



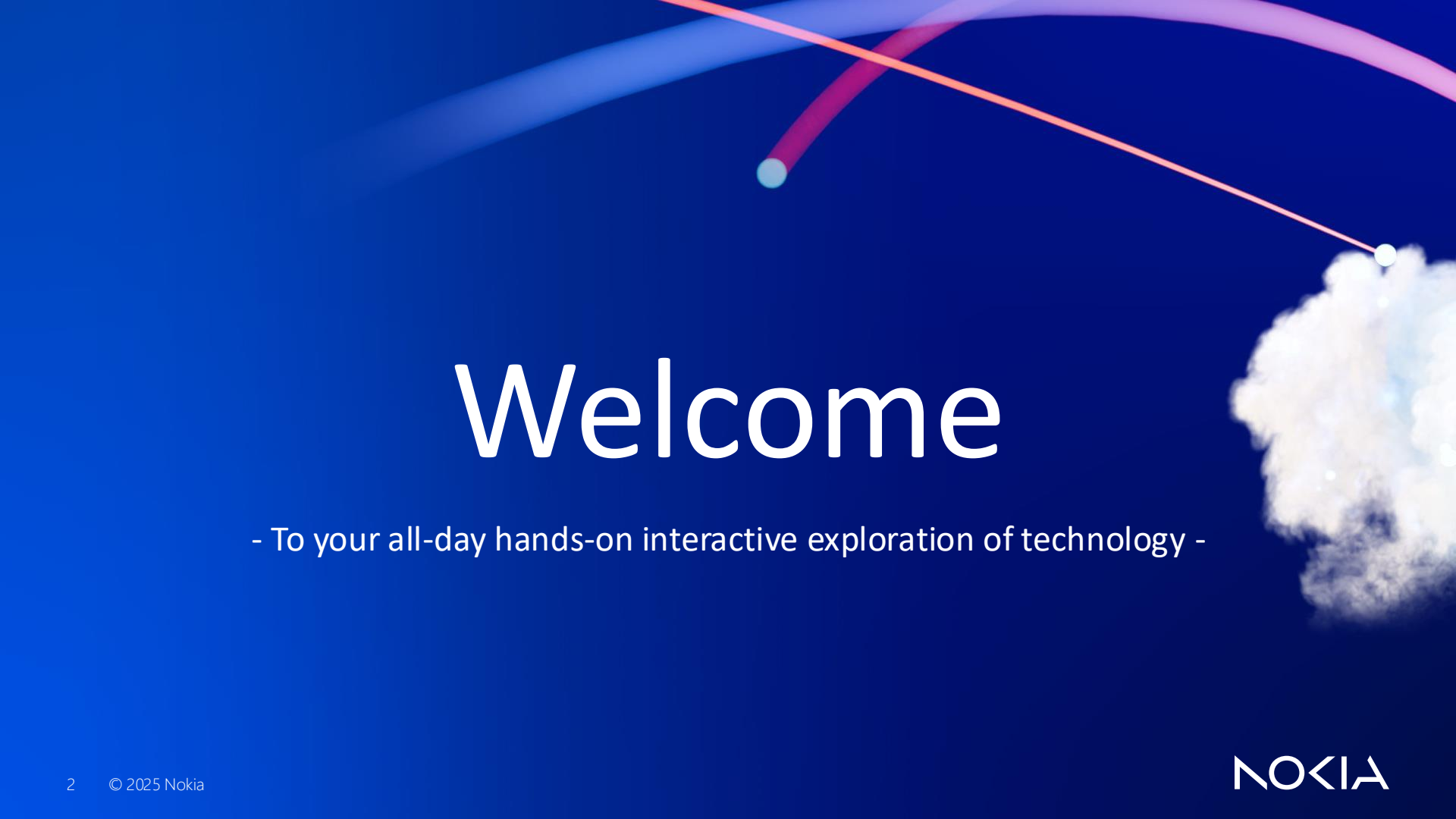
SReXperts²⁰²⁵
Network the cloud

SReXplore 2025 Primer

Network Operating Systems (NOS)

James Cumming

Bhavish Khatri



Welcome

- To your all-day hands-on interactive exploration of technology -

SReXperts²⁰²⁵

Network the cloud





SReXplore

What's in store

- Event schedule
 - **Morning: SReXplore Primer Sessions**
Equip you with the knowledge to participate in the hands on event in the afternoon.
 - **Afternoon: SReXplore**
This is why you're here. Hands on, do anything, try anything, break anything session to test your skills or learn new ones. Fun quiz to close the day.
- Event notices
 - Fire
 - Coffee
 - Restrooms



NOKIA

SReXplore

What do you need

- Yourself
- An open mind and a willingness to try new things
- Your trusty laptop (and charger)
 - SSH client
 - Web Browser
- A small amount of pre-requisite knowledge



Don't panic, that is why you are here!



NOKIA



SSID: SReXperts
Password: Noki@2025



Presentation live sharing on Teams

<https://tinyurl.com/SRX-APAC-HackathonPrimer-NOS>

Network Operating Systems Stream

SR OS & SR Linux



SReXplore Primer Agenda

1. YANG and model-driven management
2. Python
 - pySROS
 - SR Linux CLI plugins
 - NDK
3. gRPC
4. Event Handling
5. Ready for the afternoon



YANG and model-driven management



Nokia is model-driven

Enables simpler, more reliable and more efficient network management

The model-driven approach

Use of data models to define network elements. Provides a clear structured input and output framework for the network device.

Which data model

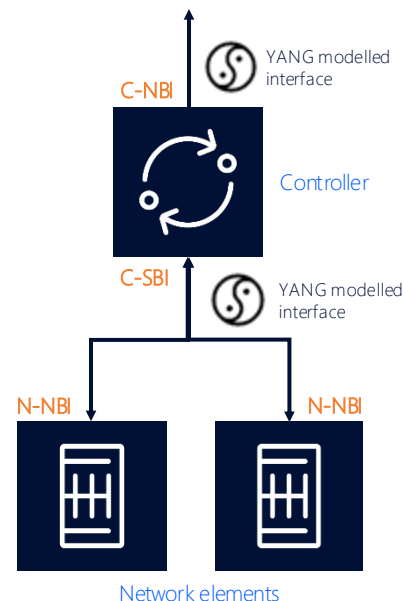
YANG is the widely used, standards based, machine and human-readable language. Models describe a rich set of data constructs for configuration, state and operations.

Drivers for model-driven

Complexity of integrating multi-vendor network elements each with their own CLI interface in the operator's management systems.

Model-driven interfaces

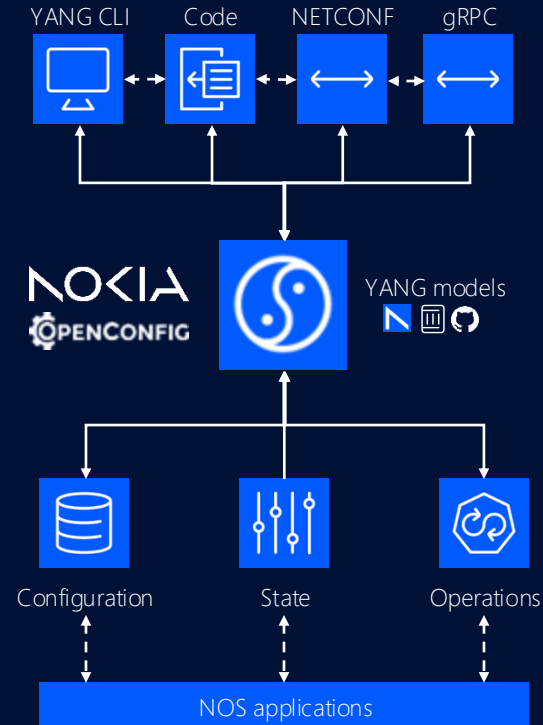
Controllers have northbound (NBI) and southbound (SBI) interfaces. Routers have northbound interfaces. C-SBI/N-NBI may be NETCONF, gRPC or model-driven CLI based.



Nokia NOS

Model-driven heart

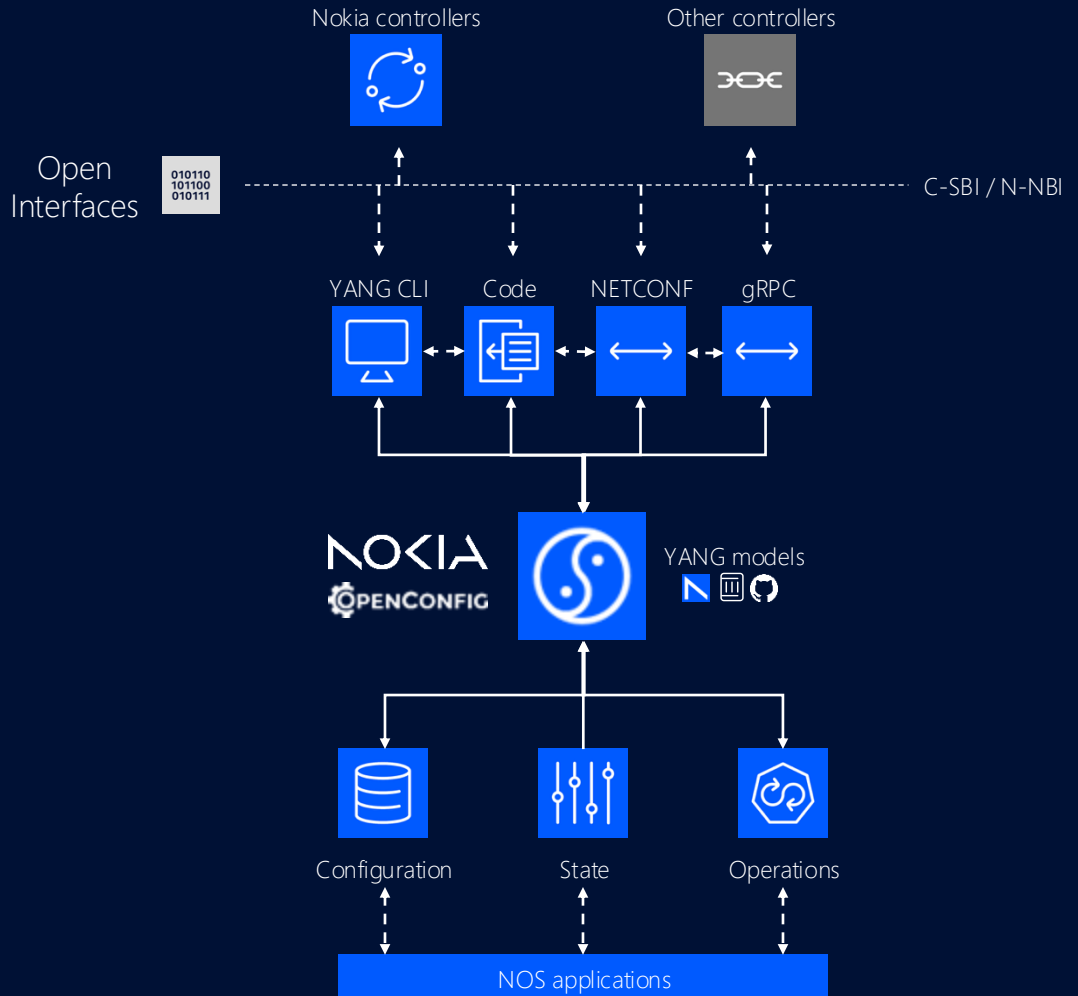
- With YANG models at the heart of the NOS all features are developed in YANG
- Mature, stable technology with coverage for all configuration, state and operational needs
- Extensive OpenConfig coverage
- Network integrity safeguarded by transactional, atomic configuration changes
- Embedded Python programming language enables automation, integration and customization
- YANG model publish-on-release philosophy suited to automation pipelines



Nokia NOS

Model-driven heart

- Open interfaces facilitate integration with network management solutions including Nokia, third-party or home-grown systems
- Feature rich NETCONF and gRPC protocols communicate on the C-SBI/N-NBI interface
- YANG based CLI provides familiar, enhanced interface for developers and operators alike
- Nokia controllers provide fully integrated and tested model-driven management solutions
- Structured data and consistency between interfaces lowers integration costs



YANG 101

SReXperts **event.yang**

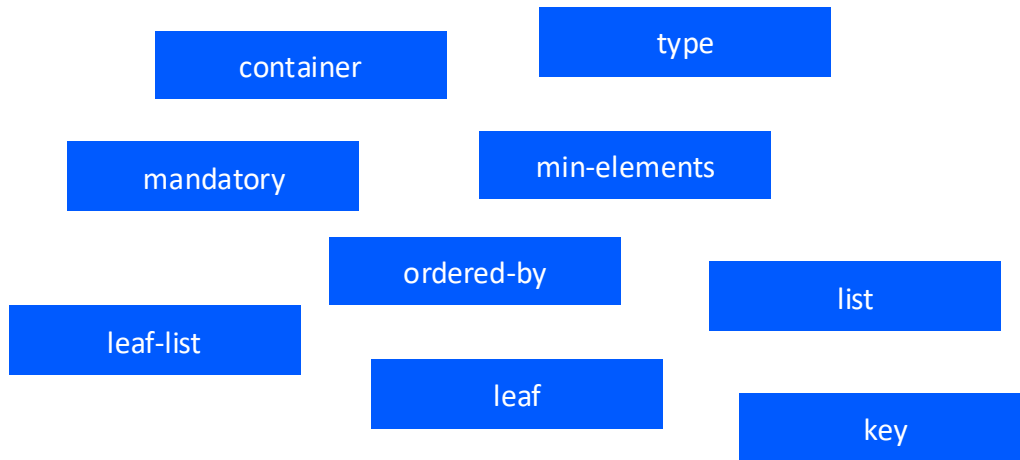
- YANG modules are human and machine readable files defining a schema
- Does not define the instance data
- Base elements
 - **module**: The name of the YANG module. Matches the filename.
 - **namespace**: A very important concept. Every YANG element is defined as the sum of the namespace and the element itself.
 - **prefix**: Locally significant identifier for this module
 - **revision**: The revision date of this module, there can be more than one revision showing the history of the module (work underway in the IETF NETMOD group to define semantic versioning)
 - A top-level container or list

```
module event {  
  yang-version "1.1";  
  namespace "urn:nokia.com:event";  
  prefix "event";  
  revision "2025-06-03";  
  container event {  
    leaf description {  
      type string;  
    }  
  }  
  list sessions {  
    key name;  
    leaf name {  
      type enumeration {  
        enum nos-primer;  
        enum nsp-primer;  
        enum eda-primer;  
        enum hands-on;  
      }  
    }  
    leaf description {  
      type string;  
      mandatory true;  
    }  
    leaf-list presenter {  
      type enumeration {  
        enum james;  
        enum thomas;  
        enum roman;  
        enum zeno;  
        enum sven;  
        enum siva;  
      }  
      ordered-by user;  
      min-elements 1;  
    }  
  }  
}
```

YANG 101

SReXperts **event.yang**

- This model describes an event



```
module event {  
  yang-version "1.1";  
  namespace "urn:nokia.com:event";  
  prefix "event";  
  revision "2025-06-03";  
  container event {  
    leaf description {  
      type string;  
    }  
    list sessions {  
      key name;  
      leaf name {  
        type enumeration {  
          enum nos-primer;  
          enum nsp-primer;  
          enum eda-primer;  
          enum hands-on;  
        }  
      }  
      leaf description {  
        type string;  
        mandatory true;  
      }  
      leaf-list presenter {  
        type enumeration {  
          enum james;  
          enum thomas;  
          enum roman;  
          enum zeno;  
          enum sven;  
          enum siva;  
        }  
        ordered-by user;  
        min-elements 1;  
      }  
    }  
  }  
}
```

YANG 101

SReXperts event.yang

- Now we have a schema, let's consider the data

```
<event xmlns="urn:nokia.com:event">
  <description>20th SReXperts Conference</description>
  <sessions>
    <name>nos-primer</name>
    <description>Network Operating Systems (NOS) Primer</description>
    <presenter>james</presenter>
    <presenter>thomas</presenter>
  </sessions>
</event>
```

