



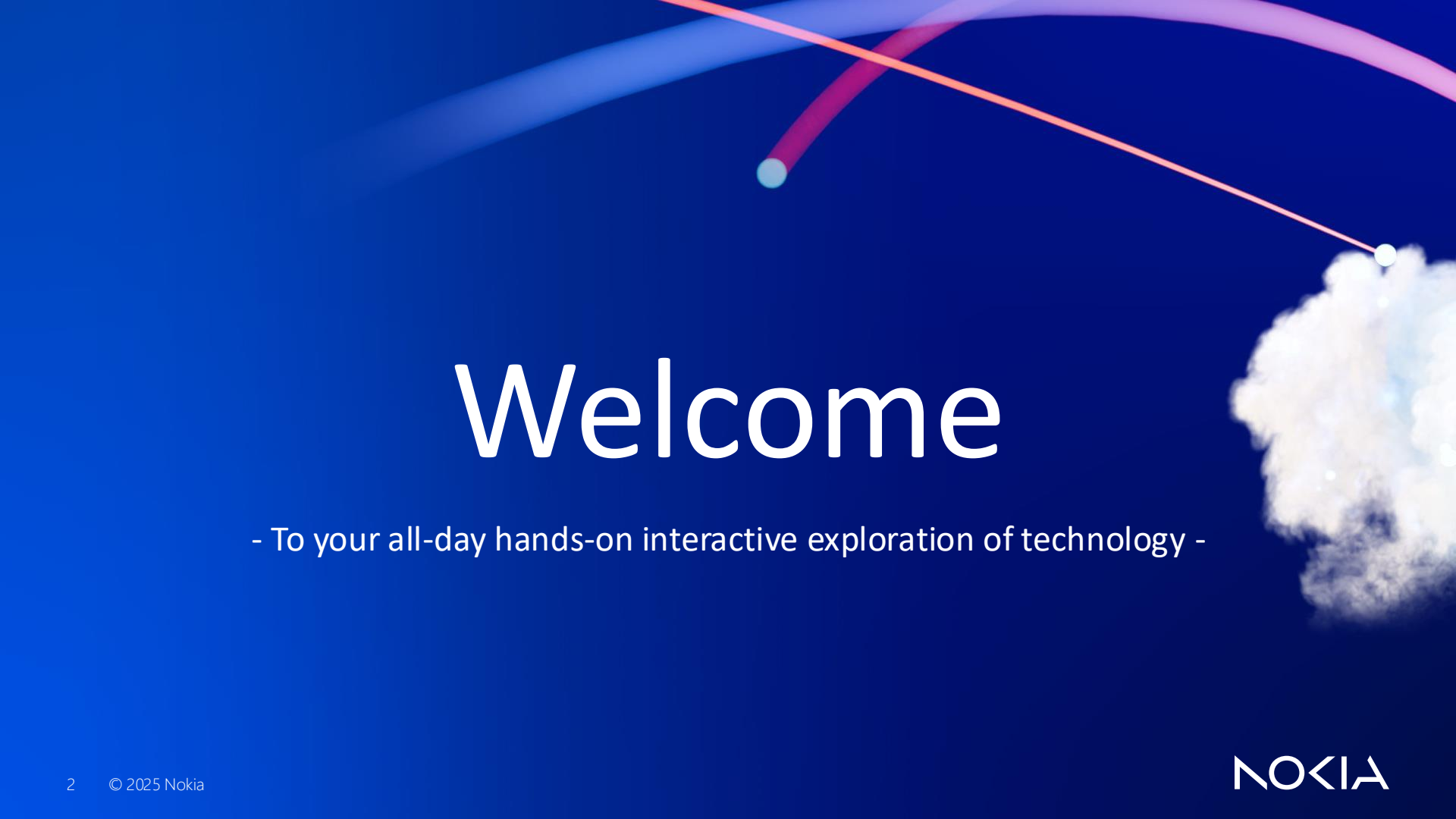
SReXperts²⁰²⁵
Network the cloud

SReXplore 2025 Primer

Network Services Platform (NSP)

Sven Wisotzky

Tim Raphael



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 main hackathon event
 - **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-NSP>

