|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | INTERNATIONAL TELECOMMUNICATION UNION  **TELECOMMUNICATION STANDARDIZATION SECTOR**  STUDY PERIOD 2017-2020 | | Focus Group on Autonomous Networks (FG-AN) | |
| AN-I-298-R1 | |
| Original: English | |
| **Question(s):** | | N/A | February 2023 (TBC) | |
| **INPUT DOCUMENT** | | | | |
| **Source:** | | FUTMinna, Nigeria | | |
| **Title:** | | Report on activities for Build-a-thon 2022 from Tech Rangers team | | |
| **Contact:** | | Ebeledike Frank Chukwubuikem  Computer Engineering Student  FUTMinna | | Email: frankcebeledike@gmail.com |
| **Contact:** | | Emmanuel Othniel Eggah  Computer Engineering Student  FUTMinna | | Email: emmanueleggah@gmail.com |
| **Contact:** | | Abel Oche Moses  Computer Engineering Student  (FUTMinna, Nigeria) | | Email: oche.m1902995@st.futminna.edu |
| **Contact**: | | Dr James Agajo (Associate Professor)  PhD (Telecommunication and Computer Engineering)  Head of Department, Computer Engineering  (FUTMinna, Nigeria) | | Email: james.agajo@futminna.edu.ng |

|  |  |
| --- | --- |
| **Keywords:** | 5G, Artificial Intelligence, build-a-thon, Challenge, closed loop, controller, hackathon, Machine Learning, Proof of concept |
| **Abstract:** | This contribution provides a report on activities by Tech Rangers team towards the Build-a-thon 2022. We analyse AN-usecase-001 [FGAN-use cases], “Import and export of knowledge in an autonomous network”, produce a design as per the reference design in the Build-a-thon repository. We also provide the corresponding code based on the reference code in the Build-a-thon 2022 repository. |

1. **References**

[FGAN-use cases] ITU-T Focus Group Autonomous Networks Technical Specification   
 “Use cases for Autonomous Networks”   
 <https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Use-case-AN.pdf>

[Build-a-thon 2022] <https://github.com/vrra/FGAN-Build-a-thon-2022>

[FG AN Arch framework] Architecture framework for Autonomous Networks,   
 <https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf>

<https://hkrtrainings.com/what-is-tosca> <https://alien4cloud.github.io/#/documentation/3.0.0/devops_guide/normative_types/tosca_concepts_types_normative_nodes.html>

[FGAN Arch] ITU-T Focus Group on Autonomous Networks Technical Specification

“Architecture framework for Autonomous Networks” <https://www.itu.int/en/ITU-T/focusgroups/an/Documents/Architecture-AN.pdf>

[TOSCA v1.3] TOSCA Simple Profile in YAML Version 1.3, OASIS Standard, 26 February  
 2020,<https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-Simple-Profile-YAML-v1.3.pdf>

[Pydroid] <https://www.makeuseof.com/install-pydroid-android/>

[WINEST Github] [https://github.com/Winest-Nigeria/fgan-I-298-R1](https://www.makeuseof.com/install-pydroid-android/)

[FGAN-Build-a-thon] ITU-T FGAN Build-a-thon report <https://extranet.itu.int/sites/itu-t/focusgroups/an/input/FGAN-I-289-R1.docx>

1. **Acronyms**

AN Autonomous Networks

AI Artificial Intelligence

ML Machine learning

POC Proof of Concept

TOSCA Topology and Orchestration Specification for Cloud Applications

1. **Introduction**

This report was written by students from Federal University of Technology Minna towards the upcoming Build a thon 2022 activity. The team has produced TOSCA YAML file corresponding to AN-usecase-001 from [FGAN-use cases] and used xOpera to parse and verify the YAML file.

The TOSCA metamodel uses the concept of service templates that describe cloud workloads as a topology template, which is a graph of node templates modelling the components a workload is made up of and of relationship templates modelling the relations between those components. TOSCA service template are instantiated at runtime using a TOSCA orchestrator (xOpera) and the order of component instantiation is based on the relationship between components. TOSCA is one of best and most used automated testing tools. It is widely employed in large-scale and small-scale applications to achieve successful outcomes.

