10 September 2021
clement.parssegny@telecom-sudparis.eu

Inventory Management Automation for Network Devices

Clément PARSSEGNY
CERN, CH-1211 Geneva, Switzerland

Keywords: Information Technologies, InforEAM, Summer Program

Summary

This document is a report of my 12 weeks long internship among the IT-CS-CE section at CERN. Besides several lectures, presentations and virtual visits of CERN experiments, I worked on a project called "Inventory Management Automation for Network Device".
The CERN IT-CS group is in charge of managing the large complex of networks of the laboratory, comprising approximately 300 routers and 4000 switches. In an effort to improve inventory management they are using InforEAM, an asset management software, for storing all our assets. The aim of the project was to ensure that the information in the InforEAM system is always accurate and up to-date for the equipment that is live (in production). My main objective was to extract inventory data from various components of network devices and use a REST interface for updating the InforEAM records.

Contents

1 Introduction

This project has been achieved in the context of the broad infrastructure that the CERN possess. With more than 50 domains corresponding to the several experiments, technical networks or campus networks, there is a need of network devices to connect every machine and thus a need for many network devices in addition to the 50 000 kilometers of optical fiber used. Those devices then demand a management capability from IT sections to keep a clear inventory of the current infrastructure and act in case of dysfunction. Moreover, the status of the research center implies a heterogeneity in the vendor of devices. Juniper, HP, Cisco and other brands are present and all of them need to be taken into account despite their difference.
CERN is already using an Enterprise Asset Management software called InforEAM to complete this task. However, it has to be kept updated with the correct state of each device. This is where the project in which I participated enters. Thereafter, the working principle of this automation project will be detailed through different aspects. Then the results of this 12 weeks work will be discussed.

2 Working Principle

This section focuses on the technical details of the inventory automation project.

2.1 Logical functioning

There were several technical constraints which needed to be taken care of :

- All the main objectives have to be implemented.
  - Get the configuration from network devices.
  - Process them into a JSON format.
  - Send the configuration to the REST API.
  - Get logs from the API to print the update's status to the user or print the different components to the user (cf. *Ressources* section).
- The implemented functionalities have to be fully integrated to the rest of the program.
- The homogeneity in the coding form with the rest of the project to make future improvements easier.

The whole project rotates around a program called *netdev* that is used to perform different operations on network devices as it can be guessed. Figure 1 illustrates the main process of updating the configuration for a device on InforEAM :

