

---

# **PyNSO Documentation**

***Release 1.1.0***

**Anthony Shaw**

June 22, 2016



<b>1</b>	<b>NED Overview</b>	<b>3</b>
<b>2</b>	<b>Authors</b>	<b>5</b>
<b>3</b>	<b>Installation (stable version)</b>	<b>7</b>
<b>4</b>	<b>Installation (development version)</b>	<b>9</b>
<b>5</b>	<b>Upgrading</b>	<b>11</b>
<b>6</b>	<b>Using it</b>	<b>13</b>
<b>7</b>	<b>Documentation</b>	<b>15</b>
7.1	Main . . . . .	15
7.1.1	Getting started . . . . .	15
7.1.2	PyNSO API - Client . . . . .	15
7.1.3	Enumerations . . . . .	16



PyNSO is a Python library to interface to Cisco Network Services Orchestration (NSO).

Cisco® Network Services Orchestrator (NSO) provides a single pane of glass for orchestrating a multivendor network. To offer an exceptional range of support for multivendor devices, it uses network element drivers (NEDs). Traditionally, device adaptors are a major roadblock, since they are not upgraded at the same pace as device interfaces. But adding support for new devices can take months. Cisco NSO NEDs, in contrast, can add new commands and devices in weeks. These drivers also provide extremely fine-grained representation of relevant configuration commands. Using NEDs, NSO also makes device configuration commands available over a networkwide, multivendor command line interface (CLI), APIs, and user interface. In addition, NSO services like VPN can configure a complex multivendor network.



---

### NED Overview

---

Network element drivers comprise the network-facing part of NSO. They communicate over the native protocol supported by the device, such as Network Configuration Protocol (NETCONF), Representational State Transfer (REST), Extensible Markup Language (XML), CLI, and Simple Network Management Protocol (SNMP).





---

### Authors

---

This Python module was developed by the Research and Development team at [Dimension Data](#). This module is provided under the [Apache 2.0 license](#) and the source code is available at [github.com](#).



---

## **Installation (stable version)**

---

PyNSO is available on PyPi. You can install latest stable version using pip:

```
pip install pynso
```



---

## **Installation (development version)**

---

You can install latest development version from our Git repository:

```
pip install -e git+https://github.com/DimensionDataCBUSydney/pynso.git@trunk#egg=pynso
```



---

## Upgrading

---

If you used pip to install the library you can also use it to upgrade it:

```
pip install --upgrade pynso
```





---

## Using it

---

Using PyNSO is simple, the datastores are based on the NETCONF standard and the translation between Python dictionaries and YANG is completed by the module automatically.

```
from pprint import pprint

from pynso.client import NSOClient
from pynso.datastores import DatastoreType

# Setup a client
client = NSOClient('10.159.91.14', 'admin', 'admin')

# Get information about the API
print('Getting API version number')
pprint(client.info()['version'])

# Get the information about the running datastore
print('Getting the contents of the running datastore')
pprint(client.get_datastore(DatastoreType.RUNNING))

# Get a data path
print('Getting a specific data path: snmp:snmp namespace and the agent data object')
pprint(client.get_data(DatastoreType.RUNNING, ('snmp:snmp', 'agent')))
```



## 7.1 Main

### 7.1.1 Getting started

In order to enable REST in NSO, REST must be enabled in `ncs.conf`. The web server configuration for REST is shared with the WebUI's config. However, the WebUI does not have to be enabled for REST to work.

Here's a minimal example of what is needed in the conf file:

```
<rest>
  <enabled>true</enabled>
</rest>
<webui>
  <enabled>>false</enabled>
  <transport>
    <tcp>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8080</port>
    </tcp>
  </transport>
</webui>
```

### 7.1.2 PyNSO API - Client

`class pynso.client.NSOClient` (*host, username, password, port=8080, ssl=False*)

**apply\_rollback** (*datastore, name*)

Apply a system rollback

**create\_data\_value** (*datastore, data\_path, data, params=None*)

Create (PUT) a data entry in a datastore

#### Parameters

- **datastore** (`DatastoreType`) – The target datastore
- **data\_path** (`list of str or tuple`) – The list of paths
- **data** (`dict`) – The new value at the given path

**Return type** `bool`

**Returns** `True` if successful, otherwise error.

**delete\_path** (*datastore, data\_path, params=None*)

Delete a data entry in a datastore

**Parameters**

- **datastore** (`DatastoreType`) – The target datastore
- **data\_path** (list of `str` or `tuple`) – The list of paths

**Return type** `bool`

**Returns** `True` if successful, otherwise error.

**get\_data** (*datastore, data\_path, params=None*)

Get a data entry in a datastore

**Parameters**

- **datastore** (`DatastoreType`) – The target datastore
- **data\_path** (list of `str` or `tuple`) – The list of paths

**get\_datastore** (*datastore, params=None*)

Get the details of a datastore

**Parameters** **datastore** (`DatastoreType`) – The target datastore

**get\_rollback** (*name*)

Get a list of stored rollbacks

**get\_rollbacks** ()

Get a list of stored rollbacks

**info** ()

Returns API information

**set\_data\_value** (*datastore, data\_path, data, params=None*)

Update (POST) a data entry in a datastore

**Parameters**

- **datastore** (`DatastoreType`) – The target datastore
- **data\_path** (list of `str` or `tuple`) – The list of paths
- **data** (`dict`) – The new value at the given path

**Return type** `bool`

**Returns** `True` if successful, otherwise error.

### 7.1.3 Enumerations

**class** `pynso.datastores.DatastoreType`

An enum of the resource types in the API.

**Variables**

- **config** – Link to the “config” resource
- **running** – Link to the “running” resource.
- **operational** – Link to the “operational” resource.

- **operations** – Container for available operations (i.e: YANG rpc statements).
- **rollbacks** – Container for available rollback files.

---

**Note:** Unless noted otherwise, all of the examples and code snippets in the documentation are licensed under the [Apache 2.0 license](#).

---



## A

`apply_rollback()` (pynso.client.NSOClient method), [15](#)

## C

`create_data_value()` (pynso.client.NSOClient method), [15](#)

## D

`DatastoreType` (class in pynso.datastores), [16](#)

`delete_path()` (pynso.client.NSOClient method), [16](#)

## G

`get_data()` (pynso.client.NSOClient method), [16](#)

`get_datastore()` (pynso.client.NSOClient method), [16](#)

`get_rollback()` (pynso.client.NSOClient method), [16](#)

`get_rollbacks()` (pynso.client.NSOClient method), [16](#)

## I

`info()` (pynso.client.NSOClient method), [16](#)

## N

`NSOClient` (class in pynso.client), [15](#)

## S

`set_data_value()` (pynso.client.NSOClient method), [16](#)