Pydantify

NetAutoberfest 2025

Urs Baumann 02.10.2025

Urs Baumann

```
>>> qr = QRCode()
>>> qr.add_data("https://www.linkedin.com/in/ubaumannch")
>>> qr.print_ascii()
```

CLI config vs NETCONF

```
sw04-pod-6#conf t
Enter configuration commands, one per line.
End with CNTL/Z.

sw04-pod-6(config)#int gi1/0/24
sw04-pod-6(config-if)#mtu ?
  <1500-9198>  MTU size in bytes

sw04-pod-6(config-if)#mtu 25000
                          ^
% Invalid input detected at '^' marker.
```

```
<rpc-error>
  <error-type>application</error-type>
  <error-tag>invalid-value</error-tag>
  <error-severity>error</error-severity>
  <error-path xmlns:t="http://example.com/schema/1.2/config">
    /t:top/t:interface[t:name="Ethernet0/0"]/t:mtu
  </error-path>
  <error-message xml:lang="en">
    MTU value 25000 is not within range 256..9192
  </error-message>
</rpc-error>
```

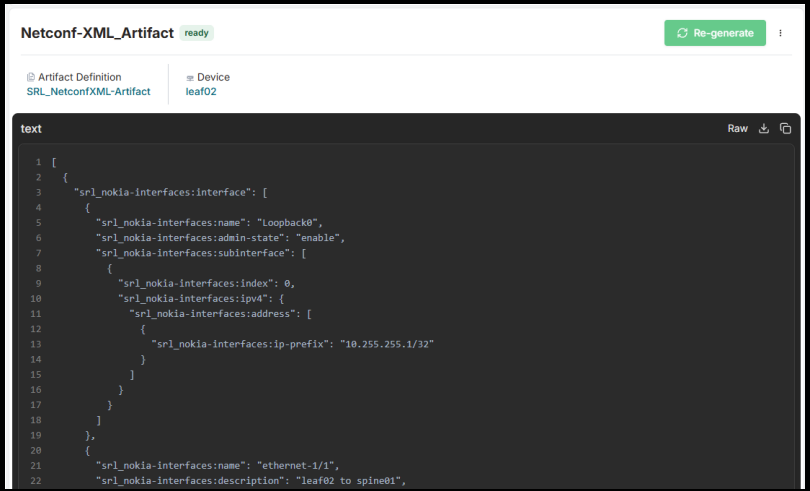
How do engineers work with YANG?

```
<config>
<if:interfaces xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
{% for interface in vlan_interfaces %}
  <if:interface>
    <if:name>Vlanif{{ interface.name }}</if:name>
    <if:description>uplink</if:description>
    <if:type xmlns:iana-if-type="urn:ietf:params:xml:ns:yang:iana-if-type">iana-if-type:propVirtual</if:type>
    <ip:ipv4 xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip">
      <ip:address>
        <ip:ip>{{ interface.ip.split("/")[0] }}</ip:ip>
        <ip:prefix-length>{{ interface.ip.split("/")[1] }}</ip:prefix-length>
      </ip:address>
    </ip:ipv4>
  </if:interface>
{% endfor %}
</if:interfaces>
</config>
```

What I want I

```python
class SRLYangPayloadHelper:
    def __init__(self) -> None:
        self.mac_vrfs: dict[int, list[str]] = defaultdict(list)

    @staticmethod
    def routed_subinterface_payload(
        ip: "IPv4Interface",
    ) -> "srl_if.SubinterfaceListEntry":
        ...

    @staticmethod
    def fabric_subinterface_payload() -> "srl_if.SubinterfaceListEntry":
        return srl_if.SubinterfaceListEntry(
            index=0,
            ipv6=srl_if.Ipv6Container(admin_state=srl_if.EnumerationEnum.enable),
        )

    def access_subinterface_payload(
        self, vlan: "NetworkVlan", interface_name: str
    ) -> "srl_if.SubinterfaceListEntry":
        # Add interface to the mac_vrf to be able to create the network-instance later
        self.mac_vrfs[vlan.vlan_id.value].append(f"{interface_name}.0")

        return srl_if.SubinterfaceListEntry(
            index=0,
            vlan=srl_if.VlanContainer(
                encap=srl_if.EncapContainer(untagged=srl_if.UntaggedContainer())
            ),
        )

    def trunk_subinterface_payload(
```

What I want II




```
[
  {
    "srl_nokia-interfaces:interface": [
      {
        "srl_nokia-interfaces:name": "Loopback0",
        "srl_nokia-interfaces:admin-state": "enable",
        "srl_nokia-interfaces:subinterface": [
          {
            "srl_nokia-interfaces:index": 0,
            "srl_nokia-interfaces:ipv4": {
              "srl_nokia-interfaces:address": [
                {
                  "srl_nokia-interfaces:ip-prefix": "10.255.255.1/32"
                }
              ]
            }
          }
        ]
      },
      {
        "srl_nokia-interfaces:name": "ethernet-1/1",
        "srl_nokia-interfaces:description": "leaf02 to spine01",
```

What I want III

- Python objects with typing, validation and documentation
- Consumable / as intuitive and user-friendly as possible
- IDE autocompletion
- IDE error checks
- IDE type-hints
- Serialization / Deserialization