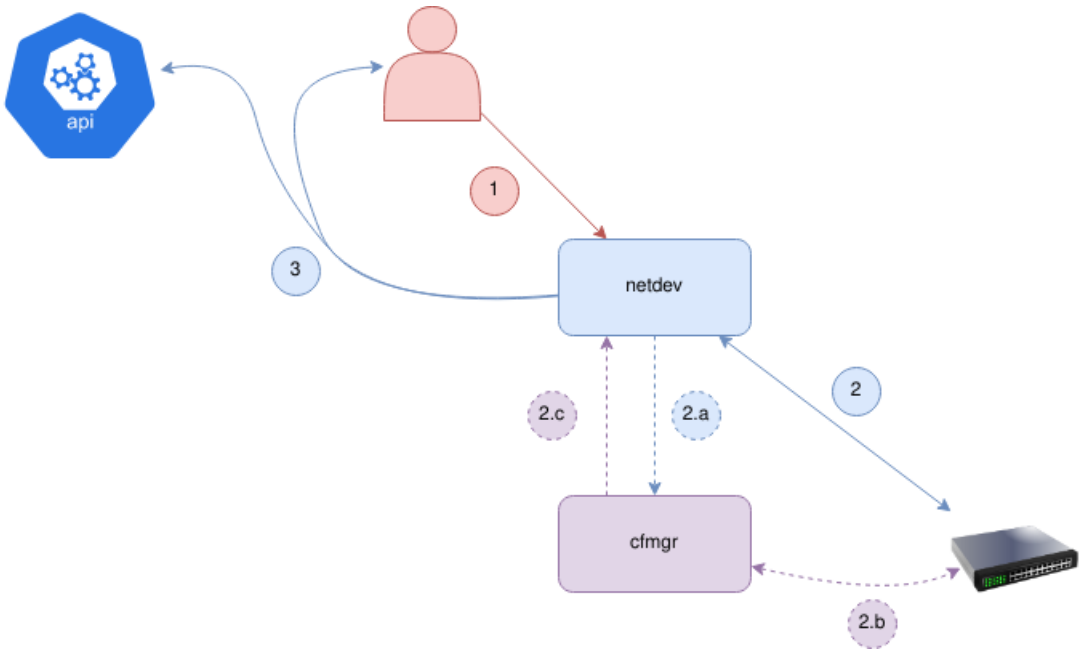


Figure 1: Automation of Network Device principle

First, the user call the netdev program to begin the update (using the *netdev update DeviceName* command). Then, depending on the device's vendor, the script may call another program called *cfmgr* (stands for configuration manager) before calling another part of netdev. This is one the remaining default of the program, caused by some devices whose data can not be processed in Python. In fact, *cfmgr* uses SNMP to connect to the device and get the inventory data while netdev has some methods to create a *network_driver* object that will retrieve all the information needed. Once the configuration is loaded from it, there is some JSON formatting and other calls to the LAN database to get other information such as the starpoint. The data is then sent to the REST API by a POST request. In the meantime, the answer delivered by the API is stocked to be finally displayed to the user as a report on the update with a table containing details on each component that have been updated.

2.2 Employed technologies

Because of the different components put through by the automation program, several languages and technologies are used for defined tasks.

2.2.1 Python

Python is a language modern, easy to understand. *netdev* program is using this language for configuration management so it has also been used for the inventory management part of the program.

2.2.2 Perl

PERL is an older language than Python that especially allow easier work with regular expressions. The *cfmgr* program is written in Perl. As some vendors are not supported in *netdev* yet, their inventory management has been integrated in *cfmgr* thus in Perl.

2.2.3 SNMP

SNMP (Simple Network Management Protocol) is a protocol used by system administrators to manage and monitor network devices remotely. Here, the protocol is used to retrieve the inventory details of the different devices involved in the command.

2.2.4 REST API

An API (Application Programming Interface) is a set of methods to connect different programs, so they can transfer data. A Representational state transfer (REST) API is based on HTTP methods to communicate. Here, the inventory is updated by POST requests.

3 Results

During these 12 weeks, several functionalities have been developed and integrated to the inventory management project. They allow more versatile tasks and a better control of what devices are involved when the script is called. Illustrations of the different processes' output implemented can be found in the Resources section.

3.1 Regular expressions

Regular expressions (also called regex) are patterns used for search some information. Here, regex are used as an input for *netdev* to operate on several devices at the same time.

For example, *netdev routers inventory update* $R\backslash d\{4\}*$ will match every router whose name begin with a R then 4 digits exactly and finally the remaining part of the device name like *R1000-S-RJUEM-2*.

This feature has been implemented using the *re* python library.

3.2 Parallel calls

One of the specifications for the automation of the inventory is to allow several operations at the same time. In fact, it is not possible to format the retrieved data. Thus, an identifier is needed to separate the different call of the *netdev* program. The decision made is to create a directory named after the PID of the *netdev* program. Thus, the configurations files retrieved are located in different places. In equation 1, an example of path for the router *R1000-S-RJUEM-2* inventory data:

$$netdev\_cfmgr\_data/\{PIDnumber\}/R1000-S-RJUEM-2/netdev-in/inventory.json \quad (1)$$

3.3 Stack Management

A stack switch is a group of different individual switches that are linked together to work as a one. It is really useful for large networks to simplify the setup and management of the network. At CERN, stacks are very common so it was necessary to adapt *netdev* to this kind of device. To do so, a sort of each component is done after retrieving all the data. After that, the different switches composing the stack and their component are differentiated by their device name. In fact, a letter is added at the end, as for LANDB. This process is illustrated in figure 2 :

