## Notes

**Cisco Configuration Backup Script**

**Introduction**
This script provides a simple yet effective way to automatically back up the running configuration of Cisco devices. It establishes a serial connection with the device, enters privileged EXEC mode if required, retrieves the running configuration, and saves it to a specified file.

**How it works**
The script utilizes the serial library to establish a connection with the Cisco device through a COM port. It sends necessary commands to the device to enter privileged EXEC mode and fetches the running configuration using the show running-config command. Finally, it saves the configuration to a specified file.

**Features**
- Establishes a serial connection with Cisco devices.
- Automatically enters privileged EXEC mode if required.
- Retrieves the running configuration.
- Saves the configuration to a specified file.

**Getting Started**
1. Connect the Cisco device to your computer using a serial cable.
2. Make sure Python is installed on your system.
3. Run the script.
4. Enter the COM port number, BAUD rate, password (if required), and output file name as prompted.
5. The running configuration will be saved to the specified file.

**Dependencies**
- Python 3.x
- pyserial library

**License**

**GitHub**
Share Link: https://github.com/Anthony-Constant/Cisco-Configuration-Backup

## PYTHON COPY & PASTED LOCAL SOURCE CODE

```
############################################################
# Script to back up cisco devices i.e. switches/routers           #
# Author: Anthony Constant                                        #
# Date: 18/04/2024                                                #
############################################################

import serial
import time
import serial.tools.list_ports

def read_until_prompt(ser, prompt, timeout=5):
    start_time = time.time()
    response = b""
    while True:
        if time.time() - start_time > timeout:
            break
        data = ser.read()
        if data:
            response += data
            if prompt in response:
                break
    return response

def backup_running_config(serial_port, baud_rate, password, output_file):
    try:
        # Serial connection setup
        print("Connecting to the device...")
        ser = serial.Serial(serial_port, baud_rate, timeout=1)
        ser.write(b"\r\n")
        print("Initiating backup of the running configuration...")

        # Give some time for the serial connection to stabilize
        time.sleep(2)
```

```python
    # Send command to the device to enter privileged EXEC mode
    print("Entering privileged EXEC mode...")
    ser.write(b"enable\r\n")

    # Check if already in enable mode
    output = read_until_prompt(ser, b"#")
    if b"Password:" in output:
        print("Sending password...")
        ser.write(password.encode('utf-8') + b"\r\n")
        read_until_prompt(ser, b"#")

    # Send command to the device to set terminal length and get the running configuration
    print("Setting terminal length and retrieving running configuration...")
    ser.write(b"terminal length 0\r\n")
    read_until_prompt(ser, b"#")
    ser.write(b"show running-config\r\n")

    # Introduce a delay to allow the device to process the command and generate output
    time.sleep(5)  # Adjust this delay as needed

    # Read the output until no data is received for a short period
    print("Waiting for configuration output...")
    config_output = b""
    while True:
        data = ser.read(4096)
        if not data:
            break
        config_output += data

    config_output = config_output.decode('utf-8')

    # Save the running configuration to the specified file
    with open(output_file, "w") as file:
        file.write(config_output)

    print("Running configuration backed up successfully.")
```

```python
        # Close serial connection
        ser.close()

    except serial.SerialException as se:
        print(f"Serial port error: {se}")
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    try:
        available_ports = list(serial.tools.list_ports.comports())
        print("Available COM ports:")
        for idx, port in enumerate(available_ports):
            print(f"{idx + 1}. {port.device}")

        selection = int(input("Enter the number corresponding to the desired COM port: "))
        selected_port = available_ports[selection - 1].device
        baud_rate = int(input("Enter the BAUD rate: "))
        password = input("Enter your password: ")
        output_file = input("Enter the name for the output file (e.g., running_config.txt): ")

        backup_running_config(selected_port, baud_rate, password, output_file)
    except (IndexError, ValueError):
        print("Invalid selection. Please enter a valid number.")
```