Birzeit University

Department of Electrical & Computer Engineering

Second Semester, 2024/2025

ENCS3130 Linux Laboratory

Shell Scripting Project – gNMI-CLI Path Verification and Data Comparison

---

Project Description

This project aims to develop a **Python-based tool** for verifying data accuracy between gNMI telemetry data and CLI command outputs for network device configurations and operational states. The tool will be implemented using Python classes and functions to ensure modularity and maintainability. Basically the same as Project1 idea just implemented in python.

Functional Requirements

1. **gNMI Query Execution**
    - Use a provided gNMI path to execute a gNMI query and retrieve the output in a structured format (e.g., JSON).
    - Implement query execution using a Python class to handle gNMI operations.
2. **CLI Command Mapping**
    - Map the gNMI path to a corresponding CLI command.
    - A Python function or class will maintain a mapping dictionary or configuration for this purpose.
3. **Data Comparison**
    - Compare the output from gNMI queries with CLI command outputs.
    - The tool should identify:
        - Values present in gNMI output but missing or differing in CLI output.
        - Fields in gNMI output that are populated but should match empty/null CLI fields or vice versa.
    - A Python class will encapsulate logic for parsing and comparing data.
4. **Report Generation**
    - Generate a discrepancy report summarizing the following:
        - Missing values in CLI or gNMI outputs.

        - Fields with mismatched values.
        - Fields that are populated in one source but empty/null in the other.
    - A dedicated Python function will handle report formatting and generation.

Implementation Approach

- Use Python classes to encapsulate functionality for gNMI operations, CLI mapping, data comparison, and report generation.
- Ensure modular design by breaking the tool into well-defined classes and methods, making it easier to understand, debug, and extend.

Evaluation Criteria for Linux Lab

1. **Functionality**
    - Ability to fetch gNMI data, execute CLI commands, and perform accurate comparisons.
    - Correct handling of discrepancies and accurate report generation.
2. **Code Quality**
    - Use of Python classes and functions for modular and maintainable code.
3. **Usability**
    - Ease of use, including clear input mechanisms and meaningful error messages.
4. **Performance**
    - Efficient processing of gNMI and CLI data, with acceptable execution time for typical inputs.
5. **Documentation**
    - Clear documentation for setup, usage, and code structure.
    - Examples or test cases included for demonstration.
6. **Testing**
    - Thorough testing of all functionality, including edge cases.
    - Validation against sample data to ensure accuracy.
7. **Report Quality**
    - Clear and detailed discrepancy reports that are easy to interpret.
8. **Robustness**
    - Resilience to invalid inputs or unanticipated errors.

By implementing the tool in Python and adhering to the criteria above, the project will provide a robust solution for identifying data mismatches and improving network telemetry reliability.

Example gNMI Paths and CLI Commands

Here's a sample mapping of gNMI paths to CLI commands, along with hypothetical outputs for comparison:

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

        {

        "in_octets": 1500000,

        "out_octets": 1400000,

        "in_errors": 10,

        "out_errors": 2

        }
    - **CLI Command**: show interfaces eth0 counters
    - **CLI Output**:

        in_octets: 1500000

        out_octets: 1400000

        in_errors: 10

        out_errors: 2
    - **Expected Comparison**: All values match; no discrepancies.
2. **Path**: /system/memory/state
    - **gNMI Output**:

        {

        "total_memory": 4096000,

        "available_memory": 1024000

        }
    - **CLI Command**: show memory
    - **CLI Output**:

        makefile

        Copy code

        total_memory: 4096000

        available_memory: 1000000

- **Expected Comparison**: available_memory differs between gNMI and CLI outputs.

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

     {

      "in_octets": 200000,

      "out_octets": 100000,

      "in_errors": 5

     }
   - **CLI Command**: show interfaces eth1 counters
   - **CLI Output**:

     in_octets: 200000

     out_octets: 100000
   - **Expected Comparison**: in_errors appears in the gNMI output but is missing in the CLI output.
4. **Path**: /system/cpu/state/usage
   - **gNMI Output**:

     {

      "cpu_usage": 65,

      "idle_percentage": 35

     }
   - **CLI Command**: show cpu
   - **CLI Output**:

     cpu_usage: 65
   - **Expected Comparison**: idle_percentage is present in the gNMI output but missing in the CLI output.
5. **Path**: /routing/protocols/protocol[ospf]/ospf/state
   - **gNMI Output**:

     {

      "ospf_area": "0.0.0.0",

      "ospf_state": "up"

     }

- **CLI Command**: show ospf status
- **CLI Output**:

  ospf_area: "0.0.0.0"

  ospf_state: "down"
- **Expected Comparison**: ospf_state differs, showing "up" in gNMI and "down" in CLI output.