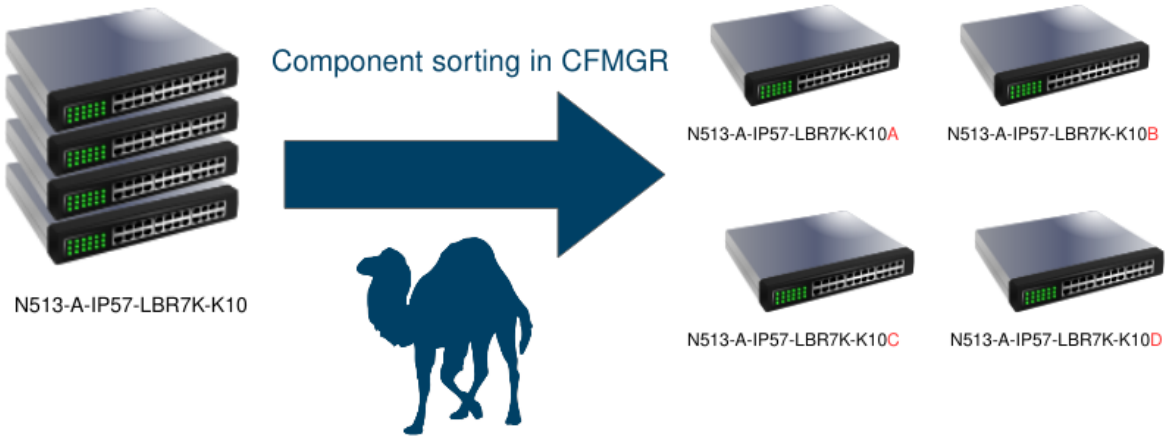


Figure 2: Stacked switches sorting principle in *cfmgr*

Finally, each switch is updated to the API separately. It allows to check the status of the component more precisely and ease maintenance operations.

3.4 Inheritance

In order to keep the whole project of inventory management easy to develop and maintain, the links between the different classes are important to understand and explain. The clearer the relationships between the different objects are, the easier the development will be. Thus, inheritance is a often used in *netdev* classes. Figure 3 is a UML diagram of the different classes to show how the inventory management has been implemented. The color of each class represents the file where it is located.