In the TOSCA Simple Profile, TOSCA service templates must always have, as the first line in its YAML file, the keyword “tosca\_definitions\_version” with an associated TOSCA Namespace Alias value.

In this contribution, we described the use case, the approach taken by the team in designing the use case and the results from our experiments. For this, we took the following steps:

(a) Analyse the use case (manual step)

(b) Create neo4j sub graph (manual step)

(c) Create YAML (using Ruamel)

(d) Parse and deploy the YAML using xOpera

NOTE – steps (c) and (d) have been successfully done using [Pydroid] and Google colab environment.

1. **Problem statement**

Due to the increasing numbers of data that needs to be processed by networks, manual techniques for management cannot handle the complexity of the network as a result there is a need for an autonomous network that runs with minimal to no human intervention—able to configure, monitor, and maintain itself independently and is required to process those data at high speed with low latency and high accuracy in other to maintain a smooth running of the system.

With the adoption of a network that collects various network data and uses AI algorithms to learn so it has a certain degree of intelligence the need for proper management and handling of data by the system is a necessity. As the requirements for intelligence become more and more specific, the amount of network data that needs to be collected increases exponentially. With limited computing resources, if data is processed only through the AI algorithm, it will be difficult for the network to provide the intelligence of self-management and control. To get out of this difficulty, it is necessary to make use of knowledge within the network and outside the network, including the data from human experience and discovered through the AI technology.

To this end, the team carried out a research on how to build a close-loop system that manages its resources and we used the TOSCA simple profile in YAML to design the close loop prototype. We studied the AN-usecase-001 from [FGAN-use cases] and implemented a PoC for the Import and export of knowledge in an autonomous network.

The Figure 1 is

Fig 3 of [FGAN-Build-a-thon], (see below), shows the representation of modules and controller descriptions.

Diagram

Description automatically generated

Figure 1: Problem statement steps [FGAN-I-289-R1].

General use case scenarios comprise of the following steps:

1. Knowledge is imported from outside or peer entities of the AN components
2. Knowledge is referred internally in the AN components, e.g., for driving evolution, driving exploration, configuration of automation loops.
3. Generate report for human consumption
4. Knowledge is stored and updated within the AN components
5. Knowledge is exported from the AN components to outside or peer entities.

As shown in Figure 2, the team selected Usecase-001 from the FG AN use case document which is titled “Import and export of knowledge in an autonomous network”. The use case was carefully studied, from which the team came up with a design (as shown in Figure 4) which was implemented in TOSCA YAML. The design was validated by parsing it using the xOpera open-source orchestrator. This was tested on the Google Colab platform. Also, the team has shown how this can be replicated on android devices [see Appendix II].

Figure 2: Problem statement steps [FGAN-I-289-R1].

Also, the team was able to show how the YAML file can be generated from the use case stored in a graph database (an example is using neo4j). To achieve this, we loaded the use case description into our Neo4j database. We used Ruamel (a python YAML editing library) to populate the service template nodes of the use case by querying the Neo4j database to retrieve our use case actors/nodes. Finally, the generated YAML was then validated by parsing it using xOpera.

These steps are shown in Figure 3.

Graphical user interface, text, application

Description automatically generated

Figure 3: Flow chart of the service template generation from the graphDB.

1. **Approach taken by the team**

The approach adopted by our team to develop the close loop are:

1. Populate Neo4j database with the actors and relationships from our use case in TOSCA.

Here we are using python to populate our Neo4j database with the actors and corresponding relationships. We also used the Ruamel.yaml python package which is a YAML parser/emitter to create a YAML file from Neo4j database.

1. Creating a YAML file which is made up of seven nodes (AN\_Orchestrator\_0, KB\_manager\_0, KB\_0, Auto\_controller\_generator\_0, OpenCN\_0, ML\_pipeline\_0 and Human\_Operator\_0).