- We can validate this against the YANG schema

```
yanglint /yang/event.yang /yang/event.xml
```

```
module event {
  yang-version "1.1";
  namespace "urn:nokia.com:event";
  prefix "event";
  revision "2025-06-03";
  container event {
    leaf description {
      type string;
    }
    list sessions {
      key name;
      leaf name {
        type enumeration {
          enum nos-primer;
          enum nsp-primer;
          enum eda-primer;
          enum hands-on;
        }
      }
      leaf description {
        type string;
        mandatory true;
      }
      leaf-list presenter {
        type enumeration {
          enum james;
          enum thomas;
          enum roman;
          enum zeno;
          enum sven;
          enum siva;
        }
        ordered-by user;
        min-elements 1;
      }
    }
  }
}
```

YANG 101

SReXperts event.yang

- Because our schema contains rules, invalid data is rejected and reported

```
<event xmlns="urn:nokia.com:event">
  <description>20th SReXperts Conference</description>
  <sessions>
    <name>nos-primer</name>
    <description>Network Operating Systems (NOS) Primer</description>
    <presenter>james</presenter>
  </sessions>
  <sessions>
    <name>nsp-info</name>
    <presenter>sven</presenter>
    <presenter>siva</presenter>
    <presenter>wilma</presenter>
  </sessions>
  <sessions>
    <name>eda-primer</name>
    <description>Event Driven Automation (EDA) Primer</description>
  </sessions>
</event>
```



This is a path
(hold that thought)

- We can validate this against the YANG schema

```
yanglint /yang/event.yang /yang/invalid.xml
libyang err : Invalid enumeration value "nsp-info". (/event:event/sessions/name) (line 9)
libyang err : Invalid enumeration value "wilma". (/event:event/sessions/presenter) (line 12)
libyang err : List instance is missing its key "name". (/event:event/sessions) (line 13)
libyang err : Too few "presenter" instances. (/event:event/sessions[name='eda-primer']/presenter)
YANGLINT[E]: Failed to parse input data file "/yang/invalid.xml".
```

```
module event {
  yang-version "1.1";
  namespace "urn:nokia.com:event";
  prefix "event";
  revision "2025-06-03";
  container event {
    leaf description {
      type string;
    }
  }
  list sessions {
    key name;
    leaf name {
      type enumeration {
        enum nos-primer;
        enum nsp-primer;
        enum eda-primer;
        enum hands-on;
      }
    }
    leaf description {
      type string;
      mandatory true;
    }
    leaf-list presenter {
      type enumeration {
        enum james;
        enum thomas;
        enum roman;
        enum zeno;
        enum sven;
        enum siva;
      }
      ordered-by user;
      min-elements 1;
    }
  }
}
```

YANG 101

SReXperts event.yang

- We can also use our schema to change formats
 - Note the order change
 - Supported encodings: XML, JSON (IETF)

```
<event xmlns="urn:nokia.com:event">
  <description>20th SReXperts Conference</description>
  <sessions>
    <name>nos-primer</name>
    <description>Network Operating Systems (NOS) Primer</description>
    <presenter>james</presenter>
    <presenter>thomas</presenter>
  </sessions>
  <sessions>
    <name>nsp-primer</name>
    <description>Network Services Platform (NSP) Primer</description>
    <presenter>sven</presenter>
    <presenter>siva</presenter>
  </sessions>
  <sessions>
    <name>eda-primer</name>
    <description>Event Driven Automation (EDA) Primer</description>
    <presenter>roman</presenter>
    <presenter>zeno</presenter>
  </sessions>
  <sessions>
    <name>hands-on</name>
    <description>The main hands on event at SReXperts SReXplore</description>
    <presenter>james</presenter>
  </sessions>
</event>
```



yanglint /yang/event.yang /yang/complete.xml -format json



```
{
  "event:event": {
    "description": "20th SReXperts Conference",
    "sessions": [
      {
        "name": "hands-on",
        "description": "The main hands on event at SReXpert SReXplore",
        "presenter": [
          "james"
        ]
      },
      {
        "name": "eda-primer",
        "description": "Event Driven Automation (EDA) Primer",
        "presenter": [
          "roman",
          "zeno"
        ]
      },
      {
        "name": "nsp-primer",
        "description": "Network Services Platform (NSP) Primer",
        "presenter": [
          "sven",
          "siva"
        ]
      },
      {
        "name": "nos-primer",
        "description": "Network Operating Systems (NOS) Primer",
        "presenter": [
          "james",
          "thomas"
        ]
      }
    ]
  }
}
```

YANG paths

SReXperts event.yang

- A path is a way of representing (on one line) the location of an element in the YANG schema
- Three common formats supported in Nokia NOS'

json-instance-path

```
/event:event/sessions[name="hands-on"]/description
```

gnmi-path

```
/event/sessions[name=hands-on]/description
```

cli-path

```
/event sessions "hands-on" description
```

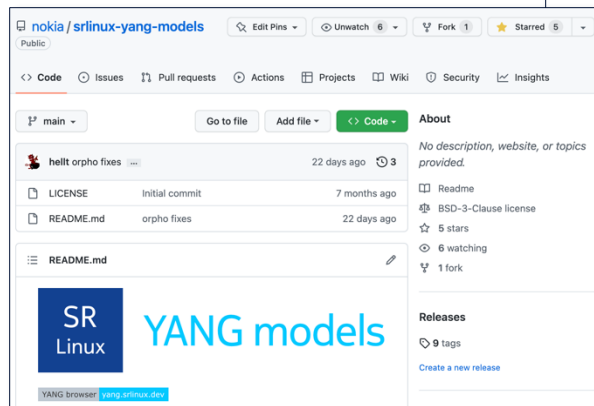
```
{
  "event:event": {
    "description": "20th SReXperts Conference",
    "sessions": [
      {
        "name": "hands-on",
        "description": "The main hands on event at SReXperts SReXplore",
        "presenter": [
          "james"
        ]
      },
      {
        "name": "eda-primer",
        "description": "Event Driven Automation (EDA) Primer",
        "presenter": [
          "roman",
          "zeno"
        ]
      },
      {
        "name": "nsp-primer",
        "description": "Network Services Platform (NSP) Primer",
        "presenter": [
          "sven",
          "siva"
        ]
      },
      {
        "name": "nos-primer",
        "description": "Network Operating Systems (NOS) Primer",
        "presenter": [
          "james",
          "thomas"
        ]
      }
    ]
  }
}
```

YANG models

Where to find SR OS, SR Linux and supported OC models

- Nokia YANG configuration, state, operations* and supported third-party models are available for every release:
 - Nokia support portal
 - Nokia GitHub repo: <https://github.com/nokia/YangModels>
 - Global YANG models repo: <https://github.com/YangModels/yang>
- Use standard Git tools to manage models for different releases
- Available directly on the device

SRBuilds Automated updating of releases. Release sros_24.3 2a28af0 · 2 months ago		
latest_sros_19.10	Automated updating of releases. R...	3 years ago
latest_sros_19.5	Automated updating of releases.	5 years ago
latest_sros_19.7	Automated updating of releases.	5 years ago
latest_sros_20.10	Automated updating of releases. R...	2 years ago



Electronic Delivery > Downloads

Please select a Product / Sub-category:

7750 SR (Service Router)

Please navigate through the hierarchy to download:

24.3

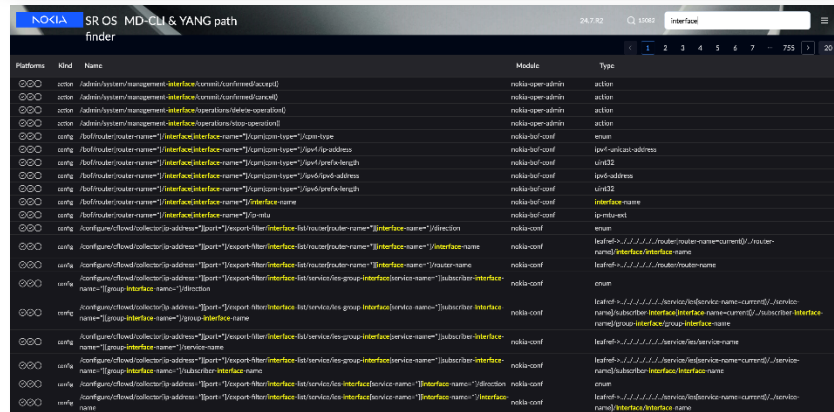
R1

Please select one or more Downloads:

Name
☐ Nokia-7750 SR-MIBs-24.3.R1
☐ Nokia-7750 SR-NISH-24.3.R1
☐ Nokia-7750 SR-PROTOS-24.3.R1
☐ Nokia-7750 SR-TIMOS-24.3.R1
☐ Nokia-7750 SR-TIMOS-ZTP-24.3.R1
☐ Nokia-7750 SR-YANG-24.3.R1
☐ Nokia-7750 SR-pySROS-24.3.R1

Point-and-click YANG navigation

- SR Linux: <https://yang.srlinux.dev>
SR OS: <https://yang.labctl.net>



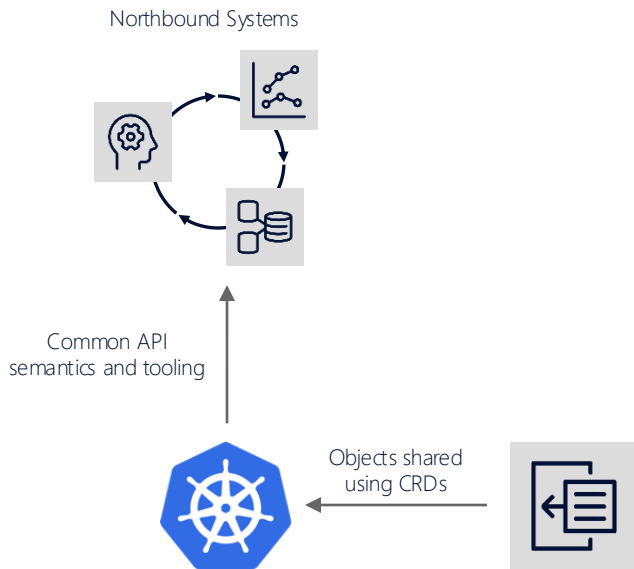
Python

...and the hands on event

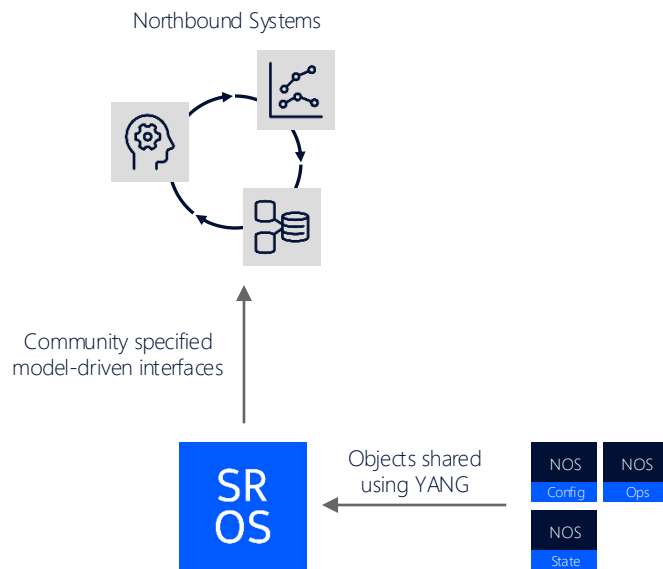
Modularization

Continuous evolution

Application world



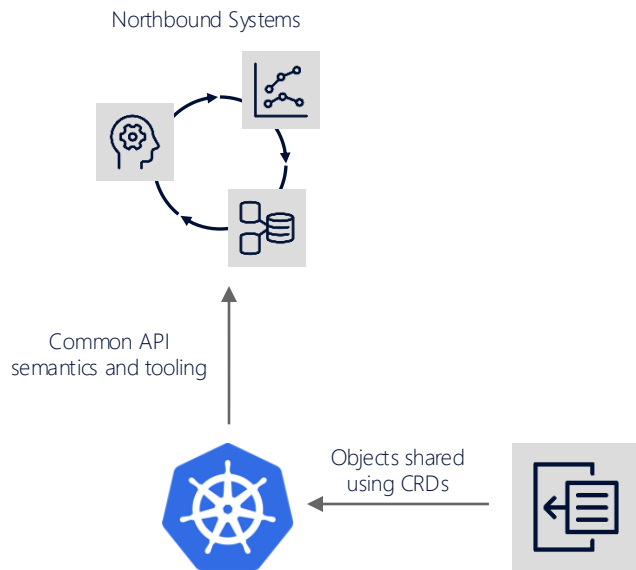
Networking world



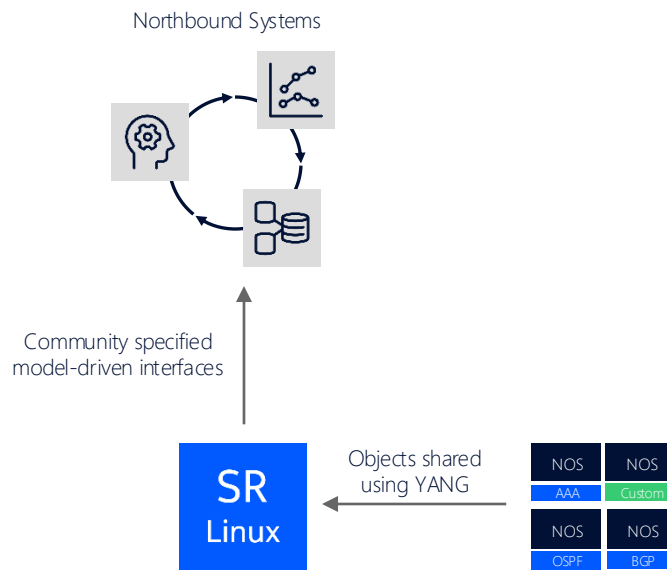
Modularization

Continuous evolution

Application world



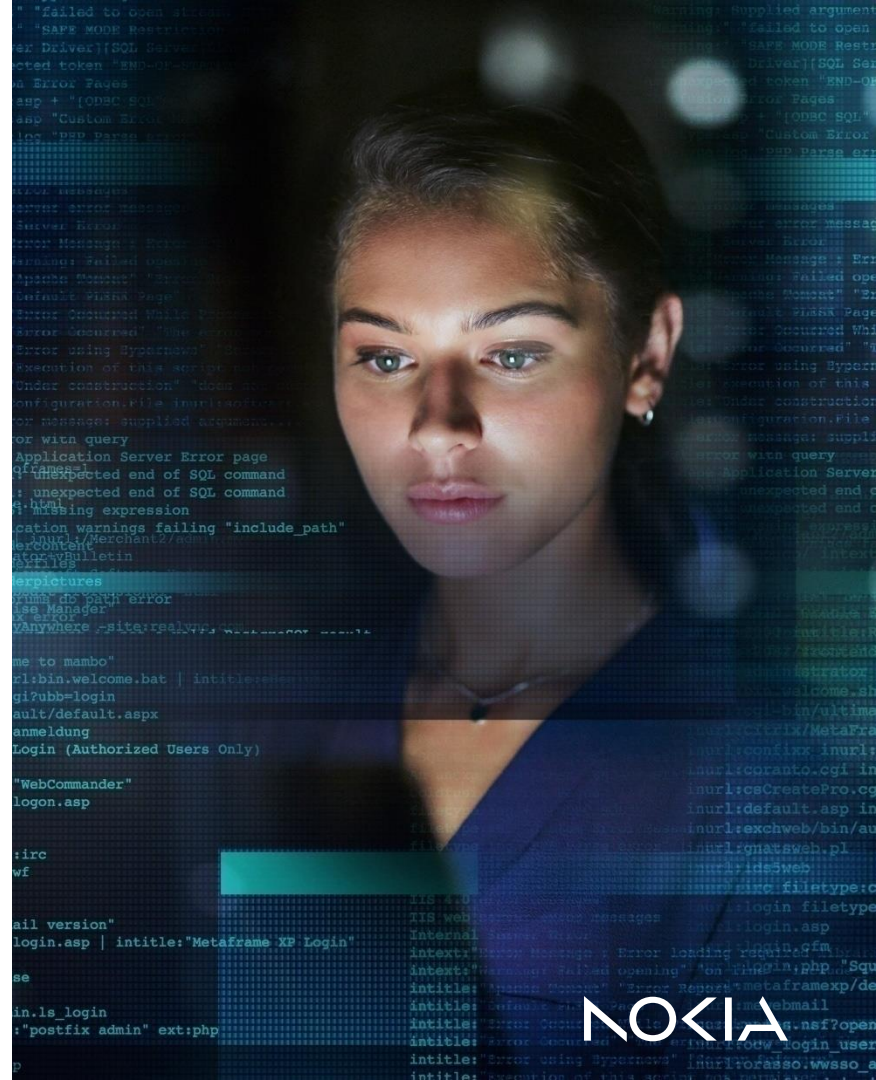
Networking world



Customization with code

Python integrated into Nokia NOS'

- Provide operators with the tools to enhance, personalize and automate
- Deliver CLI tooling to enhance the experience for network engineers and developers alike
- Personalize the CLI using Python programming
- Solve operational integration issues fast
- Create unattended reactive nodal automation
- Schedule nodal automation activities
- Manipulate inflight syslog messages
- Adjust DHCP options
- Remove transport considerations from the developer



Why Python in the hands on event

Python

- You are not required to be a programmer, developer or code guru to take part in this hands on event
- Network Operating Systems (NOS) have become more programmable
 - The operation and customization of routers requires engineers to upskill/cross-skill with some programming knowledge
 - Reflected in network engineer role profiles
- Python used in SR OS and SR Linux for many tasks
 - CLI customization
 - DHCP manipulation
 - Syslog manipulation
 - Unattended event handling
 - Native app creation

This section is not intended to be a programming course.
Just enough to get you going if you choose to undertake a Python based activity this afternoon.

