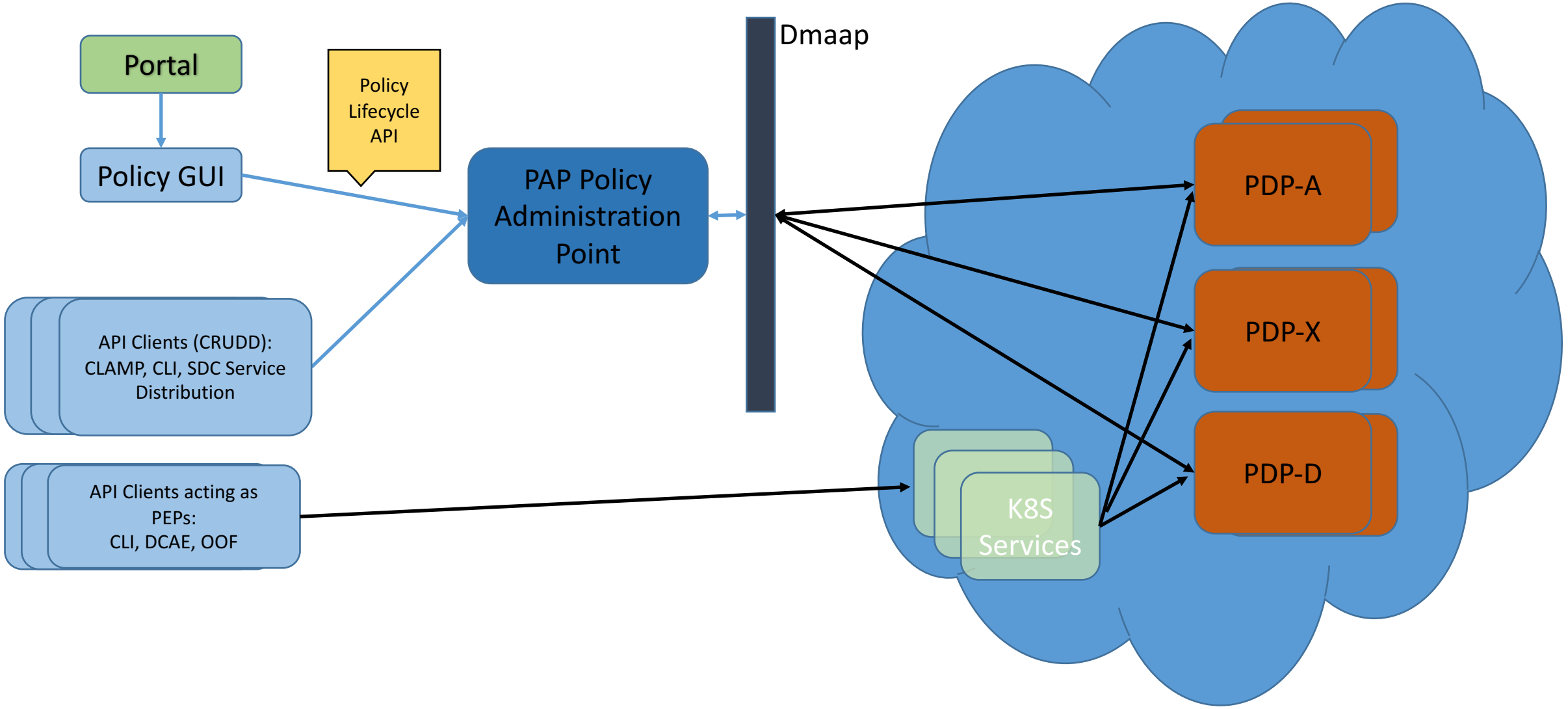
9/28/2017

Figure 3: ONAP POLICY CREATION AND DEPLOYMENT DETAILS

Current Architectural Limitations

- Policies that are specific to drools are routed through the PDP-X via BRMSGW. This is unsustainable long-term as we move to a distributed PDP micro service architecture and need to utilize a Policy Distribution API to groups of PDP's spread out over the installation of ONAP.
- BRMSGW hard codes dependencies for policies that are specific to drools. The long term architecture view is for policy artifacts to be located in a nexus repository that are pulled by appropriate PDP micro service engines. Dependencies would be set in the policy artifact jar files themselves and not hard coded into the platform as a JSON file. Having a JSON file to specify dependencies is a poor implementation architecture and there is no need to re-create Policy artifacts.
- Policy template/model/rule development environment is too tightly integrated with the Platform code and is not sustainable in the long term. The long term vision is for separate of Platform from Policies and the current architecture is too far off in its development to move to this goal.
- New policy models require too much development effort for integration with the Policy GUI. The long term goal was a platform that could ingest policy models organically and not require any development work. The Policy GUI should be to interpret any TOSCA Model ingested and flexibly present a GUI for a user to create policies from.
- Policy API is not RESTful and acts more like database query rather than request for a Policy Decision. API also hardcodes types of policies that can be created which is not the long term goal for the architecture which is to be able to create domains for policies and flexible import models for those domains.
- Policy API does not support Lifecycle management for policies that are deployed such as modes and retirement. This additional requirement is better suited for the new proposed Policy Lifecycle API than the current API.

Target Architecture (WIP)



Full Details

See https://wiki.onap.org/display/DW/The+ONAP+Policy+Framework