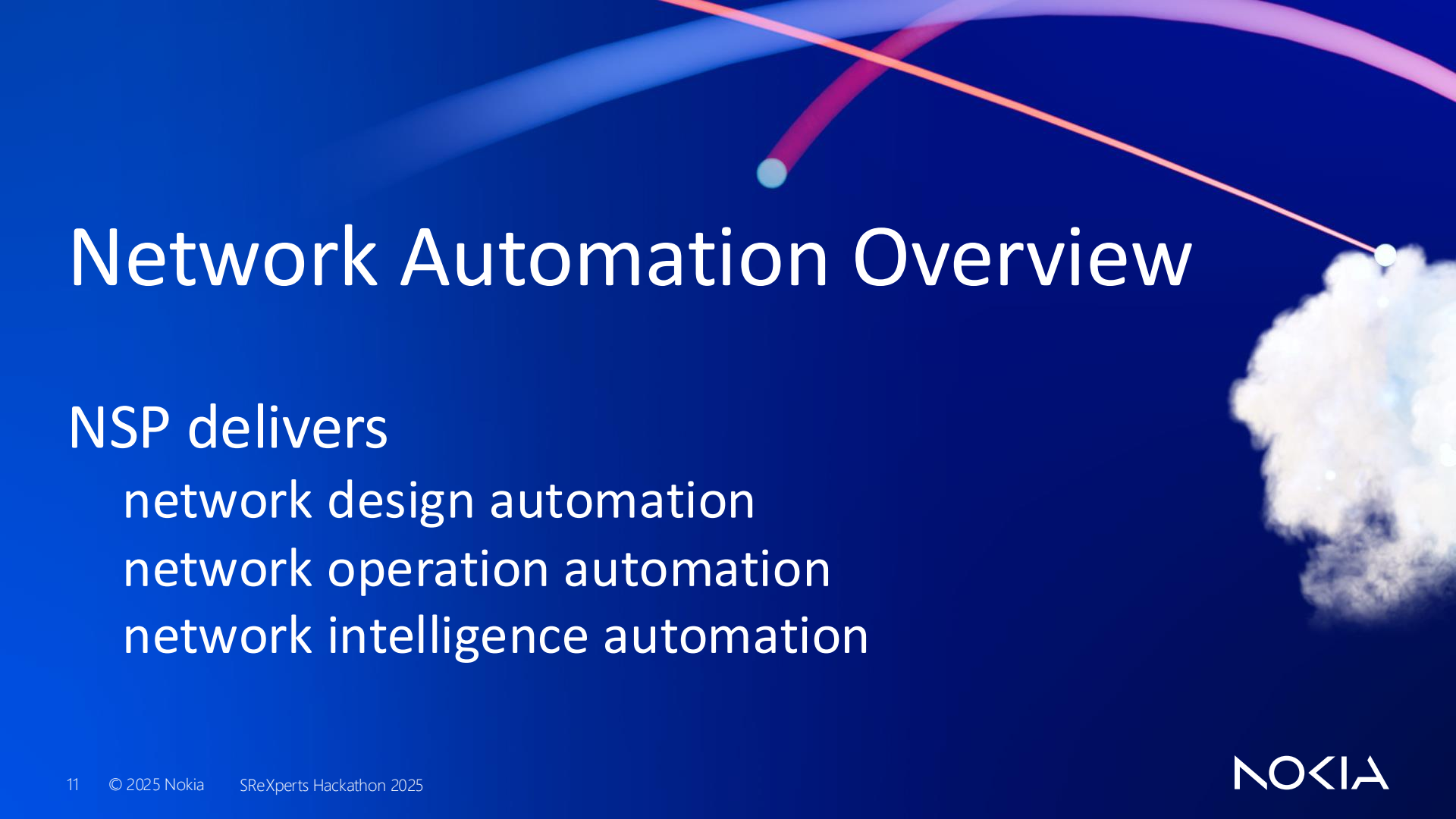
Network Services Platform Stream



Hackathon Primer Agenda

1. Network Automation Overview
2. APIs and models
3. Workflows and Operations
4. Intent-based Networking
5. Getting it out of the doors
6. Ready for the afternoon





Network Automation Overview

NSP delivers

- network design automation

- network operation automation

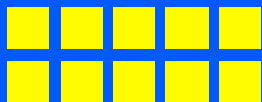
- network intelligence automation

NSP Programmable Platform

REST, RESTCONF/YANG, Kafka

SDN control

Domain Orchestration



nspOS
System services



NSP functions



Programmable
Automation



Inventory
physical, logical, ...