Hello world

Python

- You can use Python interactively (just type **python** and hit return)

```
⌘ python
Python 3.13.3 (main, Apr 8 2025, 13:54:08) [Clang 16.0.0 (clang-1600.0.26.6)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
```

- or by executing a program stored in a file (by typing **python <filename>**)

```
⌘ python hello-world.py
Hello world
```

- We're going to focus on files containing code from now on

Structure, definitions and variables

Python

- Code can be split into smaller sections, called functions. These are defined using the **def** keyword
- Common convention uses a **main** function to start the code from
 - The **main** function does not need to be at the top of the file
 - The special **if** at the bottom simply says, if I was called directly without specifically requesting another function, use the **main** function
- The **print_something** function is defined to accept an input variable called **something**. This doesn't mean the input is coming from the user, but that the input comes from somewhere else in the code

```
def print_something(something):  
    print(something)
```

```
def main():  
    print_something("Hello world")
```

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

```
python structure.py  
Hello world
```

Defaulting and variables

Python

- Arguments/variables to procedures can be defaulted
 - If the argument is not provided the default is used
 - The **something** variable is "Hello SRX" unless otherwise set
- Variables can be used inside functions as well
 - Defined with an equals (=) sign and a value (or specifically the lack of a value i.e. "" or None)
 - Here we set **my_variable** to 123
 - Used by referencing them, for example in the sum **my_variable + 1**

```
def print_something(something="Hello SRX"):  
    print(something)
```

```
def print_something_2(something="Hello SRX"):  
    print(something)
```

```
def main():  
    print_something("Hello world")  
    print_something_2()  
    my_variable = 123  
    print(my_variable + 1)
```

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

```
python structure.py  
Hello world  
Hello SRX  
124
```

Classes

Python

- Classes are a slightly more complicated version of a procedure
- Classes can be used to create objects (instances of the class)
- The **self** keyword just means this instance of this object
- Special functions are wrapped in `__` characters such as `__init__`
 - The **init** special function means “do this when creating”

```
class Presenter:
    def __init__(self, name):
        self.name = name

def main():
    presenters = ["james", "thomas"]

    presenter_objs = []
    for presenter in presenters:
        presenter_objs.append(Presenter(presenter))

    print("Presenter objects:", presenter_objs)

    print(
        "Example contents of one object:",
        dir(presenter_objs[0]),
    )

    for presenter_obj in presenter_objs:
        print(
            "Dictionary contents of object at ID",
            presenter_objs.index(presenter_obj),
            "in the list",
            presenter_obj.__dict__,
        )
        print(
            "The name of the presenter in the instance of the Presenter class:",
            presenter_obj.name,
        )

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

```
python classes.py
Presenter objects: [<__main__.Presenter object at 0x104545e80>, <__main__.Presenter object at 0x1044f3250>]
Example contents of one object: ['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__firstlineno__', '__format__',
 '__ge__', '__getattr__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__static_attributes__', '__str__',
 '__subclasshook__', '__weakref__', 'name']
Dictionary contents of object at ID 0 in the list: {'name': 'james'}
The name of the presenter in the instance of the Presenter class: james
Dictionary contents of object at ID 1 in the list: {'name': 'thomas'}
```

Importing other modules

Python

- A common thing in programming languages, including Python, is to reuse code (yours or developed and published by others)
 - We will use the pySROS (<https://github.com/nokia/pysros>) library here as an example
 - The import statement imports the library (reused code)
- You will notice the same argument variable definitions in the imported code as you saw on a previous slide
 - Python (like lots of programming languages) is "do and repeat" for a lot of things
- Before you can import a library, you need to install it. Some are included with Python and some are from third parties.

```
from pysros.management import connect

def main():
    connection_object = connect(
        host="clab-srexperits-pe1", password="mypassword"
    )

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

File Explorer:

- .github
- docs
- examples
- pysros
 - __init__.py
 - errors.py
 - exceptions.py
 - identifier.py
 - management.py
 - model.py
 - model_builder.py

```
38 from .wrappers import Container, LeafList, Annotation
39
40 __all__ = ("connect", "sros", "Connection", "Datastore", "Empty", "SrosMgmtError", "Invalid")
41 __doc__ = """This module contains basic primitives for managing an SR OS node.
42 It contains functions to obtain and manipulate configuration and state data.
43
44 .. reviewed by PLM 20210624
45 .. reviewed by TechComms 20210713
46 ***
47
48
49 def connect(*, host, port=830, username, password=None, yang_directory=None,
50             rebuild=False, transport="netconf", timeout=300, hostkey_verify=True):
51     """Create a :class:`Connection` object. This function is the main entry point for
52     model-driven management of configuration and state for a specific SR OS node using
53     the pySROS library.
54
55     .. note::
```

Environments

Python

- Python lives on your entire machine. By default, any installation of libraries would affect all users. This is sub-optimal.
- Create virtual environments that allow you to contain your work to one project.
- Two common methods
 - The inbuilt **venv** module

```
python -m venv .venv  
source .venv/bin/activate
```

- The newer (faster) **uv** program

```
uv venv  
Using CPython 3.10.0 interpreter at: /usr/local/bin/python3.10  
Creating virtual environment at: .venv  
Activate with: source .venv/bin/activate  
  
source .venv/bin/activate
```

Using pySROS

Python

- The pySROS library is used for a number of activities in the hands on event and works on any (any vendor) device that implements NETCONF according to the standards
- Using skills we've already acquired, we connect to each router in turn
- Behind the scenes we are:
 - Making a NETCONF connection to each router
 - Obtaining all the YANG models the device supports
 - Compiling the YANG into a Python-native schema
- We're also handling errors here with the **try / except** statements. If we cannot connect to a router we will skip it and carry on.

```
from pysros.management import connect
```

```
def get_connections(routers):
    connections = []
    for router in routers:
        try:
            connections.append(
                connect(
                    host=router,
                    username="admin",
                    password="SReXperts2025",
                    hostkey_verify=False,
                )
            )
            print("Successfully connected to", router)
        except Exception as error:
            print("Failed to connect to", router, ":", error)
            pass
    return connections

def main():
    routers = ["clab-srexperpts-pe1", "clab-srexperpts-pe2"]
    connections = get_connections(routers)
    print(connections)

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

```
python pysros-intro-1.py
```

```
Successfully connected to clab-srexperpts-pe1
```

```
Successfully connected to clab-srexperpts-pe2
```

```
[<pysros.management.Connection object at 0x7f1c09310790>, <pysros.management.Connection object at 0x7f1c076688d0>]
```

Using pySROS

Python

- Next we'll expand our main section
- Here you will notice the **json-instance-path** concept we explored earlier.
 - The pySROS library is YANG-native and uses the YANG paths like everything else in the YANG based NOS' from Nokia
- The steps we're performing here are
 - Print a table heading
 - Obtaining the interface configuration from each router
 - Obtaining the IP MTU data from the state of each router
 - Handling errors where no MTU/port exists
 - Printing the table rows

```
def main():
    routers = ["clab-srexper-1", "clab-srexper-2"]
    connections = get_connections(routers)
    config_path = '/nokia-conf:configure/router[router-name="Base"]/interface'
    state_path = '/nokia-state:state/router[router-name="Base"]/interface[interface-name="{0}]/oper-ip-mtu'
    print(
        "Router".ljust(25),
        "Interface Name".ljust(30),
        "Port".ljust(15),
        "IP MTU".ljust(6),
    )
    for connection in connections:
        data = connection.running.get(config_path)
        for interface in data.keys():
            try:
                port = str(data[interface]["port"])
            except:
                port = "N/A"
            if port:
                mtu = str(connection.running.get(state_path.format(interface)))
            else:
                mtu = "N/A"
            print(
                connection.running.get(
                    "/nokia-conf:configure/system/name"
                ).ljust(25),
                interface.ljust(30),
                port.ljust(15),
                mtu.ljust(6),
            )
```

python pysros-intro-2.py

Successfully connected to clab-srexper-1

Successfully connected to clab-srexper-2

Router	Interface Name	Port	IP MTU
g15-pe1	leaf21	1/1/c5/1	8690
g15-pe1	p1	1/1/c1/1	8690
g15-pe1	p2	1/1/c2/1	8690
g15-pe1	system	N/A	1500
g15-pe2	p1	1/1/c1/1	8690
g15-pe2	p2	1/1/c2/1	8690
g15-pe2	spine11	1/1/c3/1	8690
g15-pe2	spine12	1/1/c4/1	8690
g15-pe2	system	N/A	1500

SR Linux CLI Plugins

Python

- The SR Linux CLI is entirely written in Python and is user customizable
- Creating a CLI plugin uses concepts we have already covered
- Each CLI plugin is an instance of the Plugin object class
- There are a few sections to complete
 - **load**: The main section
 - **_print**: The output process
 - **_get_schema**: Create the schema used in your command
 - **_fetch_state**: Obtain the data to use with your schema from the SR Linux YANG models
 - **_populate_data**: Enter the obtained data into your schema
 - **_set_formatters**: Apply any pretty output touches

```
import subprocess

from srlinux.constants import OS_NAME
from srlinux.data import TagValueFormatter, Border, Data
from srlinux.location import build_path
from srlinux.mgmt.cli import CliPlugin
from srlinux.mgmt.server.server_error import ServerError
from srlinux.schema import FixedSchemaRoot
from srlinux.schema.yang_models import YangModels
from srlinux.syntax import Syntax
```

```
class Plugin(CliPlugin):
```

```
    def load(self, cli, **_kwargs):
        pass
```

```
    def _print(self, state, output, arguments, **_kwargs):
        pass
```

```
    def _get_schema(self):
        pass
```

```
    def _fetch_state(self, state):
        pass
```

```
    def _populate_data(self, state, arguments):
        pass
```

```
    def _set_formatters(self, data):
        pass
```

SR Linux CLI Plugins

Python

- The `load` function kicks off the others
- The `_print` function calls other functions to obtain the state data from the router, populate the schema we defined for this comment, add the beautification tweaks to the output and then print the output
- The `_get_schema` function defines our own schema, this is entirely custom and not the same as the nodes YANG schema
- The `_fetch_state` function does what you would imagine. It obtains data from the routers state database and loads it into a new variable inside the object class

```
def load(self, cli, **_kwargs):
    cli.show_mode.add_command(
        Syntax('whoami', help='Show an example CLI plugin'),
        update_location=False,
        yang_models=YangModels.SrlNokia,
        callback=self._print,
        schema=self._get_schema())

def _print(self, state, output, arguments, **_kwargs):
    self._fetch_state(state)
    result = self._populate_data(state, arguments)
    self._set_formatters(result)
    output.print_data(result)

def _get_schema(self):
    root = FixedSchemaRoot()
    root.add_child(
        'whoami',
        fields=[ 'username' ]
    )
    return root

def _fetch_state(self, state):
    username_path = build_path(['/system/aaa/authentication/session[id="*"]/username'])
    try:
        self._username_data = state.server_data_store.get_data(username_path, recursive=True)
    except ServerError:
        self._username_data = None
```

SR Linux CLI Plugins

Python

- The last two functions complete our CLI plugin
- `_populate_data` obtains a results structure in the form of our schema and fills it with the data we obtained earlier. For each branch in the YANG schema we need to call the `get()` function.

```
system.get().aaa.get().authentication.get().session.get().username or data.username
```

```
/system/aaa/authentication/session[id="*"]/username
```

- `_set_formatters` simply uses some pre-defined CLI sugar to make the output look nice with borders above and below

```
def _populate_data(self, state, arguments):
    result = Data(arguments.schema)
    data = result.whoami.create()
    data.username = '<Unknown>'
    if self._username_data:
        data.username = self._username_data.system.get().aaa.get().authentication.get().session.get().username or data.username
    return result

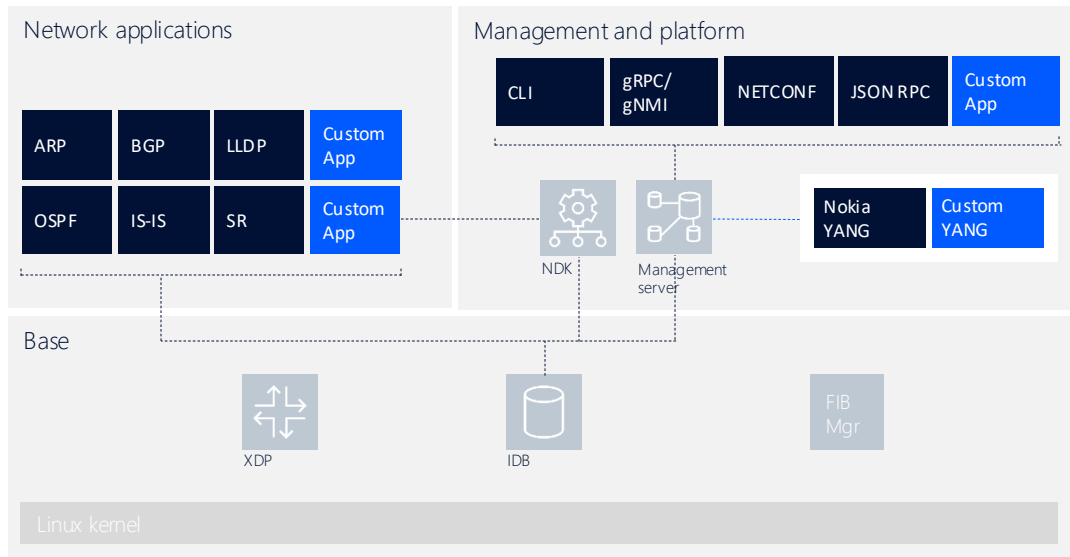
def _set_formatters(self, data):
    data.set_formatter('/whoami', Border(TagValueFormatter(), Border.Above | Border.Below))
```

