LOGICALIS

Business and technology working as one

NEP@L

Scripting: Structured vs unstructured ouput

1. Introducción

Una de las cosas más difíciles cuando se escribe código (scripting) para extraer (data collections) y manipular datos de los equipos de networking es lidiar con las respuestas que vienen de manera no estructurada.

Cuando esto sucede, hay que escribir código adicional para interpretar (parsing) dicha salida. Para esto se requieren habilidad y conocimientos avanzados.

A continuación, les mostraremos el uso de dos librerías que se relacionan para obtener resultados estructurados: Netmiko y Genie.

2. Salida no estructurada

Ejemplo comando "show versión"

Código Python:

```
#!/usr/local/bin/python3
# Uso de Genie for parsing - NEP@L
# By Ed Scrimaglia

from netmiko import ConnectHandler
import pprint

def connectToDevice(_dev_type,_ip,_user,_pass,_enable):
        fromDevice = ConnectHandler(
                device_type=_dev_type,
                ip=_ip,
                username=_user,
                password=_pass,
                secret=_enable
        )
        fromDevice.enable()
        return fromDevice

def execute(_device,_cmd):
        try:
                output = _device.send_command(_cmd)
        except ConnectionError as error:
                print("{} error de conexión".format(error))
        except Exception as error:
                print("{} error ".format(error))
        return output

def main():
        comando = "show version"
        device = connectToDevice("cisco_ios", "10.54.1X.XX ", "admin", "xxxxxxx","xxxxxxx")
        output = execute(device,comando)
        if str(type(output)) == "<class 'dict'>":
                pprint.pprint(output)
        else:
                print (output)

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

Cuando se ejecuta el código mostrado arriba, se obtiene la siguiente salida:

Caso 1

```
Cisco IOS XE Software, Version 16.03.02
Cisco IOS Software [Denali], CSR1000V Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.3.2,
RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2016 by Cisco Systems, Inc.
Compiled Tue 08-Nov-16 18:12 by mcpre

Cisco IOS-XE software, Copyright (c) 2005-2016 by cisco Systems, Inc.
All rights reserved. Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0. The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0. For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.

ROM: IOS-XE ROMMON

NEPAL_CSR1000V_WAN uptime is 3 weeks, 2 days, 5 hours, 25 minutes
Uptime for this control processor is 3 weeks, 2 days, 5 hours, 29 minutes
System returned to ROM by reload
System image file is "bootflash:packages.conf"
Last reload reason: Critical software exception,
check bootflash:crashinfo_RP_00_00_20190727-174500-UTC

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

License Level: ax
License Type: Default. No valid license found.
Next reload license Level: ax

cisco CSR1000V (VXE) processor (revision VXE) with 1077783K/3075K bytes of memory.
Processor board ID 9ZVKFD9RM6Z
4 Gigabit Ethernet interfaces
32768K bytes of non-volatile configuration memory.
3018652K bytes of physical memory.
5677055K bytes of virtual hard disk at bootflash:.
0K bytes of at webui:.

Configuration register is 0x2102
```

LOGICALIS
Business and technology working as one

Scripting: Structured vs unstructured output

En apariencia, la salida sigue un cierto formato u ordenamiento, sin embargo, la variable "output" del codigo escrito, tiene el siguiente contenido:

Caso 2

