Configuration Persistence Service in R8+

- Virtual Developer Design Forum (vDDF) / Oct 15, 2020 / version 12

NOKIA: Ben Cheung (Nokia), Marge Hillis (Nokia), Shankar N K (AT&T), Ted Johnson (AT&T), Claudio Gasparini, Zu Qiang (Ericsson), Tony Finnerty (Ericsson)

ERICSSON: Toine Siebelink (Ericsson), Michela Bevilacqua (Ericsson), Jacqueline Beaulac (Ericsson), Rishi Chail (Ericsson), Ciaran Johnston (Ericsson)

SAMSUNG: Joachim Blixt

SAMSUNG: Pawel Slowikowski (Samsung)

SAMSUNG: Swami N (Wipro)

BELL: Bruno Sakoto (Bell Canada), Philippe Leger (Bell Canada), Aditya Puthuparambil (Bell Canada), Fred Feisullin (Verizon Wireless)
# R8 DDF Presentation

<table>
<thead>
<tr>
<th>TIME</th>
<th>AUG 17, 2020 CPS REQUIREMENTS SubComm AGENDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 min</td>
<td>Context for CPS &amp; Agenda – Context for CPS &amp; Basic problems to be addressed</td>
</tr>
<tr>
<td>10 min</td>
<td>Overview of CPS – Overview to CPS</td>
</tr>
<tr>
<td>15 min</td>
<td>Model Driven CPS Proof of Concept (PoC) &amp; Demo – Overview of the Model-Driven CPS PoC for R7</td>
</tr>
<tr>
<td>5 min</td>
<td>R8 &amp; Beyond Roadmap – Model Driven Proof of Concept (PoC) in R7, way forward in R8 Honolulu, New plan &amp; roadmap</td>
</tr>
<tr>
<td>10 min</td>
<td>Use Cases using CPS Database – Overview of CPS Applications (State Mgmt)</td>
</tr>
<tr>
<td>10 min</td>
<td>Questions &amp; Answers – Q&amp;A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIME</th>
<th>Q&amp;A Session Post-Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Follow-up questions – Follow-up meetings at CPS Team Call (Fridays)</td>
</tr>
</tbody>
</table>
Context for
Configuration Persistence Service (CPS)
## Configuration Persistence Service in R8

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>PROBLEM</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Heterogeneous Data Sources</strong></td>
<td>Reconciling Multiple domains, multiple vendors, multiple functions and multiple versions</td>
<td>Model Driven Persistence</td>
</tr>
<tr>
<td><strong>Shared Data Access</strong></td>
<td>Expensive IO operations that should be shared rather than duplicated</td>
<td>Share Data access through model ownership</td>
</tr>
<tr>
<td><strong>Model Handling</strong></td>
<td>Because of the heterogeneous data sources need streamlined approach to support models without having platform life cycle events.</td>
<td>Model Driven Persistence</td>
</tr>
</tbody>
</table>
Configuration Persistence Service in R8

- Multiple vendors providing equipment / xNF
- Several telecom networks one ONAP platform
- Evolution of xNF and network standards
- ONAP applications and use cases competing for access to the same CM data
Configuration Persistence Service in R8

Shared data analogy of a teacher sharing knowledge with students
Configuration Persistence Service in R8

Model ingestion during ONAP design & run time
Model ingestion with C&PS solution (Run-time)
Configuration Persistence Service in R8

Permission & Ownership of data
Configuration Persistence Service in R8

ONAP

CPS Project

CPS

xNF Proxy

O1 compliant entity
Overview for Configuration Persistence Service (CPS)

- Business Case
- Architecture S/C
- Overview
- Technical Flows
Executive Summary - The Configuration Persistence Service (CPS) is a real-time service that is designed to serve as a data repository for Run-time Network Element (configuration) data that needs to be persistent applicable to multiple domain (RAN, Transport, and Core). This was explored as a R7 PoC. Focus on storing run-time DATA RELATED to NETWORK ELEMENT instances. In R8, this is being proposed as a stand-alone project.

Business Impact - The ability for service operators to visualize and manage network element data in a network (PNFs, VNFs, and logical constructs) with ONAP is a critical business function because they are key Life Cycle Management (LCM) and OA&M operations. The project has business impacts to enhance the operation of data-handling within ONAP by providing efficient data layer services.