Pluggable mediation

SDN control

CLI – NETCONF – gNMI – SNMP

BGP-LS – BGP-FS – PCEP – OF – RIB API – ...

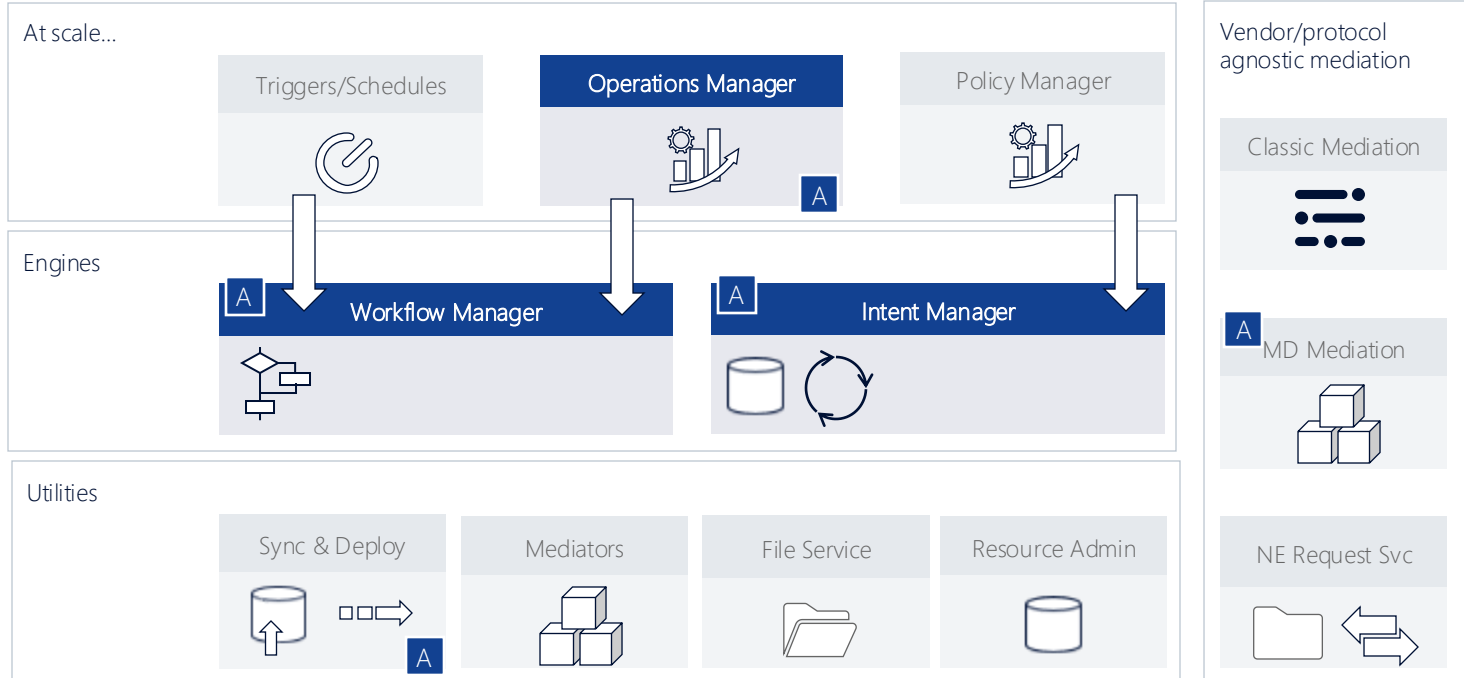
SR
OS

SR
Linux



NSP's anatomy

Programmable Automation



NSP Network Observability

Telemetry Subscriptions

Indicator Analytics

Threshold-Based KPI Monitoring

Combine telemetry data using arithmetic and aggregation logic

Define custom KPIs for operational visibility

Set thresholds to trigger:
events notifications, alarms, or email alerts

Creates dedicated indicator subscriptions
(separate from telemetry subscriptions)

Enables proactive, rule-based detection of known conditions

Baseline Analytics

Adaptive, Context-Aware, Learning-Based Anomaly Detection

Learns what “normal” looks like for each resource

Detects and reports unusual patterns that fixed thresholds would miss

Adjusts automatically to long-term gradual trends
(as traffic or usage changes over time)

Understands time pattern
(day vs. night, weekdays vs. weekends)



NSP APIs and Models

NSP APIs

NSP APIs are a set of protocols and models to...

