ARC Policy Framework Component Description - Kohn-R11

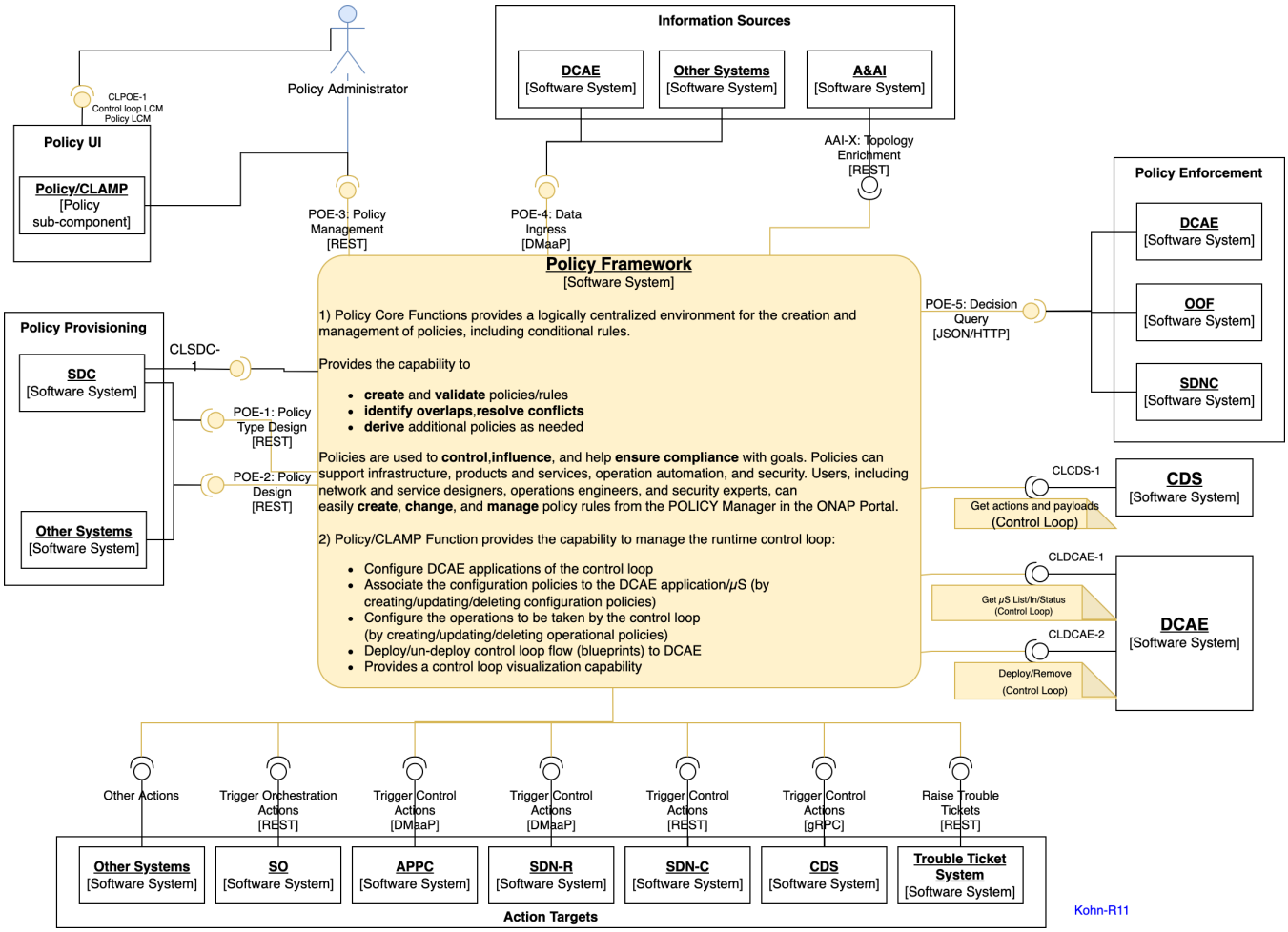
Page Status: Updated by PTL 15 Jan 2021

Component Status: Updated by PTL 15 Jan 2021 ; Pending PTL Updates and ArchCom Review

Last Reviewed:

Certified by:

1. High Level Component Definition and Architectural Relationships



In the Honolulu release the CLAMP component was successfully integrated into the Policy component as a PoC (see ~~REQ-473~~ - Merge CLAMP functionality into Policy Framework project DONE) . CLAMP's functional role to provision Policy has been enhanced to support provisioning of policies outside of the context of a Control Loop and therefore act as a Policy UI. In Istanbul release the CLAMP integration will be officially released ~~REQ-684~~ - Merge CLAMP functionality into Policy Framework project DONE , the code is already there and working.