Business Markets - This project applies to any domain (wireless, transport, optical, and wireline) that ONAP may manage. It is not a market or geographical specific capability. It is expected that scaled ONAP installations such as Edge & Core ONAP deployments will also deploy the database across each installation. Funding/Financial Impacts - This project represents a large potential Operating Expense (OPEX) savings for operators because of the ability to configure networks saving time and expenses.
CPS in Architecture Context
Configuration Persistence Service (CPS)

1. **Micro-Services & Use Cases**
   - Network Element Data
   - Read/Write

2. **Configuration Information**
   - Standards Defined VES Event
   - CPS Update

3. **Controller**
   - Read/Write

4. **Active & Available Inventory (A&AI)**
   - A&AI Sync

DU/PNF  CU/VNF

CPS Database

NETWORK ELEMENT DATA
Configuration Persistence Service (CPS)

Network Element Data
- Application Parameters
- Configuration Parameters

CPS Update

GUI Access

ONAP Micro-services, POCs & Use Cases
- Model Driven PoC
- OOF SON PCI
- E2E Network Slicing
- A1 Adaptor Extension
- State Management

U/C Data

Run-Time Operational Data
- Configuration Info
- Exo-Inventory Data
- State Information
The PNF has a parameter update to report. The update originates from the PNF and is reported through a Standards Defined VES event with a configuration NameSpace (3GPP-Provisioning).

**Standards Defined VES Event** is received by the DCAE VES Collector. DCAE publishes the VES Event onto the DMaaP Bus.

In R8+: CPS as a stand-alone component, subscribes to the DMaaP Topic and gets the DMaaP event from the DMaaP bus to update the internal database. The VES event has a Configuration namespace topic, 3GPP-Provisioning.
(1) During Network setup “getall” retrieves from A&AI the ENTIRE A&AI graph. Used to setup the initial view of CPS
(2) Updates additions/deletions of xNFs. “AAI-Event” (operation Addition / delete)

A Micro-Service, for example OOF/SON/PCI determines that an update is needed to CPSDB from operation/algorithm. It publishes to the DMaaP bus an update event.

In R8+: CPSDB is a stand-alone component. CPSDB subscribes to the DMaaP topic and gets the DMaaP event off the DMaaP bus to update itself.
The Controller (SDN-C) also sends a message to the xNF to update the parameter. This may be done via NetConf/O1/Ansible.

SO, Policy, or Control Loop has determined a parameter update is needed to the xNF. The Controller (SDN-C) eventually gets the configuration update. SDN-C publishes to the DMaaP bus a configuration update event.

R8+ The configuration & standards service as a stand-alone component subscribes to the DMaaP topic and gets the DMaaP event off the DMaaP bus to update the internal database. A configuration namespace topic is used, 3GPP-Provisioning.

Controller (SDN-C) gets an ACK back from the xNF that the parameter change was successful.

xNF would may send a Standards Defined VES unless xNF configured to suppress event on ONAP origination.

Controller DB

Configuration Persistence Service

CPS Database

Run-Time Operational Data
Configuration Info
Exo-Inventory Data
RT Logical & Physical Connections

Controller (SDN-C), VF-C, APP-C

CM Notification Change Config

Controller DB

Read/Write

Configuration NameSpace topic
3GPP-Provisioning
A&AI vs CPS

**Concepts** – A&AI conceptually stores Real-time inventory view of connected and “topology” of xNFs that ONAP sees. CPS stores Network Element Data. A&AI and CPS overlaps because they both need to know about Network Element objects so that can managed & orchestrated.
R7 Model Driven Proof of Concept for Configuration Persistence Service (CPS)
Proof Points for PoC

1. Demonstrate write/read operations for YANG data fragments using CPS and store them in a DB with a very simple generic schema

2. Demonstrate ability to deploy / upgrade YANG models at run-time

3. Demonstrate CPS behavior driven by YANG model

4. Provide architecture vision and roadmap for a target architecture, supported use cases, non-functional requirements towards an ONAP Project
R7 Model Driven CPS PoC

1. Model Driven Schema
   - R7: Select YANG model (Stretch Goal)

2. Update
   - R7: A&AI Stretch Goal

3. Synchronization
   - A&AI information to uniquely identify a xNF

4. Service Provider Interface
   - R7: Read/Write Data (Base Scope)

5. CPS API
   - R7: Model Deployed (Base Scope)
   - R7: Temporal DB Stretch
   - R7: Select YANG model (Stretch Goal)
   - R7: Model Deployed