Eco-System

- pyangbind
- ygot
- YANG Development Kit (ydk-gen)

How it started

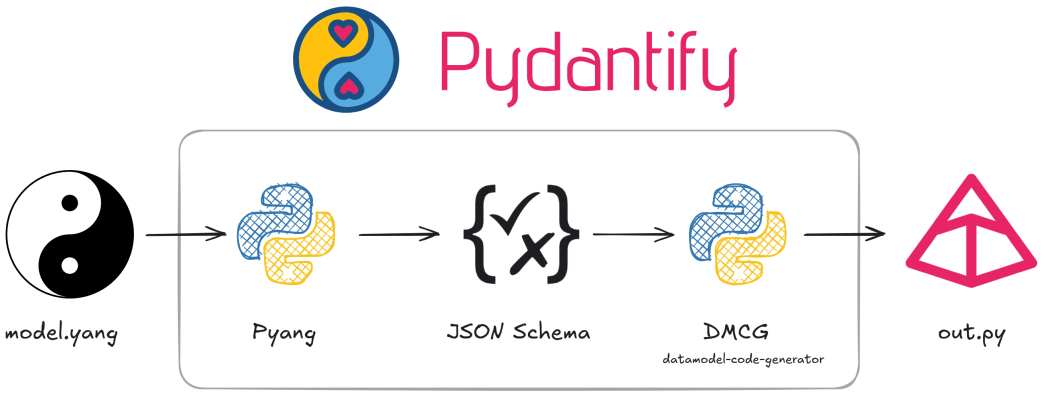
- Midterm Project @ Eastern Switzerland University of Applied Sciences
  Walther, Dominic and Jovicic, Dejan (2023)
  Make Model Driven Network Automation Pythonic
  `https://eprints.ost.ch/id/eprint/1089/`

Now

- https://pydantify.github.io/pydantify/
- https://github.com/pydantify/pydantify
- https://pypistats.org/packages/pydantify

Vendor

- `https://github.com/srl-labs/pydantic-srlinux/`
  experimental project
- `https://www.youtube.com/watch?v=CM3sT55zwt0`
  How can you use Pydantify? — Roman Dodin (Nokia)


Pydantify
model.yang
Pyang
JSON Schema
DMCG
datamodel-code-generator
out.py

YANG model

```
module my-endpoint {
  namespace
  "http://pydantify/ns/yang/pydantify-endpoint";

  prefix "ep";

  description 'Example model';

  typedef port {
    type uint16 {
        range "1 .. 65535";
      }
  }

  container endpoint {
    description "Definition of a endpoint";
    leaf address {
      type string;
      description "Endpoint address. IP or FQDN";
      mandatory true;
    }
    leaf port {
      type port;
      description "Port number between 1 and 65535";
      mandatory true;
    }
    leaf description {
      type string;
      description "Endpoint description";
    }
  }
}
```

YANG tree + JSON example

```
module: my-endpoint
  +--rw endpoint
     +--rw address        string
     +--rw port           port
     +--rw description?   string
```

```
{
  "my-endpoint:endpoint": {
    "address": "::1",
    "port": 2222,
    "description": "Port 2222 on localhost"
  }
}
```

Create JSON data the pythonic way

```python
from endpoint import Model, EndpointContainer

port = 8080
host = "localhost"

endpoint1 = Model(endpoint=EndpointContainer(address=host, port=port))
json_output = endpoint1.model_dump_json(indent=2, exclude_none=True)
```

Pydantic Model

```python
class EndpointContainer(BaseModel):
    """
    Definition of a endpoint
    """
    model_config = ConfigDict(populate_by_name=True, regex_engine="python-re")
    address: Annotated[str, Field(alias="my-endpoint:address", title="AddressLeaf")]
    """
    Endpoint address. IP or FQDN
    """
    port: Annotated[
        int, Field(alias="my-endpoint:port", ge=1, le=65535, title="PortLeaf")
    ]
    """
    Port number between 1 and 65535
    """
    description: Annotated[
        Optional[str], Field(alias="my-endpoint:description", title="DescriptionLeaf")
    ] = None
    """
    Endpoint description
    """
```

Challenges / not supported

- Pydantify: YANG > Pyyang > **JSONSchema** > DMGC > Pydantic
- Context-aware validation
- must / if / ...
- xPath
- bits / decimal64 / empty representation
- RegEx

The ”empty” type I

An ”empty” value is represented as ”[null]”, i.e., an array with the ”null” literal being its only element. For the purposes of this document, ”[null]” is considered an atomic scalar value.

This encoding of the ”empty” type was chosen instead of using simply ”null” in order to facilitate the use of empty leafs in common programming languages where the ”null” value of a member is treated as if the member is not present.

rfc7951, Section 6.9

The ”empty” type II

```
leaf foo {
    type empty;
  }
```

```
”foo”: [null]
```

The ”empty” type III

```python
class InterfaceContainer(BaseModel):
    model_config = ConfigDict(
        populate_by_name=True,
        regex_engine="python-re",
    )

    primary: Annotated[
        Optional[List[None]],
        Field(alias="interface:primary", max_length=1, min_length=1),
    ] = None
```

Questions?