2. API Definitions

2a. Exposed APIs

Interface Name	Definition	Capabilities	Version	Status	Payload Model(s)	API Spec (Swagger)
POE-1	Policy Type Design	Allows applications to create, update, delete, and query *PolicyType* entities so that they become available for use in ONAP by applications such as CLAMP.	1.0.0	production	tosca.policies.root TOSCA	https://docs.onap.org/projects/onap-policy-parent/en/latest/api/api.html#api-swagger
POE-2	Policy Design	Allows applications (such as CLAMP and Integration) to create, update, delete, and query *Policy* entities.	1.0.0	production	tosca.policies.root TOSCA	https://docs.onap.org/projects/onap-policy-parent/en/latest/api/api.html#api-swagger
POE-3	Policy Administration	Support CRUD of PDP groups and subgroups and to support the deployment and life cycles of *PolicyImpl* entities (TOSCA *Policy* and *PolicyTypeImpl* entities) on PDP sub groups and PDPs.	1.0.0	production	Embedded	https://docs.onap.org/projects/onap-policy-parent/en/latest/pap/pap.html#pap-rest-api-swagger
POE-4	Data Ingress	Listen on a DMaaP topic.		production	Messages of interest are described in the policy logic DMaaP	
POE-5	Decision Query	Policy decisions are required by ONAP components to support the policy-driven ONAP architecture. Policy Decisions are implemented using the XACML and Apex PDPs. The calling application (which may be another policy – e.g. invocation of a guard policy from PDP-D) must provide attributes in order for the PDP to return a correct decision.	NA	production	Defined by policy	https://docs.onap.org/projects/onap-policy-parent/en/latest/xacml/decision-api.html
CLPOE-1	Control Loop Lifecycle Management and Polciy Lifecycle Management Interface	A user interface (GUI) for: • Selecting the control loop flow • Entering configuration policy parameters • Entering operational policy parameters • Managing life cycle of DCAE control flow blueprint • Selecting a Service to associate to a Control Loop to be instantiated • CRUD operation on Policy (outside of Control Loop)	NA		Defined by policy	NA (GUI)

2b. Consumed APIs

Interface Name	Consumed by	Description	API Spec (Swagger)
AAF	Policy Framework	Authentication and authorization	
DMaaP	Policy Framework Policies	Policy framework uses DMaaP for SDC subscriptions and internal communication. Policies use DMaaP as a transport for contextual information from various sources	
CLSDC-1	Policy/CLAMP	• Notification of CSAR; Retrieval of CSAR • To receive the Control Loop Blueprint from SDC	https://docs.onap.org/projects/onap-sdc/en/latest/offeredapis.html

CLDCAE-1	Policy/CLAMP	• Retrieve DCAE application status • Retrieve DCAE µS lists • Retrieve DCAE µS description and blueprints	https://docs.onap.org/projects/onap-dcaegen2/en/latest/sections/apis/inventory.html
CLDCAE-2	Policy/CLAMP	Deploy/remove DCAE µS.	https://docs.onap.org/projects/onap-dcaegen2/en/latest/sections/apis/deployment-handler.html
CLCDS-1	Policy/CLAMP	Get list of operations/actions and corresponding payload for Operational Policy where selected actor is "CDS".	
AAI	Policies	Enrich ingress data with topology information	
SO	Policies	Trigger orchestration actions (policy driven)	
SDNC APPC CDS	Policies	Trigger control actions (policy driven)	
Other	Policies	Trigger any interface defined in a policy, for example, trouble ticketing	

3. Component Description

The ONAP Policy Framework is a comprehensive policy design, deployment, and execution environment. The Policy Framework is the ***decision making*** component in an ONAP system. It allows you to specify, deploy, and execute the governance of the features and functions in your ONAP system, be they closed loop, orchestration, or more traditional open loop use case implementations. The Policy Framework is the component that is the source of truth for all policy decisions

Please see the TOSCA Policy Primer page for an introduction to TOSCA policy concepts. See the Policy Design and API flow page for a description of the component interactions.

TOSCA defines a *PolicyType*, the definition of a type of policy that can be applied to a service. It also defines a *Policy*, the definition of an instance of a *PolicyType*. In the Policy Framework, we must handle and manage these TOSCA definitions and tie them to real implementations of policies that can run on PDPs.

Each TOSCA *PolicyType* must have a corresponding *PolicyTypeImpl* in the Policy Framework. The TOSCA *PolicyType* definition can be used to create a TOSCA *Policy*definition, either directly by the Policy Framework, by CLAMP, or by some other system. Once the *Policy* artifact exists, it can be used together with the *PolicyTypeImpl* artifact to create a *PolicyImpl* artifact. A *PolicyImpl* artifact is an executable policy implementation that can run on a PDP.

The TOSCA *PolicyType* artifact defines the external characteristics of the policy; defining its properties, the types of entities it acts on, and its triggers. A *PolicyTypeImpl* artifact is an XACML, Drools, or APEX implementation of that policy definition. *PolicyType* and *PolicyTypeImpl* artifacts may be preloaded, may be loaded manually, or may be created using the Lifecycle API. Alternatively, *PolicyType* definitions may be loaded over the Lifecycle API for preloaded *PolicyTypeImpl* artifacts. A TOSCA *PolicyType* artifact can be used by clients (such as CLAMP or CLI tools) to create, parse, serialize, and/or deserialize an actual Policy.

