# LAB – READING AND WRITING STRUCTURED FILES

## OBJECTIVE

In this lab, you will read connection information and write CSV and JSON files.

## PART 1

Copy the required configuration for this lab from flash into the running-configuration of the CSR1000v and R2 routers.

### STEP 1: COPY CONFIGURATION

In the CSR1kv router console, enter **privileged exec** mode and copy the additional configuration located in the **CONFIG3** file to the running-configuration using the commands:

```
CSR1kv> enable
CSR1kv# copy flash:CONFIG3 running-config
```

```
CSR1kv>enable
CSR1kv#copy flash:CONFIG3 running-config
Destination filename [running-config]?
75 bytes copied in 0.009 secs (8333 bytes/sec)
CSR1kv#
```

### STEP 2: VERIFY CONFIGURATION

Verify the new running-configuration will now allow telnet connections by using the command:

```
CSR1kv# show run | section line vty
```

**NOTE:**
To get the pipe (**|**) command, you may need to use the keys **Shift** + **#** rather than your local keyboard input.

```
CSR1kv#show run | section line vty
line vty 0 4
 logging synchronous
 login local
 transport input all
line vty 5 15
 logging synchronous
 login local
 transport input all
CSR1kv#
```

### STEP 3: COPY CONFIGURATION

In the R2 router console, enter **privileged exec** mode and copy the additional configuration located in the **CONFIG7** file to the running-configuration using the commands:

```
R2> enable
R2# copy flash:CONFIG7 running-config
```

```
R2>enable
R2#copy flash:CONFIG7 running-config
Destination filename [running-config]?
64 bytes copied in 0.024 secs (2667 bytes/sec)
R2#
```

### STEP 4: VERIFY CONFIGURATION

Verify the new running-configuration will now allow telnet connections by using the command:

```
R2# show run | section line vty
```

**NOTE:**
To get the pipe (**|**) command, you may need to use the keys **Shift** + **#** rather than your local keyboard input.

```
R2#show run ¦ section line vty
line vty 0 4
 logging synchronous
 login local
 transport input all
line vty 5 15
 logging synchronous
 login local
 transport input all
R2#
```

## PART 2

Open a terminal and switch to the lab directory

### STEP 1: OPEN A TERMINAL WINDOW

Double-click the Terminal icon on the desktop to open the terminal window for use in this lab.

### STEP 2: CHANGE DIRECTORY

Change to the directory **labs/prne/reading-and-writing-structured-files-part-3/** in the user home directory, which holds the files for the course labs.

```
~$ cd labs/prne/reading-and-writing-structured-files-part-3/
```

## PART 3

Open **Visual Studio Code**, create three new files with the filenames of **devclass.py**, **main.py** and **util.py**, ensuring to save the files in the **~/labs/prne/reading-and-writing-structured-files-part-3/** directory.
This python application will create a package that:

- Reads device information data from a CSV formatted file
- Create device objects with information for each device in the file
- Attempt to connect to the devices
- Print the results
- Output a CSV formatted file with information about each device from the list.

### STEP 1: CONFIGURE DEVCLASS MODULE

Open the module **devclass.py**

#### IMPORT REQUIRED LIBRARIES

Import the **pexpect** library

```
# Import required modules/packages/library
import pexpect
```

#### CREATE DEVICE OBJECTS

Create a device object for generic devices.

```
# Class to hold information about a generic network device
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, unknown get interfaces ---'
```

Create a device object for IOS devices.

```
# Class to hold information about an IOS network device
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    def connect(self):
        print('--- connecting IOS: telnet ' + self.ip_address)

        self.session = pexpect.spawn('telnet ' + self.ip_address,
                                     encoding='utf-8', timeout=20)
       result = self.session.expect(['Username:', pexpect.TIMEOUT, pexpect.EOF])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')

        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

        # Must set terminal length to zero for long replies, no pauses
        self.session.sendline('terminal length 0')
        result = self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

        self.interfaces = self.session.before
```