Link to Demo: <https://youtu.be/eL1-1vCSiMM>

1. **High-level flow chart**

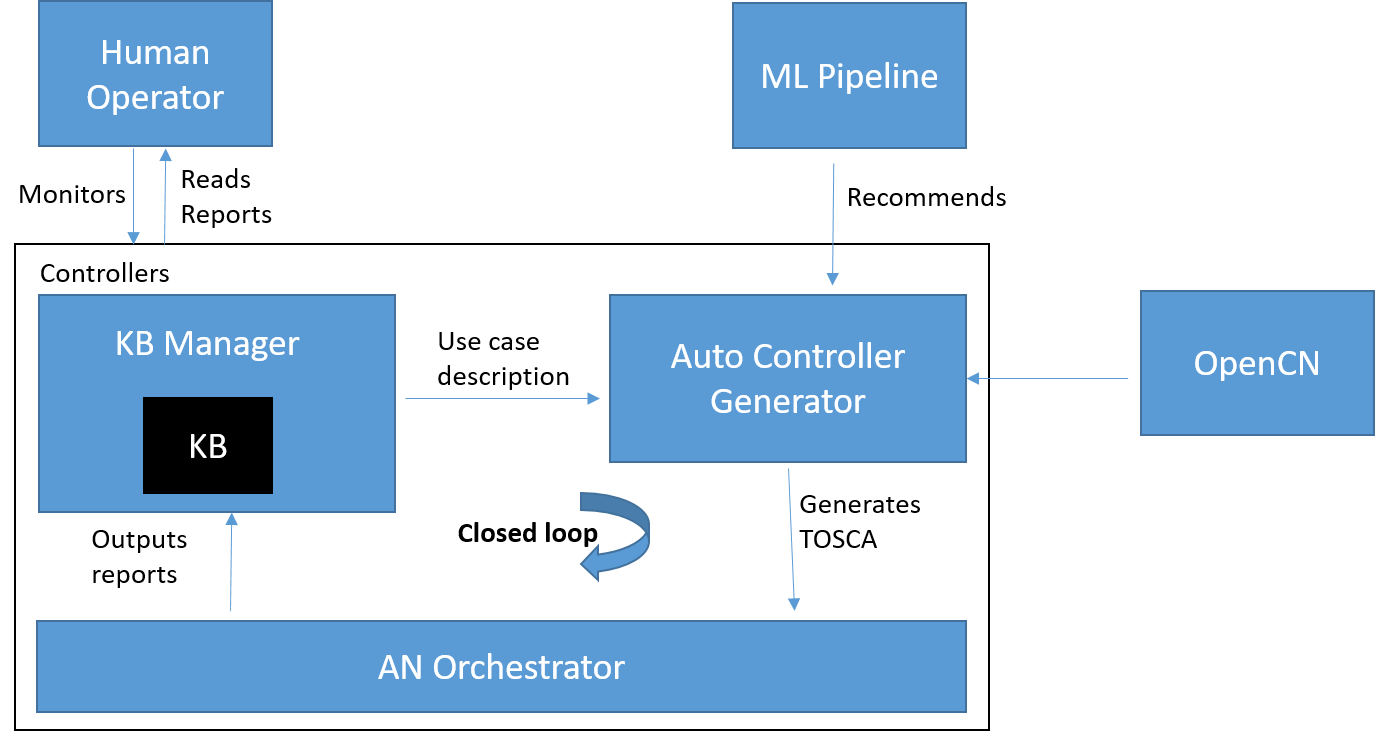
****

Figure 4: High level flow chart of use case design

**AN\_orchestrator:**

As per [FGAN Arch], AN orchestrator is the component responsible for managing workflows and processes in the AN and steps in the lifecycle of controllers. To manage the workflows and processes in AN, AN orchestrator coordinates with various other functions in the AN as well as outside the AN.

Being part of the management plane, AN orchestrator provides interface to human operators in the form of reports regarding the functioning of the AN and human interfaces for configuring the AN, where applicable.