- **Use case**
  - 3GPP-Configuration (NameSpace)
  - StndDef
  - VES ++

- **Component**
  - DCAE
  - DMaaP

- **Work required**
  - PNF
  - Temporal DB

**Use cases**
- OOF/SON/PCI
- E2E Network Slicing
- A1 Extension, policies

**Work**
- Mirroring xNF Configuration Data
- CRUD
Community Involvement & Resourcing

- Weekly ONAP meeting hosted by Ben Cheung (Nokia)
- PoC discussed with and approved by TSC
- ONAP R8 Honolulu Project proposal accepted
  - Use Cases & requirements: Configuration Persistence Service (CPS)
- CPS presentations in many fora, O-Ran PlugFest, LFN Design Developer Forum (DDF) etc.
- Resourcing
  - Ericsson 3 full-time developers
  - Bell Canada 3 part-time developers
- Jira backlog, team ceremonies like scrums etc. established
  - Developers Landing Page
PoC Ways of Working

- **Base**: N/A (new code)
  Main dependency: ODL Yang Tools 5.x (probably)

- **Design and Architecture discussions ongoing**
  [https://wiki.onap.org/display/DW/Issues+decisions+and+assumptions](https://wiki.onap.org/display/DW/Issues+decisions+and+assumptions)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 8 | 1 | MEDIUM | Existing Yang Parser  
Is there an existing Yang Parser in ONAP or OpenDayLight that can be used for C&PS  
No |
| 9 | N/A | AGREED | Location of PoC Code  
Dan Timony suggested to use and existing CCSDK repo, he mentioned ccsdk/features. As long as the PoC remains completely independent and doesn’t affect delivery of existing artifacts in the same repo.  
--  
ccsdk/features, see https://gerrit.onap.org/r/c/ccsdk/features/+/110365 (awaiting approval) |
| 10 | N/A | AGREED | Common information model, Data lake and Access control  
How will the CPS help with managing coupling between ONAP components that make use of data lake and common information model  
--  
We will start with Architectural Approach A in the PoC with the aim of fully supporting Architectural Approach C.  
I.e. access to the data lake will be conditional on permission granted by the data owner. In the PoC we will not implement the permission granting mechanism |
| 11 | 4.5 | MEDIUM | Transactional behavior  
It needs to be clear to users the level of atomic operations supported by the CPS  
Yes |
R7 Model Driven CPS PoC

Architecture designed & implemented

REST API
- org.onap.cps.rest
  - CPS Controller
    - uploadModel(modelContent)
    - validateModel(modelContent)
  - calls validateModel() and storeModel()

CPS API
- org.onap.cps.cpservice.api
  - CPSControllerImpl
    - CPServerImpl
      - validateModel(modelContent)
      - storeModel(modelContent)
      - calls storeModel()

CPS SPI
- org.onap.cps.persistence.spi
  - PersistenceService
    - storeModel(modelContent)

Data Layer
- DB Layer
  - This layer implements the actual data storage. There can be several implementations of this.
  - This actual DB can be chosen as part of the Persistence Service Implementation above.

Pluggable SPI Implementation (DB Layer)

This layer exposes REST endpoint to other services

This layer exposes public methods to be for any client use the CPS as a library

This layer defines the methods used to access and manipulate the data store
Implementation Status

- PoC being developed on Nordix Branch to avoid impacts on production code in CCSDK Repo
- Project structure set up completed
  - Following ONAP standards and conventions
  - Production code Quality (SonarQube & Checkstyle)
- ODL Yang Tools integrated
  - Parsing of Yang models
  - Parsing of Yang data, JSON or XML with validation for given Yang model
- First E2E iteration of 2 use cases completed
  - Upload & Validate Yang Model (part of proof point 2)
  - Upload JSON Data Object (part of proof point 1)
## CPS API

Configuration and Persistence Service API

Created by CPS

[Apache License Version 2.0](https://www.apache.org/licenses/LICENSE-2.0)