Create a device object for IOS-XR devices.

```
# Class to hold information about an IOS-XR network device
class NetworkDeviceXR(NetworkDevice):

    # Initialize
    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    # Connect to device
    def connect(self):

        print('--- connecting XR: ssh '+ self.username + '@' +
              self.ip_address)

        self.session = pexpect.spawn('ssh ' + self.username +
                                     '@' + self.ip_address,
                                     encoding='utf-8', timeout=20)
       result = self.session.expect(['password:', pexpect.TIMEOUT, pexpect.EOF])

        # Check for failure
        if result != 0:
            print('--- Timout or unexpected reply from device')
            return 0

        # Successfully got password prompt, logging in with password
        print('--- password:', self.password)
        self.session.sendline(self.password)
        self.session.expect(['#', pexpect.TIMEOUT, pexpect.EOF])

    # Get interfaces from device
    def get_interfaces(self):

        self.session.sendline('show interface brief')
        result = self.session.expect(['#', pexpect.TIMEOUT, pexpect.EOF])

        self.interfaces = self.session.before
```

## STEP 2: CONFIGURE UTIL MODULE

Open the module **util.py**

### IMPORT REQUIRED LIBRARIES

Import the **csv** module to read and write tabular data in CSV format and import the **pprint** function of **pprint** module to display the data well formatted and much more readable. Finally import the device object functions from the **devclass** module.

```
# Import required modules/packages/library
import csv
from pprint import pprint
from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
from devclass import NetworkDeviceXR
```

## READ DEVICES FROM CSV FILE FUNCTION

Create a function to read the device information from the **devices-02.csv** file.

```
# Read device information from CSV file
def read_devices_info(devices_file):

    devices_list = []

    # Open the CSV file
    file = open(devices_file, 'r')

    # Create the CSV reader for file
    csv_devices = csv.reader(file)

    # Iterate through all devices in our CSV file
    for device_info in csv_devices:

        # Create a device object with this data
        if device_info[1] == 'ios':

            device = NetworkDeviceIOS(device_info[0],
                                      device_info[2],
                                      device_info[3],
                                      device_info[4])

        elif device_info[1] == 'ios-xr':

            device = NetworkDeviceXR(device_info[0],
                                     device_info[2],
                                     device_info[3],
                                     device_info[4])

        else:
            device = NetworkDevice(device_info[0],
                                   device_info[2],
                                   device_info[3],
                                   device_info[4])

        # Append this device object to list
        devices_list.append(device)

    return devices_list
```

### DISPLAY DEVICE INFORMATION FUNCTION

Create a function to display the device information on connection attempt.

```
# Print information to display
def print_device_info(device):

    print('-----------------------------------------------------')
    print('    Device Name:      ', device.name)
    print('    Device IP:        ', device.ip_address)
    print('    Device username:  ', device.username,)
    print('    Device password:  ', device.password)
    print('-----------------------------------------------------')
    print('')
    print(device.interfaces)
    print('')
```

### WRITE DEVICE INFORMATION FUNCTION

Create a function to write the device information on connection attempt.

```
# Write information to file
def write_devices_info(devices_file, devices_list):

    print('---- Printing CSV output ------------------------------')

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for CSV output

    for device in devices_list:
        dev_info = [device.name, device.ip_address,
                    device.interfaces != ""]
        devices_out_list.append(dev_info)

    pprint(devices_out_list)
    print('')

    # Use CSV library to output our list of lists to a CSV file
    with open(devices_file, 'w') as file:
        csv_out = csv.writer(file)
        csv_out.writerows(devices_out_list)
```

## STEP 3: CONFIGURE MAIN MODULE

Open the module **main.py**

### IMPORT REQUIRED LIBRARIES

Import the **read_devices_info**, **print_device_info** and **write_devices_info** functions from the **util** module.

```
# Import required modules/packages/library
from util import read_devices_info
from util import print_device_info
from util import write_devices_info
```

## READ DEVICE INFORMATION FROM FILE

Read the device information from the file **devices-02.csv** using the function **read_devices_info**.

```
# read CSV info for all devices
devices_list = read_devices_info('devices-02.csv')
```