Requirement 1:

In some cases, a single **gNMI path might require multiple CLI commands to cover** all the values provided by the gNMI output, especially when the telemetry data involves comprehensive state or configuration details. Here are some examples where multiple CLI commands are needed to match all fields in the gNMI output.

Example gNMI Paths and Multiple CLI Command Mappings

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

```
{
 "admin_status": "up",
 "oper_status": "up",
 "mac_address": "00:1C:42:2B:60:5A",
 "mtu": 1500,
 "speed": 1000
}
```

   - **CLI Commands**:
     - **Command 1**: show interfaces eth0 status

       admin_status: up

       oper_status: up
     - **Command 2**: show interfaces eth0 mac-address

       mac_address: 00:1C:42:2B:60:5A
     - **Command 3**: show interfaces eth0 mtu

       mtu: 1500
     - **Command 4**: show interfaces eth0 speed

speed: 1000

- **Expected Comparison**: Each CLI command provides a subset of the values in the gNMI output, allowing complete coverage for a thorough comparison.

---

2. **Path**: /bgp/neighbors/neighbor[neighbor_address=10.0.0.1]/state
   - **gNMI Output**:

```
{
 "peer_as": 65001,
 "connection_state": "Established",
 "received_prefix_count": 120,
 "sent_prefix_count": 95
}
```

   - **CLI Commands**:
     - **Command 1**: show bgp neighbors 10.0.0.1

       peer_as: 65001

       connection_state: Established
     - **Command 2**: show bgp neighbors 10.0.0.1 received-routes

       received_prefix_count: 120
     - **Command 3**: show bgp neighbors 10.0.0.1 advertised-routes

       sent_prefix_count: 95
   - **Expected Comparison**: Each CLI command focuses on a specific aspect of the neighbor's state, allowing you to validate all fields in the gNMI output against the CLI output.

---

3. **Path**: /system/cpu/state
   - **gNMI Output**:

```
{
 "cpu_usage": 75,
 "user_usage": 45,
 "system_usage": 20,
 "idle_percentage": 25
}
```

- **CLI Commands**:
  - **Command 1**: show cpu usage

    cpu_usage: 75
  - **Command 2**: show cpu user

    user_usage: 45
  - **Command 3**: show cpu system

    system_usage: 20
  - **Command 4**: show cpu idle

    idle_percentage: 25
- **Expected Comparison**: All fields in the gNMI output are covered by executing multiple CLI commands to capture each specific metric.

---

4. **Path**: /ospf/areas/area[id=0.0.0.0]/state
   - **gNMI Output**:

```
json

Copy code

{

 "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 Commands**:
     - **Command 1**: show ospf area 0.0.0.0

```
makefile

Copy code

area_id: 0.0.0.0

active_interfaces: 4
```

lsdb_entries: 200

- **Command 2**: show ospf neighbors

yaml

Copy code

neighbor_id: 1.1.1.1, state: full

neighbor_id: 2.2.2.2, state: full

- **Expected Comparison**: Combining both commands allows you to cover all values in the gNMI output, including the detailed adjacency states for OSPF neighbors.

---

5. **Path**: /system/disk/state
   - **gNMI Output**:

