ONAP
OPEN NETWORK AUTOMATION PLATFORM

ONAP Architecture Documentation in Read The Docs

Ciaran Johnston, Ericsson
Tony Finnerty, Ericsson
Jeff Van Dam, Ericsson
Sofia Wallin, Ericsson

June 2020

Architecture Documentation

- https://wiki.onap.org/display/DW/Documenting+ONAP+Architecture
- Each project description includes a “System Context” diagram similar to that defined in the C4 model (https://c4model.com)
- Exposed and consumed interfaces are described as lollipops in the diagram
- Each interface is numbered according to the project that exposes it
  - <project-abbreviation>E-<number>
- A table is included referring to the interfaces, describing their intent, their version and (ideally) linking to the current version of the swagger documentation

- Challenges:
  - Inconsistent content from each project
  - Does not correlate directly to the information in RTD
  - Not properly version controlled
  - Projects have not taken ownership of the content

Read The Docs Documentation

- Version controlled restructured text documents stored in git and managed / owned by each project
- Includes (in some cases) an “Offered APIs” and “Consumed APIs” page under the component description
  - E.g. for DCAE
- Pages link to the specific swagger files (also stored in git)
- Interface pages can be brought together in an overall view: https://docs.onap.org/en/elalto/guides/onap-developer/apiref/index.html
- Proposal in REQ-386 to move API docs to Swagger: https://wiki.onap.org/display/DW/Developing+ONAP+API+Documentation

- Current challenges:
  - No consistency across all projects
  - No description or version of the interface
  - No “System Context View” that is consistent per project

Proposal for Next Steps

- Create a template for the component documentation to be included in the project documentation repository
  - Pick a project to try this out – e.g. policy
- Include in the template:
  - The System Context diagram – stored in the git repo as an uncompressed SVG (draw.io)
    - Include clickable links to the swagger docs in the diagram
    - Color-code interfaces which are exposed externally
  - A table describing the interfaces and referring to their versions
    - Refer to interface documentation (swagger docs)
- Define a template, try it out with a couple of projects and iterate
- If successful, roll out across the rest of the components

Migrating from the Confluence to Read The Docs

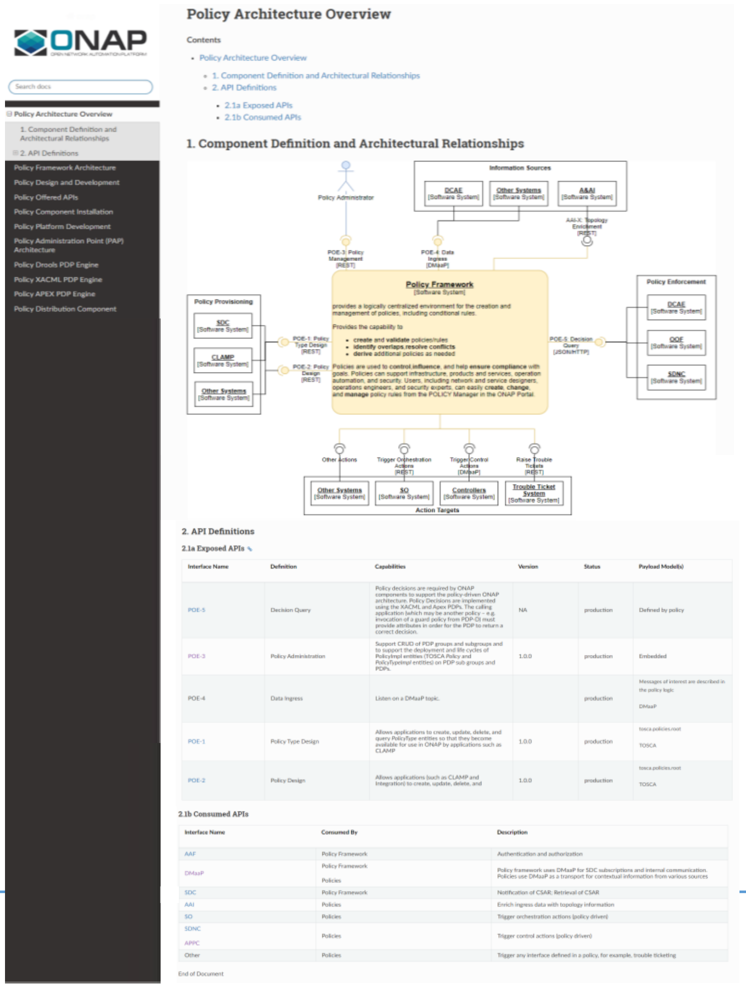
- The process:
    1. Clone repository from gerrit.onap.org
    2. Create new directories if needed
    3. Save diagram as an xml in the local repository
        1. Edit locally to add links
        2. Export as SVG in the local repository
    4. Create rst file with data from Confluence and include the SVG
    5. Test locally
    6. Submit for review