The TOSCA *Policy* artifact is used internally by the Policy Framework, or is input by CLAMP or other systems. This artifact specifies the values of the properties for the policy and specifies the specific entities the policy acts on. Policy Design uses the TOSCA *Policy* artifact and the *PolicyTypeImpl* artifact to create an executable *PolicyImpl* artifact.

Internally, Policy has three main functional areas: Policy Development; Policy Administration; Policy Decision Execution.

Policy Development abstracts persistence and supports the creation of policies/policy types.

Policy Administration also abstracts persistence (from a different user roles). It's main purposes are to ensure that policies are allocated correctly and to manage the life-cycle of policies.

Policy Decision Execution is where policy decisions are made, i.e. where the policy logic executes. Three languages are used to describe policies in ONAP: XACML; Drools; APEX. Policy designers may select the language that is most appropriate to their use case. The policies are interpreted and executed by a language specific PDP. Where necessary additional PDP may be added, thus allowing for new policy languages.

Policy Enforcement is in general not handled by the Policy Framework. Enforcement is handled by either the originator of a decision query (PDP-D does enforce guard policy decisions made in the XACML PDP), or by a reaction to a policy output (trigger).

The Policy/CLAMP functional entity provides the capability to manage runtime control loops. It provides the capability to

- Create control loop from DCAE blueprint, those blueprint are either:
  - sent by SDC to DCAE (inventory) via SDC distribution.
- Create configuration policy from the policy-model Tosca, the tosca policy-model are either:
  - sent by SDC via SDC distribution to everyone; or
  - query by CLAMP to Policy. in this case the policy-model is either pre-provisioned in Policy or separately provisioned to Policy via the REST API exposed by Policy.
- Configure DCAE applications of the control loop
- Associate µService configuration policies to the DCAE application
- Configure the operations to be taken by the control loop (by creating/updating/deleting operational policies)
- Deploy/un-deploy control loop flow (blueprints) to DCAE
- Control loop visualization.

A more detailed figure and description of the Policy/CLAMP sub-component can be found here:

https://docs.onap.org/projects/onap-policy-clamp/en/latest/index.html#master-index

4. Known System Limitations

https://docs.onap.org/projects/onap-policy-parent/en/latest/release-notes.html

5. System Deployment Architecture

https://docs.onap.org/projects/onap-policy-parent/en/latest/installation/installation.html

6. New Release Capabilities

The main new capability introduced in this release is the integration of the CLAMP component under the policy umbrella (see ~~REQ-684~~ - Merge CLAMP functionality into Policy Framework project DONE). The software has been moved under the policy/clamp repository (https://git.onap.org/policy/clamp). The OOM helm charts have also been moved as a sub-component of policy (https://git.onap.org/oom/tree/kubernetes/policy/components).

The PoC experiments with defining Control Loops fully in Tosca will continue, as well. This is covered by REQ-478 - PoC - TOSCA Defined Control Loop on Honolulu Release IN PROGRESS .

As outlined in the architecture review page (Policy Honolulu-R8 Architecture Review) the only new use case that may be supported as a stretch goal is 5G (~~REQ-429~~ - Use Case for ONAP-based SON for 5G networks DONE). Existing use cases will be supported (no change).

Conformance with ONAP-wide requirements (REQ-441 - LOGS MANAGEMENT - PHASE 1: COMMON PLACE FOR DATA TO DO , REQ-437 - COMPLETION OF PYTHON LANGUAGE UPDATE (v2.7 v3.x) IN PROGRESS , ~~REQ-398~~ - Deploy on demand ONAP through CI per use case DONE , REQ-396 - Clearly split ONAP code and use case code TO DO , and REQ-439 - CONTINUATION OF PACKAGES UPGRADES IN DIRECT DEPENDENCIES IN PROGRESS (there are some risks, see Honolulu Risks page) are already or will be supported.

Additional enhancements to the Policy components will be introduced in this release that have no external impact. These include but are not limited to:

- Stateless PAP (Keep no state in PAP runtime container, so state retrieval would be from database).
- Interoperability of Native and non-native policies in the Drools PDP.
- Healthcheck status improvements as well as statistics reporting.

7. References

1. Istanbul architecture description https://docs.onap.org/projects/onap-policy-parent/en/latest/architecture/architecture.html
2. Policy Framework API's - https://docs.onap.org/projects/onap-policy-parent/en/latest/offeredapis.html
3. Policy/CLAMP:
    a. CLAMP Overview : https://docs.onap.org/projects/onap-policy-clamp/en/latest/index.html#master-index
    b. CLAMP internal interfaces: https://docs.onap.org/projects/onap-policy-clamp/en/latest/_downloads/d25f20712a4cf2524a1cf13242349743/swagger.pdf
    c. CLAMP User Guide: https://docs.onap.org/projects/onap-policy-clamp/en/latest/user-guide.html#