- Enable external systems or scripts to communicate with NSP

- Develop custom solution with NSP (Programmable Automation Enablement)

NSP north-bound protocols

- REST, RESTCONF, Kafka

NSP modeling

- OpenAPI / JSON-schema, AsyncAPI, YANG

NSP WebUI

- YANG centric: Schema-form, View-Config

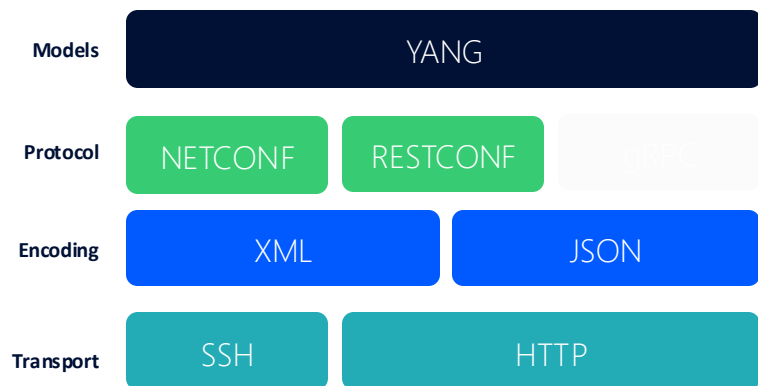
RESTCONF

IETF RFC 8040, IETF RFC 7950

RESTCONF decouples protocol/transport level capabilities from the actual data models (in YANG).

Provides separation between configuration, state and operations.

Enables linkage between objects (leafref, identityref, containment).



RESTCONF	NETCONF
GET	<get>, <get-config>
POST	<edit-config> (operation="create")
PUT	<edit-config> (operation="create/replace")
PATCH	<edit-config> (operation="merge")
DELETE	<edit-config> (operation="delete")

NSP RESTCONF | Advanced Capabilities

Find Operation (beyond RESTCONF GET)

POST `https://{nsp-server}/restconf/operations/nsp-inventory:find`

body: *examples below*

input:

`xpath-filter: /nsp-equipment:network/network-element [product="7750 SR"]` Object Selector

`fields: ne-id;ne-name;ip-address;version;location;type` Output Selector

`offset: 0`

`limit: 100`

Pagination

`include-meta: false`

Output Control

sort-by:

- name

Sorting

`xpath-filter: /nsp-equipment:network/network-element [ne-id="10.33.0.1"] /hardware-component/port [oper-state="enabled"]`

Object Selector

NSP API

Apache Kafka

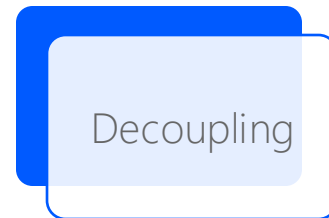
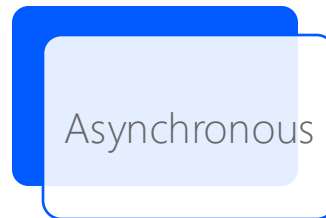
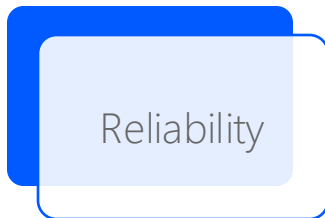
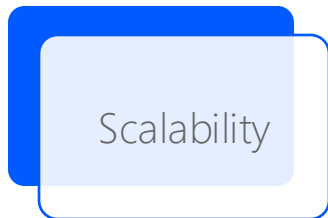
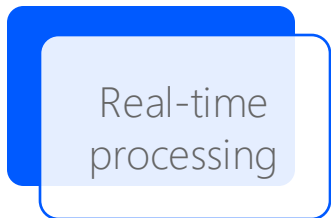
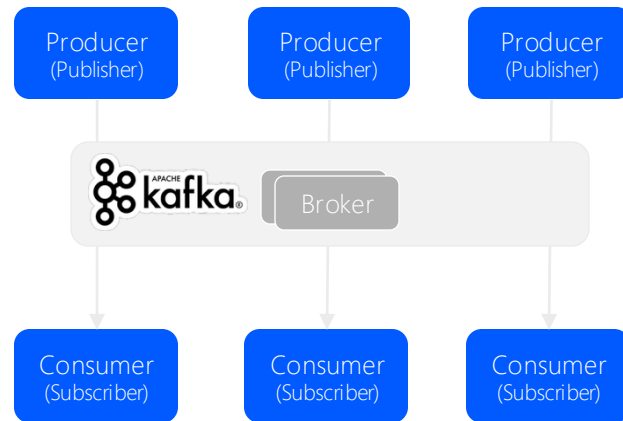
Kafka usage: telemetry streaming, notifications

