WIPRO ASSIGNMENT

DONE BY:

PRAKASH TR

**ACTIVITY1:**

RESTCONF Protocol  
• Python Activity: Develop Python scripts to perform RESTCONF operations using the requests library, such as GET, POST, PUT, and DELETE.

**PROGRAM:**

import requests

import json

from requests.auth import HTTPBasicAuth

# RESTCONF server details

BASE\_URL = "https://192.168.1.1/restconf/data"  # Replace with your device's IP and RESTCONF base path

USERNAME = "admin"  # Replace with your username

PASSWORD = "admin"  # Replace with your password

# Headers for RESTCONF requests

HEADERS = {

    "Content-Type": "application/yang-data+json",

    "Accept": "application/yang-data+json"

}

# Suppress SSL warnings (for self-signed certificates)

requests.packages.urllib3.disable\_warnings()

def get\_interfaces():

    """Performs a GET request to retrieve interface data."""

    url = f"{BASE\_URL}/ietf-interfaces:interfaces"

    response = requests.get(url, headers=HEADERS, auth=HTTPBasicAuth(USERNAME, PASSWORD), verify=False)

    if response.status\_code == 200:

        print("GET Response:")

        print(json.dumps(response.json(), indent=2))

    else:

        print(f"GET request failed with status {response.status\_code}: {response.text}")

def create\_interface(interface\_name, description="Configured via RESTCONF"):

    """Performs a POST request to create a new interface."""

    url = f"{BASE\_URL}/ietf-interfaces:interfaces/interface={interface\_name}"

    payload = {

        "ietf-interfaces:interface": {

            "name": interface\_name,

            "description": description,

            "type": "iana-if-type:ethernetCsmacd",

            "enabled": True

        }

    }

    response = requests.put(url, headers=HEADERS, data=json.dumps(payload),

                            auth=HTTPBasicAuth(USERNAME, PASSWORD), verify=False)

    if response.status\_code in (200, 201):

        print("POST Response: Interface created successfully.")

    else:

        print(f"POST request failed with status {response.status\_code}: {response.text}")

def update\_interface(interface\_name, new\_description):

    """Performs a PUT request to update an existing interface's description."""

    url = f"{BASE\_URL}/ietf-interfaces:interfaces/interface={interface\_name}"

    payload = {

        "ietf-interfaces:interface": {

            "name": interface\_name,

            "description": new\_description,

            "type": "iana-if-type:ethernetCsmacd",

            "enabled": True

        }

    }

    response = requests.put(url, headers=HEADERS, data=json.dumps(payload),

                            auth=HTTPBasicAuth(USERNAME, PASSWORD), verify=False)

    if response.status\_code in (200, 204):

        print("PUT Response: Interface updated successfully.")

    else:

        print(f"PUT request failed with status {response.status\_code}: {response.text}")

def delete\_interface(interface\_name):

    """Performs a DELETE request to remove an interface."""

    url = f"{BASE\_URL}/ietf-interfaces:interfaces/interface={interface\_name}"

    response = requests.delete(url, headers=HEADERS, auth=HTTPBasicAuth(USERNAME, PASSWORD), verify=False)

    if response.status\_code == 204:

        print("DELETE Response: Interface deleted successfully.")

    else:

        print(f"DELETE request failed with status {response.status\_code}: {response.text}")

def main():

    # GET interfaces

    print("Performing GET operation to retrieve interfaces:")

    get\_interfaces()

    # POST (create) an interface

    print("\nPerforming POST operation to create an interface:")

    create\_interface("GigabitEthernet2")

    # PUT (update) an interface

    print("\nPerforming PUT operation to update an interface description:")

    update\_interface("GigabitEthernet2", "Updated via RESTCONF")

    # DELETE an interface

    print("\nPerforming DELETE operation to delete the interface:")

    delete\_interface("GigabitEthernet2")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**ACTIVITY2:**  
• Python Activity: Create a comprehensive Python script that validates YANG models and performs NETCONF operations. Present the final project, demonstrating NETCONF management using YANG models.

**PROGRAM:**

import os

import subprocess

from ncclient import manager

from ncclient.operations.rpc import RPCError

from lxml import etree

# YANG Validation Function

def validate\_yang\_model(yang\_file\_path):

    print("Validating YANG model...")

    try:

        result = subprocess.run(['pyang', '-p', os.path.dirname(yang\_file\_path), '-f', 'tree', yang\_file\_path],

                                capture\_output=True, text=True, check=True)

        print("YANG Model Validation Successful.")

        print(result.stdout)

        return True

    except subprocess.CalledProcessError as e:

        print("YANG Model Validation Failed.")

        print(e.stderr)

        return False

# NETCONF Operations

def get\_netconf\_capabilities(mgr):

    print("\nAvailable NETCONF Capabilities:")

    for capability in mgr.server\_capabilities:

        print(capability)

def get\_config(mgr, filter\_criteria):

    print("\nFetching Configuration...")