**KB\_manager**:

As per [FGAN Arch], KB manager is a subsystem which manages storage, querying, export, import and optimization and update knowledge, including that derived from different sources including structured or unstructured data from various components or other subsystems.

KB manager is a node which optimizes and manages data available on the close\_loop but requires a node which can host its resources. Since AN\_orchestrator has the capability of hosting node resources it is designed to be a host to the KB\_ manager.

**KB**:

As per [FGAN Arch], Knowledge in AN is a collection of resources that helps in solving a specific type of problem. A knowledge base component manages knowledge derived from and used in autonomous networks. It is updated and accessed by various components in the autonomous network.

Knowledge includes metadata which is derived from the capabilities, status of AN components. This knowledge is stored and exchanged as part of interactions of AN components with knowledge base. Knowledge can be derived from different sources including structured or unstructured data

**Auto\_controller\_Generator**:

This node represent generic software component that can be managed and run by a TOSCA Compute Node Type. It generate controller specifications using the existing repository in OpenCN, the knowledge base and an analytics function aided by AI/ML

**Open Cn**:

This node stores the controllers for the AN.

**ML\_pipeline**:

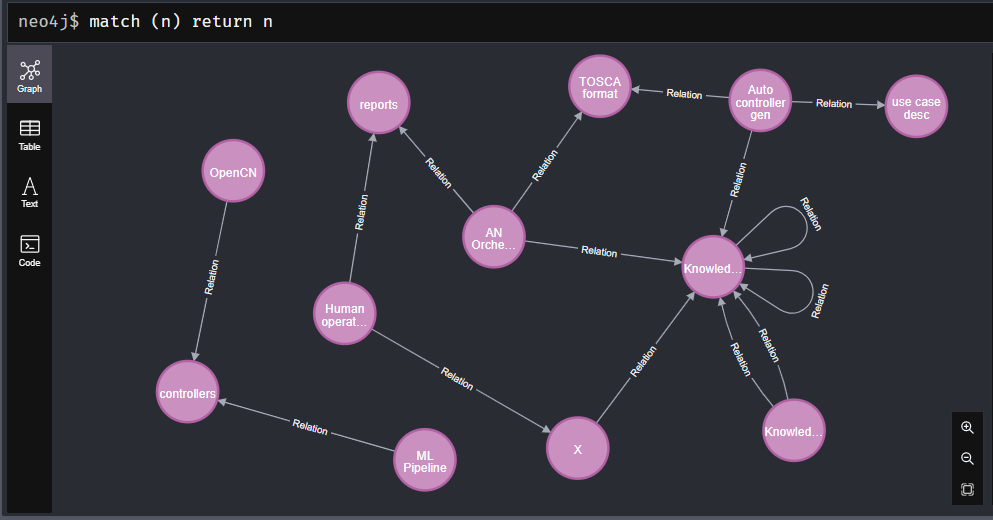
This node is able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw inference from patterns in data. It hosts analytics and recommends controllers in the AN

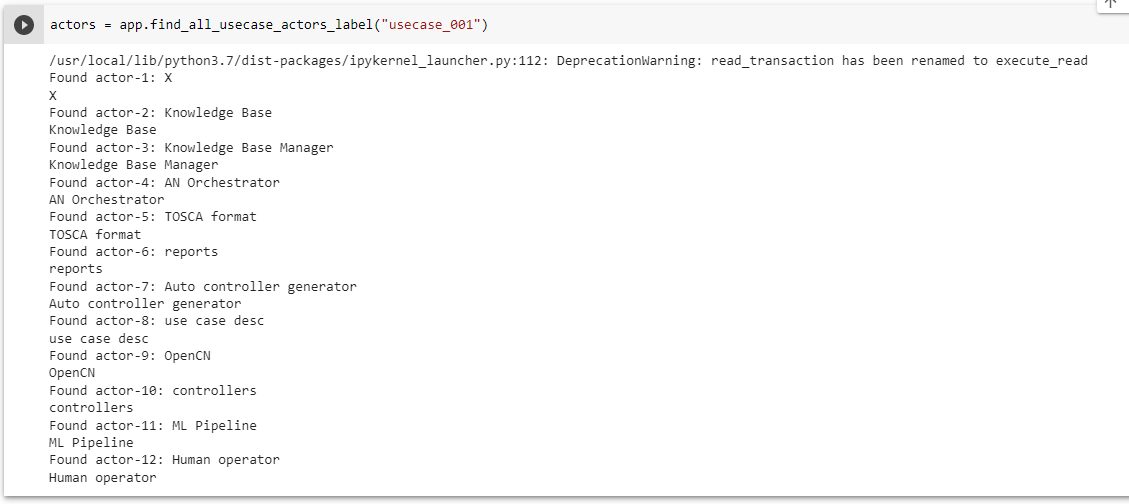
**Human\_operator**:

This node is just like a user friendly interface for human base instructions

1. **PoC: results**
2. refer to the appendix for the YAML
3. screenshot for the neo4j graph

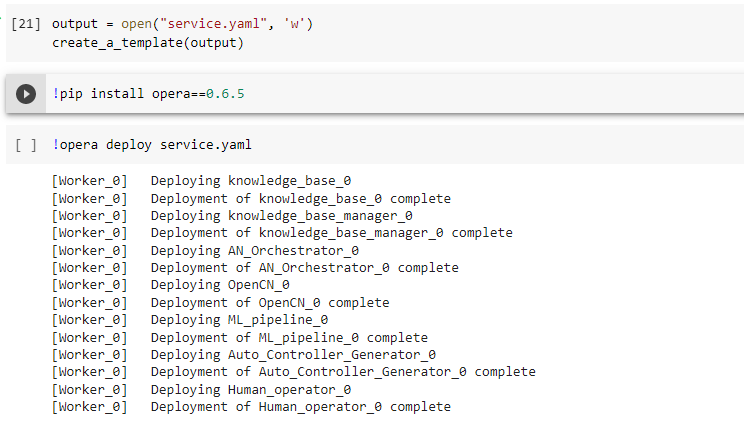






2. screenshot for the compile with xOpera





1. **Problems encountered.**

The major problem encountered during this activity are:

* Indentation of the YAML syntax
* Spelling errors due to syntax case sensitivity
* The version of YAML available on the device and the required version need to parse the files

1. **Future activities:**

This contribution provides a report on activities by Tech Rangers team towards the Build-a-thon 2022. We analyse AN-usecase-001 [Y.suppl 71], “Import and export of knowledge in an autonomous network”, produce a design as per the reference design in the Build-a-thon repository. We also provide the corresponding code based on the reference code in the Build-a-thon 2022 repository.

At the time of this report the focus was on translating AN-usecase-001 to a YAML file from Neo4j Graphs using a Python script. in the future generating a YAML file for any usecase from Neo4j will be developed.

**Appendix: YAML file**

tosca\_definitions\_version: tosca\_simple\_yaml\_1\_3

relationship\_types:

basicrelationship:

derived\_from: tosca.relationships.Root

topology\_template:

node\_templates:

Knowledge Base:

type: tosca.nodes.SoftwareComponent

description: Stores knowledge related to the AN.

requirements:

- export:

node: Knowledge Base

relationship: exports

- import:

node: Knowledge Base

relationship: imports

Knowledge Base Manager:

type: tosca.nodes.SoftwareComponent

requirements:

- export:

node: Knowledge Base

relationship: exports

- optimize:

node: Knowledge Base

relationship: optimizes

AN Orchestrator:

type: tosca.nodes.Compute

requirements:

- refer:

node: Knowledge Base

relationship: refersTo

- generates\_Tosca:

node: Auto controller generator

relationship: generates

Auto controller generator:

type: tosca.nodes.SoftwareComponent

description: Generate controller specifications

requirements:

- dependency: ML Pipeline

OpenCN:

type: tosca.nodes.SoftwareComponent

description: Stores controllers

ML Pipeline:

type: tosca.nodes.SoftwareComponent

description: Hosts analytics

Human operator:

type: tosca.nodes.SoftwareComponent

description: Reads reports and monitors

relationship\_templates:

refersTo:

type: basicrelationship

optimizes:

type: basicrelationship

exports:

type: basicrelationship

stores:

type: basicrelationship

recommends:

type: basicrelationship

imports:

type: basicrelationship

reads:

type: basicrelationship

monitors:

type: basicrelationship

inputs:

type: basicrelationship

generates:

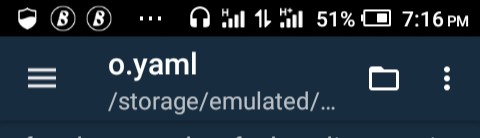
type: basicrelationship

Appendix: **How to install opera on Pydroid**

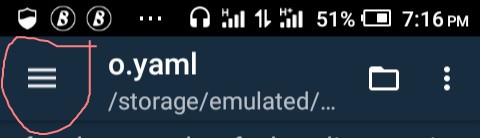
Since Pydriod is an IDE for python, to be able to parse YAML file on this platform there is a need to install a library called opera. This installation can be done on either terminal or on pip.

**Steps for pip installation**

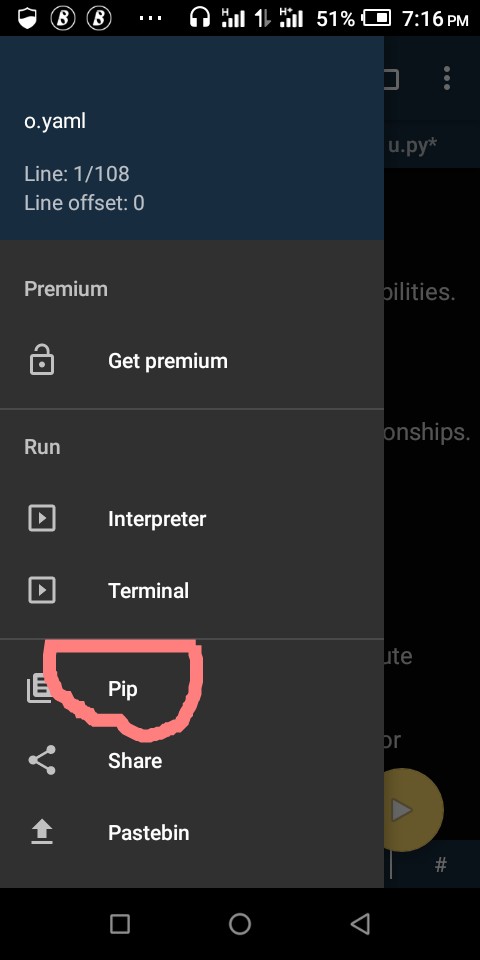
1. Open Pydriod [https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=en&gl=US] on your mobile.



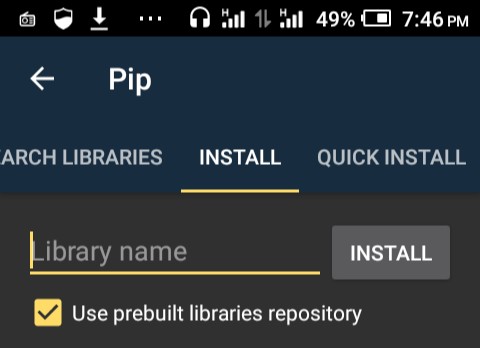
1. Click on the three horizontal lines at the top left hand side.



1. Click on pip

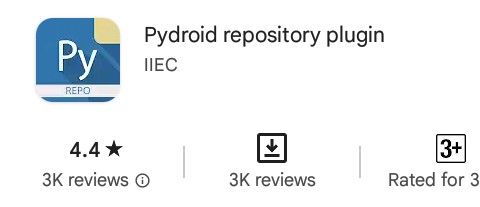


1. Type opera and click install



You may be redirected if you don't have the repository plug-in install on your phone.

To avoid this install pydroid repository plug-in from play store.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_