Topics: communication channels between producers and consumers

Kafka consumers subscribe to topics
(no need-to-know individual producers)

Multiple Kafka producers can send messages to the same topic
(enabling horizontal scale)

Kafka broker is responsible for replication and persistence



NSP API Documentation & More

Developer Portal and Lab-as-a-Service

Welcome to the Network Developer Portal

Transform operational practices with automated workflows and develop connected applications for on-premise networks using APIs and SDK for IP and optical products.

Explore our resources

NSP API Documentation and Sample Postman Collections

NSP	NSP	NSP	NSP	NSP
25.8	25.4	24.11	24.8	24.4

The tables below provides links to **NSP RESTCONF API** and **NSP REST API** resources which includes swagger documentation, Yang HTML browser and sample code collections for all of the listed NSP APIs. See [Getting Started with Postman Samples](#) for details on using POSTMAN collections.

Workflows: workflow development

Overview

A workflow consists of an orchestrated and repeatable pattern of business activity processes that transform materials, provide services, or process information. It is person or group, the work of an organization of staff, or one or more simple or complex tasks.

Workflows examples

Sample workflows are available for download from the Nokia NSP workflow repository. These workflows can be used as a starting point for your own workflow development, installation & execution instructions, NSP requirements and the product version.

There are branches for each NSP release of Workflows.

Mistral terminology

Network Services Platform (NSP) Labs

Nokia NSP Lab Catalog offers a variety of dedicated lab environments tailored to different network management use cases and suitable to meet different business objects: learn new NSP features, develop OSS solutions using NSP APIs, prepare customer demo, etc.

Get the most out of your network resources with our NSP Learning and Certification Program. Develop your knowledge and hands-on skills with eLearning and instructor-led training. [Learn more](#).

NSP Model-driven IP Management

Operations / Programmable / Model-driven SIs

This NSP lab allows the network management capabilities and the automation of network operations such as service assurance, service activation and configuration with intent-based network and service fulfillment, baseline analytics, path control and optimization, workflow automation.

Model-driven NEs network consists of both SR OS NEs and SRL Linux NEs. All NEs are deployed using Container Lab.

Versions: NSP 24.4, NSP 24.8, NSP 24.11, NSP 25.4

[View Lab Diagram](#) [Request Lab Quote](#)

NSP Classic and Model-driven IP Management

Operations / Programmable / Classic SIs / Model-driven SIs

This NSP lab allows the network management capabilities and the automation of network operations such as network infrastructure management.

Workflows

Workflow Manager

Coding Principles

Imperative := Run tasks in a sequential manner

- More expressive, flexible and natural way of solving problems in software

- Execution logic is explicitly controlled: retries, rollback, error-handling, ...