## CONNECT TO DEVICES, SHOW INTERFACES AND DISPLAY

Parsing the list of devices from the csv file, connect to each device, run the show interface summary command and display the device information on screen.

```
# Connect to device, show interface summary, display
for device in devices_list:

    print('==== Device ================================================')

    # connect to this specific device
    device.connect()

    # get interface info for specific device
    device.get_interfaces()

    # print device details for this device
    print_device_info(device)

# write CSV entry for all devices
Write_devices_info('devices-02-out.csv', devices_list)
```

## STEP 4: SAVE MODULES, RUN AND VERIFY PACKAGE

Save you application and then run it from the terminal rather than from within visual studio code.

```
~/labs/prne/reading-and-writing-structured-files-part-3$ python3 main.py
```

The output from your application will be displayed in your terminal window, verify that it is comparable to below.

```
devasc@labvm:~/labs/prne/reading-and-writing-structured-files-part-3$ python3 main.py
==== Device ================================================
--- connecting IOS: telnet 192.168.56.101
------------------------------------------------------
    Device Name:       CSR1kv
    Device IP:         192.168.56.101
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------


CSR1kv#terminal length 0
CSR1kv#show interfaces summary

 *: interface is up
 IHQ: pkts in input hold queue     IQD: pkts dropped from input queue
 OHQ: pkts in output hold queue    OQD: pkts dropped from output queue
 RXBS: rx rate (bits/sec)          RXPS: rx rate (pkts/sec)
 TXBS: tx rate (bits/sec)          TXPS: tx rate (pkts/sec)
 TRTL: throttle count

  Interface                IHQ       IQD       OHQ       OQD      RXBS      RXPS      TXBS      TXPS      TRTL
-----------------------------------------------------------------------------------------------------------------
* GigabitEthernet1           0         0         0         0         0         0         0         0         0
* GigabitEthernet2           0         0         0         0         0         0         0         0         0
CSR1kv#

==== Device ================================================
--- connecting IOS: telnet 192.168.56.130
------------------------------------------------------
    Device Name:       R2
    Device IP:         192.168.56.130
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------


R2#terminal length 0
R2#show interfaces summary

 *: interface is up
 IHQ: pkts in input hold queue     IQD: pkts dropped from input queue
 OHQ: pkts in output hold queue    OQD: pkts dropped from output queue
 RXBS: rx rate (bits/sec)          RXPS: rx rate (pkts/sec)
 TXBS: tx rate (bits/sec)          TXPS: tx rate (pkts/sec)
 TRTL: throttle count

  Interface                IHQ       IQD       OHQ       OQD      RXBS      RXPS      TXBS      TXPS      TRTL
-----------------------------------------------------------------------------------------------------------------
* GigabitEthernet1           0         0         0         0         0         0         0         0         0
* GigabitEthernet2           0         0         0         0         0         0         0         0         0
* Loopback0                  0         0         0         0         0         0         0         0         0
* Loopback1                  0         0         0         0         0         0         0         0         0
* Loopback2                  0         0         0         0         0         0         0         0         0
* Loopback3                  0         0         0         0         0         0         0         0         0
R2#

---- Printing CSV output ------------------------------
[['CSR1kv', '192.168.56.101', True], ['R2', '192.168.56.130', True]]

devasc@labvm:~/labs/prne/reading-and-writing-structured-files-part-3$
```

## PART 4

Open **Visual Studio Code**, create three new files with the filenames of **devclass.py**, **main.py** and **util.py**, ensuring to save the files in the **~/labs/prne/reading-and-writing-structured-files-part-4/** directory.
This python application will create a package that:

- Reads device information data from a JSON formatted file
- Create device objects with information for each device in the file
- Attempt to connect to the devices
- Print the results
- Output a JSON formatted file with information about each device from the list.

### STEP 1: CONFIGURE DEVCLASS MODULE

Open the module **devclass.py**

#### IMPORT REQUIRED LIBRARIES

Import the **pexpect** library

```
# Import required modules/packages/library
import pexpect
```

#### CREATE DEVICE OBJECTS

Create a device object for generic devices.

```
# Class to hold information about a generic network device
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw
        self.os_type = None

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, unknown get interfaces ---'
```

