Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENCS3130

Report

Python Project

Prepared by: Omar Hamayel 1220356

Partner: Qusay Bdier 1220649

Teaching Assistant: Loor Wael Sawalhi

Date: 1/2/2025

1- Abstract:

This report serves to compare study gNMI (gRPC Network Management Interface) against CLI (Command Line Interface) systems regarding data collection. It focuses chiefly on developing tools in Python for validating and normalizing data between both interfaces. Some central functionalities are output parsing, normalization, and comparison of data representations between gNMI and CLI. The report goes further to include the structure of the scripts, such as main execution flow, normalization functions, techniques for data comparison, and the use of data structures to map and simulate. Results show that the tool is very effective in dissecting differences and producing overall reporting for help in troubleshooting and validating tasks in network management.

This document is about a comparative study gNMI (gRPC Network Management Interface) with CLI (Command Line Interface) systems to collect data. Primarily it focuses on the development of a tool in Python that would be used to validate and normalize data from both interfaces. Some of the key functions of the tool include output parsing, a normalization function, and comparison of gNMI and CLI datasets. The report goes on to explain the structure of the scripts with a particular focus on core execution flow, normalization functions, comparison techniques, and usage of data structures for mapping and simulation.

The result indicates effectiveness on the tool's part in the dissection of differences and provides holistic reports comparatively assisting in network management while troubleshooting and validating tasks.

Table of Contents

Table of Figures

2- code explain:

2.1 Main Execution:

The principal execution logic is what brings this script together-to allow interaction on inputting an gNM path against which the comparison must be done. It first checks the input for validation against the PATH_TO_CLI dictionary whether it is a defined path and corresponds to CLI commands. If it is indeed valid, the script then calls this function: generate_report(path), which does a comparison of output to produce a report. If it is not found, then prints an error message telling the user to give a valid path, though this being an interactive script makes it useful as a diagnostic tool to facilitate a fast, efficient, and pinpoint analysis of gNMI versus CLI data for troubleshooting or validation.

2.2 Normalization Functions:

Normalization guarantees parity in data gleaned from gNMI and CLI, regardless of whether those data were received in a unique format. The function normalize_case (value) converts all string inputs to lowercase as case insensitive. This is particularly applicable to cases where CLI prints in uppercase but gNMI prints in lower case. Normalize_value: performs an all-inclusive transformation into the template that is supposed to contain the standardized "clean" data. It eliminates unwanted characters like underscores (_), units like "bytes," and symbols like percent signs (%). This function also handles unit conversion: it understands that 1K, 1M, 1G, and 1T represent numeric values in base units such as bytes. For example, 1K means 1000. The function tries to convert the resultant string to type Decimal for correct numerical comparisons. If it can't, it returns the sanitized string. All these steps are meant to ensure that this process can be used to normalize gNMI and CLI output data for comparison.

2.3 Comparison Functions:

The script has very strong functionalities to compare data for gNMI and CLI. The compare_values(key, gnmi_value, cli_value) function brings a single key-value pair from both sources and normalizes these values based on the normalize_value function. Then this function tries to compare the normalized values as numbers with a little tolerance (1e-6) for floating-point differences. If they are not numeric, it falls back on the string comparison; thus, it remains flexible and can deal with any type of data. The second one, compare_outputs(path), is extended onto at this point. Now it compares all key-value pairs at this gNMI path. It retrieves gNMI data, that is, a dictionary, identifies CLI data, then formats text of multiple lines according to this path, retrieves each key-value pair inside all gNMI data and calls that respective value from the CLI output using regex. Such as adjacency cases, which are represented in gNMI as a list of dictionaries and instead in CLI as text, so this function parses and sorts both sources before comparing. It collects and reports all mismatches that were found in the results for further investigation purposes.

