Faculty of Engineering & Technology
Electrical & Computer Engineering Department

ENCS3130

Report

Shell Script Project

Prepared by: Omar Amjad Hamayel 1220356

Partner: Qusay Bdier 1220649

Teaching Assistant: Loor Wael Sawalhi

Date: 8/12/2024

1- Abstract:

This program is designed to compare outputs from various network devices obtained through gNMI (gRPC Network Management Interface) and CLI (Command Line Interface). It generates comprehensive reports highlighting any discrepancies or missing data. The core of the script includes logic to prompt the user for the gNMI path and validate it against a mapping of gNMI paths to corresponding CLI commands. Upon receiving valid input, the program produces a comparison report.

The data fetching section consists of a function that retrieves and displays outputs from both gNMI and CLI based on the provided path. The data comparison section includes functions that parse and normalize the values from these outputs. This normalization process accommodates differences in formatting and units, particularly for critical values like bytes, KB, and MB, as well as precision for MAC addresses and other relevant data.

The program utilizes associative arrays to establish mappings between gNMI paths and their corresponding CLI commands (PATH_TO_CLI), as well as mock data for gNMI (GNMI_OUTPUTS) and CLI (CLI_OUTPUTS). The report generation component creates a detailed comparison report that clearly identifies mismatched keys, values, or any extra/missing keys between the gNMI and CLI outputs. The script also effectively handles edge cases, such as missing data or discrepancies in key-value pairs, ensuring a thorough comparison of both data sources.

Table of Contents

Table of Figures

2- code explain:

2.1 main.sh:

The Bash script is designed to compare gNMI (gRPC Network Management Interface) data with outputs generated from the CLI (Command Line Interface) on network devices. It accomplishes this by sourcing four external script files: `gnmi_cli_config.sh` for configuration settings, `data_fetcher.sh` for data retrieval functions, `data_comparator.sh` for comparison logic, and `report_generator.sh` for generating reports.

Upon execution, the script prompts the user to enter a gNMI path, which could refer to specific data points such as interface counters or system status. It then verifies the validity of the provided path by checking it against the `PATH_TO_CLI` associative array for the corresponding CLI commands. If the path is valid, the `generate_report` function is called to create a comparison report. Conversely, if the path is invalid, the user receives an error message indicating that a valid gNMI path must be entered. This approach ensures that only valid paths initiate the comparison process, while invalid entries are flagged for correction.

2.2 data_fetcher.sh:

The `fetch_data()` function is designed to accept a path and return outputs from both gNMI and CLI commands associated with that path. It takes a single argument, `gnmi_path`, which specifies the data points to be retrieved.

Initially, the function checks if the provided `gnmi_path` exists in the `GNMI_OUTPUTS` associative array, which contains the outputs of gNMI commands. If the path is found, the corresponding gNMI output is printed; if not, a message is displayed indicating that the gNMI output is not available for the given path.

Next, the function performs a similar check for the `CLI_OUTPUTS` array, which holds the CLI command outputs corresponding to the same gNMI paths. If the path exists in `CLI_OUTPUTS`, the relevant CLI output is printed; otherwise, a message is shown stating that the CLI output does not exist for that specific path.

This approach ensures that both gNMI and CLI outputs are retrieved and displayed, along with appropriate messages for any missing data.

2.3 data_comparator.sh:

The revolutionary gscript is designed for comparing data from gNMI (gRPC Network Management Interface) and CLI (Command Line Interface). It features functionalities for parsing and normalizing data, including functions like `parse_json_keys()` and `parse_json_value()`, which utilize regular expressions to extract keys and values from JSON data. Additionally, it can extract MAC addresses as needed.

The `normalize_case()` function converts strings to lowercase, while `convert_to_bytes()` handles various units such as KB, MB, and GB. The `normalize_value()` function standardizes data by removing underscores, converting units, and preserving percentage signs.