Create a device object for IOS devices.

```
# Class to hold information about an IOS network device
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios'

    def connect(self):
        print('--- connecting IOS: telnet ' + self.ip_address)

        self.session = pexpect.spawn('telnet ' + self.ip_address,
                                     encoding='utf-8', timeout=20)
       result = self.session.expect(['Username:', pexpect.TIMEOUT, pexpect.EOF])

        self.session.sendline(self.username)
        result = self.session.expect('Password:')

        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

        # Must set terminal length to zero for long replies, no pauses
        self.session.sendline('terminal length 0')
        result = self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect(['>', pexpect.TIMEOUT, pexpect.EOF])

        self.interfaces = self.session.before
```

Create a device object for IOS-XR devices.

```
# Class to hold information about an IOS-XR network device
class NetworkDeviceXR(NetworkDevice):

    # Initialize
    def __init__(self, name, ip, user='cisco', pw='cisco123!'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios-xr'

    # Connect to device
    def connect(self):

        print('--- connecting XR: ssh '+ self.username + '@' +
              self.ip_address)

        self.session = pexpect.spawn('ssh ' + self.username +
                                     '@' + self.ip_address,
                                     encoding='utf-8', timeout=20)
       result = self.session.expect(['password:', pexpect.TIMEOUT, pexpect.EOF])

        # Check for failure
        if result != 0:
            print('--- Timout or unexpected reply from device')
            return 0

        # Successfully got password prompt, logging in with password
        print('--- password:', self.password)
        self.session.sendline(self.password)
        self.session.expect(['#', pexpect.TIMEOUT, pexpect.EOF])

    # Get interfaces from device
    def get_interfaces(self):

        self.session.sendline('show interface brief')
        result = self.session.expect(['#', pexpect.TIMEOUT, pexpect.EOF])

        self.interfaces = self.session.before
```

## STEP 2: CONFIGURE UTIL MODULE

Open the module **util.py**

### IMPORT REQUIRED LIBRARIES

Import the **json** module to read and write tabular data in JSON format and import the **pprint** function of **pprint** module to display the data well formatted and much more readable. Finally import the device object functions from the **devclass** module.

```
# Import required modules/packages/library
import json
from pprint import pprint
from devclass import NetworkDevice
from devclass import NetworkDeviceIOS
from devclass import NetworkDeviceXR
```

## READ DEVICES FROM CSV FILE FUNCTION

Create a function to read the device information from the **devices-02.csv** file.

```
# Read device information from JSON file
def read_devices_info(devices_file):

    # Create empty list
    devices_list = []

    # Open the device file with JSON data
    json_file = open(devices_file, 'r')

    # Create the JSON reader for the file
    json_device_data = json_file.read()

    # Convert JSON string into python data structure
    devices_info_list = json.loads(json_device_data)

    # Iterate through all devices in our JSON file
    for device_info in devices_info_list:

        # Create a device object with this data
        if device_info['os'] == 'ios':

            device = NetworkDeviceIOS(device_info['name'],
                                      device_info['ip'],
                                      device_info['user'],
                                      device_info['password'])

        elif device_info['os'] == 'ios-xr':

            device = NetworkDeviceXR(device_info['name'],
                                     device_info['ip'],
                                     device_info['user'],
                                     device_info['password'])

        else:
            device = NetworkDevice(device_info['name'],
                                   device_info['ip'],
                                   device_info['user'],
                                   device_info['password'])

        # Append this device object to list
        devices_list.append(device)

    return devices_list
```

### DISPLAY DEVICE INFORMATION FUNCTION

Create a function to display the device information on connection attempt.

```
# Print information to display
def print_device_info(device):

    print('----------------------------------------------------')
    print('    Device Name:      ', device.name)
    print('    Device IP:        ', device.ip_address)
    print('    Device username:  ', device.username,)
    print('    Device password:  ', device.password)
    print('----------------------------------------------------')
```

### WRITE DEVICE INFORMATION FUNCTION

Create a function to write the device information on connection attempt.

```
# Write information to file
def write_devices_info(devices_file, devices_list):

    print('---- Printing JSON output -------------------------------')

    # Create the list of lists with devices and device info
    devices_out_list = []  # create list for JSON output

    for device in devices_list:
        dev_info = {'name': device.name, 'ip': device.ip_address,
                    'os': device.os_type,
                    'user': device.username,
                    'password': device.password}
        devices_out_list.append(dev_info)

    pprint(devices_out_list)
    print('')

    # Convert the python device data into JSON for output to the file
    json_device_data = json.dumps(devices_out_list)

    # Output the JSON string to the file
    with open(devices_file, 'w') as json_file:
        json.file.write(json_device_data)
```