2.4 Data Structures:

The interpretation of the script for data-and-relationship major mappings includes three key dictionaries: PATH_TO_CLI dictionary, which is a mappings table that gNMI paths associated with the commands used through CLI; for example, gNMI path, /interfaces/interface[name=eth0]/state/counters, is mapped to the CLI command show interfaces eth0 counters, both producing the same data. Now, It could get this data from either of those locations to make comparisons. Subsequently, the GNMI_OUTPUTS dictionary would provide some JSON-like strings simulating the outputs of gNMI queries. They will consist of formatted key-value pairs, such as: {"in_octets": 1500000, "out_octets": 1400000}, which mean that these are not APIs, but simply specific data points returned as responses from gNMI. Like the GNMI_OUTPUTS dictionary, the CLI_OUTPUTS dictionary would contain strings that would hold multi-line text, simulating the raw output of any CLI command. Some formatting would be added, or slight differences would be seen in how the data was returned in these two simulated outputs. Therefore, with these two dictionaries, the scripts are also capable of mimicking data retrieval in the real world and comparison across sources.

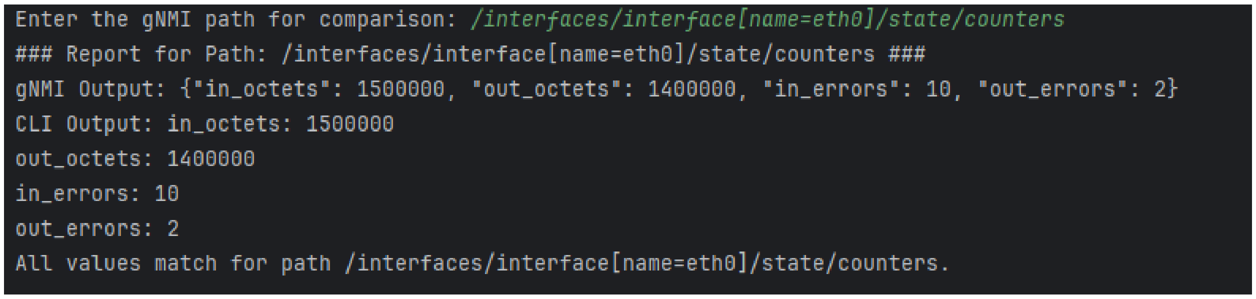
2.5 Report Generation:

Generating report function (path), the same serves as the entry point for preparing a comparison-gNMI based detailed report for a specific gNMI path. At this point, it is providing raw gNMI and CLI outputs for the path so that it can justify comparison. Next, calling the compare_outputs(path) function literally states "all results with respect to that path are matched with direct description as gNMI and CLI outputs consistent" if all values are found equal for Function is thereby wrapping full comparison process for users to get the complete picture analysis for the data in any specific path.

3- Result:

3.1 Requirement 1:

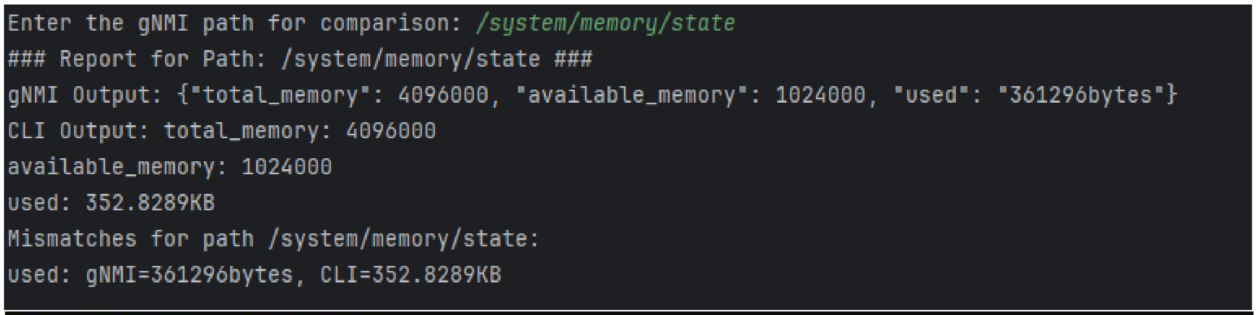
- Path 1: **/interfaces/interface[name=eth0]/state/counters**