```
--{+ running }--{ }--
A:g15-leaf11# show whoami
-----
username: admin
-----
```

Applications and agents with the NDK

SR Linux

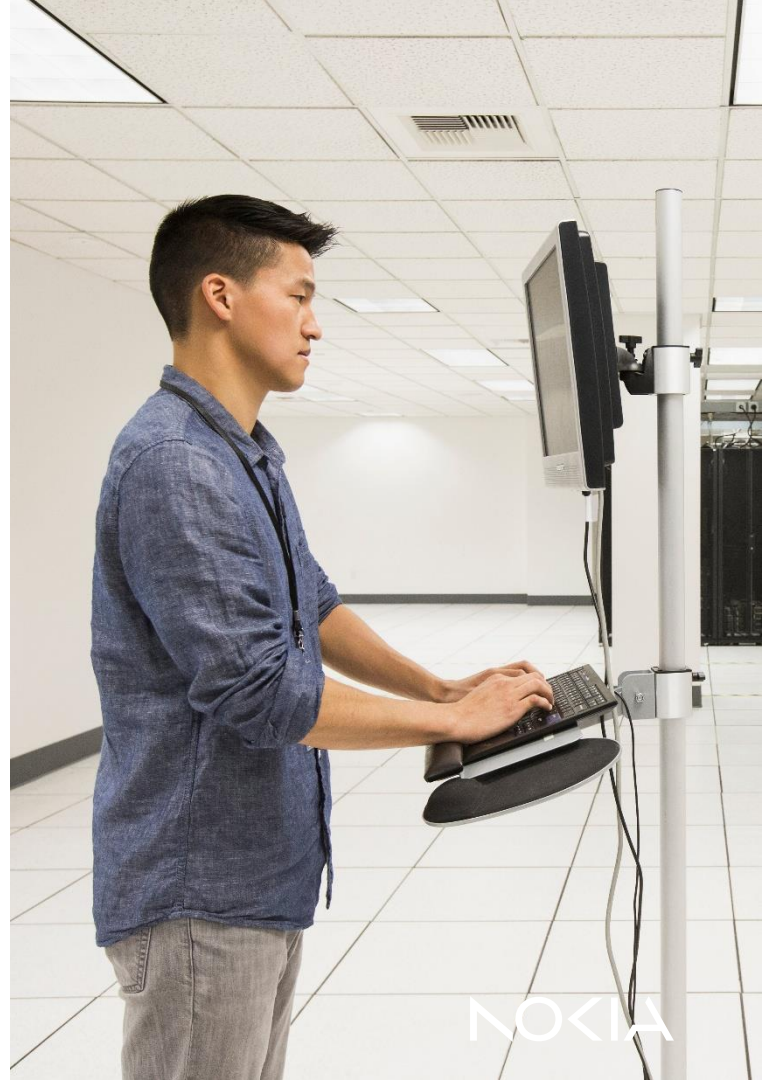
- Going beyond simple customization
- SR Linux CLI provided as an application written in Python and fully customizable
- Nokia consumables (BGP, IS-IS, etc.) provided as applications interfacing with common touch-points and APIs
 - Each application has its own YANG and state
 - Interface between NDK applications and the SR Linux foundations is gRPC
- Fully customizable by operators to develop and ship your own applications that run natively on SR Linux as first-class citizens
- Allows operators to create tailor-made agents to run unattended on the NOS



Codeamation

Network Operating Systems

- Network Operating Systems (NOS) are evolving
- As they become more customizable programming capabilities increase
- Skillsets for network engineers become more diverse
- SR OS allows CLI customization, manipulation of certain in-flight messages (such as DHCP), syslog manipulation and event handling applications
- SR Linux allows for CLI customization, event handling applications and completely customized agents running as native apps on the NOS
- The hands on event has both programming and non-programming activities for you to attempt
- The programming activities in the hands on event this year are all Python based (whether they are on SR OS or SR Linux) [Although the NDK on SR Linux supports multiple programming languages]



gRPC



What is it?

gRPC

- gRPC is an open source framework for implementing a remote procedure call (RPC) application/API
- Originally developed by Google and delivered by the Cloud Native Computing Foundation
- Provides the ability to create and manage both client and server implementations in multiple programming languages
 - C, C++, C#, Dart, Go, Java, Kotlin, Node.js, Objective-C, PHP, Python and Ruby
- Allows the specifying of the RPC service in a simple textual format (called proto)
- Provides a development kit to code-gen the supporting code in the chosen language from the proto

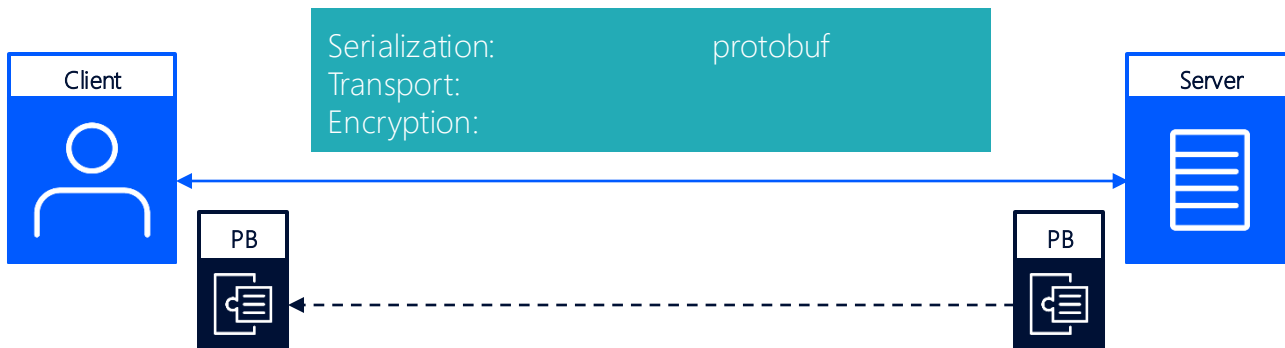


**CLOUD NATIVE
COMPUTING FOUNDATION**

How was it built?

gRPC

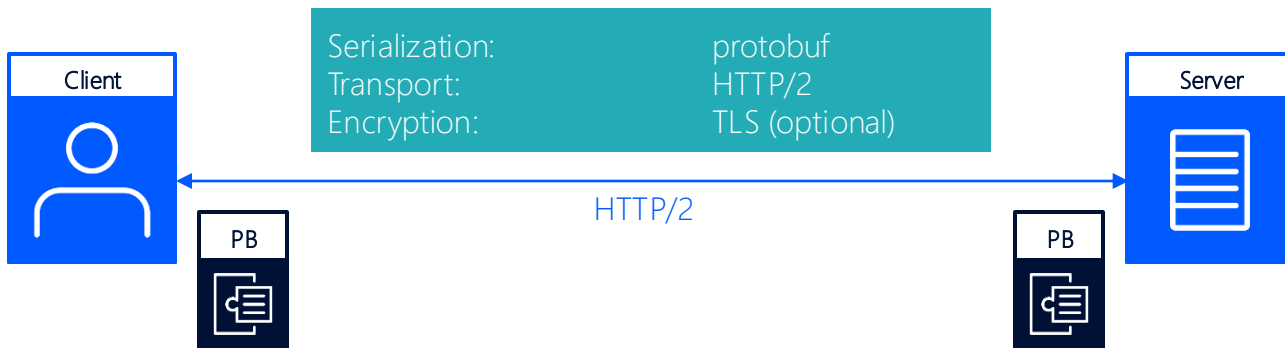
- Message serialization is done with Protocol Buffers (Protobuf or PB)
 - Originally developed at Google in 2001
 - Services are created and published using the protobuf format by servers
 - Clients use the protobuf definition to communicate with the servers
 - Data encoded using the specific NOS' YANG model and configurable encoding



How was it built?

gRPC

- Message serialization is done with Protocol Buffers (Protobuf or PB)
- HTTP/2 is used for transport
- TLS for encryption is strongly recommended in live networks



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities

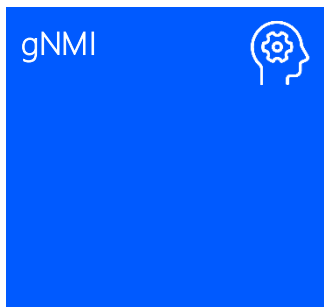


Services

gRPC



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities



The **gRPC Network Management Interface** consists of a single gRPC service with RPCs for configuration management and monitoring. Often used for telemetry as it includes methods for periodically receiving data pushed by the server.

Services

gRPC



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities

gNMI



Applications: Telemetry,
configuration management

Defines the Get, Set, Subscribe,
Capabilities RPCs under the
gNMI service

gRIBI



gNSI



gNOI



The **gRPC Routing Information Base Interface** allows clients to inject routes directly into the RIB of a compatible network device, to override or be used in addition to routing information learned via routing protocols. It is built with a single service implementing a request/response design based on several RPCs.

Services

gRPC



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities

gNMI



Applications: Telemetry,
configuration management

Defines the Get, Set, Subscribe,
Capabilities RPCs under the
gNMI service

gRIBI



Applications: Override or
augment available routing
information

Defines the Get, Flush and
Modify RPCs under the gRIBI
service

gNSI



gNOI



The **gRPC Network Security Interface** defines several gRPC services that are necessary for safely and securely operating an OpenConfig platform.

Services

gRPC



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities

gNMI



Applications: Telemetry,
configuration management

Defines the Get, Set, Subscribe,
Capabilities RPCs under the
gNMI service

gRIBI



Applications: Override or
augment available routing
information

Defines the Get, Flush and
Modify RPCs under the gRIBI
service

gNSI



Applications: authorization,
accounting, certificate and key
management and console
access management

Defines various RPCs and
services including authz, certz
and several others

gNOI



Lastly, the **gRPC Network Operations Interface** defines a set of gRPC services that can be used for executing operational commands on network devices.

Services

gRPC



- We will focus on **four** gRPC-based network management interfaces that are part of this afternoon's activities

gNMI



Applications: Telemetry,
configuration management

Defines the Get, Set, Subscribe,
Capabilities RPCs under the
gNMI service

gRIBI



Applications: Override or
augment available routing
information

Defines the Get, Flush and
Modify RPCs under the gRIBI
service

gNSI



Applications: authorization,
accounting, certificate and key
management and console
access management

Defines various RPCs and
services including authz, certz
and several others

gNOI



Applications: device lifecycle and
operational management

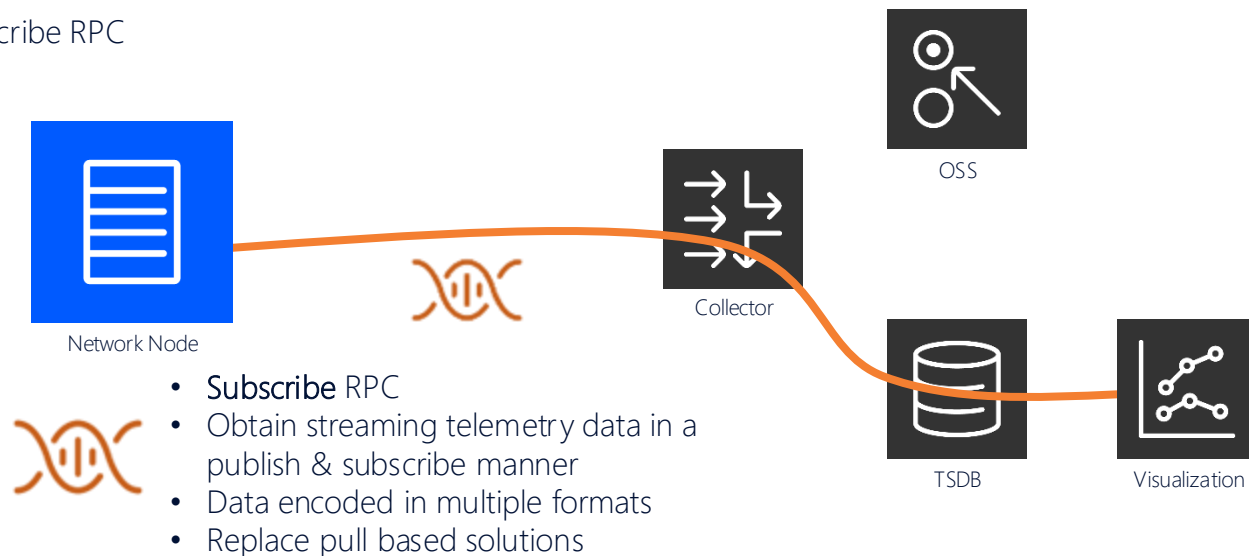
Defines various RPCs and
services, including **File** (Get, Put),
System (Ping and Traceroute)
and others

Applications

gNMI service



- Focus on the configuration management and telemetry landscape
- Provides **Capabilities** RPC to deliver YANG model and gNMI version information
- Provides Get RPC to retrieve configuration and state
- Provides the Set RPC to manage configuration
- Provides the Subscribe RPC



gNMlc

gNMI service



- Focus on the configuration management and telemetry landscape
- Provides **Capabilities** RPC to deliver YANG model and gNMI version information
- For changing configuration and collecting specific datapoints, the **Set** and **Get** RPCs can be used
 - Easy to achieve with **gNMlc**, a tool originally written at Nokia and donated to OpenConfig for interacting with gNMI services

```
~$ gnmic -a clab-srexperfs-p1:57400 -u admin get --path /state/system/oper-name --insecure
[
  {
    "source": "clab-srexperfs-p1:57400",
    "timestamp": 1748688057221608601,
    "time": "2025-05-31T12:40:57.221608601+02:00",
    "updates": [
      {
        "Path": "state/system/oper-name",
        "values": {
          "state/system/oper-name": "p1"
        }
      }
    ]
  }
]
```

gNMI Services

Proto File Definition

- The gRPC server role is fulfilled by SR OS / SR Linux
- A widely used application of gRPC in networking
 - Relies on gnmi.proto

<https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto>

```
service gNMI {  
    // Capabilities allows the client to retrieve the set of  
    // capabilities that is supported by the target. This allows  
    // the target to validate the service version that is  
    // implemented and retrieve the set of models that the target  
    // supports. The models can then be specified in subsequent  
    // RPCs to restrict the set of data that is utilized.  
    // Reference: gNMI Specification Section 3.2  
    rpc Capabilities(CapabilityRequest)  
        returns (CapabilityResponse);  
    // Retrieve a snapshot of data from the target. A Get RPC  
    // requests that the target snapshots a subset of the data  
    // tree as specified by the paths included in the message and  
    // serializes this to be returned to the client using the  
    // specified encoding.  
    // Reference: gNMI Specification Section 3.3  
    rpc Get(GetRequest)  
        returns (GetResponse);  
    // Set allows the client to modify the state of data on the  
    // target. The paths to be modified along with the new values  
    // that the client wishes to set the value to.  
    // Reference: gNMI Specification Section 3.4  
    rpc Set(SetRequest)  
        returns (SetResponse);  
    // Subscribe allows a client to request the target to send it  
    // values of particular paths within the data tree. These  
    // values may be streamed at a particular cadence (STREAM),  
    // sent one off on a long-lived channel (POLL), or sent as a  
    // one-off retrieval (ONCE).  
    // Reference: gNMI Specification Section 3.5  
    rpc Subscribe(stream SubscribeRequest)  
        returns (stream SubscribeResponse);  
}
```

Streaming telemetry

gNMI service

- gRPC messages / structure is sent using protobuf
- Values transmitted (inputs / outputs) via gNMI may use different encodings depending on the client's request and the server's supported formats. Valid formats are:
 - JSON
 - JSON_IETF
 - PROTO
 - BYTES
 - ASCII

```
message SubscribeRequest {
  oneof request {
    SubscriptionList subscribe = 1;
    // Specify the paths within a subscription.
    Poll poll = 3;
    // Trigger a polled update.
  }
  repeated gnm1_ext.Extension extension = 5;
  reserved 4;
  reserved "aliases";
}

message SubscriptionList {
  Path prefix = 1;
  // Prefix used for paths.
  repeated Subscription subscription = 2;
  // Set of subscriptions to create.
  QOSMarking qos = 4;
  // DSCP marking to be used.
  enum Mode {
    STREAM = 0; // Values streamed by the target
    ONCE = 1; // Values sent once-off by the target
    POLL = 2; // Values sent in response to a poll request
  }
  Mode mode = 5;
  bool allow_aggregation = 6;
  repeated ModelData use_models = 7;
  Encoding encoding = 8;
  bool updates_only = 9;
  reserved 3;
  reserved "use_aliases";
}

message Subscription {
  Path path = 1; // The data tree path
  SubscriptionMode mode = 2; // The subscription mode to use
  uint64 sample_interval = 3; // ns between samples in SAMPLE mode
  bool suppress_redundant = 4;
  uint64 heartbeat_interval = 5;
}
```

```
gnmic -a dab-srexperts-p1:57400 \
-u admin \
subscribe \
--path /state/router[router-name="Base"]/interface[interface-name="pe1"]/statistics/ip-in-octets \
--insecure \
--mode once
```

gNMI service

In this packet capture, we see a subscription response using protobuf for the structure and JSON for the returned data value.



