```
from netmiko import ConnectHandler
from getpass import getpass
import time



def connect_to_device(ip, username, password):
    """
    Establish a connection to the device using Netmiko.

    :param ip: IP address of the switch
    :param username: Username for the switch login
    :param password: Password for the switch login
    :return: A connection object if successful, None otherwise
    """
    device = {
        'device_type': 'cisco_ios',
        'ip': ip,
        'username': username,
        'password': password,
    }
    try:
        print("Connecting to the device...")
        connection = ConnectHandler(**device)
        print("Connected successfully.")
        return connection
    except Exception as e:
        print(f"Failed to connect to the device: {e}")
        return None
```

```
def execute_commands(connection, commands):
    """
    Execute a list of commands on the connected device.

    :param connection: Netmiko connection object
    :param commands: List of commands to execute
    :return: Output of the commands
    """
    try:
        print("Sending configuration commands...")
        output = connection.send_config_set(commands)
        print(output)
        connection.send_command("end")
        return output
    except Exception as e:
        print(f"Failed to execute commands: {e}")
        return None



def change_switch_vlan(connection, interface, vlan):
    """
    Configures a switch interface with the specified VLAN and performs a shutdown/no shutdown sequence.

    :param connection: Netmiko connection object
    :param interface: Interface to configure (e.g., "Fa0/5")
    :param vlan: VLAN ID to assign (e.g., "64")
    """
```

```
    commands = [
        f"interface {interface}",
        f"switchport access vlan {vlan}",
        "shutdown",
        "no shutdown",
    ]
    print(f"Configuring interface {interface} to VLAN {vlan}...")
    execute_commands(connection, commands)
    print(f"Process Success: Changed VLAN to {vlan} - Successful")


def clear_port(connection, interface):
    """
    Clears the sticky port configuration on the specified interface.

    :param connection: Netmiko connection object
    :param interface: Interface to clear
    """
    commands = [
        f"interface {interface}",
        "shutdown",
        "no shutdown",
        "exit",
    ]
    print(f"Performing Clear Port for interface {interface}...")
    execute_commands(connection, commands)
    print(f"Process Success: Clear Port - Successful")
```

```
def clear_sticky_port_security(connection, interface):
    """
    Clears port-security sticky configuration on the specified interface.

    :param connection: Netmiko connection object
    :param interface: Interface to clear sticky port-security
    """
    clear_command = f"clear port-security sticky int {interface}"
    print(f"Executing: {clear_command}")
    try:
        output = connection.send_command(clear_command)
        print(output)

        # Wait for 10 seconds
        print("Waiting for 10 seconds...")
        time.sleep(10)

        # Perform shutdown and no shutdown on the interface
        commands = [
            f"interface {interface}",
            "shutdown",
            "no shutdown",
            "exit"
        ]
        print(f"Performing interface reconfiguration...\nCommands: {commands}")
        output = connection.send_config_set(commands)
        print(output)

        # Exit configuration mode
```

```
        connection.send_command("end")
        print("Configuration complete.\n")

        connection.disconnect()
        print("Disconnected from the device.")


    except Exception as e:
        print(f"Failed to clear port-security sticky: {e}")


def sh_err(connection, interface):
    """
    Clears the sticky port configuration on the specified interface.

    :param connection: Netmiko connection object
    :param interface: Interface to clear
    """
    commands = [
        f"sh err rec",
        "exit"
    ]
    execute_commands(connection, commands)
    print(f"Process Success: Clear Port - Successful")



def main():
    print("Choose an action:")
```

```
print("1. Change Switch VLAN")
print("2. Clear Port")
print("3. Clear Port Security Sticky")
print("4. Show Err Rec")

choice = input("Enter the number corresponding to your choice: ")


ip = f"10.16.0.{input('Enter the device IP (last octet): ')}"


#ip = input("Enter the device IP: ")
#username = input("Enter your username: ")
#password = getpass("Enter your password: ")

username = "a.acosta"
password = "MS043ms-64.,"
# Establish connection
connection = connect_to_device(ip, username, password)
if not connection:
    return

if choice == '1':
    interface = input("Enter the interface (e.g., G1/0/6 - Fa0/6): ")
    vlan = input("Enter the VLAN ID (e.g., 64): ")
    change_switch_vlan(connection, interface, vlan)
elif choice == '2':
    interface = input("Enter the interface (e.g., G1/0/6 - Fa0/6): ")
    clear_port(connection, interface)
```

```
    elif choice == '3':
        interface = input("Enter the interface to clear sticky (e.g., G1/0/6 - Fa0/6): ")
        clear_sticky_port_security(connection, interface)
    elif choice == '4':
        interface = input("Enter the interface to clear sticky (e.g., G1/0/6 - Fa0/6): ")
        sh_err(connection, interface)
    else:
        print("Invalid choice. Please select a valid option (1, 2, or 3).")

    # Disconnect from the device
    connection.disconnect()
    print("Disconnected from the device.")


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