```
>>> output
'Cisco IOS XE Software, Version 16.03.02\nCisco IOS Software [Denali], CSR1000V Software
(X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)\nTechnical Support:
http://www.cisco.com/techsupport\nCopyright (c) 1986-2016 by Cisco Systems, Inc.\nCompiled Tue
08-Nov-16 18:12 by mcpre\n\n\nCisco IOS-XE software, Copyright (c) 2005-2016 by cisco Systems, Inc.\nAll
rights reserved. Certain components of Cisco IOS-XE software are\nlicensed under the GNU General Public
License ("GPL") Version 2.0. The\nsoftware code licensed under GPL Version 2.0 is free software that
comes\nwith ABSOLUTELY NO WARRANTY. You can redistribute and/or modify such\nGPL code under the
terms of GPL Version 2.0. For more details, see the\ndocumentation or "License Notice" file accompanying the
IOS-XE software,\nor the applicable URL provided on the flyer accompanying the
IOS-XE\nsoftware.\n\n\nROM: IOS-XE ROMMON\n\nNEPAL_CSR1000V_WAN uptime is 3 weeks, 2 days, 5
hours, 56 minutes\nUptime for this control processor is 3 weeks, 2 days, 6 hours, 0 minutes\nSystem returned
to ROM by reload\nSystem image file is "bootflash:packages.conf"\nLast reload reason: Critical software
exception, check bootflash:crashinfo_RP_00_00_20190727-174500-UTC\n\n\n\nThis product contains
cryptographic features and is subject to United\nStates and local country laws governing import, export,
transfer and\nuse. Delivery of Cisco cryptographic products does not imply\nthird-party authority to import,
export, distribute or use encryption.\nImporters, exporters, distributors and users are responsible
for\ncompliance with U.S. and local country laws. By using this product you\nagree to comply with applicable
laws and regulations. If you are unable\nto comply with U.S. and local laws, return this product
immediately.\n\nA summary of U.S. laws governing Cisco cryptographic products may be found
at:\nhttp://www.cisco.com/wwl/export/crypto/tool/stqrg.html\n\nIf you require further assistance please
contact us by sending email to\nexport@cisco.com.\n\nLicense Level: ax\nLicense Type: Default. No valid
license found.\nNext reload license Level: ax\n\ncisco CSR1000V (VXE) processor (revision VXE) with
1077783K/3075K bytes of memory.\nProcessor board ID 9ZVKFD9RM6Z\n4 Gigabit Ethernet
interfaces\n32768K bytes of non-volatile configuration memory.\n3018652K bytes of physical
memory.\n5677055K bytes of virtual hard disk at bootflash:.\n0K bytes of at webui:.\n\nConfiguration register
is 0x2102\n'
>>>
```

La salida en el Caso 1 es lo que produce el comando Print interpretando los caracteres especiales y formateando el resultado. El Caso 2 contiene el resultado de la variable en estado puro, la que se debe manipular si se quiere, por ejemplo, encontrar la versión de IOS que está en el equipo. Por esto es por lo que, como comentábamos en la introducción, hace falta cierta habilidad y conocimiento para poder escribir código (parsing) y encontrar un valor determinado.

La solución a este problema es utilizar una librería llamada Genie, que se encarga de estructurar (parsear) la salida y devolver una estructura JSON, a partir de la cual es simple encontrar un valor determinado.

3. Salida estructurada

Ejemplo comando "show versión"

Código Python:

```
#!/usr/local/bin/python3
# Uso de Genie for parsing - NEP@L
# By Ed Scrimaglia

from netmiko import ConnectHandler
import pprint

def connectToDevice(_dev_type,_ip,_user,_pass,_enable):
        fromDevice = ConnectHandler(
                device_type=_dev_type,
                ip=_ip,
                username=_user,
                password=_pass,
                secret=_enable
        )
        fromDevice.enable()
        return fromDevice

def execute(_device,_cmd):
        try:
                output = _device.send_command(_cmd, use_genie=True) ***

        except ConnectionError as error:
                print("{} error de conexión".format(error))
        except Exception as error:
                print("{} error ".format(error))
        return output

def main():
        comando = "show version"
        device = connectToDevice("cisco_ios", "10.54.1X.XX ", "admin", "xxxxxxx","xxxxxxx")
        output = execute(device,comando)
        if str(type(output)) == "<class 'dict'>":
                pprint.pprint(output)
        else:
                print (output)