```
message SubscribeResponse {
  oneof response {
    Notification update = 1; // Changed or sampled value for a path.
    bool sync_response = 3;
    Error error = 4 [deprecated = true];
  }
  repeated gnmi_ext.Extension extension = 5;
}
```

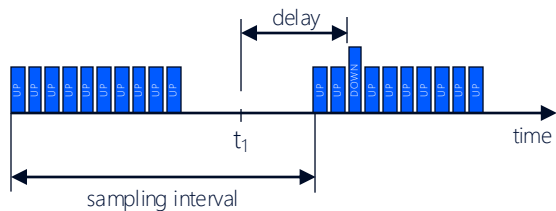
```
message Notification {
    int64 timestamp = 1; // Timestamp in nanoseconds since Epoch.
    Path prefix = 2; // Prefix used for paths in the message.
    repeated Update update = 4; // Data elements that have changed values.
    repeated Path delete = 5; // Data elements that have been deleted.
    bool atomic = 6;
    reserved "alias";
    reserved 3;
}
```

```
gnmic -a clab-srexports-p1:57400 \
-u admin \
subscribe \
--path /state/router[router-name="Base"]/interface[interface-name="pe1"]/statistics/ip/in-octets \
--insecure \
--mode once
```

Finding a Balance

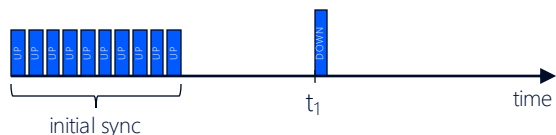
gNMI service

- There is a balance to be found between the following
 - Volume of data
 - Frequency of data
 - CPU performance of the network node
 - Storage in the time series database (TSDB)



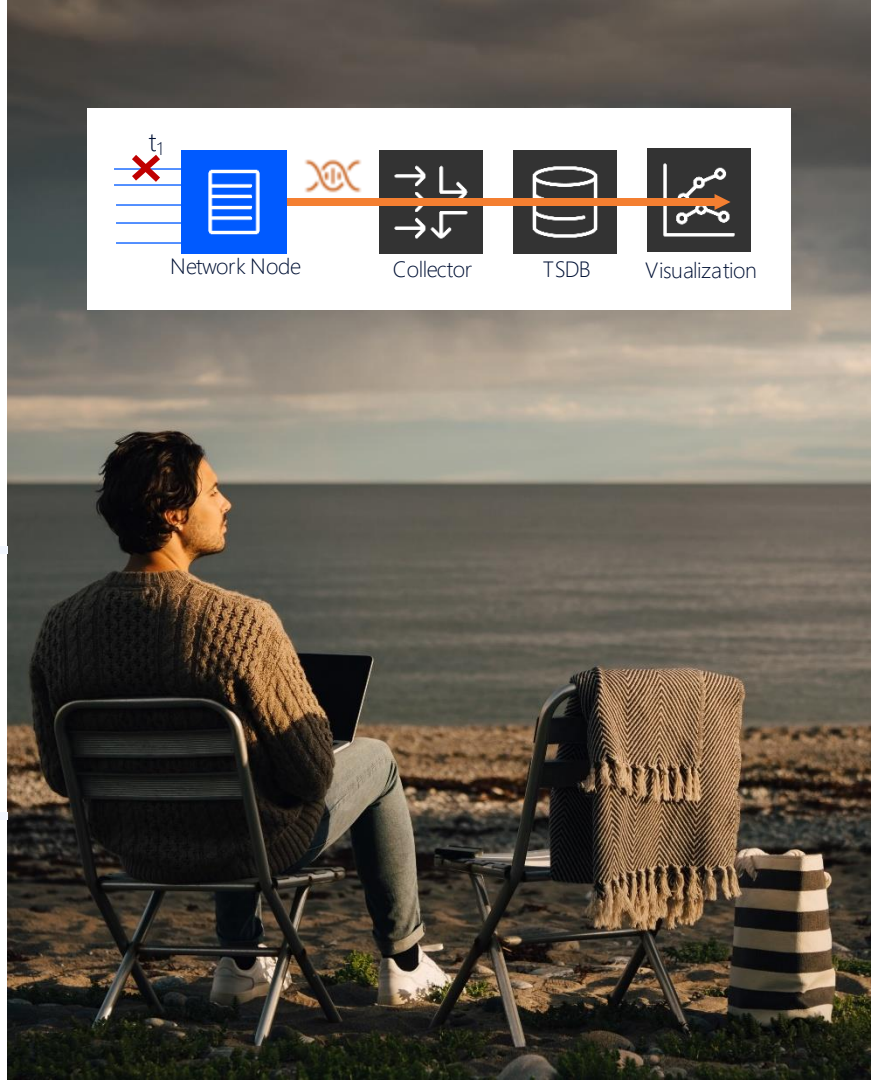
SAMPLE

- Increased volume of data
- Delay in receiving specific events
- ≥ 1 second intervals



ON_CHANGE

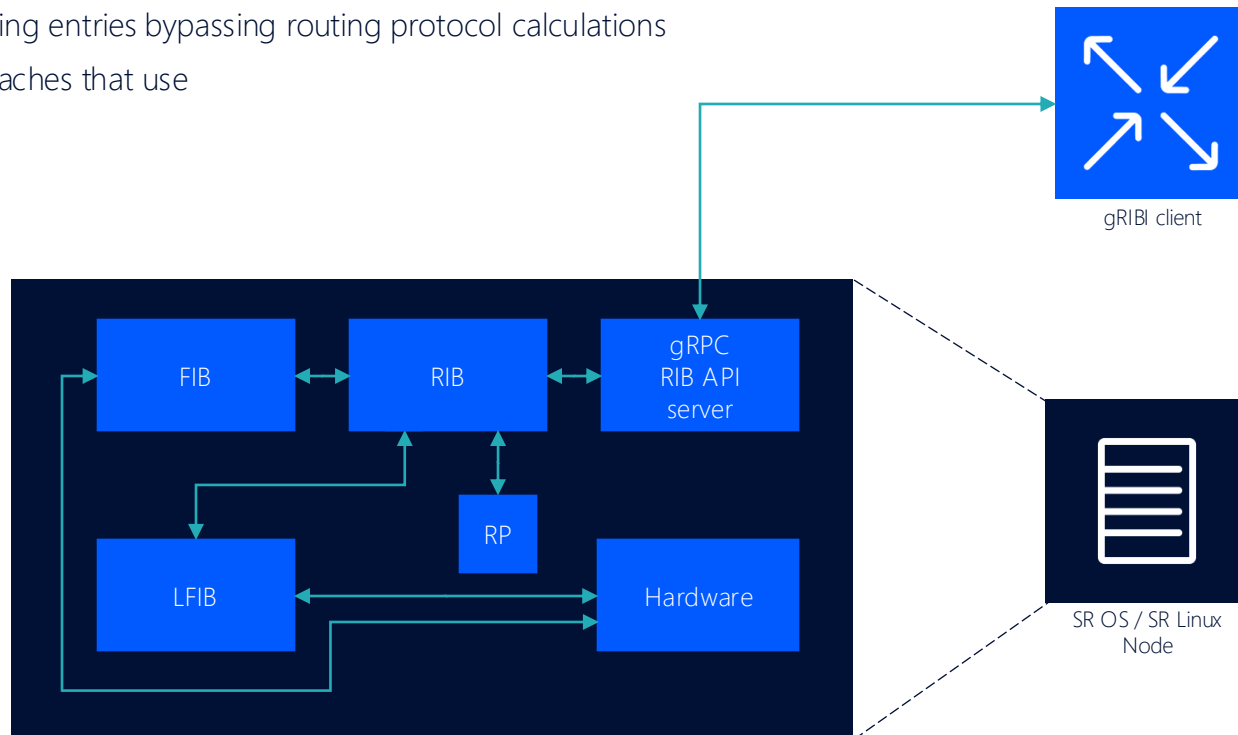
- Yields less data
- Faster notification of specific events



Applications

gRIBI service

- Programmatically create routing entries bypassing routing protocol calculations
- Alternative for existing approaches that use
 - OpenFlow
 - Routing protocol modifications
 - Vendor-specific SDK
 - PCEP

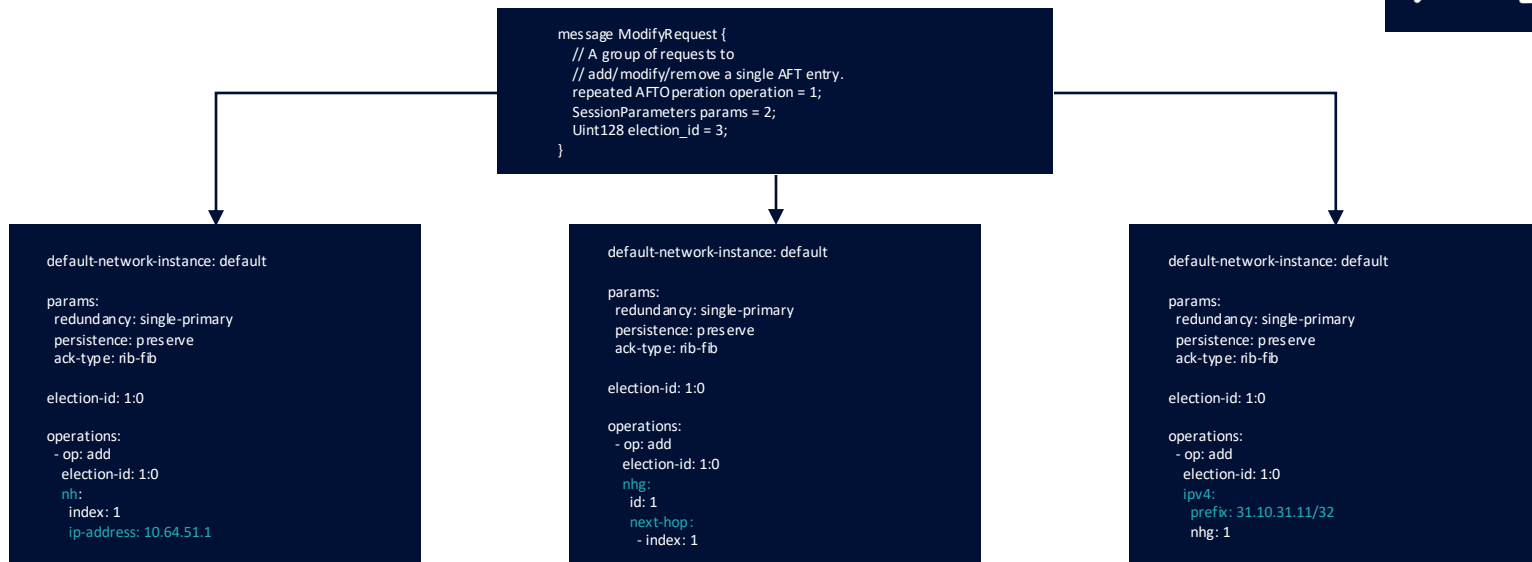
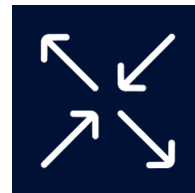


Payload Structure

gRIBI service



- Message format defined in openconfig gRIBI proto
 - Uses OpenConfig's Abstract Forwarding Table (AFT) vendor-agnostic representation
 - Prefix Lookup → points to Next Hop Group → points to next hop(s)

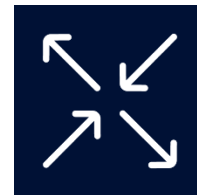


gRIBIc

gRIBI service



- Applications include scrubbing, troubleshooting and installing offline calculated backup paths
- Defined RPCs are
 - Modify
 - Get
 - Flush
- **gRIBIc** is a tool written at Nokia that can be used to test and learn about the gRIBI service



```
~$ gribic -a clab-srexperfs-peering2:57400 -insecure -u admin modify --input-file file.yml
...
status: RIB_PROGRAMMED
status: FIB_PROGRAMMED
```

```
message ModifyRequest {
  // A group of requests to add/modify/remove
  // a single AFT entry.
  repeated AFTOperation operation = 1;
  SessionParameters params = 2;
  UInt128 election_id = 3;
}
```

Applications

gNSI service

- A suite of services used to secure network infrastructure based on gRPC
- gNSI defines **five** gRPC services, **two** of which are available in SR Linux



Authz



gRPC services for per
RPC user access

Certz



gRPC services for the
manipulation of
certificates and TLS
profile configuration

Credentialz



gRPC services for user,
group and SSH server
management

Acctz



gRPC services for
controllers to stream
accounting events

Pathz



gRPC services native
RBAC support

<https://github.com/openconfig/gnsi/>

Certz

gNSI service



- A suite of services focusing on the security of network devices
- **Certz** service has the following RPCs
 - AddProfile
 - DeleteProfile
 - GetProfileList
 - CanGenerateCSR
 - Rotate
- **gNSIc** (currently in beta) is a tool written at Nokia that can act as a CLI client that lets you make calls to gNSI services

Certz

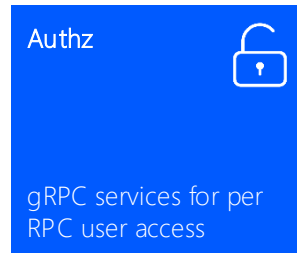


gRPC services for the manipulation of certificates and TLS profile configuration

```
~$ gnsic -a clab-srexperts-leaf11:57401 -u admin -p <password> --skip-verify certz get-profile-list
```

```
+-----+-----+
| Target Name      | SSL Profile ID(s) |
+-----+-----+
| clab-srexperts-leaf11:57401 | __default__      |
+-----+-----+
```

- A suite of services focusing on the security of network devices
- **Authz** service has the following RPCs
 - Rotate
 - Probe
 - Get
- **gNSIc** (currently in beta) is a tool written at Nokia that can act as a CLI client that lets you make calls to gNSI services



