ONAP - Control Loop in TOSCA PoC

Ericsson
Michela Bevilacqua - Liam Fallon
We need to think about **Control Loop** information

- Control Loops as first class citizens in ONAP
- Management of Control Loops at **design time** and **run time**
- Control loop participants need to use **common** knowledge
Catalogues: Native TOSCA in Long Term

- SDC
- DCAE-MOD
- CLAMP

Control Loop Deployment and Instantiation

- Control Loop Run Time Catalogue
  - TOSCA Artifacts

- DCAE
- Policy
- Controllers

Managed Domain
Deployment of Control Loops in Long Term

Control Loop Design

CSAR with Control Loop Definition
- Control Loop Descriptor (TOSCA)
- Control Loop Artifacts & Metadata

CLAMP

Existing Services
- Control Loop Component Micro Services
- CL Metadata
- CL Metadata
- CL Metadata
- Demonstrate Control loops can be defined and deployed using TOSCA

- Use a design time catalogue for Control Loops for a complete storage of all the artifacts from different DT systems

- Show design time systems can populate the Design Time control loop catalogue
  - DCAE-MOD interacting with the design time catalogue
  - SDC interacting with the design time catalogue

- Show TOSCA defined control loops being onboarded and deployed
We don’t need in TOSCA to always have a well-known capability at one of the ends

- Can be implied using a TOSCA requirement directly
TOSCA ARTIFACTS

- **BASIC types** (cloop_base_types.yaml)
  - Fundamental types including the event producer and consumer capabilities, and base app type

- **DCAE ms artefact** (cloop_dcae_types.yaml)
  - Types modelled in DCAE, TCA_Gen_2 or PMSH *mapped to TOSCA (by DCAE)*
  - Component composition not covered in Rel G

- **Policy** (cloop_other_types.yaml)
  - Other types used in the closed loop example (e.g. Policy)
  - *Policy_Framework and Apex_Policy_Framework mapped to TOSCA (by SDC)*

- **Control Loop** (cloop_dcae_example.yaml)
  - Control Loop example containing TCA_Gen_2 and Apex_Policy Framework app
  - *Cloop created by CLAMP Designer*
Reference Design Time Flow

**Design Time**
- DCAN-MOD
- CLAMP
- DCAN-DBH
- DCAN_Deployment
- DCAN_Inventory
- DCAN_Policy_Handlers

**Post Workflow**

**Current Flow**
1. Create Policy Model
2. Service Creation (attaching DCAN with blueprint)
3. Service Creation (without DCAN with blueprint)
4. Create Control Loop (TOSCA file)

**Proposed Flow**
1. Create Policy Model and artifacts (TOSCA file)
2. Service Creation (without DCAN blueprint)
3. Create DCAN/Blueprint and DCAN APP artifact (TOSCA file)

**Relax E2E Service and CL Artifacts dependency**

- Store DCAN Service Blueprint in DCAN_inv
- Store DCAN Service Blueprint ID
- Create DCAN APP artifact (TOSCA file)
- DCAN service blueprint storage
- Control Loop CT
- Control Loop CT Catalogue

**Improved**
- Single CL DT Catalogue
Reference Run Time Flow

CL Template Deployment/CL Instance Configuration

CL Instance Deployment or Instantiation

Reading CLT from CL DT Catalogue

CL Instance Configuration steps

Store CL Instance in CL RT Catalogue
• REST API for DT CL Catalogue
  • CRUD operations from SDC or DCAE_MOD or manually by the operator
  • Get operation from CLAMP

• CL Template Deployment: Get CL template information, progress CL configuration then store CL instance in RT CL Catalogue

• CL Instance Deployment/Instantiation: Get CL instance from RT CL Catalogue and execute
• Continue Control Loop PoC in Rel H to investigate further:
  • Integration with SDC, DCAE-MOD and CLAMP
  • Deployment/Instantiation of CL with any CL components/APP
    • CL component as DCAE ms instances (covered by current CLs)
    • CL component as metadata in existing ONAP component instances (e.g. CDS blueprint)
    • CL component as external component to ONAP
  • CL Monitoring
• Generalize the concept of Control Loop component to include external component to ONAP
• An external CL Component/Application is a micro-service oriented and with a model-based deployment. An Application can be deployed dynamically
• An Application has a specific purpose, use case or domain but there is no intent to constraint an app to a specific function (e.g. Data collection or Analytic function). Application deployment must be Application logic agnostic
• An Application may interwork with ONAP platform/components to perform Network Automation
• An Application may be part of a ONAP Control Loop
• The support in ONAP of Application deployment and its involvement in ONAP Control Loop will:
  • provide a new capability to the platform expanding its boundaries
  • reinforce ONAP Architecture Principles about Integration Friendly / Standard API: Various service providers and users of ONAP should be able to quickly integrate ONAP with their existing OSS (and OSS applications)