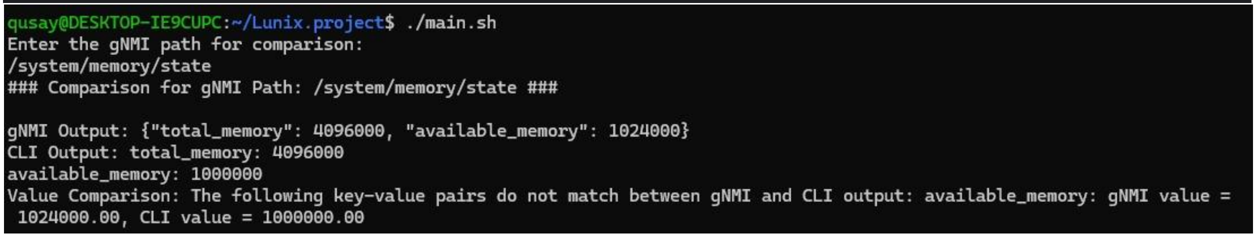
```
Enter the gNMI path for comparison: /interfaces/interface[name=eth0]/state/counters
### Report for Path: /interfaces/interface[name=eth0]/state/counters ###
gNMI Output: {"in_octets": 1500000, "out_octets": 1400000, "in_errors": 10, "out_errors": 2}
CLI Output: in_octets: 1500000
out_octets: 1400000
in_errors: 10
out_errors: 2
All values match for path /interfaces/interface[name=eth0]/state/counters.
```

Figure 1:/interfaces/interface[name=eth0]/state/counters

➢ Path 2: **/system/memory/state**



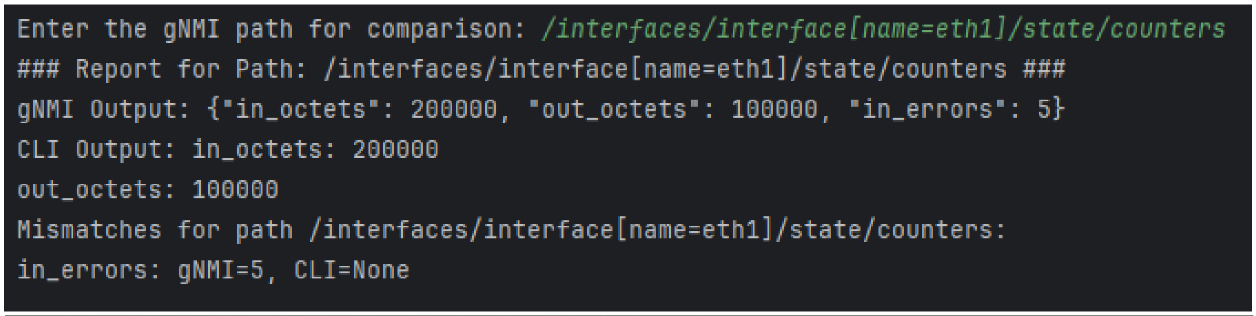
```
Enter the gNMI path for comparison: /system/memory/state
### Report for Path: /system/memory/state ###
gNMI Output: {"total_memory": 4096000, "available_memory": 1024000, "used": "361296bytes"}
CLI Output: total_memory: 4096000
available_memory: 1024000
used: 352.8289KB
Mismatches for path /system/memory/state:
used: gNMI=361296bytes, CLI=352.8289KB
```



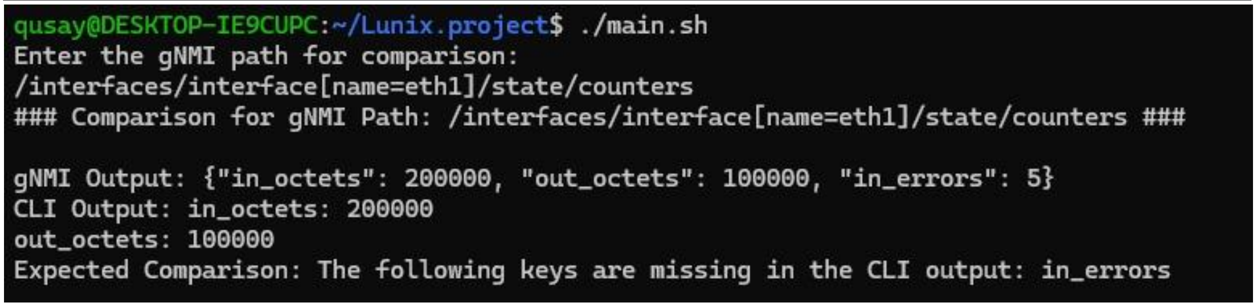
```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/system/memory/state
### Comparison for gNMI Path: /system/memory/state ###

gNMI Output: {"total_memory": 4096000, "available_memory": 1024000}
CLI Output: total_memory: 4096000
available_memory: 1000000
Value Comparison: The following key-value pairs do not match between gNMI and CLI output: available_memory: gNMI value =
 1024000.00, CLI value = 1000000.00
```