### cps-resource: cps Resource

<table>
<thead>
<tr>
<th>Method</th>
<th>URL Path</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GET</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/anchors</td>
<td>getAnchors</td>
</tr>
<tr>
<td><strong>POST</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/anchors</td>
<td>createAnchor</td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/anchors/{anchor-id}</td>
<td>deleteAnchor</td>
</tr>
<tr>
<td><strong>GET</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/anchors/{anchor-id}</td>
<td>getAnchor</td>
</tr>
<tr>
<td><strong>GET</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/modules</td>
<td>getModule</td>
</tr>
<tr>
<td><strong>POST</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/modules</td>
<td>createModule</td>
</tr>
<tr>
<td><strong>GET</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/modules/namespace/{namespace-id}</td>
<td>getModuleForNamespaceDataspaces</td>
</tr>
<tr>
<td><strong>GET</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/modules/namespace/{namespace-id}/revision/{revision}</td>
<td>getModuleForNamespaceDataspacesRevision</td>
</tr>
<tr>
<td><strong>POST</strong></td>
<td>/api/cps/v1/dataspaces/{dataspaces-id}/nodes</td>
<td>createNode</td>
</tr>
</tbody>
</table>

[BASE URL: /, API VERSION: 1.0]
R7 Model Driven CPS PoC

Generic Schema Design Finalized

- Can store any modelled data
- Portable, can easily be adapted to any DB technology
Early Performance Indicators (DB)
(proof point 4)

• Insert
  • 450 records/sec (up to 2M records)

• Queries
  • XPath Query (while loading new data) < 100ms
  • Tree-Walk (75K objects) count with condition < 1,000ms

• Delete
  • Delete Tree of 75K objects incl. constraint checks ~ 6 sec
Demos

1. JSON Data
   1. Upload valid JSON, store in DB
   2. Handle invalid JSON

2. Yang Model File (modules)
   1. Valid model file, store in DB
   2. Handle invalid model

3. Split JSON data for given model
   1. Split into ‘fragments’ ready for generic DB
   2. Handle model change
## R7 Model Driven CPS PoC

<table>
<thead>
<tr>
<th>TOPIC</th>
<th>PoC RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SEED CODE</strong></td>
<td>Seed Code to serve as a basis for CPS has been developed</td>
</tr>
</tbody>
</table>
| **PROOF POINTS**             | 1. Create/read CRUD operations using YANG fragments using a simple schema or schema-less repository  
2. Deploy & upgrade YANG model fragments at run-time  
3. Validate based on YANG Constraints  
4. Provide architecture vision and roadmap for a target architecture. |
| **ARCHITECTURE AGREEMENTS**  | Resolve key architectural Issues necessary for CPS as a stand-alone project   |
| **CPS CORE FUNCTIONALITY**  | Can demonstrate some key CPS operations; basic E2E & integration of Yang parser |
| **PERFORMANCE LIMITS**       | Early performance indicators tested using generic schema with json-blob storage for attributes |
Roadmap for Configuration Persistence Service (CPS)
**Configuration Persistence Service (CPS) Roadmap –**

**CPS 1.0**
- **R6 CPS**
  - CC-SDK/SDN-C solution
  - Evolution of “ConfigDB”

**Supporting R6 Use Cases:**
- SON/OOF/PCI U/C

**CPS 1.1**
- **R6 CPS Extensions**
  - Evolution of CC-SDK/SDN-C solution REQ322
  - **Supporting R7 Use Cases:**
    - SON/OOF/PCI U/C
    - 5G E2E Network Slicing
    - A1 Policy extension (Ericsson)

**Model-Driven PoC**
- Write “base” CPS
- Write NE Data
- Read NE Data
- Access Control

**State Management PoC**
- State Management PoC (BellCA) self-contained

**CPS 2.0**
- **R8 CPS stand-alone project proposal**
  - Deprecate CPS 1.0 & 1.1
  - Project proposals TSC/Architecture S/C
  - Setup Project Repo

**CPS FUNCTIONALITY:**
- Data Recovery
- Model Adaption (Dynamic Schema)

---

Legend:
RED text is CC-SDK/SDN-C solution
BLUE text is the PoC & stand-alone project
**CPS Roadmap & R8-R10 Plan**

**Configuration Persistence Service (CPS) Roadmap –**

**R8 Honolulu**
- **CPS 2.0**
- **R8 CPS stand-alone project proposal**
  - Deprecate CPS 1.0 & 1.1
  - Project proposals TSC/Architecture S/C
  - Setup Project Repo

**CPS FUNCTIONALITY:**
- Data Recovery
- Model Adaption (Dynamic Schema)

**Rx (future) development**
- June 2021

**Legend:**
- RED text is CC-SDK/SDN-C solution
- BLUE text is the PoC & stand-alone project

**R9 Istanbul**
- **CPS FUNCTIONALITY:**
  - Data Auditing Rules Driven
  - Topology Traversal
  - Data Syncing

**Rx (future) development**
- December 2021

**R10 Kyoto**
- **CPS FUNCTIONALITY:**
  - Data Auditing Model driven
  - Data History
  - Roll-Back
  - Database Backup
  - Performance Optimization (Scaling)

**Rx (future) development**
- June 2021
State Management Use Case for Configuration Persistence Service (CPS) in R8
State Management PoC (Bell Canada)

**Actors**
- Role: Act upon policy execution and instructions to perform management tasks on the network elements.

**Message bus**
- Role: From processed events; decides which actor should be leveraged to perform what action in the network.