```
{
 "total_space": 1024000,
 "used_space": 500000,
 "available_space": 524000,
 "disk_health": "good"
}
```

   - **CLI Commands**:
     - **Command 1**: show disk space

       total_space: 1024000

       used_space: 500000

       available_space: 524000
     - **Command 2**: show disk health

       disk_health: good
   - **Expected Comparison**: The two commands together provide all values needed for complete coverage of the gNMI data.

Requirement 2:

handle discrepancies in units and formats between gNMI and CLI outputs, the script can include conversion and normalization logic. Here's an approach to managing these common types of formatting differences:

1. **Case Normalization:** Handle cases where values differ only in case or format. For example:
    - LINK_UP (gNMI) vs. LinkUp (CLI)
    - ACTIVE (gNMI) vs. Active (CLI)
    - **Solution**: Normalize both values to lowercase before comparison.
2. **Unit Parsing and Conversion:** Detect and convert different unit representations to a consistent format. For example:
    - 400 (gNMI) vs. 400G (CLI)
    - 361296 (gNMI, in bytes) vs. 352.97 KB (CLI, in kilobytes)
    - **Solution**: Add logic to parse and convert units as necessary. For numeric values with units, extract and standardize the units (e.g., convert KB to bytes).
3. **Decimal Handling:** Handle differences in decimal precision:
    - 31 (gNMI) vs. 31.0% (CLI)
    - 43 (gNMI) vs. 43.00 (CLI)
    - **Solution**: Round both values to a common precision or remove insignificant decimal places before comparison.

Additional Workflow Steps to handle discrepancies in units and formats

1. **Normalize Case**: Convert both gNMI and CLI values to lowercase to ensure case-insensitive comparisons.
2. **Extract Units**: Identify if a CLI output has units attached (e.g., "400G," "352.97 KB") and separate them from the numeric value. Map units for a standard representation.
3. **Convert Units**: Implement a conversion function that converts different units to a standardized base unit (e.g., bytes for data size, Mbps for bandwidth, percentage without a decimal). For example:
    - 400G (CLI) → 400 * 10^9 (standardized as bytes)
    - 352.97 KB (CLI) → 361296 bytes
    - 31.0% (CLI) → 31%
4. **Adjust Precision**: If values differ in decimal places (e.g., 43 vs. 43.00), round or truncate values to a common precision before comparison.
5. **Generate Enhanced Report**: The comparison report will now include notes about adjustments made for unit and format consistency, providing transparency on discrepancies.

Example Enhanced Comparison with gNMI Paths and CLI Commands

1. **Path**: /interfaces/interface[name=eth0]/state/oper-status
    - **gNMI Output**: LINK_UP
    - **CLI Command**: show interfaces eth0 status
        - **CLI Output**: LinkUp
    - **Normalized Comparison**: link_up (gNMI) vs. linkup (CLI)
    - **Result**: Match after normalization.
2. **Path**: /interfaces/interface[name=eth0]/state/admin-status
    - **gNMI Output**: ACTIVE
    - **CLI Command**: show interfaces eth0 admin-status
        - **CLI Output**: Active
    - **Normalized Comparison**: active (gNMI) vs. active (CLI)
    - **Result**: Match after normalization.
3. **Path**: /interfaces/interface[name=eth0]/state/speed
    - **gNMI Output**: 400
    - **CLI Command**: show interfaces eth0 speed
        - **CLI Output**: 400G
    - **Unit Conversion**: 400 Mbps (gNMI) vs. 400 * 10^9 bps (CLI)
    - **Result**: Match after conversion.
4. **Path**: /system/memory/state/used
    - **gNMI Output**: 361296 bytes
    - **CLI Command**: show memory used
        - **CLI Output**: 352.97 KB
    - **Unit Conversion**: 361296 bytes (gNMI) vs. 352.97 * 1024 bytes (CLI)
    - **Result**: Match after conversion.
5. **Path**: /system/cpu/state/utilization
    - **gNMI Output**: 31
    - **CLI Command**: show cpu utilization
        - **CLI Output**: 31.0%
    - **Precision Adjustment**: 31 (gNMI) vs. 31.0 (CLI)
    - **Result**: Match after adjusting precision.
6. **Path**: /system/storage/state/used

- **gNMI Output**: 43
- **CLI Command**: show storage usage
  - **CLI Output**: 43.00
- **Precision Adjustment**: 43 (gNMI) vs. 43.00 (CLI)
- **Result**: Match after adjusting precision.

Submission:

Please submit the following:

1. Python script program
2. Report: The report must include the following components:

- The code: Provide the complete shell script code.
- Report : explain your test cases, implementation and output screen shot with examples.

Notes:

- Write the code for the python to satisfy the specifications described above.
- Make sure your code is clean and well indented; variables have meaningful names, etc.
- Make sure your script has enough comments inserted to add clarity.
- Work in groups of at most two students
- Deadline**: Sunday, 5 January 2025 at 11:59pm**. Please submit your project (code + report) through Ritaj as a reply to this message.
- This project is per group effort: instances of cheating will result in you failing the lab.