Figure 2:/system/memory/state

➢ Path 3: **/interfaces/interface[name=eth1]/state/counters**



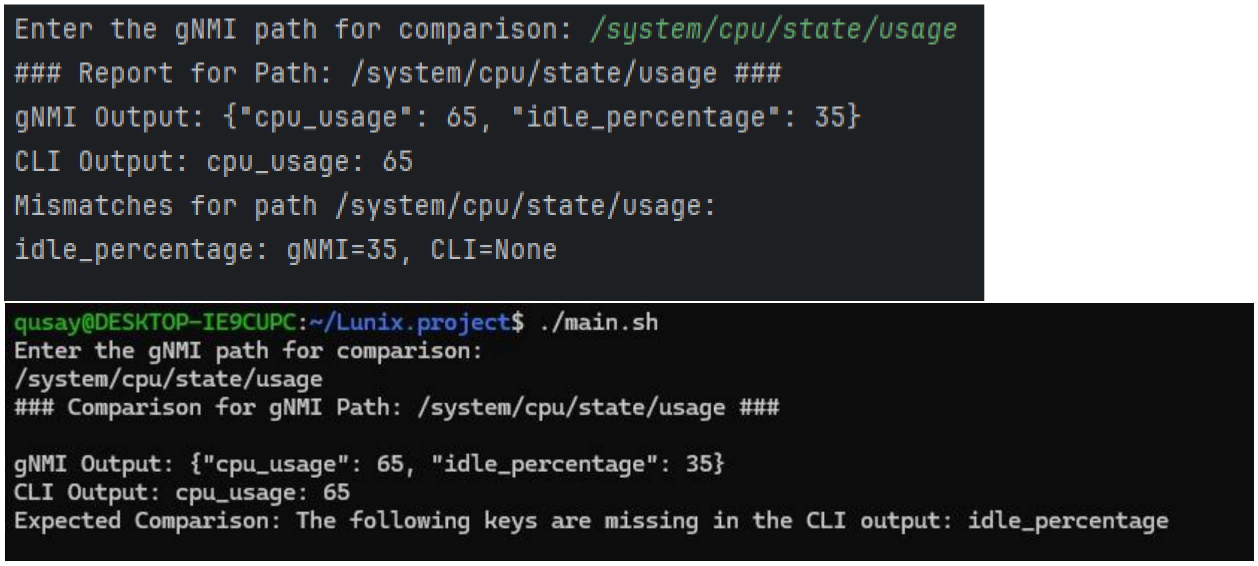
```
Enter the gNMI path for comparison: /interfaces/interface[name=eth1]/state/counters
### Report for Path: /interfaces/interface[name=eth1]/state/counters ###
gNMI Output: {"in_octets": 200000, "out_octets": 100000, "in_errors": 5}
CLI Output: in_octets: 200000
out_octets: 100000
Mismatches for path /interfaces/interface[name=eth1]/state/counters:
in_errors: gNMI=5, CLI=None
```