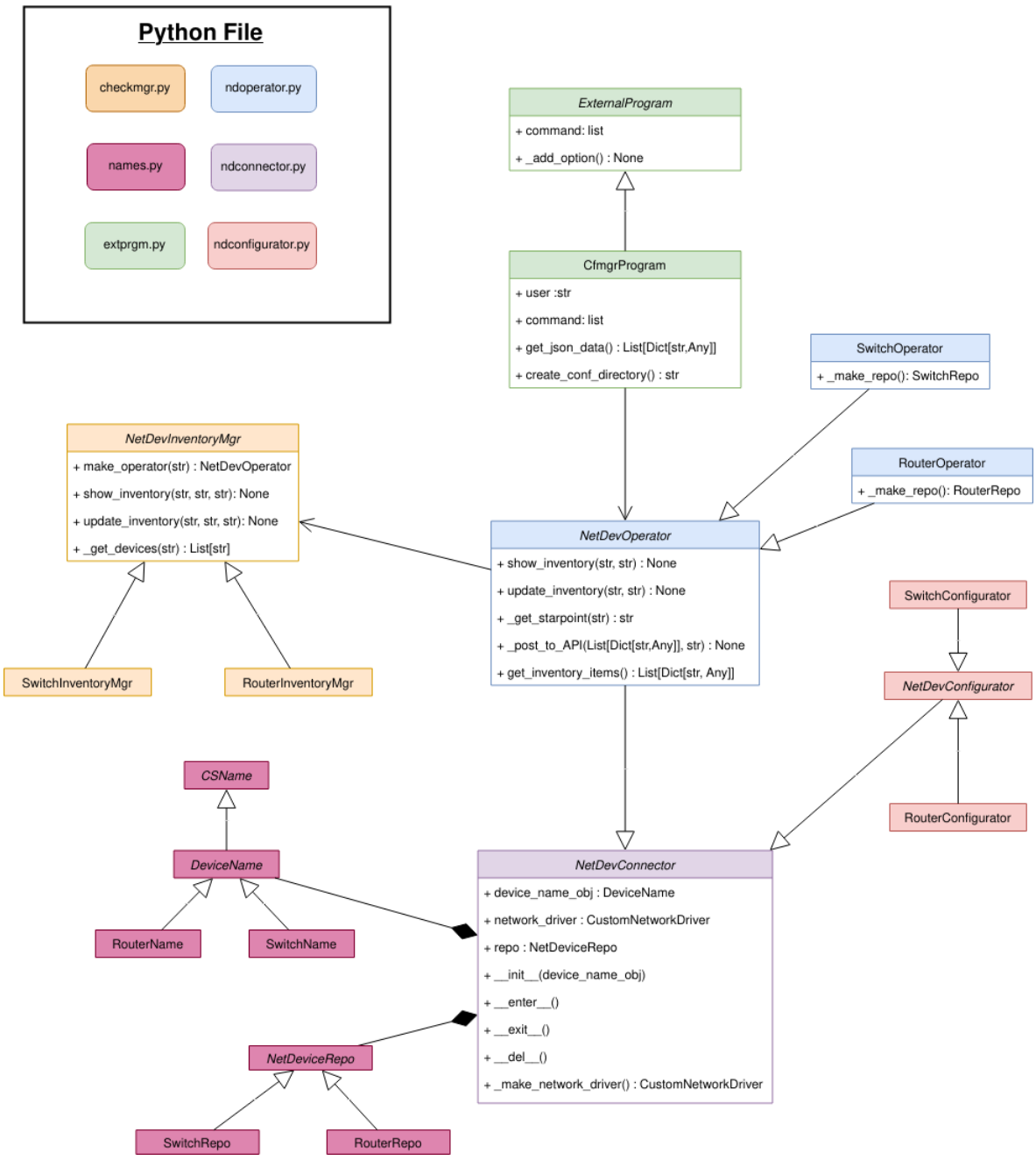


Figure 3: UML diagram of the Inventory Management structure

4 Conclusion

This project gave me the opportunity to get familiar with REST interfaces and requests at a production environment scale. Moreover, the heterogeneity of the network devices vendors (Juniper, HP, Paloalto, etc.) consisted an interesting challenge for me. I learned a lot about programming good habits and insights about various network management protocols.

This project has been separated into two major steps : the proof of work and the actual integration of the process in the production environment. The first part wasn't the most difficult but was necessary for the integration which represented a real challenge. This project structure taught me, in addition to technical knowledge, an example of how software development can be organised in time.

Acknowledgments

This Summer Program has been a wonderful experience in both professional and personal sides. But this was possible only because of the great people I had the opportunity to meet.

I want to thank Hannu Kamarainen for his tutoring during this internship. He taught me a lot of useful things that will help me in my future career.
I also want to thank Stefan Stancu and the whole IT-CS-CE section for their help and their warm welcome. It has been a real pleasure to work for them.
Finally, I want to thank the Summer Program Team for their support who worked really hard to make this edition real despite the sanitary situation.

