Policy Lifecycle API

Pamela Dragosh – AT&T

December 13, 2017

Abstract

This presentation will be an introduction to the upcoming release of the Policy Lifecycle API for Beijing. The Policy Lifecycle API is being introduced to support a richer lifecycle for creating, updating, deleting, querying and deploying policy during design-time and runtime. In addition, the API will be designed and developed using ONAP RESTful API best practices.

For design-time, new features will be added to support a lifecycle that mirrors current SDC functionality such as: check-in/check-out, submission for certification, and lastly submission for distribution. These new features will move the policy API one step closer to: 1) integration into the SDC Design-time environment, 2) ability to develop and integrate a Policy Testing/Certification platform that can support flexible environment for doing conflict detection/resolution.

For run-time, new features will be added to support runtime deployment of policies that support different modes of operation such as deploying policies in different modes: active, inactive, and safe mode. Platform operations team will have the ability to easily deploy and un-deploy policies to/from the desired mode. In addition, functionality will be added to give the ability for an operations team to retire selected policies at a given date.

We would like to use this session to present an overview of the Lifecycle API and garner feedback from the community on the design.

Disclaimer

- Policy Lifecycle API is a draft, work-in-progress feature

Goals and Motivation

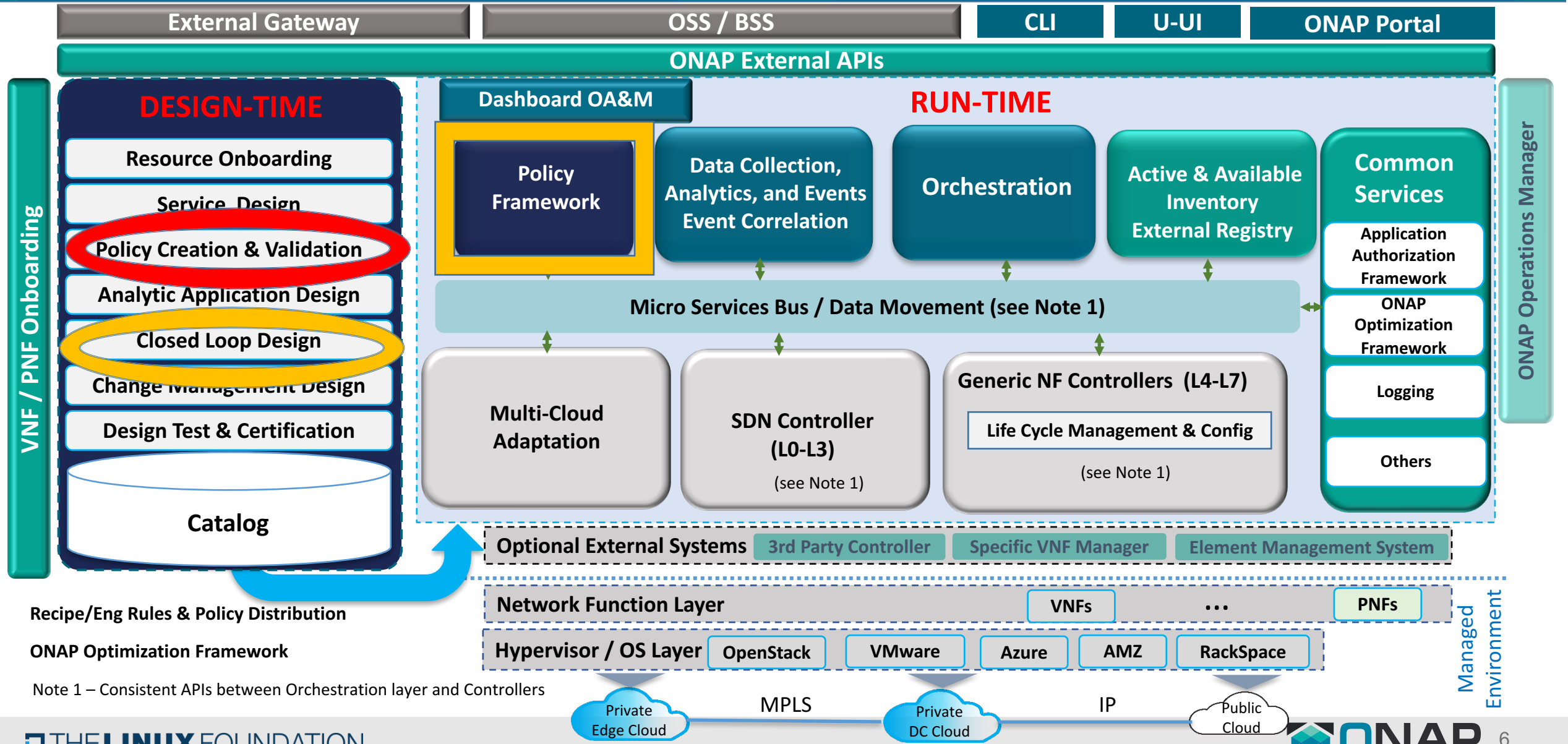
- Goal for the Policy Lifecycle API is gets the Policy Framework Platform a step closer to support Policy Driven self-serve needs of running instance(s) of the ONAP Platform.
- Motivation is to better utilize Models to assist the Policy Framework in driving the ONAP Platform.
- A clear and concise API can go a long way towards implementing a zero-touch platform.