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/interfaces/interface[name=eth1]/state/counters
### Comparison for gNMI Path: /interfaces/interface[name=eth1]/state/counters ###

gNMI Output: {"in_octets": 200000, "out_octets": 100000, "in_errors": 5}
CLI Output: in_octets: 200000
out_octets: 100000
Expected Comparison: The following keys are missing in the CLI output: in_errors
```

Figure 3:/interfaces/interface[name=eth1]/state/counters

➢ Path 4: **/system/CPU/state/usage**

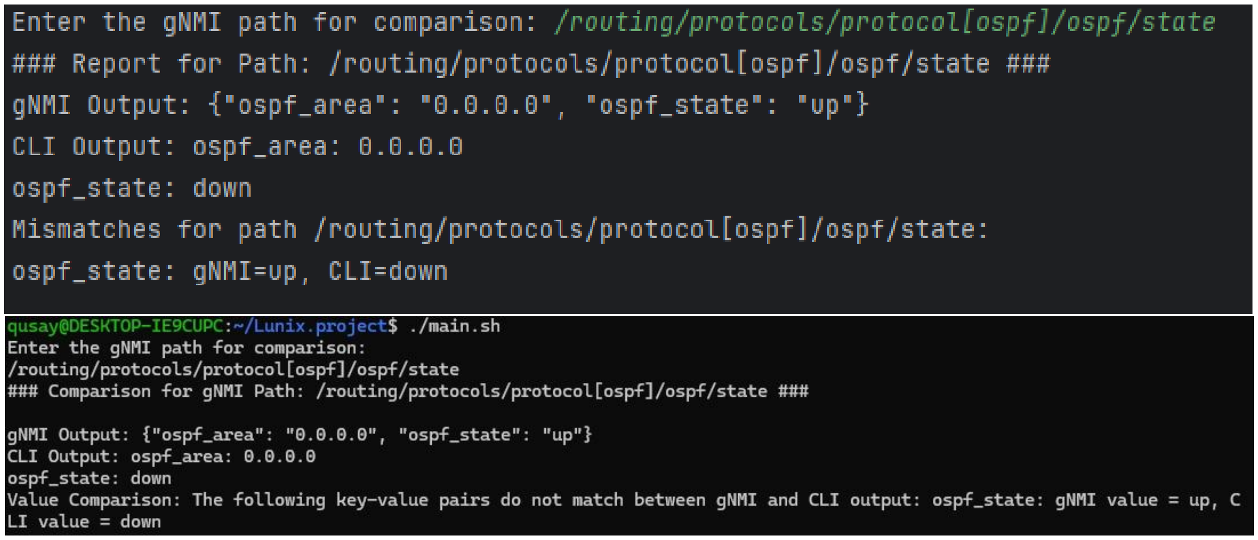


```
Enter the gNMI path for comparison: /system/cpu/state/usage
### Report for Path: /system/cpu/state/usage ###
gNMI Output: {"cpu_usage": 65, "idle_percentage": 35}
CLI Output: cpu_usage: 65
Mismatches for path /system/cpu/state/usage:
idle_percentage: gNMI=35, CLI=None
```

```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/system/cpu/state/usage
### Comparison for gNMI Path: /system/cpu/state/usage ###

gNMI Output: {"cpu_usage": 65, "idle_percentage": 35}
CLI Output: cpu_usage: 65
Expected Comparison: The following keys are missing in the CLI output: idle_percentage
```

Figure 4:/system/CPU/state/usage

➢ Path 5: **/routing/protocols/protocol[ospf]/ospf/state**



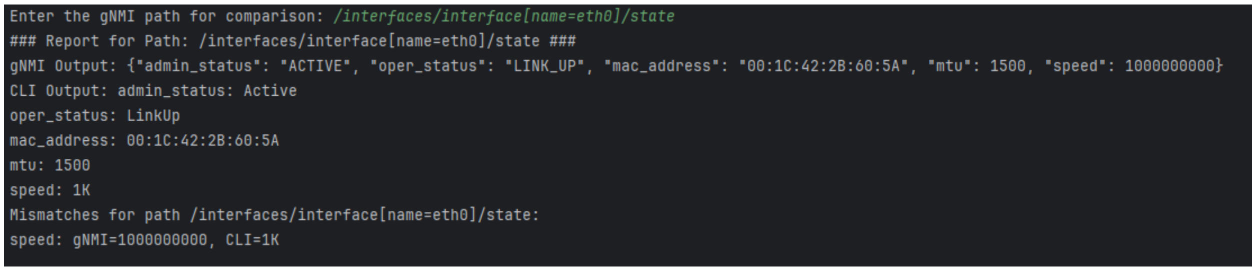
```
Enter the gNMI path for comparison: /routing/protocols/protocol[ospf]/ospf/state
### Report for Path: /routing/protocols/protocol[ospf]/ospf/state ###
gNMI Output: {"ospf_area": "0.0.0.0", "ospf_state": "up"}
CLI Output: ospf_area: 0.0.0.0
ospf_state: down
Mismatches for path /routing/protocols/protocol[ospf]/ospf/state:
ospf_state: gNMI=up, CLI=down
```