- Attractive to traditional software and script developers

More than just PUSH operations

- Access to a variety of data-sources

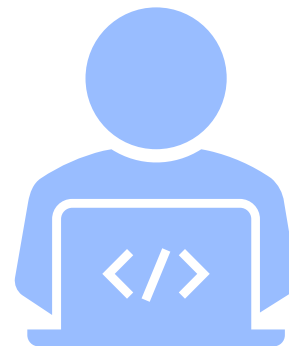
- Pause workflows to wait for user-input

Solution flexibility (using popular languages/technologies)

- Talk to managed and unmanaged devices (SSH, CLI, NETCONF, REST/RESTCONF)

- Technologies: YAML, JSON, YAQL, Jinja2, YANG, SchemaFrom, Mistral DSL, JavaScript, Python

Allows for idempotency: $f(x) = f(f(x))$



Workflow Manager

Learn by examples | Starting simple

```
---  
version: '2.0'  
  
learning:  
  type: direct  
  
tasks:  
  pingTask:  
    action: nsp.ping  
    input:  
      host: localhost  
      duration: 5  
...  

```



Workflow

Represents a process to achieve a goal or implement a procedure interesting to the operator.

It consists of tasks (at least one) describing what steps must be executed.



Task

Defines a specific computational step in the workflow. The functional step can be implemented with an action or another workflow.

Each task may require input and produce output.



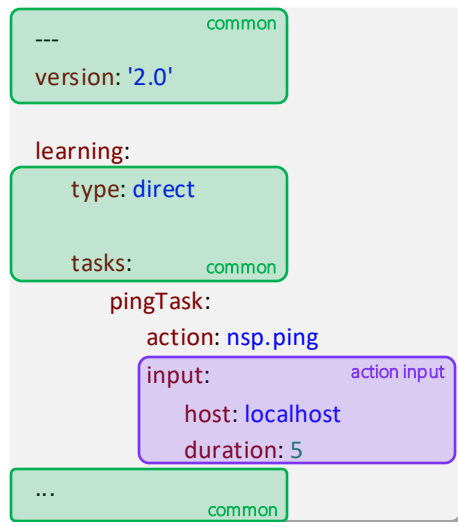
Action

Like functions in general purpose programming languages like Python.

It has a name and parameters. Mistral distinguishes *system actions* and *ad-hoc actions*.

Workflow Manager

Learn by examples | Starting simple



Main components:

Workflows are identified by a unique name

(example: learning)

Workflows are made of tasks identified by unique names

(example: pingTask)

Tasks are calling actions (or other workflows)

(example: nsp.ping)

Actions may require input and produce output

(example: host, duration)

Built-in action catalogue is extensible (at runtime)

Workflow Manager

Learn by examples | Documentation

```
---
version: '2.0'

learning:
  type: direct

  description: Primer Session Example

  tags:
    - SReXperts2025

  tasks:
    # single task only
    pingTask:
      action: nsp.ping
      input:
        host: localhost
        duration: 5
...
```

Always document your workflows:

- Add description
- Add detailed documentation (using README.md)
- Use tags to categorize workflows w/ labels (certain tags have special meaning)
- Add comments

Workflow Manager

Learn by examples | Workflow Input/Output

```
--  
version: '2.0'  
  
learning:  
  type: direct  
  
  input:  
    - host: localhost  
  
  output:  
    result: done  
  
  output-on-error:  
    result: failed  
  
  tasks:  
    pingTask:  
      action: nsp.ping  
      input:  
        host: <% $.host %>  
      duration: 5  
  ...
```

Add input variables

A list of input variables can be provided

Default-values for inputs are optional

YAQL or Jinj2 to use input variables

Add output variables

Two separate section for successful and failed executions

Output is a dictionary (or html)

Dynamic mapping is possible: YAQL or Jinja2

YAQL is a query language on objects to apply filtering, grouping and aggregation of data.

Jinja2 is a templating language used to generate text by embedding logic (like loops and conditions) to convert objects into content like HTML or JSON documents.

Workflow Manager

Learn by examples | Input Forms, Output Processing

```
--
version: '2.0'

learning:
  type: direct

  input:
    - mgmtIP

  output:
    success: <% task(pingTask).result.success %>

  output-on-error:
    success: <% task(pingTask).result.success %>

  tasks:
    pingTask:
      action: nsp.ping
      input:
        host: <% $.mgmtIP %>
      duration: 5
  ...
```

Input form

Targeting operators using NSP WebUI

Defined in schema-from (YAML or JSON)

Auto-generation is possible, based on default-values and variable names provided

mgmtIP becomes an auto-complete component (suggest) to pick a valid management IP-address (from device inventory)

Dynamic components (picker/suggest) retrieve data via workflow actions

Output processing

Use `task().result` within YAQL expressions to access the results of a task execution.