Current State

- Current API supports policies using config, decision and brms models.
  - Models for Control Loop Config and Operational Policies
  - Models to support Optimization, generic Drools rules, etc.
- Does not conform to RESTful best practices
- Not optimally a model-driven platform
- Not integrated with SDC
  - Eg: Want the ability to express and capture VNF homing and placement policies during Service Design
- Deployment “mode” is not supported
  - Active, Inactive, Safe

ONAP Target Architecture for R2 and Beyond

(High-Level Functional View)



User Experience

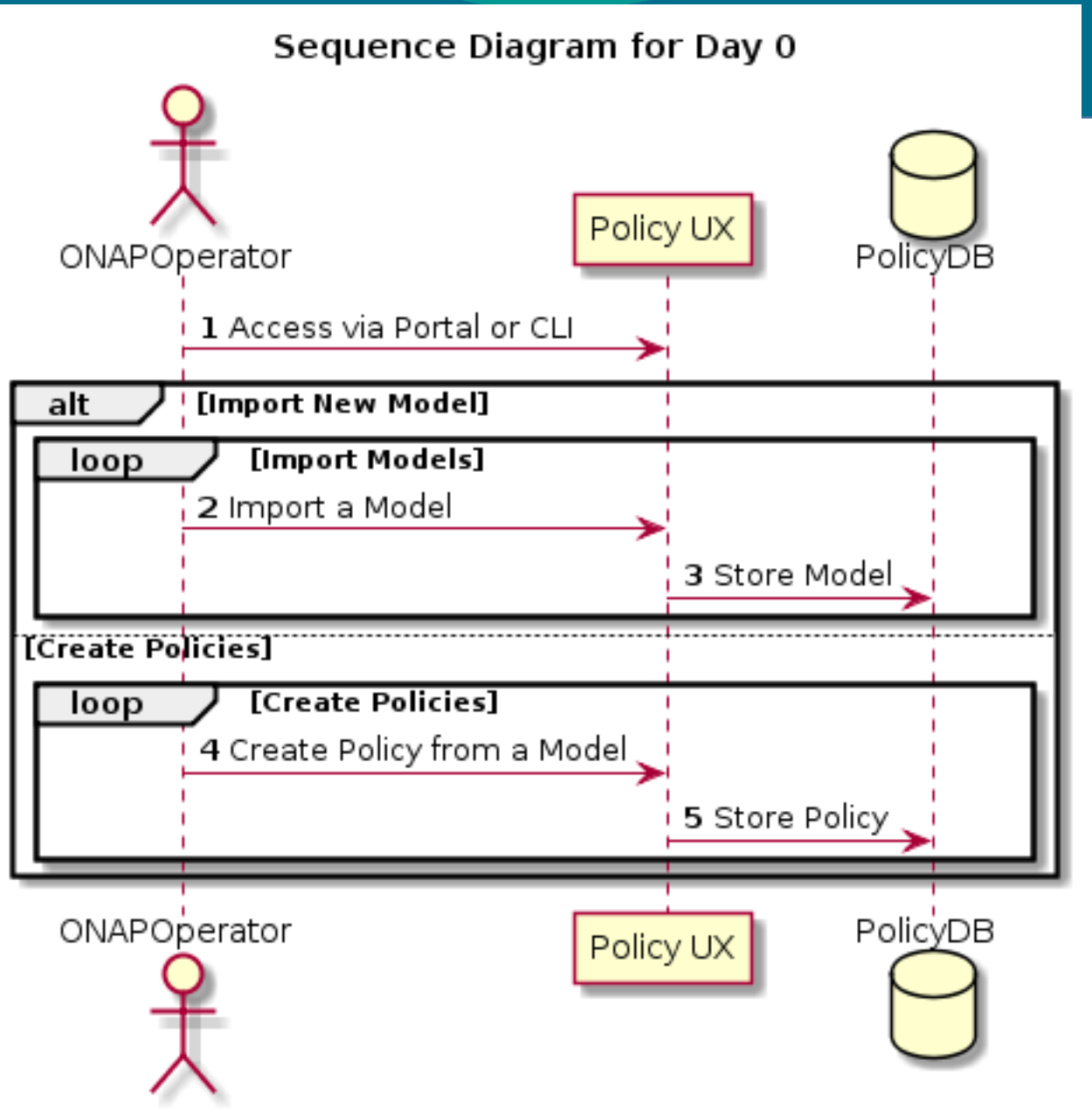
- User experience is important, the Policy Lifecycle API should adhere to RESTful best practices as recommended by the ONAP Developer Best Practices.
- RESTful Path Design is important to allow ONAP Operators be able to rapidly perform CRUD on Policies for different domains, models, etc.
- API can be developed in a way such that a Policy Design GUI can easily be built to drive the API, as well as other components in ONAP (eg. CLAMP)

