Notes

Cisco Switch Configurator

Introduction
The Cisco Switch Configurator allows users to interact with Cisco switches through a graphical interface, enabling the execution of commands and configuration tasks with ease.

How it works
The script establishes a serial connection with the Cisco switch using the specified COM port and BAUD rate. Users can then choose to execute custom commands or predefined command sequences on the switch. The output from the switch is displayed in the GUI, providing options to copy the output to the clipboard or save it to a file.

Dependencies
Python 3.x
Required Python packages: serial, tkinter, getpass

Features
- Serial Communication: Establishes a serial connection with the Cisco switch using the specified COM port and BAUD rate.
- Command Execution: Executes custom commands or predefined command sequences on the switch.
- Output Handling: Displays the output from the switch in the GUI and provides options to copy the output to the clipboard or save it to a file.
- Error Handling: Provides error messages for invalid inputs or connection issues.
- Changelog Display: Shows the version history and changelog within the GUI.

Getting Started
- Run the script.
- Select the COM port and BAUD rate for the serial connection.
- Choose a command from the predefined list or enter a custom command in the designated field.
- Click the "Configure" button to execute the selected command(s) on the switch.
- View the output in the text box provided. You can copy the output to the clipboard or save it to a file using the respective buttons.

License

GitHub
Share Link: https://github.com/Anthony-Constant/Cisco-Switch-Configurator

PYTHON COPY & PASTED LOCAL SOURCE CODE

```
# Import required modules
import serial
import time
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from getpass import getpass

# Changelog
CHANGELOG = """
Changelog:
- Version 1.6 (2024-04-09):
  - Added support for entering and sending custom commands with an Enter key press.
  - Added support for entering custom commands in the command box.
  - Added button to execute "term len 0" and "show run" and save output to log file.
  - Improved error handling and messages.
  - Added BAUD rate selection feature.
  - Added "Executions" dropdown menu for predefined command executions.
  - Added functionality to send multiple carriage returns before executing commands.
  - Added documentation for functions and the main program.
- Version 1.5 (2024-04-09):
  - Added COM Port dropdown menu.
  - Added default COM Ports to the dropdown menu.
  - Improved GUI layout and styling for better readability and organization.
  - Added documentation for functions and the main program.
- Version 1.4 (2024-04-09):
  - Added loading message when configuring the switch.
  - Improved GUI layout and styling for better readability and organization.
  - Added documentation for functions and the main program.
- Version 1.3 (2024-04-09):
  - Integrated fix for capturing larger output from the switch.
- Version 1.2 (2024-04-08):
  - Initial GUI improvements.
```

```
- Version 1.1 (2024-04-08):
  - Initial release.

  Copyright © AnthonyConstant.co.uk 2024

"""

# Well-known Cisco commands
CISCO_COMMANDS = [
    "show version",
    "show interfaces",
    "show ip interface brief",
    "show interfaces status",
    "show running-config",
    "show vlan brief",
    "show mac address-table",
    "show spanning-tree",
    "show arp",
    "show cdp neighbors",
    "show logging",
    # Add more commands as needed
]

# Default COM Ports
DEFAULT_COM_PORTS = ["COM1", "COM2", "COM3", "COM4", "COM5", "COM6", "COM7", "COM8", "COM9", "COM10"]

# Available BAUD rates
BAUD_RATES = [9600, 19200, 38400, 57600, 115200]

# Executions
EXECUTIONS = [
    "Execute term len 0 and show run",
    # Add more executions as needed
]


def configure_switch(com_port, baud_rate, commands):
    """
```

```
    Configure the switch with provided commands.

    Args:
        com_port (str): The COM port to connect to.
        baud_rate (int): The BAUD rate for the serial connection.
        commands (list of str): List of commands to send to the switch.

    Returns:
        str: Output message from the switch.
    """
    try:
        # Establish serial connection
        ser = serial.Serial(com_port, baudrate=baud_rate, timeout=1)

        # Wait for the prompt to appear
        time.sleep(2)
        ser.reset_input_buffer()
        # Send multiple carriage returns
        ser.write(b"\r\n")
        output = b''

        # Send commands
        for command in commands:
            ser.write(command.encode() + b"\r\n")
            time.sleep(2)  # Wait for the switch to process the command

        # Send an additional carriage return to simulate pressing Enter
        ser.write(b"\r\n")

        # Read output in chunks until no more data is available
        while True:
            chunk = ser.read(4096)  # Adjust chunk size as needed
            if not chunk:
                break
            output += chunk  # No need to decode chunk

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

```
        # Decode the byte object to string before returning
        output = output.decode()

        return output

    except Exception as e:
        return str(e)


def copy_to_clipboard(text):
    """
    Copy text to the clipboard.