AUTO GENERATE UI

```
type: object
properties:
  - name: mgmtIP
    title: MGMT IP
    description: description for mgmtIP
    columnSpan: 4
    newRow: true
    readOnly: false
    required: false
    type: string
    component:
      input: autoComplete
      suggest
      action: nspWebUI.nelList
    name:
      - mgmtIP
```

Form

MGMT IP

192.168.97.10
192.168.99.7
192.168.96.86
192.168.96.49
192.168.96.233
192.168.97.135
192.168.96.40
192.168.97.6
192.168.98.132
192.168.98.13

10 of 17

Workflow Manager

Learn by examples | Lists and Loops

```
---
version: '2.0'

learning:
  type: direct

  input:
    - nodes

  tasks:
    pingTask:
      with-items: node in <% $.nodes %>
      concurrency: 1
      action: nsp.ping
      input:
        host: <% $.node.get("ip-address") %>
        duration: 5
      publish:
        result: <% task().result %>

...
```

Input may contain complex attributes (lists, dicts)

Example (⬅️): list of nodes; every nodes is an object (dict) by itself.

UI picker is automatically generated when using the keyword *nodes*.

Usability for WebUI operators get improved, while API usage is impacted

Flow control

The example (⬅️) iterates over the list of nodes, one after another.

Concurrency is disabled (one node at a time).

Every task can publish variables to environment

Tasks may succeed or fail, potentially returning different structures

Use statements “publish” and “publish-on-error” as needed

Workflow Manager

Learn by examples | NSP APIs w/ Flow Control

```
---
version: '2.0'

learning:
  type: direct

input:
  - neld

vars:
  baseUrl: https://restconf-gateway/restconf/data/nsp-equipment:network

tasks:
  getNodeDetails:
    action: nsp.https
    input:
      url: <% $.baseUrl %>/network-element=<% $.neld %>?fields=ip-address&include-meta=false
      method: GET
    publish:
      nodeInfo: <% switch(task().result.content.get('nsp-equipment:network-element').len())>0 => task().result.content.get('nsp-equipment:network-element').first() %>
    on-error:
      - nodeNotFound
    on-success:
      - nodeNotFound: <% not task().result.content %>
      - pingTargetDevice: <% task().result.content %>
```



Control Flow

Conditional logic and transitions that define the flow between tasks in a workflow.



Data Flow

Data passing procedure (published data) between tasks in a workflow.

```
pingTargetDevice:
  action: nsp.ping
  input:
    host: <% $.nodeInfo.get('ip-address') %>
    duration: 5
  publish:
    result: <% task().result %>
  on-error:
    - pingFailed
```

```
nodeNotFound:
  action: std.fail
  publish-on-error:
    result: "ERROR: Failed to get node details"
```

```
pingFailed:
  action: std.fail
  publish-on-error:
    result: "ERROR: Ping failed"
```

...

Workflow Manager

Learn by examples | Inventory Find

```
---
version: '2.0'

learning:
  type: direct

input:
  - neName

vars:
  baseUrl: https://restconf-gateway/restconf/operations

tasks:
  getNodeDetails:
    action: nsp.https
    input:
      url: <% $.baseUrl %>/nsp-inventory:find
      method: POST
      body:
        input:
          xpath-filter: /nsp-equipment:network/network-element[ne-name=<% $.neName %>']
          fields: ne-id;ne-name;ip-address;version;location;type;product
          include-meta: false
          resultFilter: $.content.get('nsp-inventory:output').data
    publish:
      nodeInfo: <% switch(task().result.content.len())>0 => task().result.content.first() %>
    on-error:
      - nodeNotFound
    on-success:
      - nodeNotFound: <% not task().result.content %>
      - pingTargetDevice: <% task().result.content %>
```