- Currently 3 architecture overviews ready for review
    - Policy
    - SDC
    - SO

- Challenges:
    - Inconsistent structure of repositories
    - Inconsistent structure of rst files

Policy Architecture Overview (Local Test)

High Level Component Definition and Architectural Relationships


ONAP
OPEN NETWORK AUTOMATION PLATFORM
Search docs
Policy Architecture Overview
1. Component Definition and Architectural Relationships
2. API Definitions
Policy Framework Architecture
Policy Design and Development
Policy Offered APIs
Policy Component Installation
Policy Platform Development
Policy Administration Point (PAP) Architecture
Policy Drools PDP Engine
Policy XACML PDP Engine
Policy APEX PDP Engine
Policy Distribution Component

Policy Architecture Overview

Contents

- Policy Architecture Overview
    - 1. Component Definition and Architectural Relationships
    - 2. API Definitions
        - 2.1a Exposed APIs
        - 2.1b Consumed APIs

1. Component Definition and Architectural Relationships

Policy Administrator
Information Sources
DCAE [Software System]
Other Systems [Software System]
A&AI [Software System]
AAI-X: Topology Enrichment [REST]
POE-3: Policy Management [REST]
POE-4: Data Ingress [DMaaP]
Policy Provisioning
SDC [Software System]
CLAMP [Software System]
Other Systems [Software System]
POE-1: Policy Type Design [REST]
POE-2: Policy Design [REST]
Policy Framework [Software System]
POE-5: Decision Query [JSON/HTTP]
Policy Enforcement
DCAE [Software System]
OOF [Software System]
SDNC [Software System]
Other Actions
Trigger Orchestration Actions [REST]
Trigger Control Actions [DMaaP]
Raise Trouble Tickets [REST]
Other Systems [Software System]
SO [Software System]
Controllers [Software System]
Trouble Ticket System [Software System]
Action Targets

provides a logically centralized environment for the creation and management of policies, including conditional rules.

Provides the capability to

- **create** and **validate** policies/rules
- **identify overlaps**,**resolve conflicts**
- **derive** additional policies as needed

Policies are used to **control**,**influence**, and help **ensure compliance** with goals. Policies can support infrastructure, products and services, operation automation, and security. Users, including network and service designers, operations engineers, and security experts, can easily **create**, **change**, and **manage** policy rules from the POLICY Manager in the ONAP Portal.

2. API Definitions

2.1a Exposed APIs

Interface Name	Definition	Capabilities	Version	Status	Payload Model(s)
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
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
POE-4	Data Ingress	Listen on a DMaaP topic.		production	Messages of interest are described in the policy logic DMaaP
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
POE-2	Policy Design	Allows applications (such as CLAMP and Integration) to create, update, delete, and	1.0.0	production	tosca.policies.root TOSCA

2.1b Consumed APIs

Interface Name	Consumed By	Description
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
SDC	Policy Framework	Notification of CSAR; Retrieval of CSAR
AAI	Policies	Enrich ingress data with topology information
SO	Policies	Trigger orchestration actions (policy driven)
SDNC APPC	Policies	Trigger control actions (policy driven)
Other	Policies	Trigger any interface defined in a policy, for example, trouble ticketing

End of Document

Discussion …

- Include this as a checkpoint in the review process
  - Keep the SVG approach
  - Need to compare previous and proposed version during review

- Review with the PTLs

- Relevant to the previous session on how we document these components within ONAP

- C4 level diagrams are good – need to look at the information flows as well

- Do the same presentation in the PTL call