```
~$ gnsic -a clab-srexperts-leaf11:57401 -u admin --skip-verify authz rotate --policy "JSON POLICY BODY"
...
INFO[0000] "clab-srexperts-leaf11:57401": got UploadResponse
```

Services and RPCs

gNOI service

- A collection of services focusing on operational interactions with network devices
- Largest gRPC service in terms of services and RPCs supported
- gNOI predates gNSI and included a **cert** service that is now deprecated
- While many services are defined, SR Linux and SR OS both support a subset



Healthz 	System 	File 	OS 	Link-Qualification 	FactoryReset 
gRPC services that can check on system component health	gRPC services for upgrades, time, rebooting, ping and traceroute	gRPC services for manipulating files and filesystems	gRPC services for system upgrades	gRPC services for qualifying links	gRPC services for restoring systems to factory state
Get	Ping	Get	Install	Create	Start
List	Traceroute	TransferToRemote	Activate	Get	
Acknowledge	Time	Put	Verify	Capabilities	
Artifact	SwitchControlProcessor	Stat		Delete	
Check	Reboot	Remove		List	
	RebootStatus				
	CancelReboot				
	KillProcess				

gNOIc

gNOI service



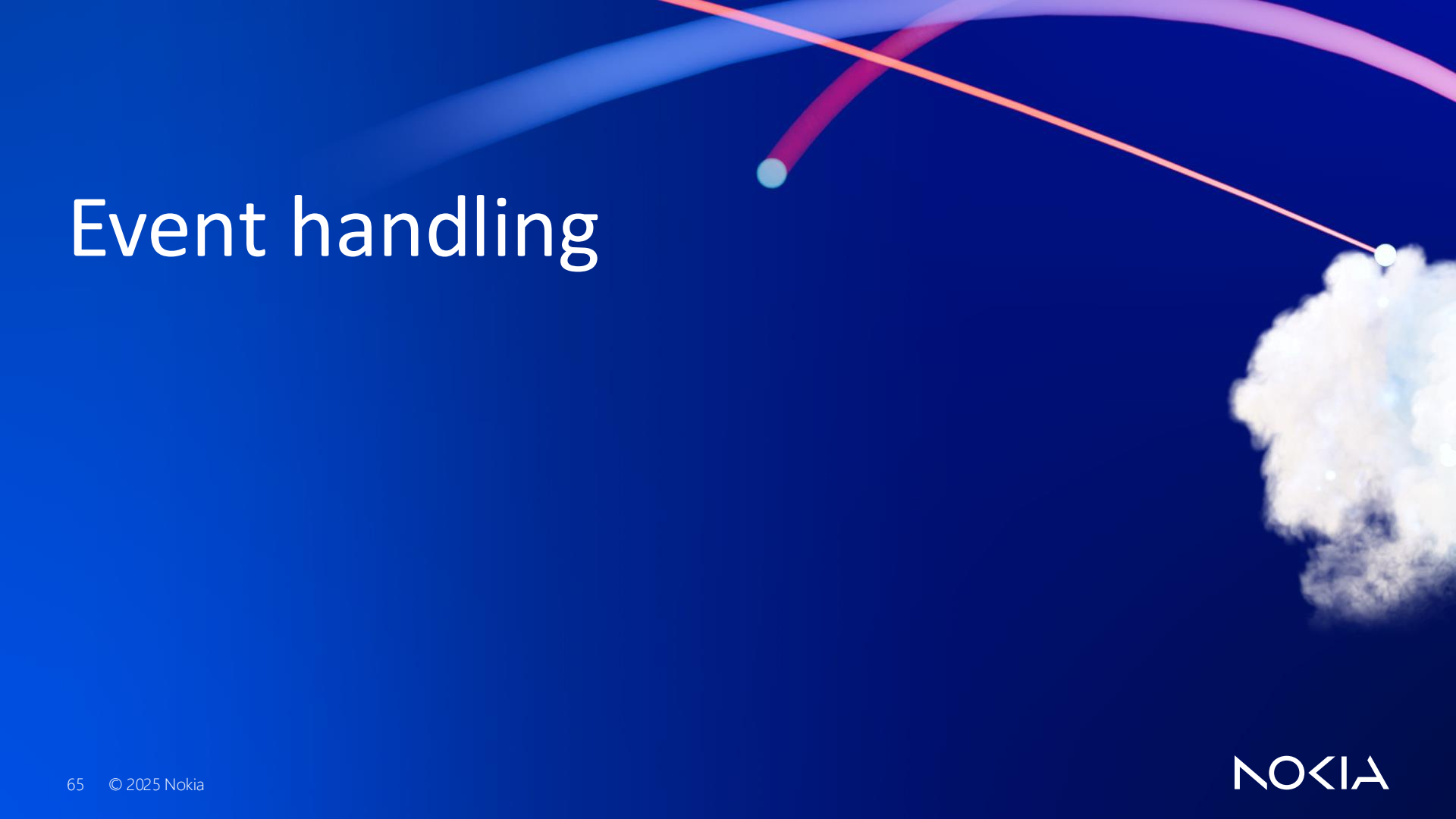
- gNOIc is a tool written at Nokia that can act as a CLI client that lets you make calls to gNOI services



```
~$ gnoic -a clab-srexperfs-p1:57400 -u admin --insecure system time
+-----+-----+-----+-----+
| Target Name | Time | Timestamp | |
+-----+-----+-----+-----+
| clab-srexperfs-p1:57400 | 2025-05-30 18:21:46.054121527 +0200 CEST | 1748622106054121527 |
+-----+-----+-----+-----+
```

Healthz 	System 	File 	OS 	Link-Qualification 	FactoryReset 
gRPC services that can check on system component health	gRPC services for upgrades, time, rebooting, ping and traceroute	gRPC services for manipulating files and filesystems	gRPC services for system upgrades	gRPC services for qualifying links	gRPC services for restoring systems to factory state
Get	Ping	Get	Install	Create	Start
List	Traceroute	TransferToRemote	Activate	Get	
Acknowledge	Time	Put	Verify	Capabilities	
Artifact	SwitchControlProcessor	Stat		Delete	
Check	Reboot	Remove		List	
	RebootStatus				
	CancelReboot				
	KillProcess				

Event handling

An abstract graphic on a dark blue background. It features several curved lines in shades of blue and purple. A thin red line with a small white dot at its end extends from the upper right towards the center. In the lower right corner, there is a white, fluffy cloud-like shape.

An event in your network

- Configuration change
- Malicious actors
- Miscommunication

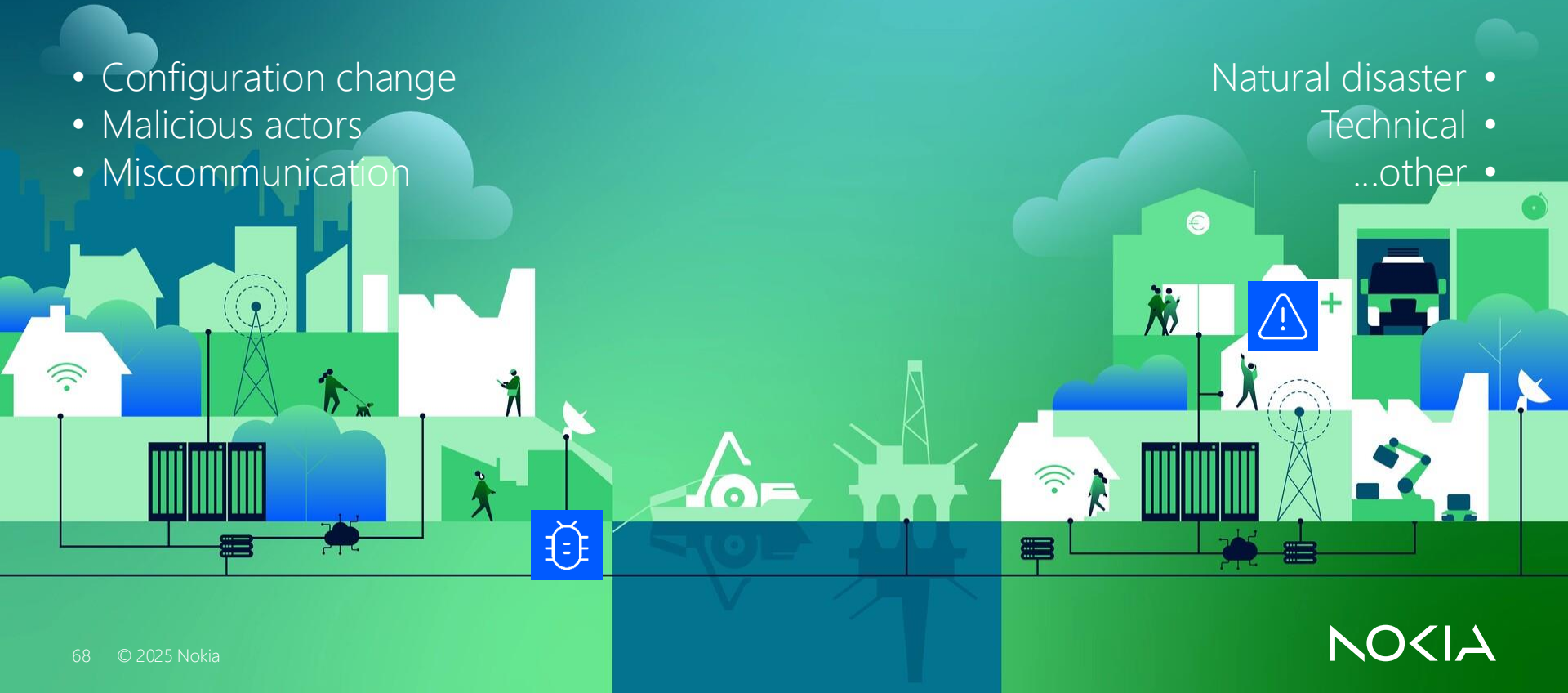
- Natural disaster •
- Technical •
- ...other •



An event in your network

- Configuration change
- Malicious actors
- Miscommunication

- Natural disaster •
- Technical •
- ...other •



An event in your network

- Configuration change
- Malicious actors
- Miscommunication

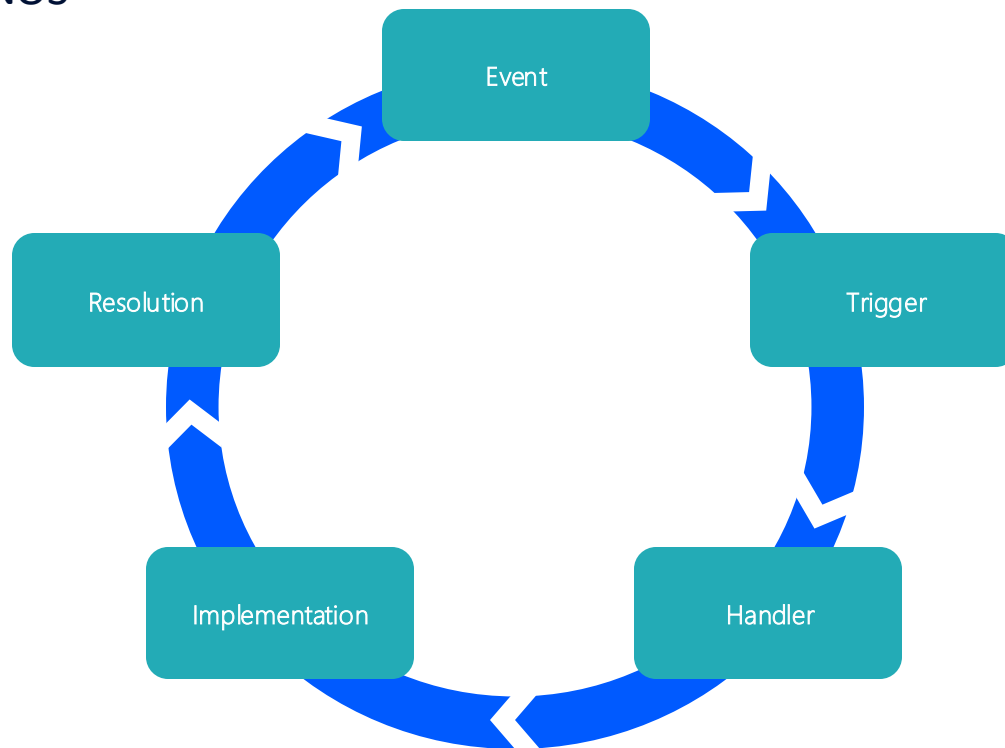
... can have dire consequences!

- Natural disaster •
- Technical •
- ...other •



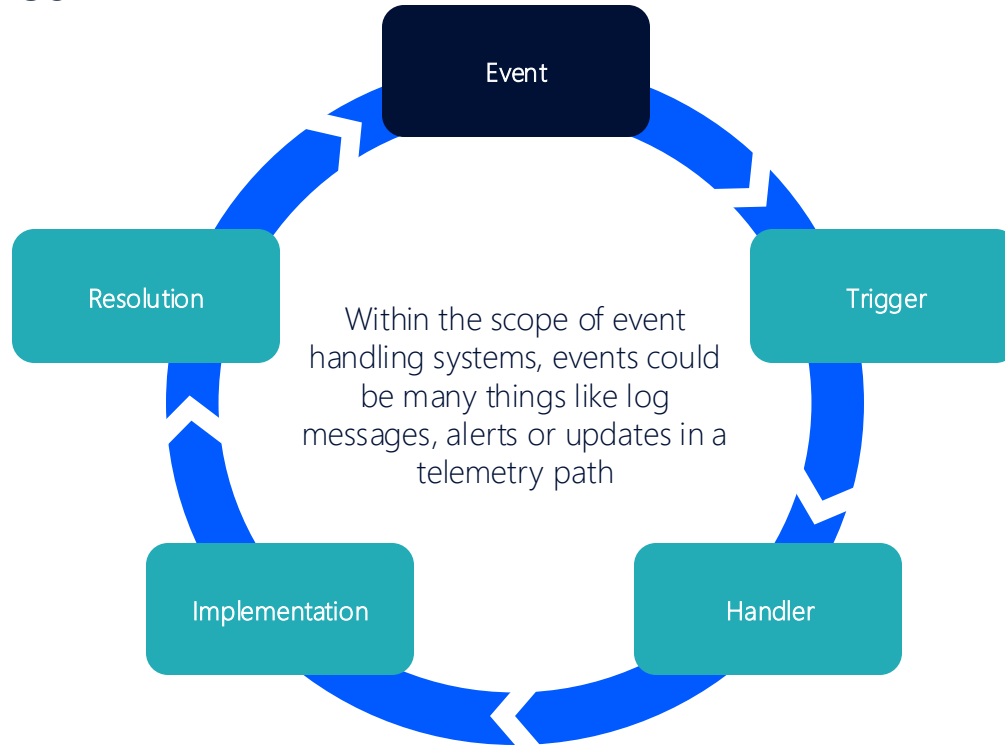
Components

Event Handling in NOS



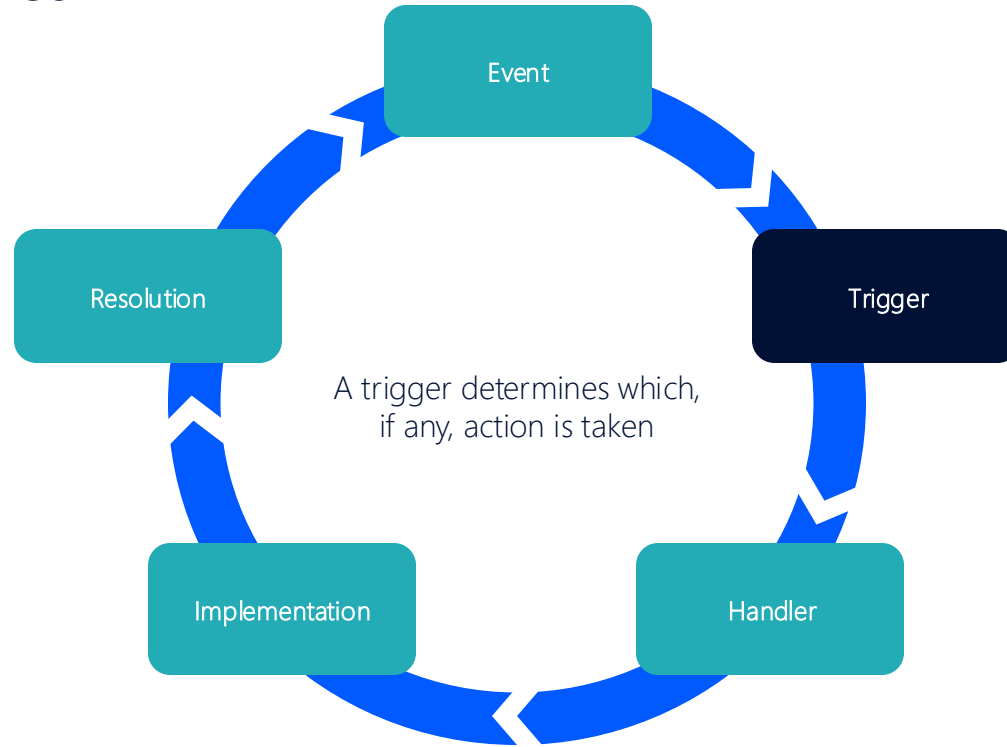
Components

Event Handling in NOS



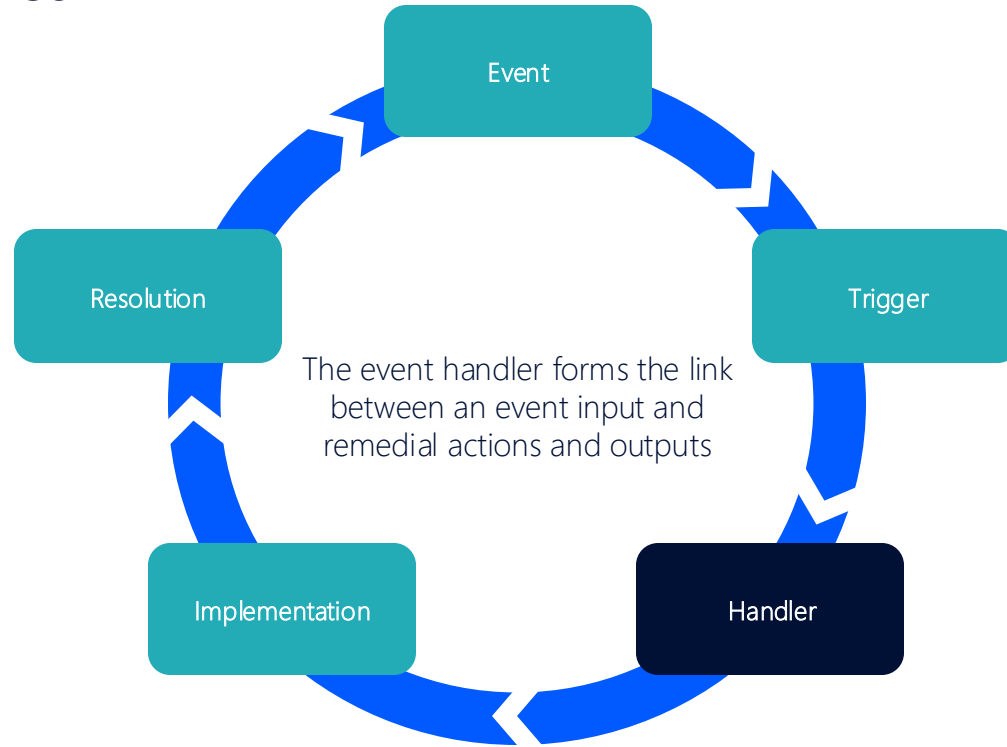
Components

Event Handling in NOS



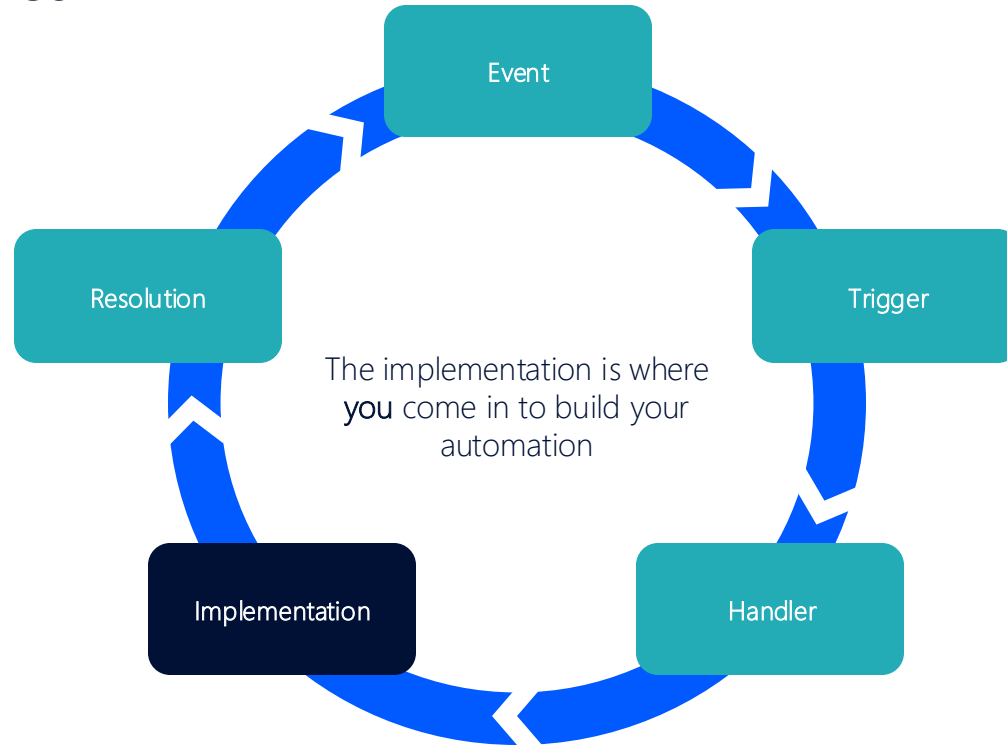
Components

Event Handling in NOS



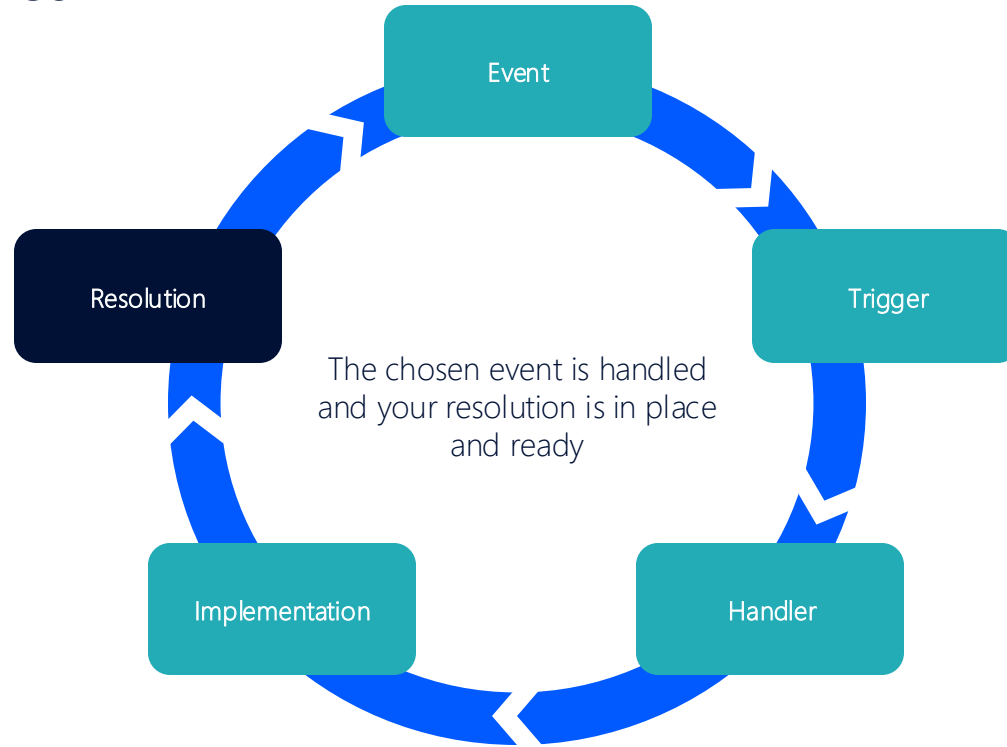
Components

Event Handling in NOS



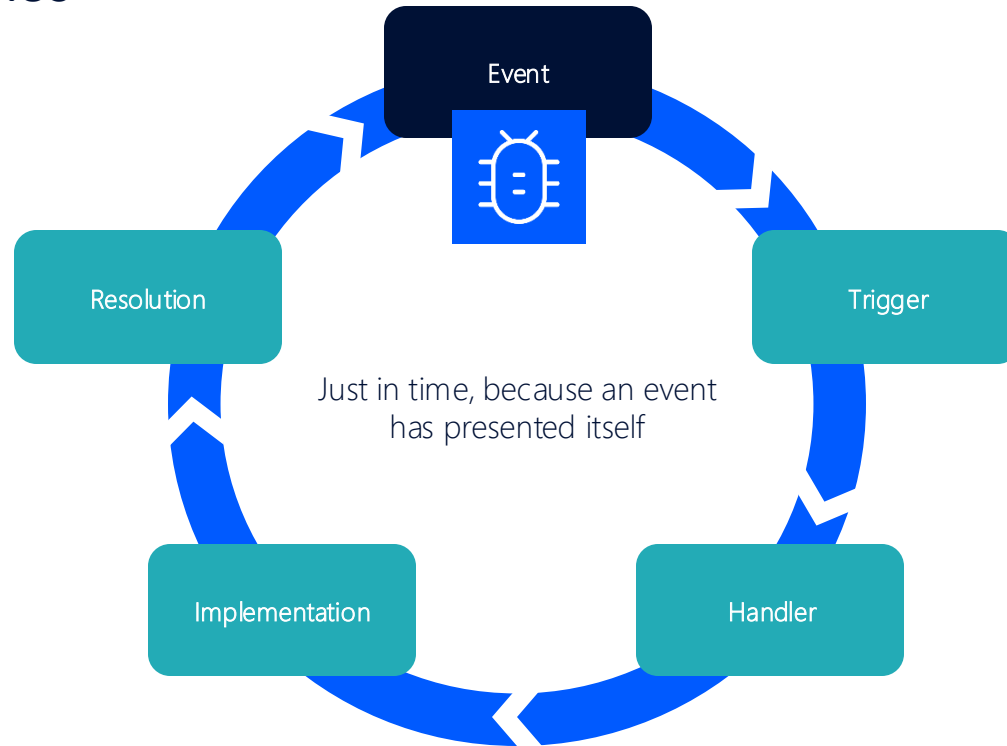
Components

Event Handling in NOS



Components

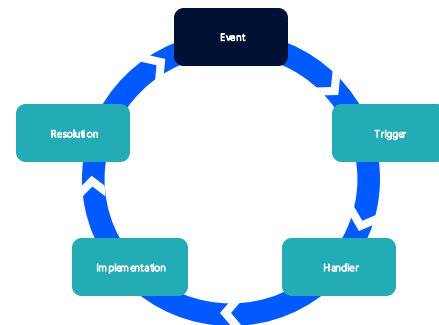
Event Handling in NOS



Details: SR OS

Event Handling in NOS

- Using existing system defined events
- Identify the event you are interested in
 - Based on the documentation
 - Using the Log Events Search Tool
 - Retroactively

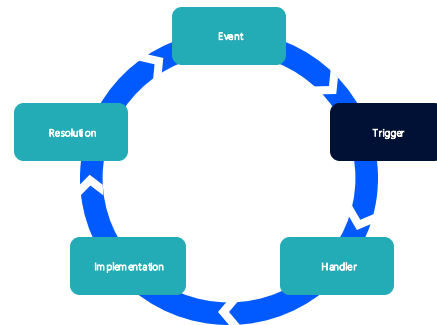


Event name	tmnxLldpRemEntryPeerUpdated
Application name	LLDP
Event ID	2102
Message format string	LLDP Remote peer updated, local port-id \$ifIndex\$, dest-mac-type \$tmnxLldpRemLocalDestMACAddress\$, remote system name \$tmnxLldpRemSysName\$, remote chassis-id \$tmnxLldpRemChassisId\$, remote port-id \$tmnxLldpRemPortId\$, remote-index \$tmnxLldpRemIndex\$
Cause	The tmnxLldpRemEntryPeerUpdated notification is generated when a tmnxLldpRemSysName changes for an existing peer
Effect	N/A
Recovery	N/A

Details: SR OS

Event Handling in NOS

- For your chosen event, create an event-trigger
 - Combination of application and event
 - Points to a **handler** and **filter** to apply
- Multiple entries can be defined

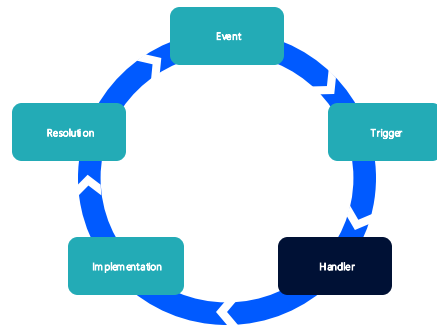