Resources

```
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/python /opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/netdev routers inventory show R1000-S-RJUEM-2
Loading conf repo from '/tftpboot/router-conf/prod/r1000-s-rjuem-2'
+----------------+-----------+------------------------------+-----------------+--------------+------------+
| deviceName     | class     | descr                        | name            | serialNumber | partNumber |
+----------------+-----------+------------------------------+-----------------+--------------+------------+
| R1000-S-RJUEM-2 | chassis   | EX9208                       | Chassis         | J       8RFB | None       |
| R1000-S-RJUEM-2 | backplane | EX9208-BP3                   | Midplane        | A            | 7        50 |
| R1000-S-RJUEM-2 | other     | Front Panel Display          | FPM Board       | C            | 7       13 |
| R1000-S-RJUEM-2 | other     | PS 1.4-2.52kW; 90-264V AC in | PEM 0           | Q       7C0  | 7       46 |
| R1000-S-RJUEM-2 | other     | PS 1.4-2.52kW; 90-264V AC in | PEM 1           | Q       76A  | 7       46 |
| R1000-S-RJUEM-2 | other     | RE-S-EX9200-2X00x6           | Routing Engine 0 | C            | 7       58 |
| R1000-S-RJUEM-2 | other     | EX9200-SF2                   | CB 0            | C            | 7       52 |
| R1000-S-RJUEM-2 | module    | EX9200-12QS                  | FPC 0           | C            | 7       26 |
| R1000-S-RJUEM-2 | cpu       | SMPC PMB                     | CPU             | C            | 7       77 |
| R1000-S-RJUEM-2 | port      | QSFP+-4X10G-SR               | port 0/0/0      | X            | 7       53 |
| R1000-S-RJUEM-2 | port      | QSFP+40GE-LX4                | port 0/1/0      | 1       3DG  | 7       05 |
| R1000-S-RJUEM-2 | module    | EX9200-MPC                   | FPC 5           | C            | 7       64 |
| R1000-S-RJUEM-2 | cpu       | RMPC PMB                     | CPU             | C            | 7       19 |
| R1000-S-RJUEM-2 | other     | 20X1GE SFP MACSEC            | MIC 0           | C            | 7       61 |
| R1000-S-RJUEM-2 | port      | SFP-SX                       | MIC 0/0/2       | M       SN   | NON    R   |
| R1000-S-RJUEM-2 | port      | SFP+-10G-SR                  | MIC 0/0/4       | C       0Y   | NON-JNPR   |
| R1000-S-RJUEM-2 | port      | SFP-LX10                     | MIC 0/1/4       | G       051  | 7       50 |
| R1000-S-RJUEM-2 | other     | 20X1GE SFP MACSEC            | MIC 1           | C            | 7       61 |
| R1000-S-RJUEM-2 | fan       | Enhanced Left Fan Tray       | Fan Tray        | None         | None       |
+----------------+-----------+------------------------------+-----------------+--------------+------------+

Process finished with exit code 0
```

Figure 4: Output example of the *netdev routers inventory show* command for a Juniper router

```
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/python3 /opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/netdev routers inventory show G773-E-FPA32-90
Loading conf repo from '/tftpboot/router-conf/prod/g773-e-fpa32-90'
+----------------+---------+--------------------------+-------------+--------------+------------+
| deviceName     | class   | descr                    | name        | serialNumber | partNumber |
+----------------+---------+--------------------------+-------------+--------------+------------+
| G773-E-FPA32-90 | chassis | PA-3220                  | Chassis     | 01         7 | None       |
| G773-E-FPA32-90 | fan     | PA-3200-FANTRAY          | FANTRAY1    | 01         3 | None       |
| G773-E-FPA32-90 | power   | D1U54P-W-650-12-HB4C-PAN | PS1 (left)  | M2         3 | None       |
| G773-E-FPA32-90 | power   | D1U54P-W-650-12-HB4C-PAN | PS2 (right) | MZ         5 | None       |
+----------------+---------+--------------------------+-------------+--------------+------------+

Process finished with exit code 0
```

Figure 5: Output example of the *netdev routers inventory show* command for a Paloalto firewall