At the core of the script are the comparison functions. The `compare_outputs()` function evaluates the keys and values between gNMI and CLI outputs, identifying any missing or extra keys, as well as value mismatches. This comparison is enhanced by normalizing the values from both outputs.

Example functions like `extract_gnmi_adjacencies()` and `extract_cli_adjacencies()` are responsible for extracting adjacency data from both results and performing immediate comparisons. The `compare_outputs2()` function further extracts and compares specific elements such as `area_id`, `active_interfaces`, `lsdb_entries`, and `adjacencies`, reporting any differences found.

In summary, this script effectively compares structured gNMI and CLI data, normalizes values for consistency, and generates reports detailing variations or agreements in the comparisons.

2.4 gnmi_cli_config.sh:

Three associative arrays are utilized to store data for comparing gNMI and CLI outputs: `PATH_TO_CLI`, `GNMI_OUTPUTS`, and `CLI_OUTPUTS`.

The `PATH_TO_CLI` associative array establishes a mapping between gNMI paths and their corresponding CLI commands. This mapping enables the script to determine which CLI commands to execute based on a specific gNMI path.

The `GNMI_OUTPUTS` array contains responses in JSON format from gNMI, reflecting various operational states of a network device, such as interface counters, memory usage, and CPU statistics. In contrast, the `CLI_OUTPUTS` array holds the CLI output for the same paths, formatted as strings with key-value pairs typically separated by colons.

These arrays are designed to facilitate the comparison of similar parameters from a network device through both gNMI and CLI. For example, gNMI data for an interface may provide a detailed JSON object that specifies octet counts and error counts, while the CLI data presents this information in a less structured format as a string of key-value pairs.

The script compares these outputs by normalizing the values—such as converting units or removing extraneous characters—and identifying discrepancies in keys or values. In summary, this comparison process enhances confidence in the consistency between gNMI and CLI outputs, which is essential for troubleshooting and validating the status of network devices.

2.5 report_generator.sh:

The script features a function named `generate_report`, which is designed to accept a gNMI path as input and produce a comprehensive comparison report between the gNMI output and the corresponding CLI output. By utilizing the provided gNMI path, the function retrieves the relevant outputs from the `GNMI_OUTPUTS` and `CLI_OUTPUTS` associative arrays.

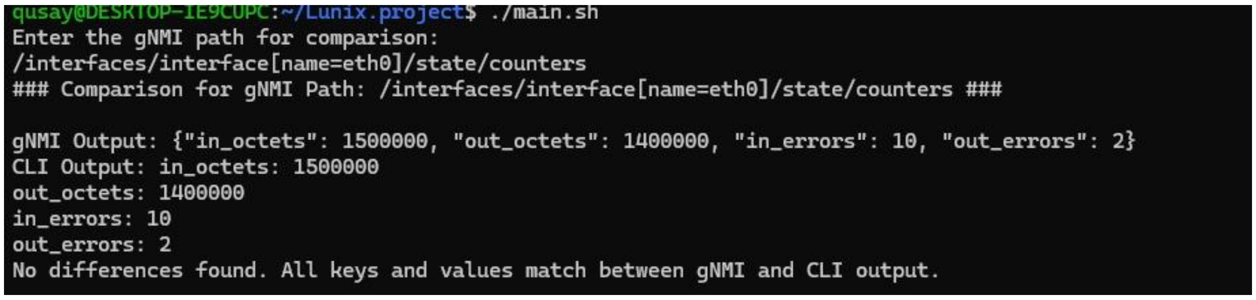
Initially, the function prints the gNMI path along with the respective outputs for both gNMI and CLI. It then checks whether the specified gNMI path is `/ospf/areas/area[id=0.0.0.0]/state`. If the path does not match this specific one, the function proceeds to call `compare_outputs` to perform the comparison. However, if the path corresponds to the specific OSPF area, it invokes `compare_outputs2`, which is tailored to handle the unique comparison logic required for OSPF area data.

