



# Generate a Python script that extracts a network configuration from the local host and saves this information on file with date and time stamp.

Here's a Python script that extracts network configuration information from the local host using the subprocess module to execute system commands and saves the output to a file with a date and time stamp in its name.

## Python Script: Extract Network Configuration

```
import subprocess
import datetime

def extract_network_configuration():
    # Generate a unique filename with date and time stamp
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
    output_file = f"network_config_{timestamp}.txt"

    try:
        # Execute the command to get network configuration (platform-dependent)
        if subprocess.os.name == "nt":  # Windows
            command = ["ipconfig", "/all"]
        else:  # Linux/Mac
            command = ["ifconfig"]

        # Run the command and capture the output
        result = subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                               encoding='utf-8')

        # Check if the command executed successfully
        if result.returncode == 0:
            # Write the output to the file
            with open(output_file, 'w') as file:
                file.write(result.stdout)
            print(f"Network configuration has been saved to: {output_file}")
        else:
            print(f"Error occurred while fetching network configuration: {result.stderr}")

    except Exception as e:
        print(f"An error occurred: {e}")

# Run the function
extract_network_configuration()
```

## How It Works

### 1. Command Selection Based on OS:

- For **Windows**, it uses `ipconfig /all` to fetch network configuration details.
- For **Linux/Mac**, it uses `ifconfig`. (You can replace `ifconfig` with `ip addr` for modern Linux systems.)

### 2. Timestamp-Based Filename:

- The script generates a unique filename using the current date and time (e.g., `network_config_2025-04-04_21-45-30.txt`).

### 3. Saving Output:

- The script captures the command's output using `subprocess.run()` and writes it to a file.

### 4. Error Handling:

- If there's an error (e.g., invalid command or permission issues), it prints an error message.

## Example Output

If you run this script on a Windows machine, it will create a file named something like `network_config_2025-04-04_21-45-30.txt` containing detailed network information, such as:

```
Windows IP Configuration

Host Name . . . . . : My-PC
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No

Ethernet adapter Ethernet:

Connection-specific DNS Suffix :
Description. . . : Intel(R) Ethernet Connection
Physical Address. : 00-1A-2B-3C-4D-5E
DHCP Enabled. : Yes
IPv4 Address. : 192.168.1.100(Preferred)
```

On Linux/Mac, it will include details like:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
            ether 00:1a:2b:3c:4d:5e txqueuelen 1000 (Ethernet)
```

## **Customizations**

### **1. Change Command:**

- For modern Linux systems, replace ifconfig with ip addr show.

### **2. Add More Details:**

- You can modify the script to include additional commands (e.g., xroute, netstat) for more detailed network information.

### **3. Cross-Platform Library:**

- For more advanced use cases, consider using Python libraries like psutil or netifaces for cross-platform network information retrieval.

This script is a simple yet effective way to extract and save local network configurations for documentation or troubleshooting purposes!