User Experience - Day 0

- Starting from an initial minimal instance of an ONAP Platform, an ONAP Operator must be able to easily create and administer policies BEFORE any virtual resource/service is on-boarded.
  - Infrastructure Policies
  - Platform component Policies
  - Identity Policies
  - Guard Policies
- How should the user experience for this be?

User Experience - Day 0

- Policy UX has 2 options:
  - Policy GUI via Portal
  - Interactive Command Line Interface
- Either UX use the same Policy Lifecycle API



API path design options

Start from the perspective of models, in which the goal is for the Policy Framework to be dynamically model-driven.

/models
/models/{model-id}
/models/{model-id}/policies
/models/{model-id}/policies/{policy-id}

Storage would be directly into the Policy Platform Database itself.

API path design options

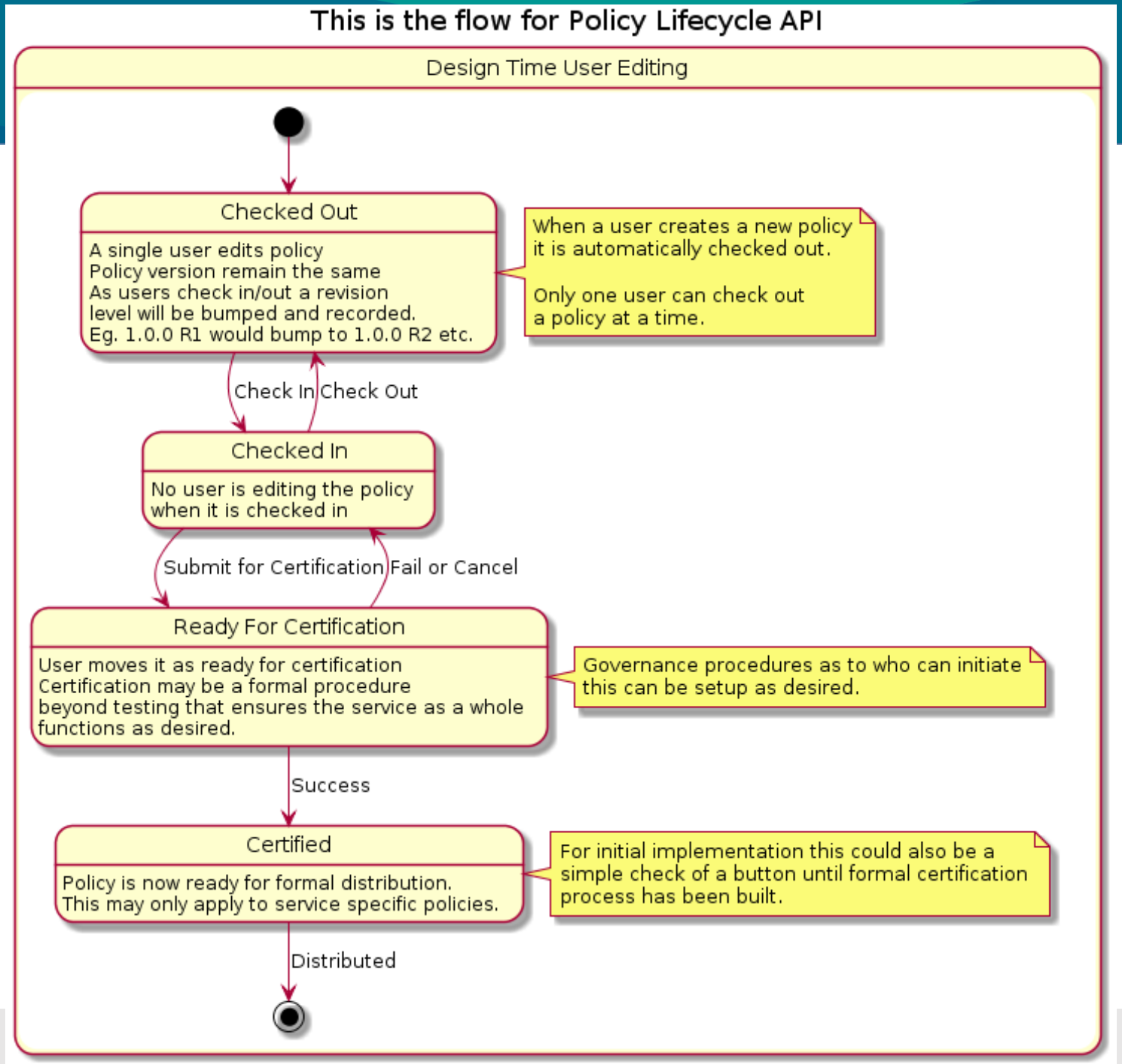
Start from the perspective of domains: governance, ONAP component, identity. Models are already defined and developed per ONAP Release, but the flexibility remains to dynamically create new domain models.