```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/routing/protocols/protocol[ospf]/ospf/state
### Comparison for gNMI Path: /routing/protocols/protocol[ospf]/ospf/state ###

gNMI Output: {"ospf_area": "0.0.0.0", "ospf_state": "up"}
CLI Output: ospf_area: 0.0.0.0
ospf_state: down
Value Comparison: The following key-value pairs do not match between gNMI and CLI output: ospf_state: gNMI value = up, C
LI value = down
```

Figure 5:/routing/protocols/protocol[ospf]/ospf/state

3.2 Requirement 2:

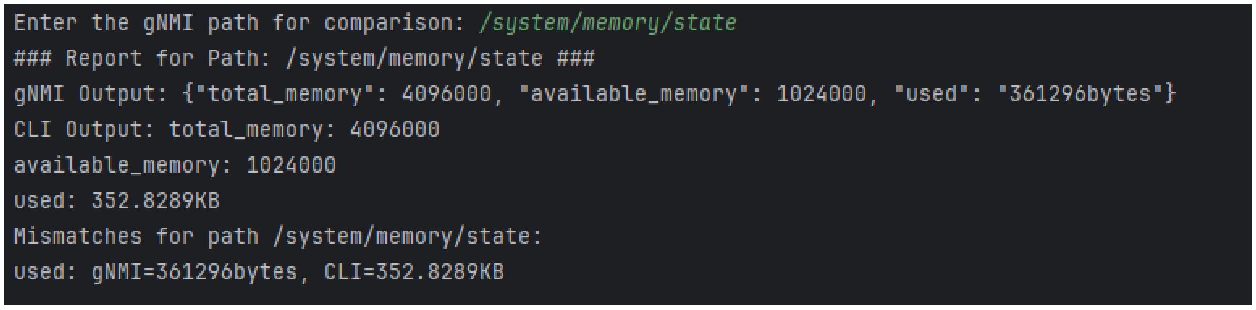
- Path 1: **/interfaces/interface[name=eth0]/state**



```
Enter the gNMI path for comparison: /interfaces/interface[name=eth0]/state
### Report for Path: /interfaces/interface[name=eth0]/state ###
gNMI Output: {"admin_status": "ACTIVE", "oper_status": "LINK_UP", "mac_address": "00:1C:42:2B:60:5A", "mtu": 1500, "speed": 1000000000}
CLI Output: admin_status: Active
oper_status: LinkUp
mac_address: 00:1C:42:2B:60:5A
mtu: 1500
speed: 1K
Mismatches for path /interfaces/interface[name=eth0]/state:
speed: gNMI=1000000000, CLI=1K
```

Figure 6:/interfaces/interface[name=eth0]/state

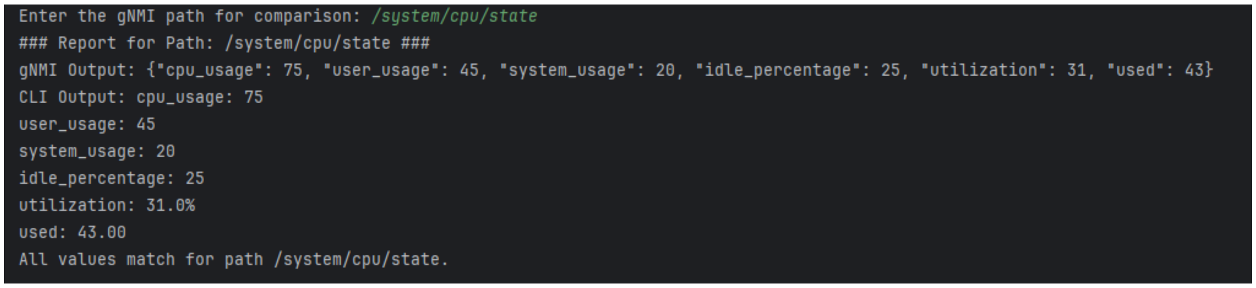
- Path 2: **/system/memory/state**