    try:

        config = mgr.get\_config(source='running', filter=('subtree', filter\_criteria)).data\_xml

        print("Configuration Retrieved:")

        print(config)

        return config

    except RPCError as e:

        print(f"Failed to fetch configuration: {e}")

        return None

def edit\_config(mgr, config\_xml):

    print("\nApplying Configuration Change...")

    try:

        mgr.edit\_config(target="running", config=config\_xml)

        print("Configuration Updated Successfully.")

    except RPCError as e:

        print(f"Failed to edit configuration: {e}")

def main():

    # Step 1: Validate YANG Model

    yang\_model\_path = "ietf-interfaces.yang"  # Path to your YANG model file

    if not validate\_yang\_model(yang\_model\_path):

        return

    # Step 2: Connect to NETCONF Server

    print("\nConnecting to NETCONF server...")

    with manager.connect(

        host="192.168.1.1",  # Replace with your NETCONF device IP

        port=830,

        username="admin",

        password="admin",

        hostkey\_verify=False

    ) as mgr:

        print("NETCONF connection established.")

        # Display NETCONF Capabilities

        get\_netconf\_capabilities(mgr)

        # Step 3: Retrieve Configuration Using NETCONF

        interface\_filter = """

            <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">

                <interface>

                    <name>GigabitEthernet1</name>

                </interface>

            </interfaces>

        """

        config = get\_config(mgr, interface\_filter)

        # Step 4: Edit Configuration

        new\_config = """

            <config>

                <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">

                    <interface>

                        <name>GigabitEthernet1</name>

                        <description>Configured via NETCONF</description>

                        <enabled>true</enabled>

                    </interface>

                </interfaces>

            </config>

        """

        edit\_config(mgr, new\_config)

        # Step 5: Verify Configuration Change

        config\_after\_edit = get\_config(mgr, interface\_filter)

        print("\nConfiguration after edit:")

        print(config\_after\_edit)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**ACTIVITY3:**  
o Develop test cases to validate correct replacement of placeholders with given hostname and IP address.  
O Test scenarios:  
1. Valid Hostname and IP Address  
2. Empty Hostname  
3. Empty IP Address

**PROGRAM:**

public class PlaceholderReplacer {

    private String template;

    public PlaceholderReplacer(String template) {

        this.template = template;

    }

    public String replacePlaceholders(String hostname, String ipAddress) {

        String result = template;

        if (hostname != null && !hostname.isEmpty()) {

            result = result.replace("{hostname}", hostname);

        }

        if (ipAddress != null && !ipAddress.isEmpty()) {

            result = result.replace("{ip}", ipAddress);

        }

        return result;

    }

    // Unit tests

    public static void main(String[] args) {

        PlaceholderReplacer replacer = new PlaceholderReplacer("Connecting to {hostname} at IP {ip}");

        // Test Case 1: Valid Hostname and IP Address

        String result1 = replacer.replacePlaceholders("exampleHost", "192.168.1.1");

        System.out.println("Test 1 - Valid Hostname and IP Address: " + result1);

        // Test Case 2: Empty Hostname

        String result2 = replacer.replacePlaceholders("", "192.168.1.1");

        System.out.println("Test 2 - Empty Hostname: " + result2);

        // Test Case 3: Empty IP Address

        String result3 = replacer.replacePlaceholders("exampleHost", "");

        System.out.println("Test 3 - Empty IP Address: " + result3);

        // Test Case 4: Both Empty Hostname and IP Address

        String result4 = replacer.replacePlaceholders("", "");

        System.out.println("Test 4 - Both Empty Hostname and IP Address: " + result4);

    }

}

**ACTIVITY4:**  
• Tasks:   
o Save the generated configuration to a file.  
o Handle file open errors gracefully.  
o Test scenarios:  
1. Successful File Save  
2. File Open Error (simulate by restricting write permissions)

**PROGRAM:**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

public class FileHandler {

    public boolean saveToFile(String content, String filePath) {

        try (FileWriter writer = new FileWriter(filePath)) {

            writer.write(content);

            System.out.println("Configuration saved successfully.");

            return true;

        } catch (IOException e) {

            System.err.println("Error saving to file: " + e.getMessage());

            return false;

        }

    }

    // Main method to run tests

    public static void main(String[] args) {

        FileHandler fileHandler = new FileHandler();

        String filePath = "config.txt";

        // Test Case 1: Successful File Save

        String configContent = "hostname=exampleHost\nip=192.168.1.1";

        System.out.println("Test 1 - Successful File Save:");

        boolean isSaved = fileHandler.saveToFile(configContent, filePath);

        System.out.println("Result: " + (isSaved ? "Pass" : "Fail"));

        // Test Case 2: File Open Error (simulate by making the file read-only)

        System.out.println("\nTest 2 - File Open Error:");

        File file = new File(filePath);

        file.setReadOnly(); // Simulate restricted write permissions

        boolean isSavedError = fileHandler.saveToFile(configContent, filePath);

        System.out.println("Result: " + (!isSavedError ? "Pass" : "Fail"));

        // Clean up by resetting file permissions if necessary

        file.setWritable(true);

    }

}