**Policies**

**Analytics**
- Role: Standardizes, transforms, enriches and analyzes the collector data to publish events out of it.

**Collectors**
- Role: Collects fault, performance, configuration data from network devices and then publishes (raw) data for other components to consume.

**StateDB**
- Role: Keeps operational data from the network and its services. Supports and optimizes Analytics applications for their analysis.
State Management PoC (Bell Canada)

Api Documentation

inventory-controller Inventory Controller

model-controller Model Controller

state-controller State Controller

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/states</td>
<td>retrieveByFilter</td>
</tr>
<tr>
<td>POST</td>
<td>/states</td>
<td>add</td>
</tr>
<tr>
<td>GET</td>
<td>/states/search</td>
<td>retrieveByQuery</td>
</tr>
</tbody>
</table>

Models

- InventoryDto
- ModelDto

StateDto
```java
{  
  data
  modelId
  resourceId
  resourceInventoryId
  timestamp
  version
  ...
}
```
State Management PoC (Bell Canada)

PoC

CPS
Read and write persisted Configuration Management data complying to generic YANG models.

MVP

Release

CPS
Production ready component to store, retrieve and keep track of network operational data complying to generic YANG models.

State DB
Over time, keep track of configuration and state data, complying to generic YANG models.

State DB and CPS both have the same high level vision:
- To implement the missing run-time operational store to support other ONAP components
- To leverage YANG models to provide a generic structure for data instances

Opportunity to converge
Use Cases for
Configuration Persistence Service (CPS)

Use Cases

Proof of Concept
CPS Use Cases and Proof of Concepts in R8

<table>
<thead>
<tr>
<th>5G USE CASE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE MANAGEMENT POC</td>
<td>Bell Canada led PoC for State tracking and State management using CPS</td>
</tr>
<tr>
<td></td>
<td>Integration with CPS (as a platform). Have the State management S/W now work</td>
</tr>
<tr>
<td></td>
<td>with CPS using available swaggers/APIs.</td>
</tr>
<tr>
<td>OOF - SON (5G)</td>
<td>Optimization and SON functions for 5G RAN. Self-optimization, Self-Healing,</td>
</tr>
<tr>
<td></td>
<td>Self-configuration.</td>
</tr>
<tr>
<td>NETWORK SLICING (5G Use Case)</td>
<td>Network Slicing defines Slices for 5G RAN systems. Network Slicing is a long-</td>
</tr>
<tr>
<td></td>
<td>lead (multi-release) development. (will be presented in its own lecture at</td>
</tr>
<tr>
<td></td>
<td>the Virtual Face to Face)</td>
</tr>
<tr>
<td>MOBILITY STANDARDS HARMONIZATION/</td>
<td>A1 adapter: Enhancing the A1 adapter/interface capabilities in ONAP to manage</td>
</tr>
<tr>
<td>A1 adapter</td>
<td>A1 Policies, support multiple A1 targets in the RAN and multi-version A1</td>
</tr>
<tr>
<td></td>
<td>interface for different A1 targets, introduce secure TLS communication.</td>
</tr>
</tbody>
</table>
**OOF / SON / PCI Use Case**

- **Query to allocate PCI for new cell (optional)**

**Diagram Elements:**
- **OOF**
- **SON**
- **PCI Handler**
- **MS**
- **DCAE**
- **SDN - C**
- **Policy**
- **ConfigDB**
- **Simulated RAN**
- **REST API**
- **DMaaS Message**
- **Netconf/Yang**

**Key Points:**
- PCI enhancements
- ANR function

**Network Flows:**
1. Query to allocate PCI for new cell (optional)
2. Config Value Change Notification
3. Config Value Change
4. Simulated RAN
   - ~2000 Cells
   - Nbr_list, PCI values