To enhance clarity, an empty line is printed at the end, indicating the conclusion of the report for that particular path. The primary goal of this function is to automate the comparison of gNMI and CLI outputs, reflecting the configurations and status of network devices, and to help identify any discrepancies that may arise during configuration.

3- Result:

3.1 Requirement 1:

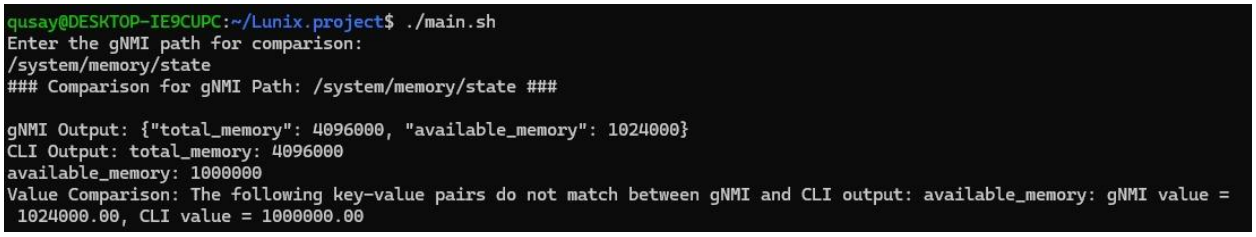
- Path 1: **/interfaces/interface[name=eth0]/state/counters**



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/interfaces/interface[name=eth0]/state/counters
### Comparison for gNMI 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
No differences found. All keys and values match between gNMI and CLI output.
```

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

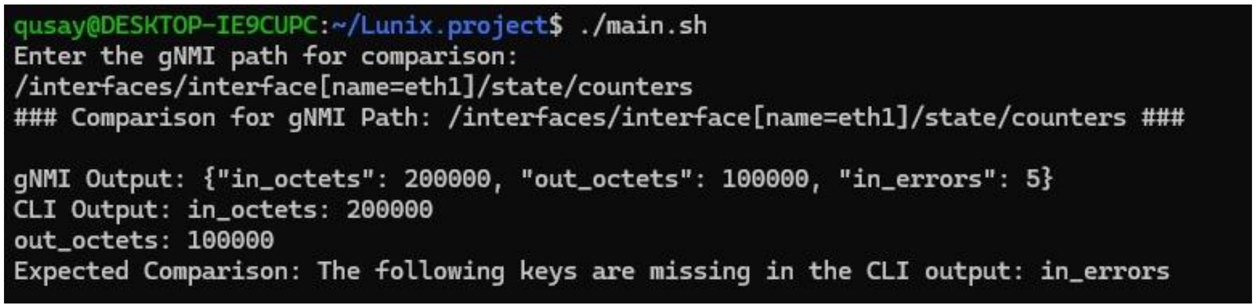
- Path 2: **/system/memory/state**