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

LOGICALIS
Business and technology working as one

Scripting: Structured vs unstructured output

Cuando se ejecuta el código mostrado anteriormente, se obtiene la siguiente salida:

```
{'version': {
                'chassis': 'CSR1000V',
                'chassis_sn': '9ZVKFD9RM6Z',
                'curr_config_register': '0x2102',
                'disks': {'bootflash:.': {'disk_size': '5677055',
                                          'type_of_disk': 'virtual hard disk'},
                'webui:.': {'disk_size': '0', 'type_of_disk': ''}},
                'hostname': 'NEPAL_CSR1000V_WAN',
                'image_id': 'X86_64_LINUX_IOSD-UNIVERSALK9-M',
                'image_type': 'production image',
                'last_reload_reason': 'Critical software exception, check '
                'bootflash':'crashinfo_RP_00_00_20190727-174500-UTC',
                'license_level': 'ax',
                'license_type': 'Default. No valid license found.',
                'main_mem': '1077783',
                'mem_size': {'non-volatile configuration': '32768',
                                          'physical': '3018652'},
                'next_reload_license_level': 'ax',
                'number_of_intfs': {'Gigabit Ethernet': '4'},
                'os': 'IOS-XE',
                'platform': 'CSR1000V',
                'processor_type': 'VXE',
                'rom': 'IOS-XE ROMMON',
                'rtr_type': 'CSR1000V',
                'system_image': 'bootflash:packages.conf',
                'uptime': '3 weeks, 2 days, 6 hours, 13 minutes',
                'uptime_this_cp': '3 weeks, 2 days, 6 hours, 17 minutes',
                'version': '16.3.2,',
                'version_short': '16.3'
          }
}
```

El formato de la salida es precisamente el formato JSON.

Para obtener el valor de la imagen que está ejecutando el equipo (en este caso, un CSR1000V) simplemente hay que ejecutar la instrucción:

Print (version[‘system_image]), y el resultado será **'bootflash:packages.conf'.**

Scripting: Structured vs unstructured output

El cambio que se ha hecho aquí es simplemente usar la librería Genie, específicamente la opción "use_genie=True" en el comando "send_command" de la librería Netmiko.

El contenido de la variable "output" es en este caso:

```
>>> output

{'version': {'version_short': '16.3', 'platform': 'CSR1000V', 'version': '16.3.2,', 'image_id':
'X86_64_LINUX_IOSD-UNIVERSALK9-M', 'image_type': 'production image', 'os': 'IOS-XE', 'rom': 'IOS-XE
ROMMON', 'hostname': 'NEPAL_CSR1000V_WAN', 'uptime': '3 weeks, 2 days, 6 hours, 13 minutes',
'uptime_this_cp': '3 weeks, 2 days, 6 hours, 17 minutes', 'system_image': 'bootflash:packages.conf',
'last_reload_reason': 'Critical software exception, check
bootflash:crashinfo_RP_00_00_20190727-174500-UTC', 'license_level': 'ax', 'license_type': 'Default. No valid
license found.', 'next_reload_license_level': 'ax', 'chassis': 'CSR1000V', 'main_mem': '1077783',
'processor_type': 'VXE', 'rtr_type': 'CSR1000V', 'chassis_sn': '9ZVKFD9RM6Z', 'number_of_intfs': {'Gigabit
Ethernet': '4'}, 'mem_size': {'non-volatile configuration': '32768', 'physical': '3018652'}, 'disks': {'bootflash:.':
{'disk_size': '5677055', 'type_of_disk': 'virtual hard disk'}, 'webui:.': {'disk_size': '0', 'type_of_disk': ''}},
'curr_config_register': '0x2102'}}
```

Luego de ordenar esta salida según el formato JSON, se obtiene el resultado mostrado en el ejemplo anterior. Cuando la salida de un comando esté en un formato como JSON o XML, YAML, etc., se dice la salida está en formato estructurado, y el código que hay que escribir para obtener un valor determinado es simplemente un comando.

4. Prerequisitos

Librerias Python:

- Netmiko v 2.4.1
- Genie v 19.4.0

5. Scope de Genie

Los comandos que Genie puede estructurar estan en el siguiente URL, por plataforma

https://pubhub.devnetcloud.com/media/pyats-packages/docs/genie/genie_libs/#/parsers

LOGICALIS

Business and technology working as one

www.la.logicalis.com /LogicalisLatam