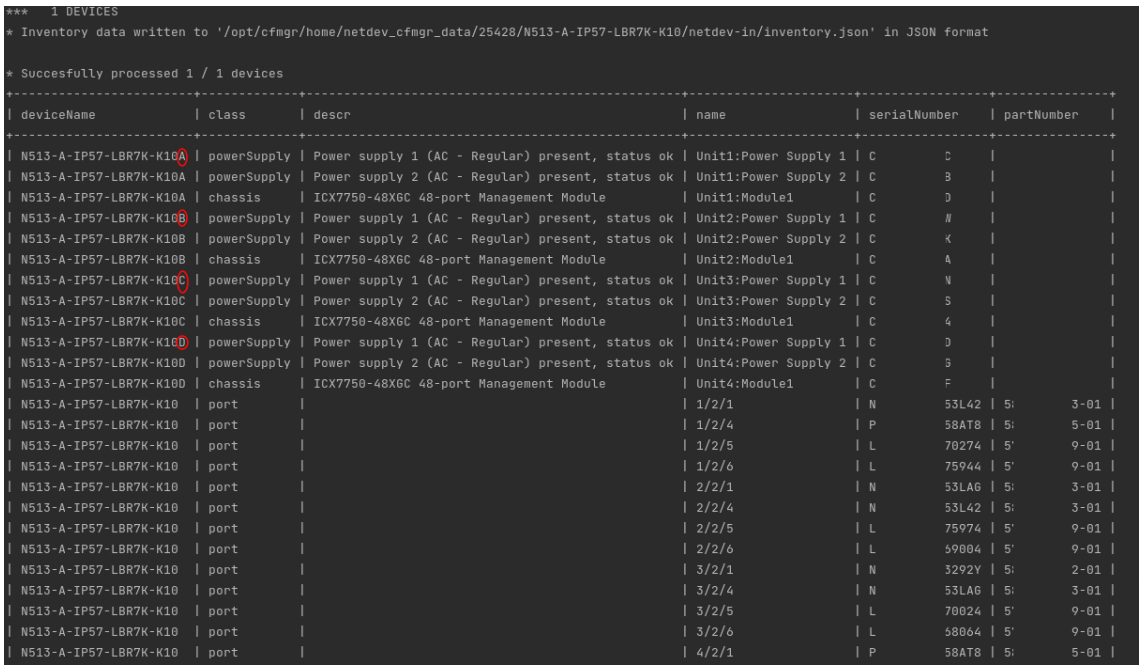
```
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/python3 /opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/netdev switches inventory show N513-A-IP57-LBR7K-K10
Loading conf repo from '/tftpboot/switch-conf/prod/n513-a-ip57-lbr7k-k10'
['/opt/dev-cfmgr/cparsseg/main/cfmgr.pl', '-interactive', 'switches', 'ops', 'inventory', 'N513-A-IP57-LBR7K-K10', '--out-file=/opt/cfmgr/home/netdev_cfmgr_data/25428/N513-A-IP57-LBR7K-K10/netdev-in/inventory.json']

**** CFMGR DEV SYSTEM

**** All modules will be loaded from /opt/dev-cfmgr/cparsseg/main/cfmgr_modules

cfmgr: Initializing.........
* Extracting device information from LANDB...
*** MATCHES
* N513-A-IP57-LBR7K-K10             *
***   1 DEVICES
* Inventory data written to '/opt/cfmgr/home/netdev_cfmgr_data/25428/N513-A-IP57-LBR7K-K10/netdev-in/inventory.json' in JSON format
```

Figure 6: Example of the *netdev switches inventory show* command for a switch stack