```
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

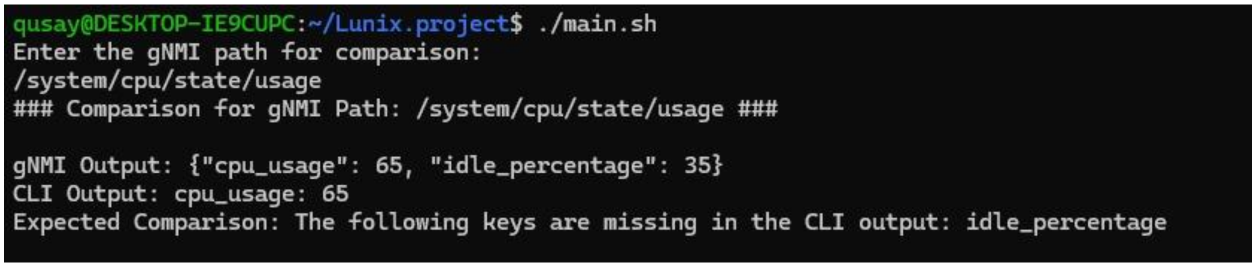
- Path 3: **/interfaces/interface[name=eth1]/state/counters**



```
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

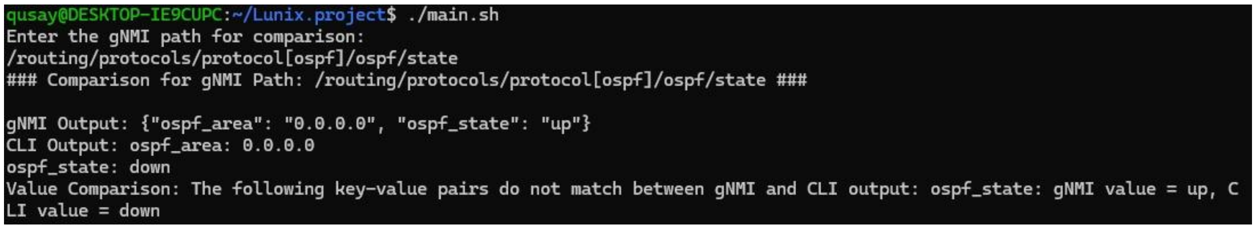
- Path 4: **/system/CPU/state/usage**



```
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**

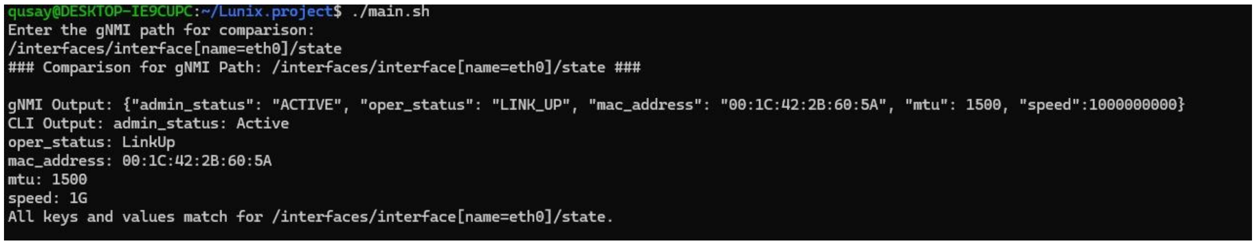


```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/routing/protocols/protocol[ospf]/ospf/state
### Comparison for gNMI Path: /routing/protocols/protocol[ospf]/ospf/state ###

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
```

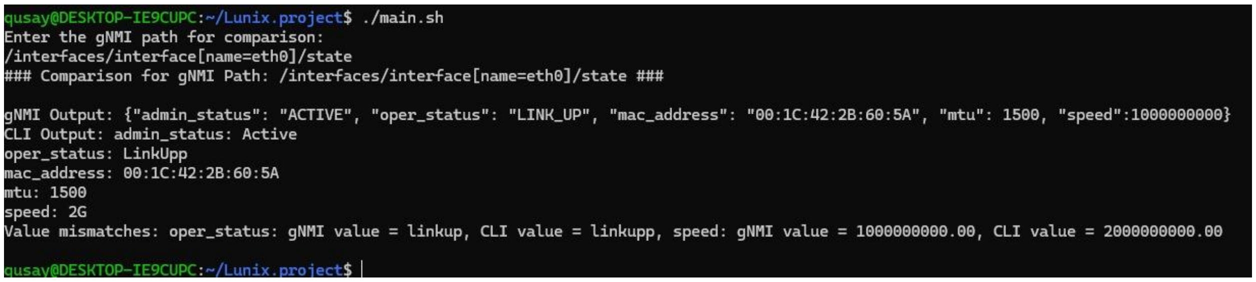
Figure 5:/routing/protocols/protocol[ospf]/ospf/state