**Related Features:**
- **FM/PM Collector and Database**
OOF / SON / PCI Use Case

- Config DB (MariaDB) used by PCI-H-MS (step 4b) and OOF (step 7)
- Query API (swagger JSON spec) exposed to other ONAP modules
- cellId needs to be globally unique (assumed eCGI) and align with ONAP YANG model, ORAN, 3GPP
- pnf-name indicates netconf server to be used for interactions regarding cells
- Pnf object (pnf-name, pnf-id) to be aligned with A&AI (A&AI/ConfigDB interaction to be finalized in Dublin release)

<table>
<thead>
<tr>
<th>Cell (Object)</th>
<th>pnf (Object)</th>
<th>ConfigDB API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Format</td>
<td>Attribute</td>
</tr>
<tr>
<td>networkId</td>
<td>string</td>
<td>pnf-name</td>
</tr>
<tr>
<td>cellId</td>
<td>string</td>
<td>cells</td>
</tr>
<tr>
<td>pciValue</td>
<td>uint64</td>
<td>lastModifiedTS</td>
</tr>
<tr>
<td>nbrList</td>
<td>list of cellId</td>
<td></td>
</tr>
<tr>
<td>lastModifiedTS</td>
<td>timestamp</td>
<td></td>
</tr>
<tr>
<td>pnf-name</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET cellList</td>
<td>networkId, ts</td>
<td>List of cellIds</td>
</tr>
<tr>
<td>GET PCI</td>
<td>cellId, ts</td>
<td>PCI Value</td>
</tr>
<tr>
<td>GET nbrList</td>
<td>cellId, ts</td>
<td>List of cellIds and their PCI values</td>
</tr>
<tr>
<td>GET pnf-name</td>
<td>cellId, ts</td>
<td>pnf-name</td>
</tr>
</tbody>
</table>
End to End Network Slicing Use Case

1. OSS/BSS/Apps
   - Standard APIs
   - Internal call
   - NSSMF(s) (ONAP)
   - xNFs

2. OSS/BSS/Apps
   - Standard APIs
   - Internal call
   - NSSMF(s) (ONAP)
   - xNFs

3. OSS/BSS/Apps
   - Standard APIs
   - Internal call
   - NSSMF(s) (ONAP)
   - xNFs

4. OSS/BSS/Apps
   - TMF APIs
   - Internal call
   - NSSMF(s) (ONAP)
   - xNFs

5. OSS/BSS/Apps
   - Standard APIs
   - NSSMF(s) (ONAP)
   - xNFs

3rd party component
## End to End Network Slicing Use Case

<table>
<thead>
<tr>
<th>Component</th>
<th>Network Slice NRM</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetworkSlice</td>
<td>operationalState</td>
<td></td>
</tr>
<tr>
<td>NetworkSlice</td>
<td>administrativeState</td>
<td></td>
</tr>
<tr>
<td>NetworkSlice</td>
<td>serviceProfileList</td>
<td></td>
</tr>
<tr>
<td>NetworkSlice</td>
<td>networkSliceSubnetRef</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>operationalState</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>administrativeState</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>nsInfo</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>sliceProfileList</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>managedFunctionRef</td>
<td></td>
</tr>
<tr>
<td>NetworkSliceSubnet</td>
<td>networkSliceSubnetRef</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>serviceProfileId</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>sNSSAIList</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>pLMNIdList</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>perfReq</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>maxNumberOfUEs</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>coverageAreaTAList</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>uEMobilityLevel</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>resourceSharingLevel</td>
<td></td>
</tr>
<tr>
<td>ServiceProfile</td>
<td>availability</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>sliceProfileId</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>sNSSAIList</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>pLMNIdList</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>perfReq</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>maxNumberOfUEs</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>coverageAreaTAList</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>latency</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>uEMobilityLevel</td>
<td></td>
</tr>
<tr>
<td>SliceProfile</td>
<td>resourceSharingLevel</td>
<td></td>
</tr>
</tbody>
</table>
Executive Summary - This requirement enhances the A1 adapter/interface capabilities provided in Rel 6 as part of 5G/ORAN & 3GPP Standards Harmonization requirement (REQ-38). O-RAN has defined A1 interface specification in the context of the management of 5G RAN elements to provide intent based policies for optimization of the RAN network performance. Planned enhancements for Rel 7 include additional support for managing A1 Policies, multiple A1 targets in the RAN, multi-version support for different A1 targets, and secure TLS communication.
ACCESS TO CPS Database (READ/WRITE):

READ ONLY - Run-Time parameters can be READ by any ONAP platform component and any ONAP plug-in. Examples of ONAP platform components are A&AI, SDC, SDNC etc.