/domain
/domain/{domain-id}
/domain/{domain-id}/models
/domain/{domain-id}/models/{model-id}
/domain/{domain-id}/models/{model-id}/policies
/domain/{domain-id}/models/{model-id}/policies/{policy-id}

Storage would be directly into the Policy Platform Database itself.

Lifecycle Flow Diagram

- For Policies NOT created during SDC Design, there may be governance required to ensure policies injected into the platform meet certification/testing standards.
- Open for discussion as to whether this level of functionality would be necessary for the Policy Framework

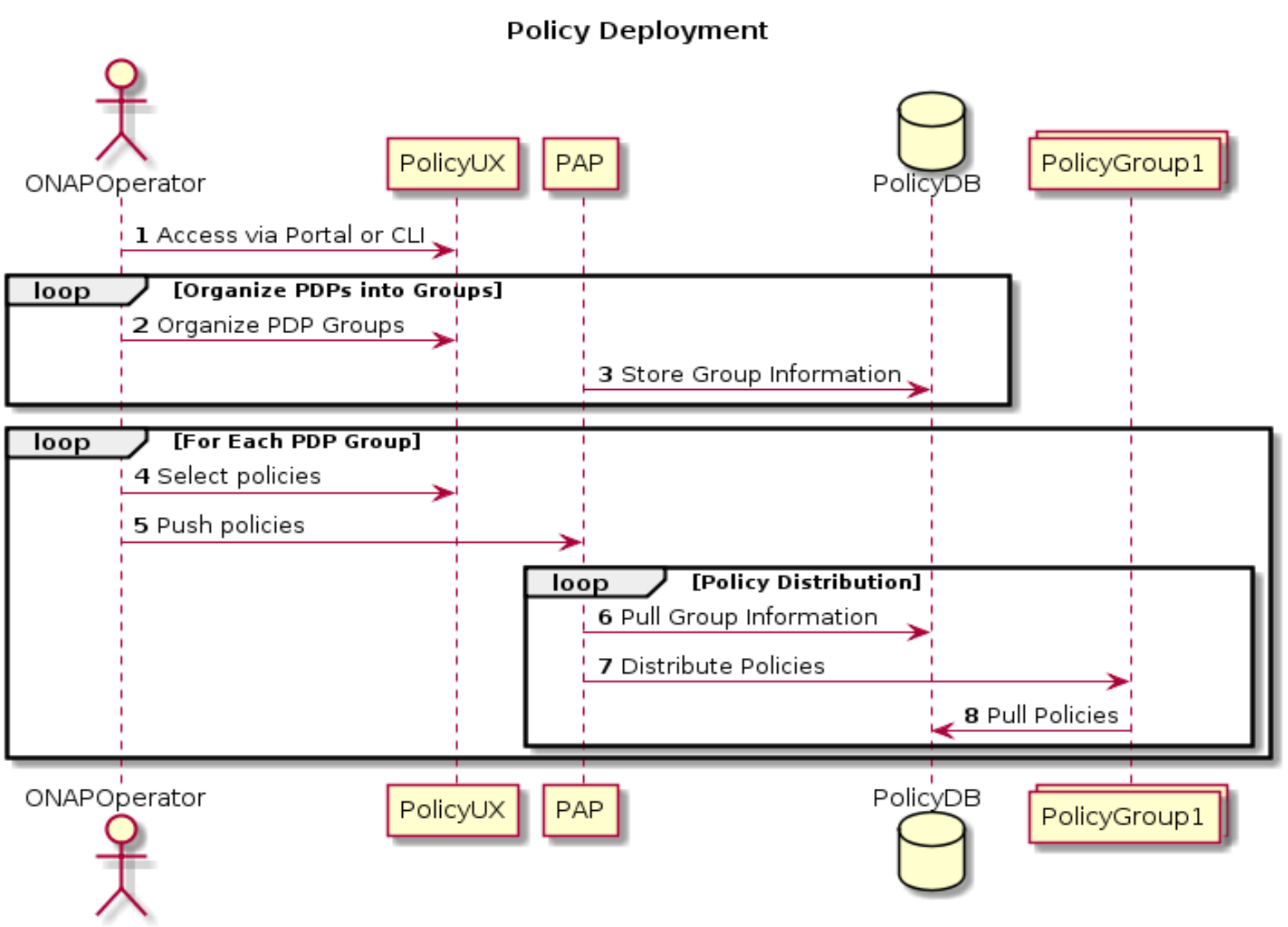


Runtime Policy Deployment

- Policy Lifecycle API should support Deployment of Policy to Groups of PDP's
- Day 0 there would be an initial deployment of Policy PDP engines to support Platform Level Policies
- Policy Lifecycle API would support the ability to group the PDP engines to support desired subsets of policies

Runtime Policy Deployment

- A Policy PDP Group could be a combination of different PDP Engines or could consist of a set of the same PDP engines.