3.2 Requirement 2:

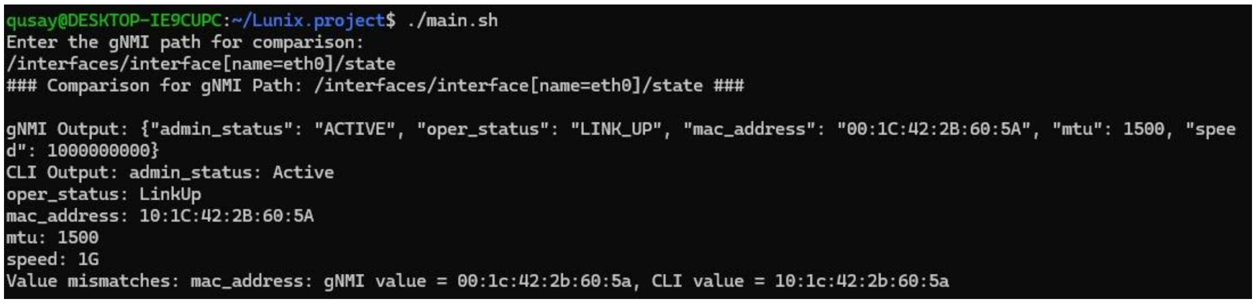
- Path 1: **/interfaces/interface[name=eth0]/state**



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/interfaces/interface[name=eth0]/state
### Comparison for gNMI 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: 1G
All keys and values match for /interfaces/interface[name=eth0]/state.
```



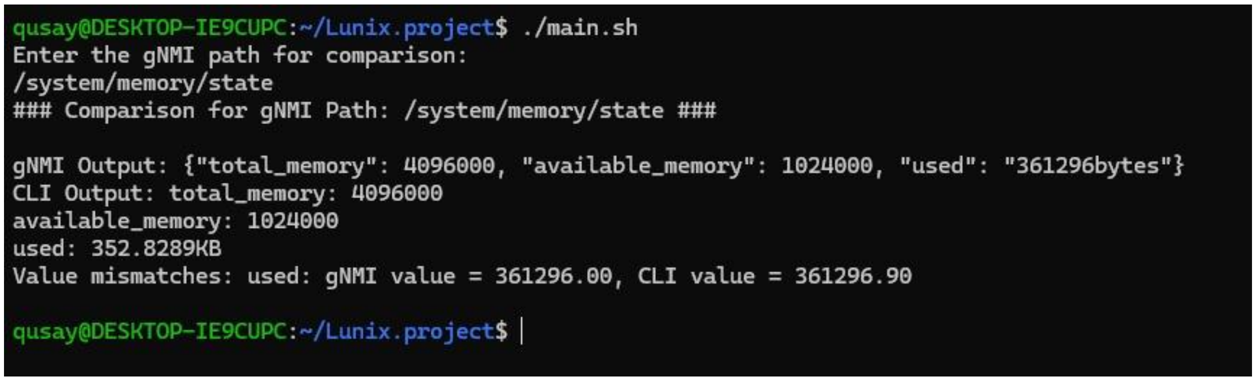
```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/interfaces/interface[name=eth0]/state
### Comparison for gNMI Path: /interfaces/interface[name=eth0]/state ###

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: LinkUpp
mac_address: 00:1C:42:2B:60:5A
mtu: 1500
speed: 2G
Value mismatches: oper_status: gNMI value = linkup, CLI value = linkupp, speed: gNMI value = 1000000000.00, CLI value = 2000000000.00

qusay@DESKTOP-IE9CUPC:~/Lunix.project$ |
```



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/interfaces/interface[name=eth0]/state
### Comparison for gNMI 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, "spee
d": 1000000000}
CLI Output: admin_status: Active
oper_status: LinkUp
mac_address: 10:1C:42:2B:60:5A
mtu: 1500
speed: 1G
Value mismatches: mac_address: gNMI value = 00:1c:42:2b:60:5a, CLI value = 10:1c:42:2b:60:5a
```

