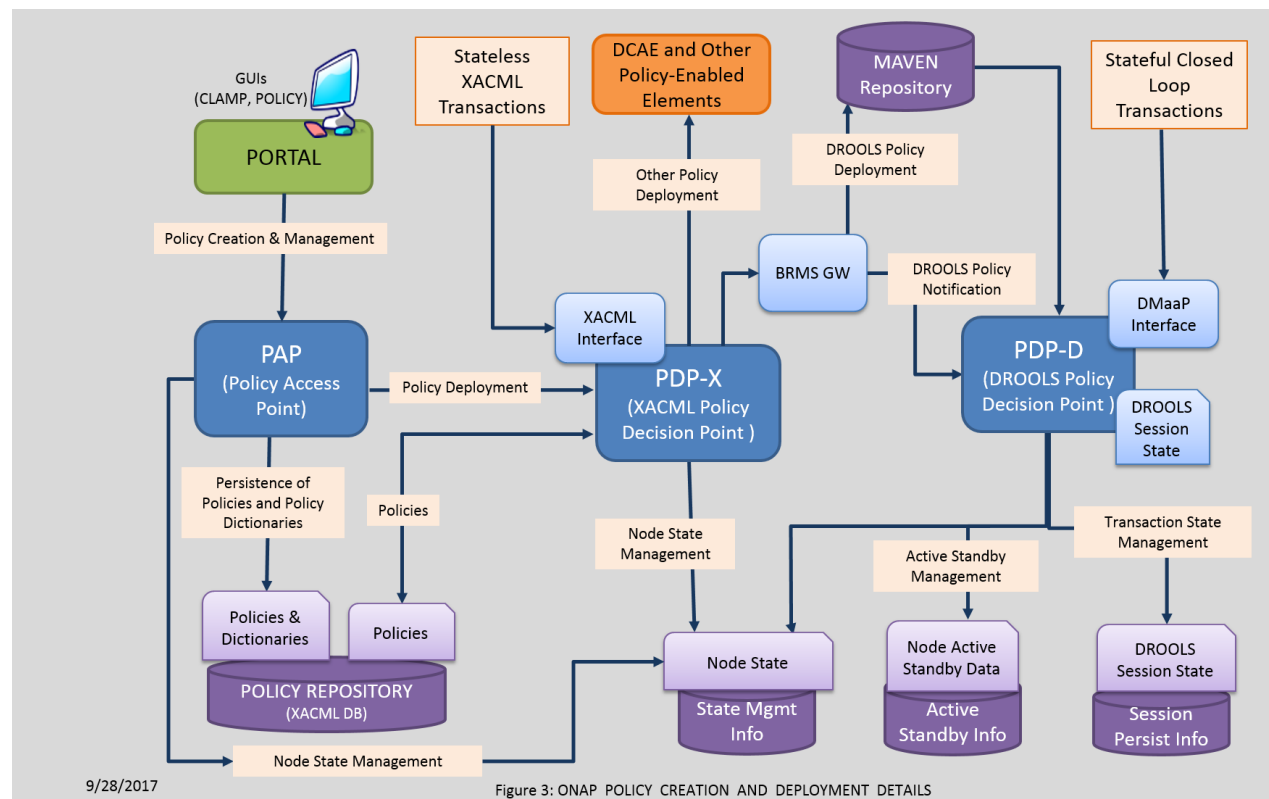


The ONAP Policy Framework: Architectural Improvements for Dublin

The desired ONAP Policy Framework is described on [The ONAP Policy Framework](#) page. This page describes the architectural improvements that will be undertaken in the Dublin release of ONAP as the ONAP Policy Framework evolves towards its desired architecture.

Current Architecture

The seed architecture of the Policy Framework is described on the wiki page [Policy Seed Code Architecture](#) and the current architecture is located [here](#).




Current Architecture 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.

In addition, please see this [Control Loop Operation and Improvements](#). for more details on the problems and suggested improvements for operational policies to be addressed in Dublin. Some of this work will be fixed in the current Seed Code architecture as the Drools PDP will be able to support both architectures in Dublin.

Work Proposal to Fix Current Architecture Reliability Issues with policy CRUD and deployment

Casablanca stability testing for SDC Service Distribution exposed the following erratic behavior of the PDP engine with respect to policy CRUD:

 **POLICY-1277** - policy config takes too long time to become retrievable in PDP CLOSED

This is a list of work proposal to help fix the reliability of the Policy Engine/API component piece of the Policy Seed Code Architecture for Dublin required Medium-sized T-Shirt effort:

- Database layer changes to deal with failures and timeouts
 - The code doesn't handle DB layer failure well, we need to inspect and enhance the code for all DB interactions.
- Network layer changes to deal with failures and timeouts
 - Same is true for communication between components.
- Consistency checking code will be added and invoked at certain intervals and on demand
 - Need to be able to audit that all components are in sync on the fly.
- Retry logic will be inserted where applicable
 - Retry code is non-existent for some DB and component interactions, need to add.
- Tier2 alerting will be inserted where applicable
 - Integration from logfiles to internal systems.
- Health checks will be instituted to monitor component health
 - Health checks are basic, add more checks where possible.
- Deployment processes will be hardened to ensure environment-specific configurations and properties files are deployed
 - Need post-deployment auditing scripts created and run automatically on deployments.
- Improve the policy creation process that removes the need to update every time you push a policy and also does not require repackaging of the rules jar as it just inserts facts in to the working memory. It takes a 9 step process down to about 2 steps.
 - Will be removed, covered by ONAP changes.
- Recovery tools to reduce the outage interval (stretch goal)
 - Tooling to help aid in recovery which is manual today.

However, what this work doesn't do is fix Drools PDP to become a first class citizen as a PDP nor does it integrate new Apex PDP. Goal for Policy Framework is support any PDP desired, not just the ones supported by ONAP.