API Path design options

/groups/

/groups/{group-id}

/groups/{group-id}/policies

/groups/{group-id}/policies/{policy-id}

/groups/{group-id}/pdps

/groups/{group-id}/pdps/{pdp-id}

/groups/{group-id}/pdps/{pdp-id}/policies

/groups/{group-id}/pdps/{pdp-id}/policies/{policy-id}

Next steps

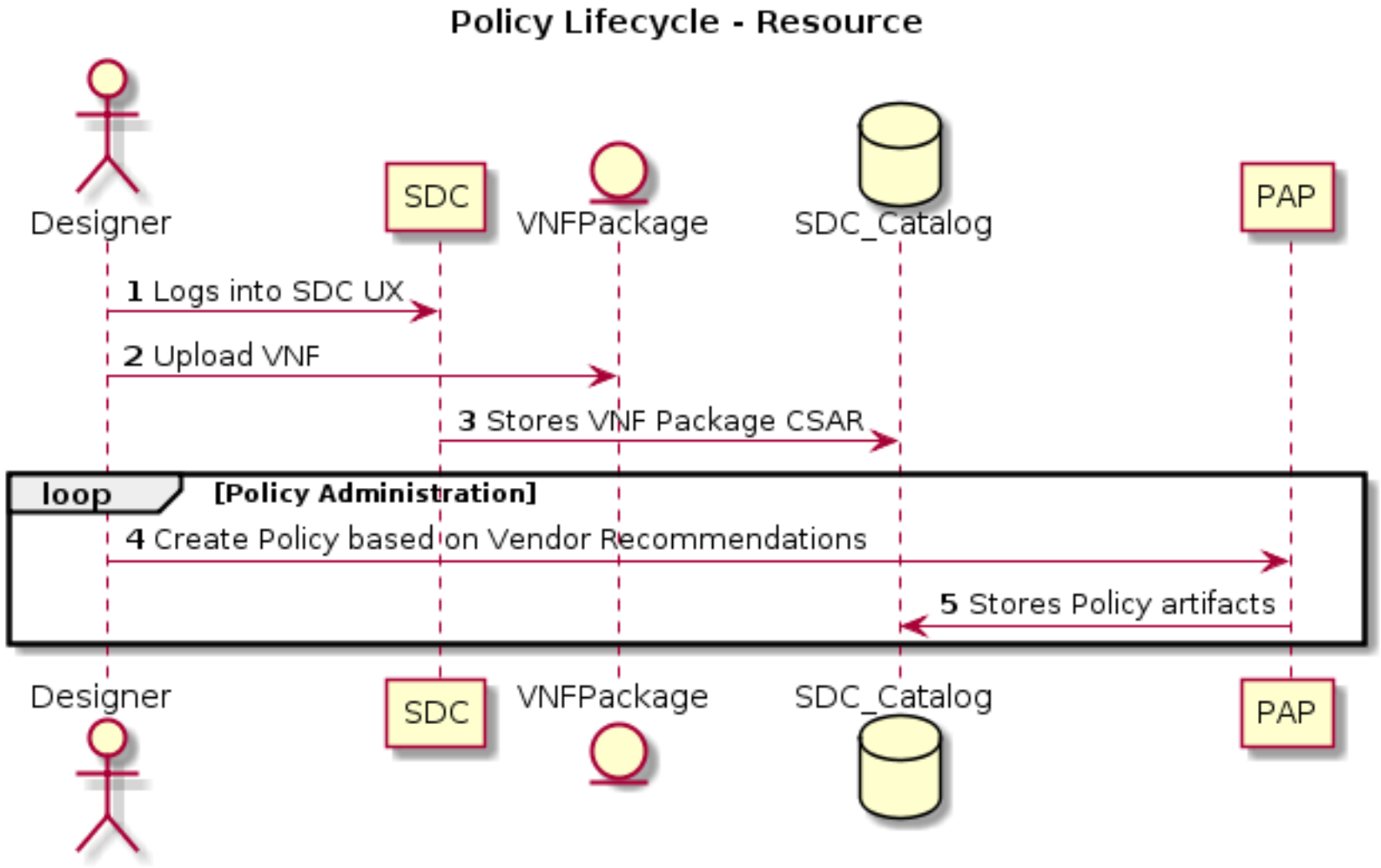
- ONAP Operator has done the following:
  - Instantiated a minimal ONAP Platform that had at least the Policy Platform loaded (Portal, AAF, Dmaap, an initial set of PDPs, etc.)
  - Uploaded any desired Policy Models
  - Created desired Policies
  - Deployed Policies to PDP Engines
- Ready to start Onboarding VNFs and Designing Services

Resource Onboarding via SDC

- Vendor VNF Policies already defined, packaged and uploaded into the SDC Catalog
  - Some policies should not be changed, whereas other policies are “Recommended”
- Provide Policy Lifecycle API to translate some Vendor Policies
  - Eg. Vendor recommended self-serve policies need to be translated into Operational Policies. Optimization can be performed
- Provide ability to provide additional policies for the VNF
- Storage would be in the SDC Catalog, follow SDC design governance

Resource Onboarding Flow

- Policy artifacts may simply be models, not actual runtime policies
  - Affinity and Anti-Affinity
  - Homing and Placement
  - Operational Policies



API Path design options

- Policy Lifecycle API could provide a simple interface to the SDC Catalog to present list of Vendor Policies

/resources
/resources/{resource-id}
/resources/{resource-id}/vendor-policies
/resources/{resource-id}/vendor-policies/{policy-id}

- Policy Lifecycle API add ability to add Policies and/or update policies. Open question as to what the design for this would be.