## STEP 3: CONFIGURE MAIN MODULE

Open the module **main.py**

### IMPORT REQUIRED LIBRARIES

Import the **read_devices_info**, **print_device_info** and **write_devices_info** functions from the **util** module.

```
# Import required modules/packages/library
from util import read_devices_info
from util import print_device_info
from util import write_devices_info
```

### READ DEVICE INFORMATION FROM FILE

Read the device information from the file **devices-14.json** using the function **read_devices_info**.

```
# Read JSON info for all devices
devices_list = read_devices_info('devices-14.json')
```

### CONNECT TO DEVICES, SHOW INTERFACES AND DISPLAY

Parsing the list of devices from the csv file, connect to each device, run the show interface summary command and display the device information on screen.

```
# Connect to device, show interface summary, display
for device in devices_list:

    print('==== Device ================================================')

    # Connect to this specific device
    device.connect()

    # Get interface info for specific device
    device.get_interfaces()

    # Print device details for this device
    print_device_info(device)

# Write JSON entry for all devices
write_devices_info('devices-14-out.json', devices_list)
```

## STEP 4: READ OUTPUT FILE TO VERIFY OUTPUT FILE IS CORRECT

Read from the output file and run the commands again to connect to the devices, show interfaces and display the output. To verify the output data.

```
# Do it all again, reading from the output file, to prove all correct
print('------------------------------------------------------------')
print('---------- Reading from the output file, to test ----------')

# Read JSON info for all devices
devices_list = read_devices_info('devices-14.json')

# Connect to device, show interface, display
for device in devices_list:

    print('==== Device ================================================')

    # Connect to this specific device
    device.connect()

    # Get interface info for specific device
    device.get_interfaces()

    # Print device details for this device
    print_device_info(device)
```

## STEP 5: SAVE MODULES, RUN AND VERIFY PACKAGE

Save you application and then run it from the terminal rather than from within visual studio code.

```
~/labs/prne/reading-and-writing-structured-files-part-4$ python3 main.py
```

The output from your application will be displayed in your terminal window, verify that it is comparable to below.

```
devasc@labvm:~/labs/prne/reading-and-writing-structured-files-part-4$ python3 main.py
==== Device ================================================
--- connecting IOS: telnet 192.168.56.101
------------------------------------------------------------
    Device Name:       CSR1kv
    Device IP:         192.168.56.101
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------------
==== Device ================================================
--- connecting IOS: telnet 192.168.56.130
------------------------------------------------------------
    Device Name:       R2
    Device IP:         192.168.56.130
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------------
---- Printing JSON output ------------------------------
[{'ip': '192.168.56.101',
  'name': 'CSR1kv',
  'os': 'ios',
  'password': 'cisco123!',
  'user': 'cisco'},
 {'ip': '192.168.56.130',
  'name': 'R2',
  'os': 'ios',
  'password': 'cisco123!',
  'user': 'cisco'}]

------------------------------------------------------------
---------- Reading from the output file, to test ----------
==== Device ================================================
--- connecting IOS: telnet 192.168.56.101
------------------------------------------------------------
    Device Name:       CSR1kv
    Device IP:         192.168.56.101
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------------
==== Device ================================================
--- connecting IOS: telnet 192.168.56.130
------------------------------------------------------------
    Device Name:       R2
    Device IP:         192.168.56.130
    Device username:   cisco
    Device password:   cisco123!
------------------------------------------------------------
devasc@labvm:~/labs/prne/reading-and-writing-structured-files-part-4$
```

### STEP 6: CHALLENGE (OPTIONAL)

The show interfaces commands are not showing in the output, adjust the relevant module to print out the interface summaries.

## PART 5 (OPTIONAL BUT HIGHLY RECOMMENDED)

As this lab is completed in NETLAB+ and your code files will be erased when the reservation ends, it is advisable to save your files in GitHub under your repository for this course.