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

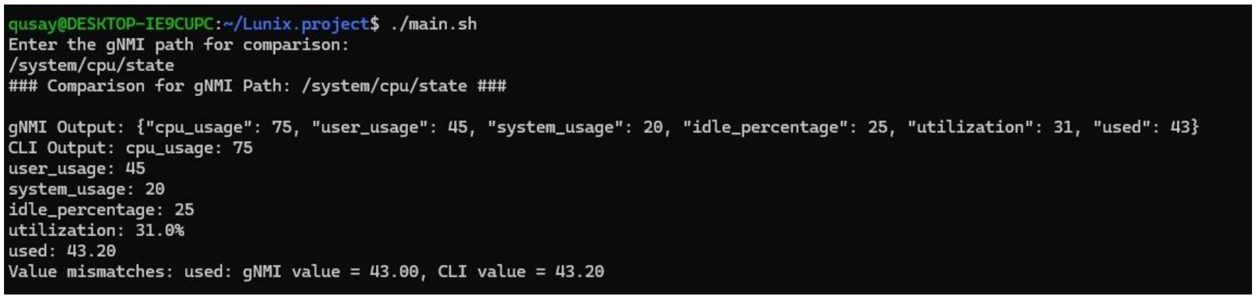
- Path 2: **/system/memory/state**



```
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, "used": "361296bytes"}
CLI Output: total_memory: 4096000
available_memory: 1024000
used: 352.8289KB
Value mismatches: used: gNMI value = 361296.00, CLI value = 361296.90

qusay@DESKTOP-IE9CUPC:~/Lunix.project$ |
```

Figure 7:/system/memory/state

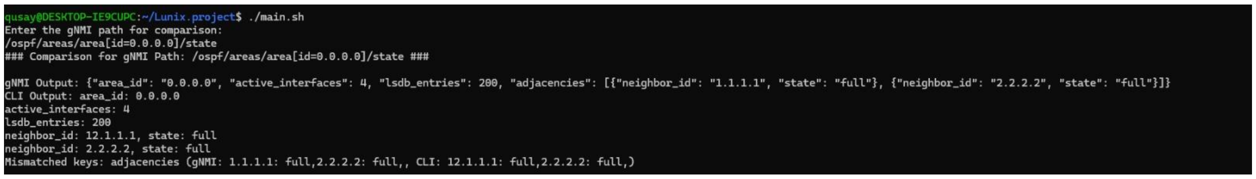
- Path 3: **/system/cpu/state**



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/system/cpu/state
### Comparison for gNMI 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: 43.20
Value mismatches: used: gNMI value = 43.00, CLI value = 43.20
```

Figure 8:/system/cpu/state

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



```
qusay@DESKTOP-IE9CUPC:~/Lunix.project$ ./main.sh
Enter the gNMI path for comparison:
/ospf/areas/area[id=0.0.0.0]/state
### Comparison for gNMI 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: 12.1.1.1, state: full
neighbor_id: 2.2.2.2, state: full
Mismatched keys: adjacencies (gNMI: 1.1.1.1: full,2.2.2.2: full,, CLI: 12.1.1.1: full,2.2.2.2: full,)
```

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

4- Conclusion:

The bridging script introduced here effectively integrates gNMI and CLI outputs, creating a platform-agnostic approach for comparing and validating network device data. By utilizing associative arrays, data normalization techniques, and advanced comparison logic, the script thoroughly identifies mismatches, missing values, and other inconsistencies in the reported data.

Currently, the script is modularized into distinct functions for data fetching, comparison, and report generation, making it scalable and adaptable for various network environments. This modularity not only enhances the script's flexibility but also aids in troubleshooting network issues, ensuring the integrity and reliability of device configurations and operational states.