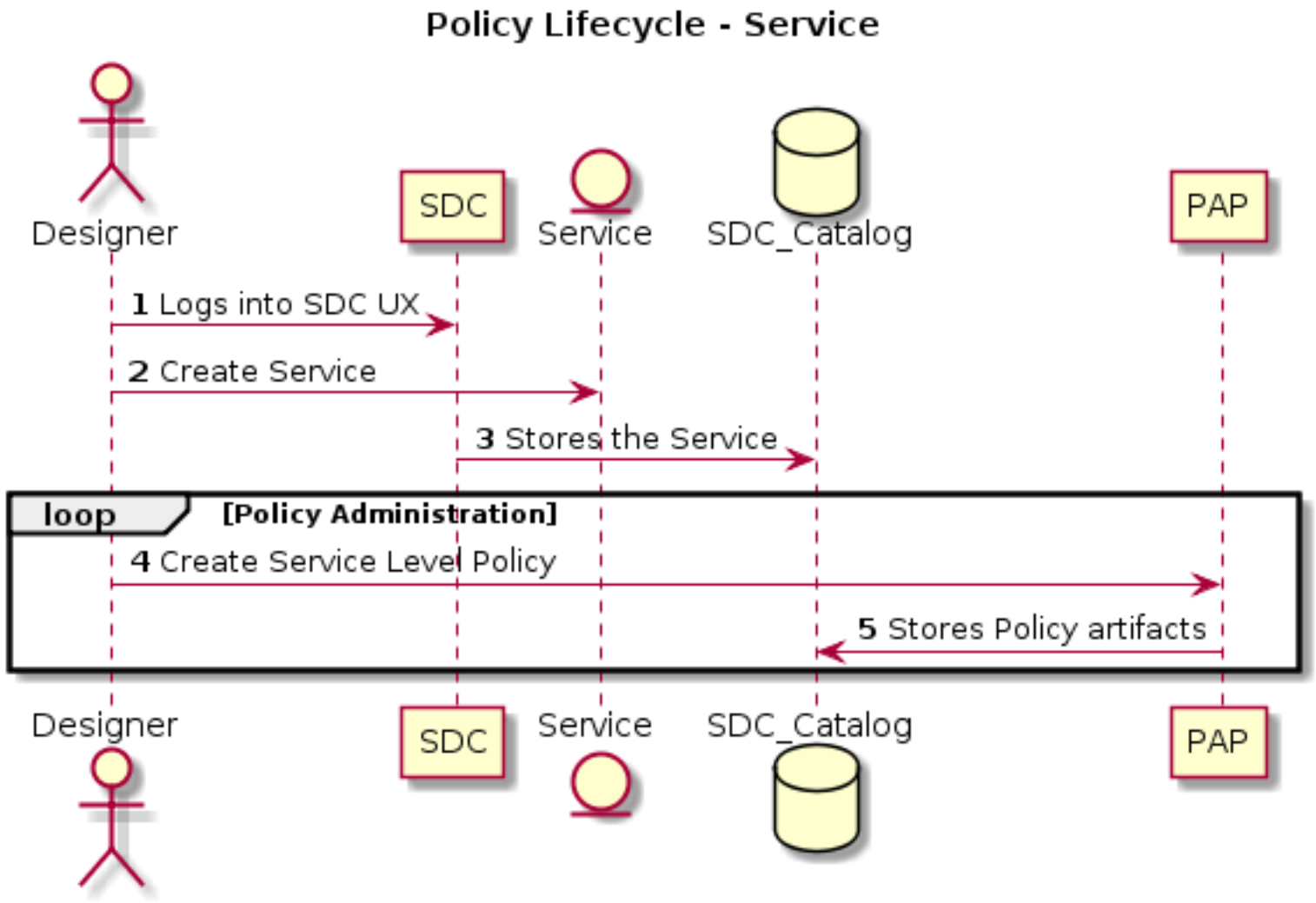
/resources/{resource-id}/policies
/resources/{resource-id}/policies/{policy-id}

Service Design via SDC

- Service Designers would be utilizing the Policy Lifecycle API to CRUD Service Level policies.
- Policy Lifecycle API will be able to take into account all the Policies for the VNF's and Service
- Stored in the SDC Catalog as artifacts
  - Control Loop Config and Operational Policies
  - Homing and Placement Policies