```
configure {  
  log {  
    event-trigger {  
      lldp event tnmxlldpRemEntryPeerUpdated {  
        entry 10 {  
          filter "primer-filtre"  
          handler "primer-handler"  
        }  
      }  
    }  
  }  
}
```

Details: SR OS

Event Handling in NOS

- Create the filter that was referred to
 - Either blacklist or whitelist
 - Can use a combination of event attributes to filter on
- And add the handler to the configuration
 - Assigns the event to your implementation
 - Multiple scripts can be activated by a single handler



```
configure {  
  log {  
    event-handling {  
      handler "primer-handler" {  
        entry 10 {  
          script-policy {  
            name "primer-policy"  
            owner "admin"  
          }  
        }  
      }  
    }  
  }  
  filter "primer-filter" {  
  }  
}}
```

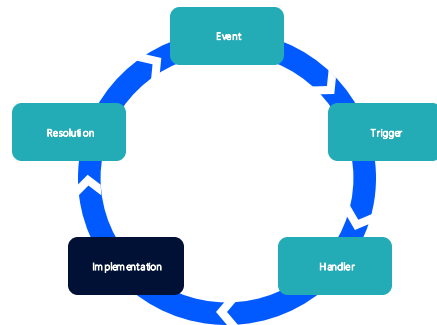
Details: SR OS

Event Handling in NOS

- Create a python-script object referring to your specific solution
 - This object is the in-memory representation of your Python file
- And point to it from a script-policy
 - specifies results directory, limits on runtime and history

```
configure {  
  python {  
    python-script "primer-script" {  
      admin-state enable  
      urls ["cf1:/primer-script.py"]  
      version python3  
    }  
  }  
  system {  
    script-control {  
      script-policy "primer-policy" owner "admin" {  
        admin-state enable  
        results "cf1:/primer-policy-results/"  
        python-script {  
          name "primer-script"  
        }  
      }  
    }  
  }  
}
```

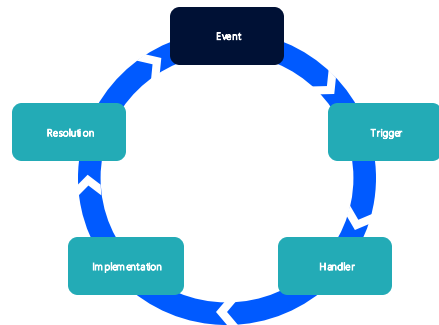
```
def handle_event():  
    event = get_event()  
    print(event.eventparameters["tmnxUdpRemChassisId"]  
  
if __name__ == "__main__":  
    foo()
```



Details: SR Linux

Event Handling in NOS

- Any change in a YANG path is an event
 - One of the benefits of a gRPC-first network operating system
- Identify the YANG-path you are interested in
 - By looking it up in using the CLI
 - Using the SR Linux YANG browser
 - Retroactively



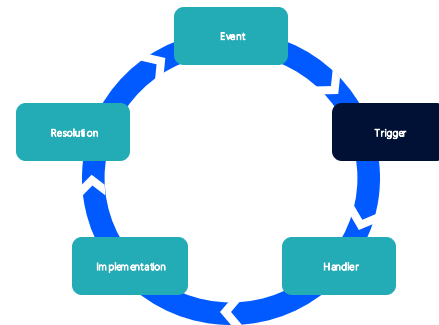
Path Details

Data:	state
Type:	string
Path:	/system/lldp/interface[name=*/neighbor[id=*/system-description The system description of the remote neighbor
Description:	The system description field shall contain an alpha-numeric string that is the textual description of the network entity. The system description should include the full name and version identification of the system's hardware type, software operating system, and networking software. If implementations support IETF RFC 3418, the sysDescr object should be used for this field.

Details: SR Linux

Event Handling in NOS

- Start by creating an event-handler instance
 - Specify which YANG paths should be monitored for changes
- These changes will trigger your automation

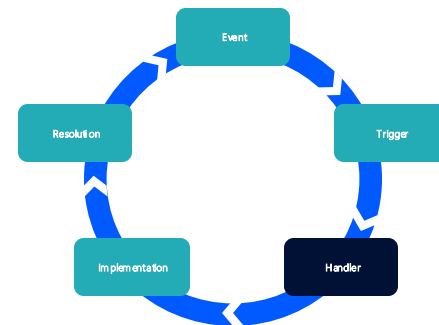