READ/WRITE - Parameters can be READ/WRITE from Controllers, DCAE (future), VES Collector/DMaaP, A&AI, Policy/CLAMP (future) and other components with permission settings.

DEFAULT - SO (future), DCAE, A&AI (indirectly), Controllers (CDS, APPC, SDNC) will have default read/write access to CPS Database.

DEFINABLE - Other components will have default read-only access to Configuration Persistence Service but can be given Read/Write access on a per record basis.

SYNCING NEW xNF ADDED or DELETED (A&AI):

ELEMENT SYNC - Software keeps the A&AI elements with the elements in the RunTime Config DB in Sync. When the network first being established, a GetAllPNFs function from A&AI can be used on startup.

A&AI - A&AI is still the master of valid entities in the network and provides a dynamic view of the assets (xNFs) available to ONAP.

CPS Database - The CPS Database is a master of the associate (exo-inventory) data associated with the entities.

DYNAMIC VIEW - When a xNF appears or is removed from the system, CPS Database records will be added/removed based on A&AI entries.

LOGIC - When a xNF appears is removed there is logic to determine how and when something is to be updated. There is some intelligence to know what elements of update.

INDEXING:

INDEXING - Data Records will be indexed by xNF (VNF, PNF, ANF). It would be an objective to have a similar indexing mechanism as A&AI. May also need an index to be a logical object ID.

RETRIEVAL - How are data records retrieved efficiently. This relates how the records are indexed.
DEPENDENCIES – need to operate

SDC Yang Model (to load schema)
  ability to process & translate yang models into schemas
AAF (intra-ONAP security)
Database implementation for Data Persistence
  (for example MariaDB)

DEPENDENCIES – value added

DMaaP (some use cases to work / indirect dependency)
PNF #106

**Configuration Persistence Service (Run-Time)**

- **PNF #106**
- **VES Event**
  - **DCAE VES Collector**
- **DCAE Analytics**
- **Controller**
  - SDN-R, VF-C, APP-C
- **Other Module**
  - OOF/SON/PCI
- **RunTimeDB**
  - **RECORD ELEMENT**
  - Temperature = 98
  - Source of Truth,
  - "Access",
  - **TAG** = xNF association,
  - Platform ONAP
  - (App-C, ms#10105)
  - (for PNF #106)
  - or Flex-Index
  - PNF #106
A&AI correlated/Index to RunTimeDB
Publish changes in A&AI, notification on DMaaP

Indices into Configuration Persistence Service may also use Flex-Index (such as CellID)
CPS Database (Run-Time View)

RECORD ELEMENT
INDEX = PNF #106
Parameter #1
Parameter #2
Parameter #3
Logical object, Cell #1
  Cell Parameter #1
  Cell Parameter #2
  Cell Parameter #3

RECORD ELEMENT
INDEX = Logical Object #111
Parameter #1
Parameter #2
Parameter #3
State Info X.733
Associations
  { Logical Object #111 Cell #2 }
Cardinality Rules
Linking Restrictions

PNF #106

Cell #1 – Logical Object
Cell #2
Cell #3
CPS Information Model Design R8

Proposed new description:
This is an abstract class that extends the RootEntity class and represents characteristics and behaviors of concepts that pertain to the environment managed by ONAP.

Proposed description:
A template which describes a resource in terms of deployment and operational behavior requirements.

Needs to be subclassed for a particular model language. The information in the subclass must be sufficient to catalogue the models in a meaningful way for microservices.

Proposed new description:
An active entity managed by ONAP and described by a DesignEntity.

Proposed description (moved from ResourceDesc):
Represents physical and non-physical (virtual) components which are owned / managed by the business or provided by a Supplier and are used (directly or indirectly) to construct services.

Could be any kind of data, e.g., CM, PM, FM...

ResourceDescHasResourceAppModel

ResourceInstHasData

ResInstDataDescribedByResAppModel

ResourceDesc

ResourceInstance

ResourceInstanceHasData

ResourceInstanceAppData

ResourceDescHasResourceAppModel

ResourceInstDescribedByResourceDesc

Future

ResourceAppModel

NetworkFunctionDesc

isSharable

NetworkFunctionInstance

type

role

Experimental

The Linux Foundation

ONAP