    Args:
        text (str): Text to copy to the clipboard.

    Returns:
        bool: True if copy operation succeeds, False otherwise.
    """
    try:
        # Initialize Tkinter
        root = tk.Tk()
        # Hide the main window
        root.withdraw()
        # Set clipboard data
        root.clipboard_clear()
        root.clipboard_append(text)
        # Update clipboard
        root.update()
        # Destroy Tkinter instance
        root.destroy()
        return True
    except:
        return False


def save_to_file(text, filename):
```

```
    """
    Save text to a file.

    Args:
        text (str): Text to save.
        filename (str): Name of the file to save the text.

    Returns:
        bool: True if save operation succeeds, False otherwise.
    """
    try:
        with open(filename, "w") as file:
            file.write(text)
        return True
    except Exception as e:
        return False


def configure_switch_gui():
    """Create and display the GUI for configuring the switch."""
    def on_configure():
        com_port = com_port_combobox.get()
        baud_rate = int(baud_rate_combobox.get())
        command = commands_entry.get("1.0", tk.END).strip()

        if not com_port:
            messagebox.showerror("Error", "Please select a COM Port.")
            return

        if not command:
            messagebox.showerror("Error", "Please enter a command.")
            return

        if command in EXECUTIONS:
            if command == "Execute term len 0 and show run":
                commands = [
                    "term len 0",
                    "show run"
```

```
            ]
            # Execute and save to log file
            output = configure_switch(com_port, baud_rate, commands)
            output_text.delete("1.0", tk.END)
            output_text.insert(tk.END, output)
            if save_to_file(output, "switch_config.log"):
                messagebox.showinfo("Success", "Output saved to switch_config.log.")
            else:
                messagebox.showerror("Error", "Failed to save output to file.")
    else:
        # Regular command execution without saving to log file
        output = configure_switch(com_port, baud_rate, [command])
        output_text.delete("1.0", tk.END)
        output_text.insert(tk.END, output)

def on_copy():
    output = output_text.get("1.0", tk.END)
    if copy_to_clipboard(output):
        messagebox.showinfo("Success", "Output copied to clipboard.")
    else:
        messagebox.showerror("Error", "Failed to copy output to clipboard.")

def on_cisco_command_selected(event):
    selected_command = cisco_commands_combobox.get()
    commands_entry.delete("1.0", tk.END)
    commands_entry.insert("1.0", selected_command)

def on_execution_selected(event):
    selected_execution = executions_combobox.get()
    commands_entry.delete("1.0", tk.END)
    commands_entry.insert("1.0", selected_execution)

# Create and display the GUI for configuring the switch
window = tk.Tk()
window.title("Cisco Switch Configurator")

# Notebook
notebook = ttk.Notebook(window)
```

```
notebook.pack(fill='both', expand=True)

# Configure Tab
configure_frame = ttk.Frame(notebook)
notebook.add(configure_frame, text='Configure')

# COM Port
ttk.Label(configure_frame, text="COM Port:", font=("Helvetica", 10)).grid(row=2, column=0, padx=5, pady=5, sticky="e")
com_port_combobox = ttk.Combobox(configure_frame, values=[""] + DEFAULT_COM_PORTS, width=30)
com_port_combobox.grid(row=2, column=1, padx=5, pady=5)
com_port_combobox.current(0)  # Set default COM Port

# BAUD Rate
ttk.Label(configure_frame, text="BAUD Rate:", font=("Helvetica", 10)).grid(row=3, column=0, padx=5, pady=5, sticky="e")
baud_rate_combobox = ttk.Combobox(configure_frame, values=[""] + BAUD_RATES, width=30)
baud_rate_combobox.grid(row=3, column=1, padx=5, pady=5)
baud_rate_combobox.current(0)  # Set default BAUD Rate