Service Onboarding Flow

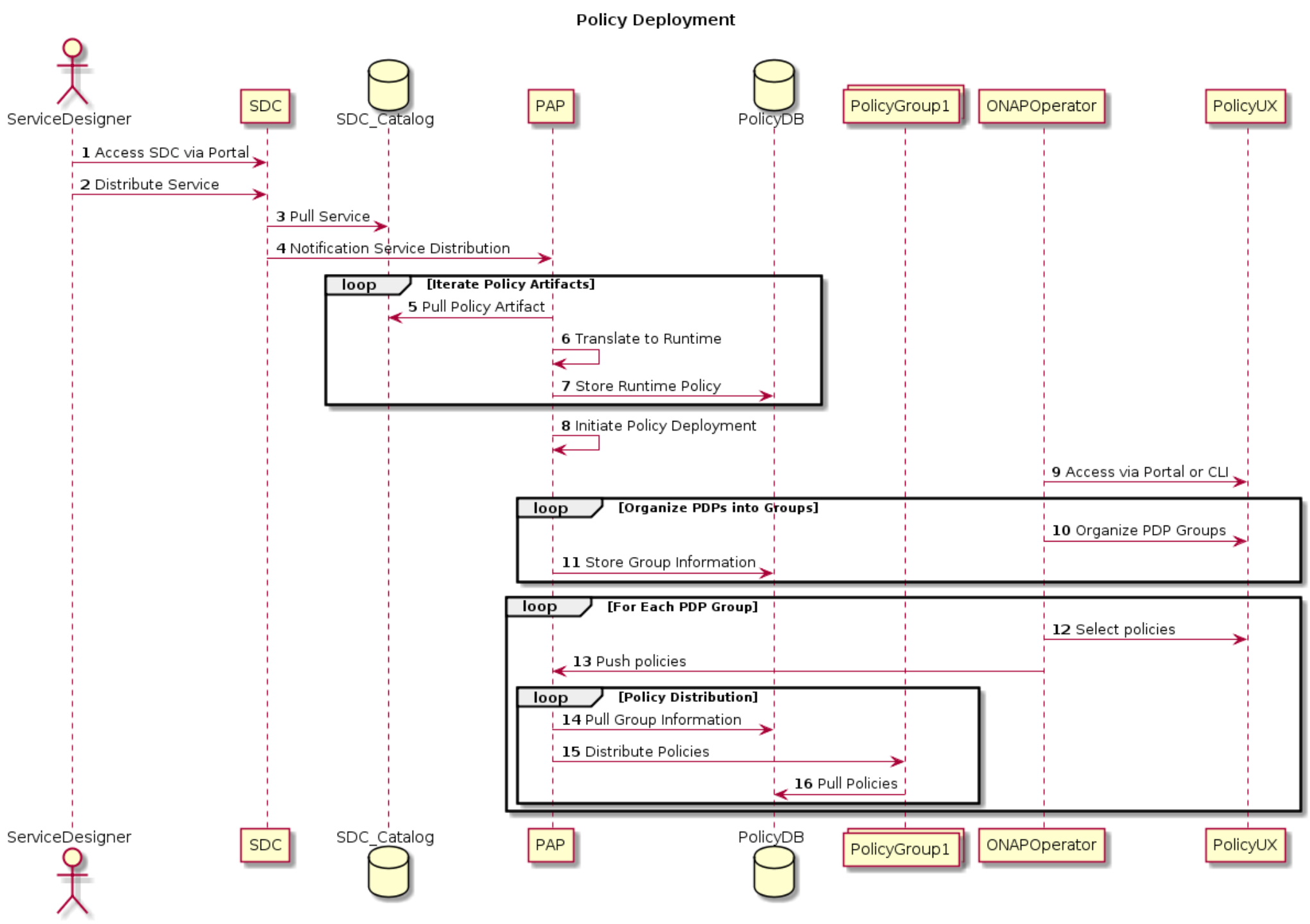
- Policy artifacts may simply be models, not actual runtime policies
  - Homing and Placement Policies
  - Control Loop Policies (via CLAMP)
- Stored in the Service CSAR
- NOTE: Service design would also include composition of certified VNF Resources. Not shown.



Resource and Service Policy Distribution and Deployment

- Policy Platform would require integration with SDC Distribution API
- Policy Lifecycle API will provide the ability to translate Policy artifacts in a service into runtime policy (if necessary)
- Policy Lifecycle API would deploy Resource and Service policies to PDP Groups
  - 2 Options for PDP Groups (open for discussion)
    1. May be deployed before the Service is distributed
    2. Deployed as part of Service distribution

Runtime Flow



Thank You!

Bio

- Pamela Dragosh has 27 years experience in designing and building software platforms in AT&T Research. Her projects have ranged from speech recognition, text-to-speech, digital rights management, music encryption, big data, location-based services, software defined networking, and policy platform. Pam has open sourced several projects including XACML 3.0 Policy Engine, OpenAZ Apache, and is currently the Project Technical Lead in ONAP for the Policy Framework Project.