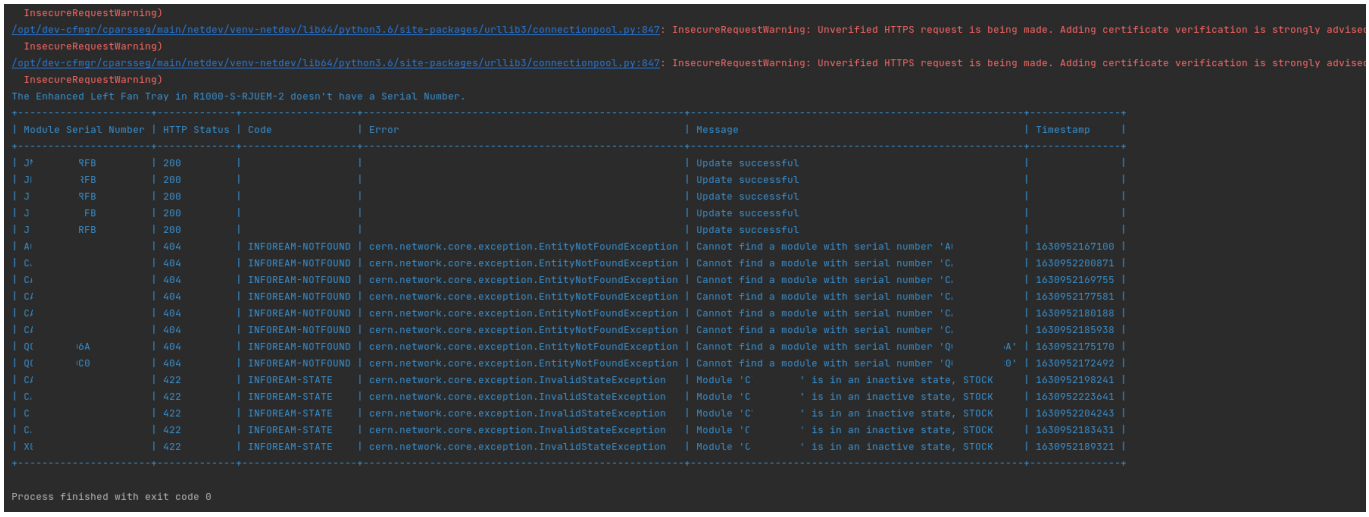
```
***   1 DEVICES
* Inventory data written to '/opt/cfmgr/home/netdev_cfmgr_data/25428/N513-A-IP57-LBR7K-K10/netdev-in/inventory.json' in JSON format

* Succesfully processed 1 / 1 devices
+------------------------+-------------+--------------------------------------------------+----------------------+-----------------+---------------+
| deviceName             | class       | descr                                            | name                 | serialNumber    | partNumber    |
+------------------------+-------------+--------------------------------------------------+----------------------+-----------------+---------------+
| N513-A-IP57-LBR7K-K10A | powerSupply | Power supply 1 (AC - Regular) present, status ok | Unit1:Power Supply 1 | C          D    |               |
| N513-A-IP57-LBR7K-K10A | powerSupply | Power supply 2 (AC - Regular) present, status ok | Unit1:Power Supply 2 | C          B    |               |
| N513-A-IP57-LBR7K-K10A | chassis     | ICX7750-48XGC 48-port Management Module          | Unit1:Module1        | C          D    |               |
| N513-A-IP57-LBR7K-K10B | powerSupply | Power supply 1 (AC - Regular) present, status ok | Unit2:Power Supply 1 | C          N    |               |
| N513-A-IP57-LBR7K-K10B | powerSupply | Power supply 2 (AC - Regular) present, status ok | Unit2:Power Supply 2 | C          K    |               |
| N513-A-IP57-LBR7K-K10B | chassis     | ICX7750-48XGC 48-port Management Module          | Unit2:Module1        | C          A    |               |
| N513-A-IP57-LBR7K-K10C | powerSupply | Power supply 1 (AC - Regular) present, status ok | Unit3:Power Supply 1 | C          N    |               |
| N513-A-IP57-LBR7K-K10C | powerSupply | Power supply 2 (AC - Regular) present, status ok | Unit3:Power Supply 2 | C          5    |               |
| N513-A-IP57-LBR7K-K10C | chassis     | ICX7750-48XGC 48-port Management Module          | Unit3:Module1        | C          4    |               |
| N513-A-IP57-LBR7K-K10D | powerSupply | Power supply 1 (AC - Regular) present, status ok | Unit4:Power Supply 1 | C          D    |               |
| N513-A-IP57-LBR7K-K10D | powerSupply | Power supply 2 (AC - Regular) present, status ok | Unit4:Power Supply 2 | C          5    |               |
| N513-A-IP57-LBR7K-K10D | chassis     | ICX7750-48XGC 48-port Management Module          | Unit4:Module1        | C          F    |               |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 1/2/1                | N          53L42 | 5          3-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 1/2/4                | P          58AT8 | 5          5-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 1/2/5                | L          70274 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 1/2/6                | L          75944 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 2/2/1                | N          53LAG | 5          3-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 2/2/4                | N          53L42 | 5          3-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 2/2/5                | L          75974 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 2/2/6                | L          59004 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 3/2/1                | N          3292Y | 5          2-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 3/2/4                | N          53LAG | 5          3-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 3/2/5                | L          70024 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 3/2/6                | L          58064 | 5          9-01 |
| N513-A-IP57-LBR7K-K10  | port        |                                                  | 4/2/1                | P          58AT8 | 5          5-01 |
```