# Cisco Commands
ttk.Label(configure_frame, text="Cisco Commands:", font=("Helvetica", 10)).grid(row=4, column=0, padx=5, pady=5, sticky="e")
cisco_commands_combobox = ttk.Combobox(configure_frame, values=[""] + CISCO_COMMANDS, width=47)
cisco_commands_combobox.grid(row=4, column=1, padx=5, pady=5)
cisco_commands_combobox.bind("<<ComboboxSelected>>", on_cisco_command_selected)

# Executions
ttk.Label(configure_frame, text="Executions:", font=("Helvetica", 10)).grid(row=5, column=0, padx=5, pady=5, sticky="e")
executions_combobox = ttk.Combobox(configure_frame, values=[""] + EXECUTIONS, width=47)
executions_combobox.grid(row=5, column=1, padx=5, pady=5)
executions_combobox.bind("<<ComboboxSelected>>", on_execution_selected)


# Commands
ttk.Label(configure_frame, text="Commands:", font=("Helvetica", 10)).grid(row=6, column=0, padx=5, pady=5, sticky="e")
commands_entry = tk.Text(configure_frame, height=10, width=50)
commands_entry.grid(row=6, column=1, padx=5, pady=5)

# Output Text
ttk.Label(configure_frame, text="Output:", font=("Helvetica", 10)).grid(row=7, column=0, padx=5, pady=5, sticky="e")
```

```
    output_text = tk.Text(configure_frame, height=10, width=50)
    output_text.grid(row=7, column=1, padx=5, pady=5)

    # Buttons
    configure_button = ttk.Button(configure_frame, text="Configure", command=on_configure)
    configure_button.grid(row=8, column=0, padx=5, pady=5)

    copy_button = ttk.Button(configure_frame, text="Copy", command=on_copy)
    copy_button.grid(row=8, column=1, padx=5, pady=5)

    # Version Tab
    version_frame = ttk.Frame(notebook)
    notebook.add(version_frame, text='Version')

    # Display changelog
    ttk.Label(version_frame, text="Changelog:", font=("Helvetica", 10, "bold")).grid(row=0, column=0, columnspan=2, padx=5, pady=5,
sticky="w")
    ttk.Label(version_frame, text="Cisco Switch Configurator Version 1.6", font=("Helvetica", 10, "bold")).grid(row=0, column=0,
columnspan=2, padx=5, pady=5, sticky="w")
    ttk.Label(version_frame, text=CHANGELOG, font=("Helvetica", 8), wraplength=600, justify="left").grid(row=1, column=0, columnspan=2,
padx=5, pady=5, sticky="w")

    # Help Tab
    help_frame = ttk.Frame(notebook)
    notebook.add(help_frame, text='Help')

    # Readme text
    readme_text = tk.Text(help_frame, height=20, width=80)
    readme_text.insert(tk.END, """
   Cisco Switch Configurator

This Python script provides a graphical user interface (GUI) for configuring Cisco switches via serial connection.

Features

- Serial Communication: Establishes a serial connection with the Cisco switch using the specified COM port and BAUD rate.

- Command Execution: Executes custom commands or predefined command sequences on the switch.
```

```
- Output Handling: Displays the output from the switch in the GUI and provides options to copy the output to the clipboard or save it to a
file.

- Error Handling: Provides error messages for invalid inputs or connection issues.

- Changelog Display: Shows the version history and changelog within the GUI.

Requirements
- Python 3.x
- Required Python packages: serial, tkinter, getpass

Usage
1. Run the script.
2. Select the COM port and BAUD rate for the serial connection.
3. Choose a command from the predefined list or enter a custom command in the designated field.
4. Click the "Configure" button to execute the selected command(s) on the switch.
5. View the output in the text box provided. You can copy the output to the clipboard or save it to a file using the respective buttons.

License
Copyright © AnthonyConstant.co.uk 2024
    """)

    readme_text.config(state='disabled')  # Disable editing
    readme_text.grid(row=0, column=0, padx=5, pady=5)



    window.mainloop()


if __name__ == "__main__":
    configure_switch_gui()
```