```
system {  
  event-handler {  
    instance primer {  
      admin-state enable  
      paths [  
        "system lldp interface * neighbor * port-description"  
        "system lldp interface * neighbor * system-description"  
      ]  
    }  
  }  
}
```

Details: SR Linux

Event Handling in NOS

- Specify the location of your Python script
 - Nokia includes several example scripts within the SR Linux software image
- Optionally, add additional input parameters as **options**



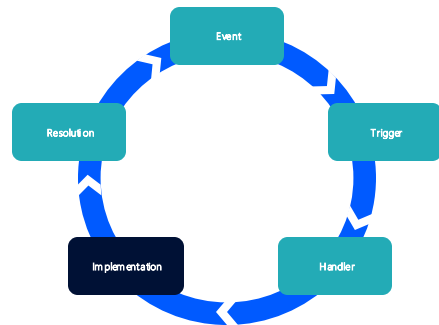
```
system {
  event-handler {
    instance primer {
      admin-state enable
      upython-script primer.py
      paths [
        "system lldp interface * neighbor * port-description"
        "system lldp interface * neighbor * system-description"
      ]
    }
    options {
      object interfaces {
        values [
          ethernet-1/{1..200}
        ]
      }
    }
  }
}
```

Details: SR Linux

Event Handling in NOS

- Write your Python code and make it available to your SR Linux machine
 - Inputs to your script are passed as JSON
 - The system looks for a function **event_handler_main** to call
 - Output actions are returned by this function as JSON
- Persistent data allows you to persist data throughout script executions
- Available output actions

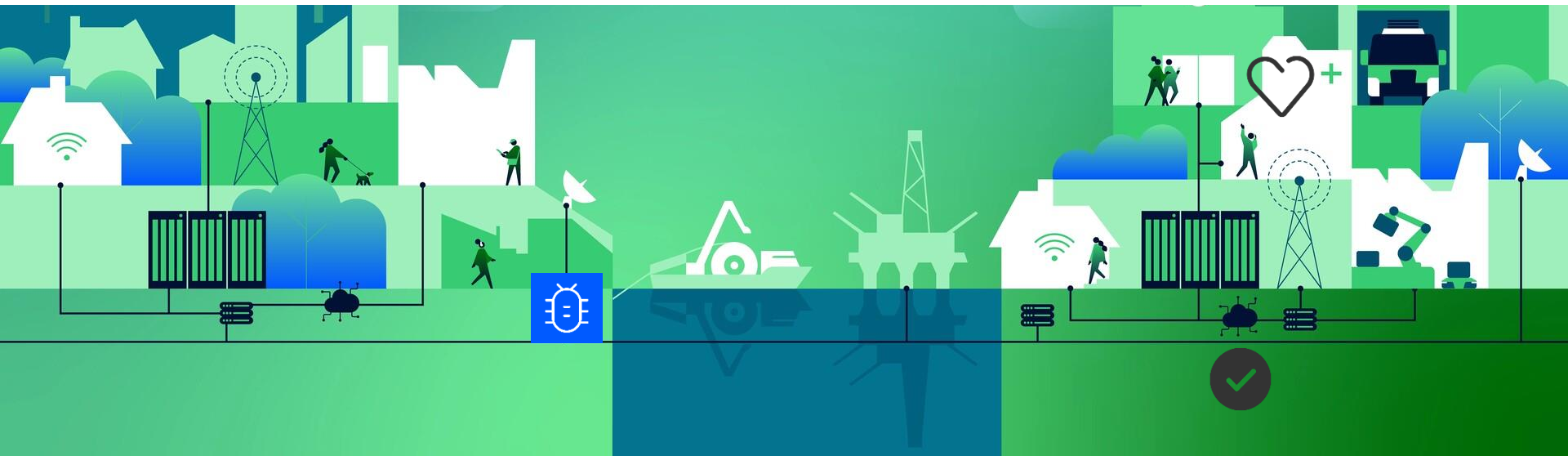
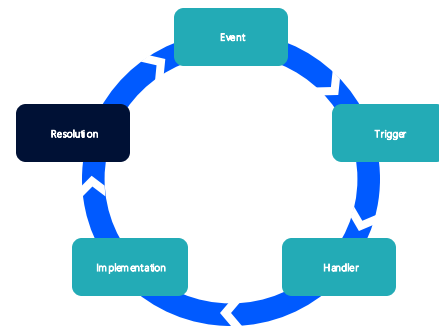
set-ephemeral-path	run-script
set-cfg-path	reinvoke-with-delay
delete-cfg-path	always-execute
set-tools-path	



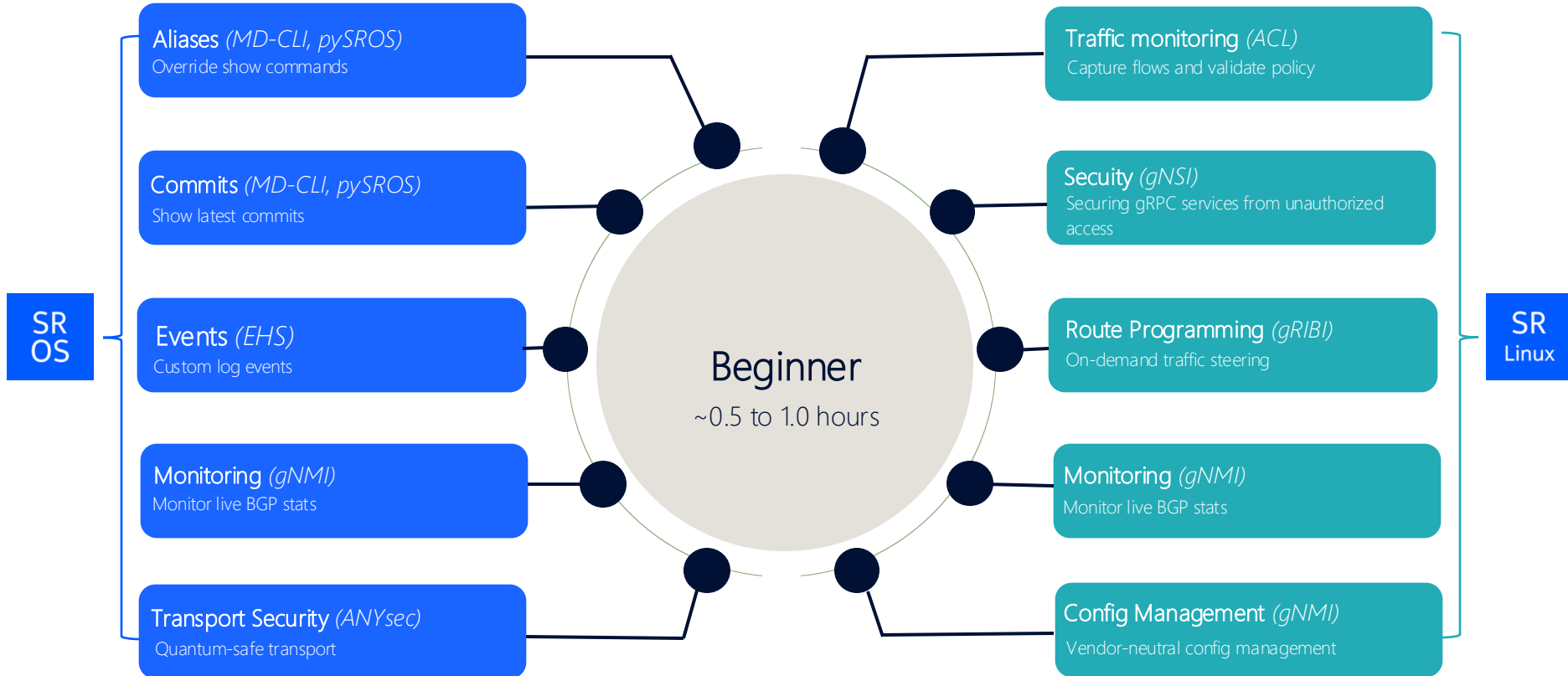
Details: SR OS and SR Linux

Event Handling in NOS

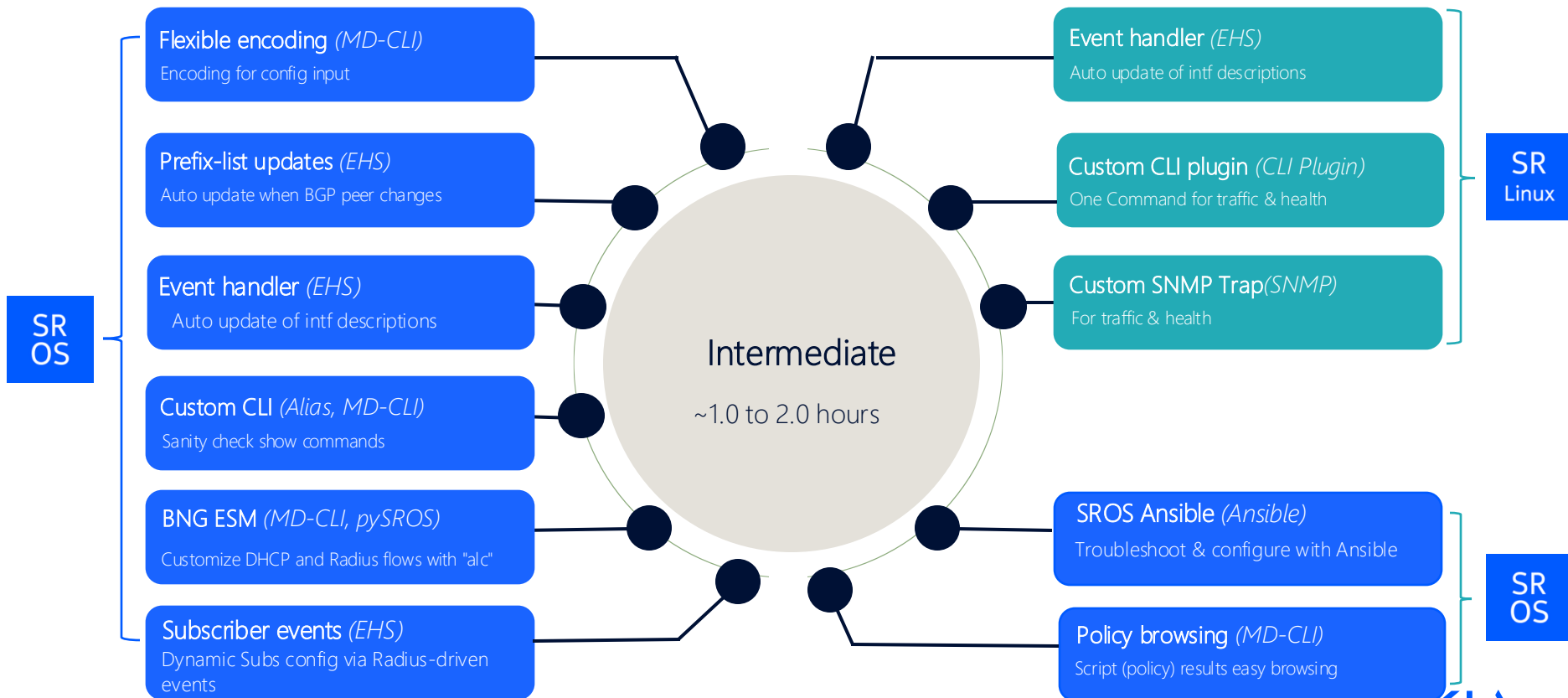
- After some testing and verification, your resolution is now in place and is prepared to handle the next event with confidence.



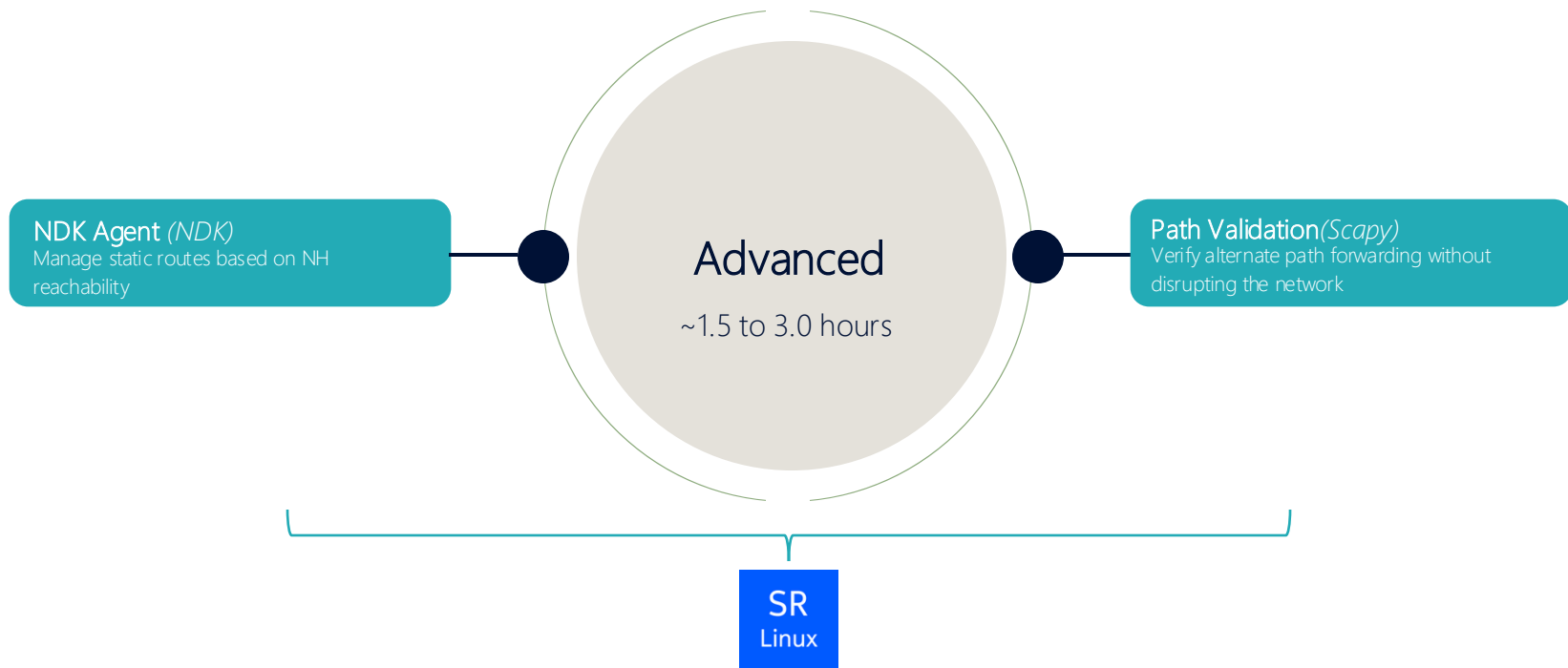
Afternoon activities



Afternoon activities



Afternoon activities



You are ready for the afternoon!

- The SReXplore hands on event is what you've been waiting for
- A full afternoon of hands-on challenges and learning
- All experience levels and all interests

14:00-18:00hrs

- You will need
 - Yourself
 - An open mind and a willingness to throw yourself into things
 - Your laptop (With either an SSH client and/or a web browser)



Support each other

Achieve together

See you at 14:00hrs



Copyright and confidentiality

The contents of this document are proprietary and confidential property of Nokia. This document is provided subject to confidentiality obligations of the applicable agreement(s).

This document is intended for use by Nokia's customers and collaborators only for the purpose for which this document is submitted by Nokia. No part of this document may be reproduced or made available to the public or to any third party in any form or means without the prior written permission of Nokia. This document is to be used by properly trained professional personnel. Any use of the contents in this document is limited strictly to the use(s) specifically created in the applicable agreement(s) under which the document is submitted. The user of this document may voluntarily provide suggestions, comments or other feedback to Nokia in respect of the contents of this document ("Feedback"). Such Feedback may be used in Nokia products and

related specifications or other documentation. Accordingly, if the user of this document gives Nokia Feedback on the contents of this document, Nokia may freely use, disclose, reproduce, license, distribute and otherwise commercialize the feedback in any Nokia product, technology, service, specification or other documentation.

Nokia operates a policy of ongoing development. Nokia reserves the right to make changes and improvements to any of the products and/or services described in this document or withdraw this document at any time without prior notice.

The contents of this document are provided "as is". Except as required by applicable law, no warranties of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, are made in relation to the accuracy, reliability or contents

of this document. NOKIA SHALL NOT BE RESPONSIBLE IN ANY EVENT FOR ERRORS IN THIS DOCUMENT or for any loss of data or income or any special, incidental, consequential, indirect or direct damages howsoever caused, that might arise from the use of this document or any contents of this document.

This document and the product(s) it describes are protected by copyright according to the applicable laws.

Nokia is a registered trademark of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.