Figure 7: Output example of the *netdev switches inventory show* command for a switch stack

```
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/python3 /opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/bin/netdev routers inventory update R1000-S-RJUEM-2
Loading conf repo from '/tftpboot/router-conf/prod/r1000-s-rjuen-2'
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/lib64/python3.6/site-packages/urllib3/connectionpool.py:847: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See:
  InsecureRequestWarning)
```

Figure 8: Example of the *netdev routers inventory update* command for a Juniper router



```
  InsecureRequestWarning)
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/lib64/python3.6/site-packages/urllib3/connectionpool.py:847: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See:
  InsecureRequestWarning)
/opt/dev-cfmgr/cparsseg/main/netdev/venv-netdev/lib64/python3.6/site-packages/urllib3/connectionpool.py:847: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See:
  InsecureRequestWarning)
The Enhanced Left Fan Tray in R1000-S-RJUEM-2 doesn't have a Serial Number.
+----------------------+-------------+------------------+-----------------------------------------------+-----------------------------------------------------+---------------+
| Module Serial Number | HTTP Status | Code             | Error                                         | Message                                             | Timestamp     |
+----------------------+-------------+------------------+-----------------------------------------------+-----------------------------------------------------+---------------+
| J*       ?FB         | 200         |                  |                                               | Update successful                                   |               |
| J        ?FB         | 200         |                  |                                               | Update successful                                   |               |
| J        ?FB         | 200         |                  |                                               | Update successful                                   |               |
| J         FB         | 200         |                  |                                               | Update successful                                   |               |
| J        RFB         | 200         |                  |                                               | Update successful                                   |               |
| A                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'A          | 1630952167100 |
| C                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'C          | 1630952200871 |
| C                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'C          | 1630952169755 |
| C                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'C          | 1630952177581 |
| C                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'C          | 1630952180188 |
| C                    | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'C          | 1630952185938 |
| Q(       6A          | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'Q          A' | 1630952175170 |
| Q(       C0          | 404         | INFOREAM-NOTFOUND | cern.network.core.exception.EntityNotFoundException | Cannot find a module with serial number 'Q          0' | 1630952172492 |
| C                    | 422         | INFOREAM-STATE    | cern.network.core.exception.InvalidStateException | Module 'C          ' is in an inactive state, STOCK | 1630952198241 |
| C                    | 422         | INFOREAM-STATE    | cern.network.core.exception.InvalidStateException | Module 'C          ' is in an inactive state, STOCK | 1630952223641 |
| C                    | 422         | INFOREAM-STATE    | cern.network.core.exception.InvalidStateException | Module 'C          ' is in an inactive state, STOCK | 1630952204243 |
| C                    | 422         | INFOREAM-STATE    | cern.network.core.exception.InvalidStateException | Module 'C          ' is in an inactive state, STOCK | 1630952183431 |
| X                    | 422         | INFOREAM-STATE    | cern.network.core.exception.InvalidStateException | Module 'C          ' is in an inactive state, STOCK | 1630952189321 |
+----------------------+-------------+------------------+-----------------------------------------------+-----------------------------------------------------+---------------+

Process finished with exit code 0
```

Figure 9: Output example of the *netdev routers inventory update* command for a Juniper router