```
Enter the gNMI path for comparison: /system/memory/state
### Report for Path: /system/memory/state ###
gNMI Output: {"total_memory": 4096000, "available_memory": 1024000, "used": "361296bytes"}
CLI Output: total_memory: 4096000
available_memory: 1024000
used: 352.8289KB
Mismatches for path /system/memory/state:
used: gNMI=361296bytes, CLI=352.8289KB
```

Figure 7:/system/memory/state

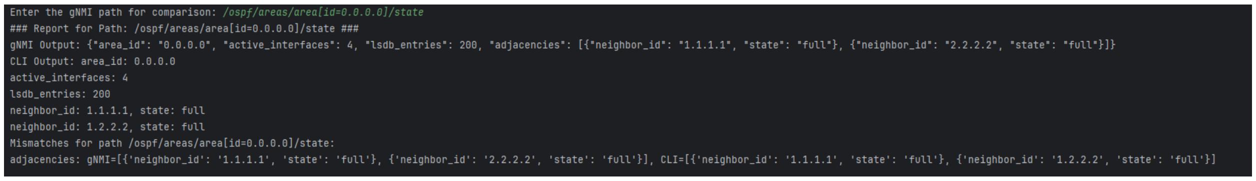
- Path 3: **/system/cpu/state**



```
Enter the gNMI path for comparison: /system/cpu/state
### Report for Path: /system/cpu/state ###
gNMI Output: {"cpu_usage": 75, "user_usage": 45, "system_usage": 20, "idle_percentage": 25, "utilization": 31, "used": 43}
CLI Output: cpu_usage: 75
user_usage: 45
system_usage: 20
idle_percentage: 25
utilization: 31.0%
used: 43.00
All values match for path /system/cpu/state.
```

Figure 8:/system/cpu/state

- Path 3: **/ospf/areas/area[id=0.0.0.0]/state**



```
Enter the gNMI path for comparison: /ospf/areas/area[id=0.0.0.0]/state
### Report for Path: /ospf/areas/area[id=0.0.0.0]/state ###
gNMI Output: {"area_id": "0.0.0.0", "active_interfaces": 4, "lsdb_entries": 200, "adjacencies": [{"neighbor_id": "1.1.1.1", "state": "full"}, {"neighbor_id": "2.2.2.2", "state": "full"}]}
CLI Output: area_id: 0.0.0.0
active_interfaces: 4
lsdb_entries: 200
neighbor_id: 1.1.1.1, state: full
neighbor_id: 1.2.2.2, state: full
Mismatches for path /ospf/areas/area[id=0.0.0.0]/state:
adjacencies: gNMI=[{'neighbor_id': '1.1.1.1', 'state': 'full'}, {'neighbor_id': '2.2.2.2', 'state': 'full'}], CLI=[{'neighbor_id': '1.1.1.1', 'state': 'full'}, {'neighbor_id': '1.2.2.2', 'state': 'full'}]
```

Figure 9:/ospf/areas/area[id=0.0.0.0]/state

4- Conclusion:

This developed tool proves to be a very efficient bridge connecting the outcomes received through gNMI as well as CLI output. This is done by providing a fairly strong base on which data comparison and validation can be carried out. The tool enables accurate and reliable analyses even when data formats differ or representation of tests is achieved through normalization and tolerance-based methods. The detail reports generated from this tool become an asset while diagnosing mismatches to ensure network integrity. In this way, troubleshooting is made easy, reliability improved in data, and good practice follows an efficient way of managing the network. Future work could address broadening the range of supported paths and further optimization of the software in respect to real-time applications.