Example uses the UI-centric ne-name as input
Requires xpath-filter: ne-name is NOT the key
nsp-inventory:find request is doing the magic
It has additional processing capabilities to keep
workflow engine processing simpler

```
pingTargetDevice:
  action: nsp.ping
  input:
    host: <% $.nodeInfo.get('ip-address') %>
    duration: 5
  publish:
    result: <% task().result %>
  on-error:
    - pingFailed

nodeNotFound:
  action: std.fail
  publish-on-error:
    result: "ERROR: Failed to get node details"

pingFailed:
  action: std.fail
  publish-on-error:
    result: "ERROR: Ping failed"

...
```

Workflow Manager

Learn by examples | Jinja Templating

```
consolidate:
  action: std.echo
  input:
  output:
    svcName: <% $.svcName %>
    neSvcId: <% $.neSvcId %>
    svcKind: <% $.svcKind %>
    tunnelHops: <% $.tunnelHops %>
  publish:
    tunnelInfo: <% task().result %>
  on-success:
    - prettyPrint

prettyPrint:
  action: nsp.jinja_template
  input:
    name: render-to-report
    data: <% $.tunnelInfo %>
  publish:
    result: <% task().result.template %>
```

Example uses the std.echo action to frame the input data required for rendering the jinja template. This flow generates a html table report of the data consolidated.

Workflow Manager

Learn by examples | The power is in the actions

```
scanTargetDevice:  
  action: nsp.shell  
  input:  
    command: nmap -p21-23,53,179,646,830,3455,3784,57400 -oF - <% $.mgmtIP %>  
  publish:  
    scanresult: <% task().result.response %>
```

```
cliChecks:  
  action: nsp.mdm_cli  
  input:  
    neld: <% $.neld %>  
    cmds:  
      - show system information  
      - tools dump system-resources  
      - show service service-using  
      - show service sap-using  
      - show port  
  publish:  
    clidump: <% task().result.responses %>
```

```
getConfigSROS:  
  action: nsp.https  
  input:  
    method: GET  
    url: https://restconf-gateway/restconf/data/network-device-mgr:network-devices/network-device=<% $.neld %>/root/nokia-conf:configure  
  publish:  
    jsonConfig: <% json_dump(task().result.content) %>
```

Workflow Manager

INPUT FORM | AUTO GENERATION RULEZ

Default-values data-types (boolean, integer, string) will be used

Special handling using **RegEx** if default-value is any of the following:

IPv4 address/subnet, MAC address, URL or e-mail address

Certain keywords will result in the usage of a password component (any case, supporting kebab-case and camelCase):

PASS, PASSWD, PASSWORD, SECRET, AUTH, AUTHORIZATION, TOKEN, CRED, CREDENTIALS, LOGIN

Use keywords **neld**, **neName**, and **mgmtIP** to suggest a device

Suggests also exists for following keywords:

port, workflow, intentType, intent

Multi-selection component is added for keywords:

nodes, ports, workflows, intentTypes



Workflow Manager

Best practices for workflow development

We recommend to follow best practices when designing workflows. Here some examples:

- Validate workflow input variables

- Use consistent names for input/output variables

- Handle corner-cases and exceptions properly

- Execute functional, performance, and(!) scale testing

- Avoid hard-coding IP addresses, usernames, passwords

- Use server-side filtering rather than post-filtering

 - If server-side filtering is not available, use filtering capabilities in nsp.https

 - Store extra-large responses as file

 - Include inventory meta only if needed

- Start simple, add complexity and error handling later, refactor/modularize as needed.

- Remind: Keep things simple!

- Design YAQL expressions carefully

- Consider: empty unset properties and empty lists

- Use DNS to resolve NSP service endpoints

- Avoid: NSP external IP, location service

- Use join for clean execution graphs

- Use nsp.python instead of std.js

- Avoid unnecessary REST/RESTCONF calls

- Use system-user or revoke authentication tokens

- Add documentation

https://network.developer.nokia.com/learn/25_4/programming/development-best-practices

https://network.developer.nokia.com/learn/25_4/programming/workflows/wfm-workflow-development/wfm-best-practices

Operations



Device Operations

Introduction

Operator-centric engine to allow mass-execution of workflows

- Catalogue of operation-types to be executed

- Operation targets: managed network elements

- WebUI to plan, execute and monitor campaigns

Value-add

- Close WebUI integration with Device Management

- Tracking of execution progress

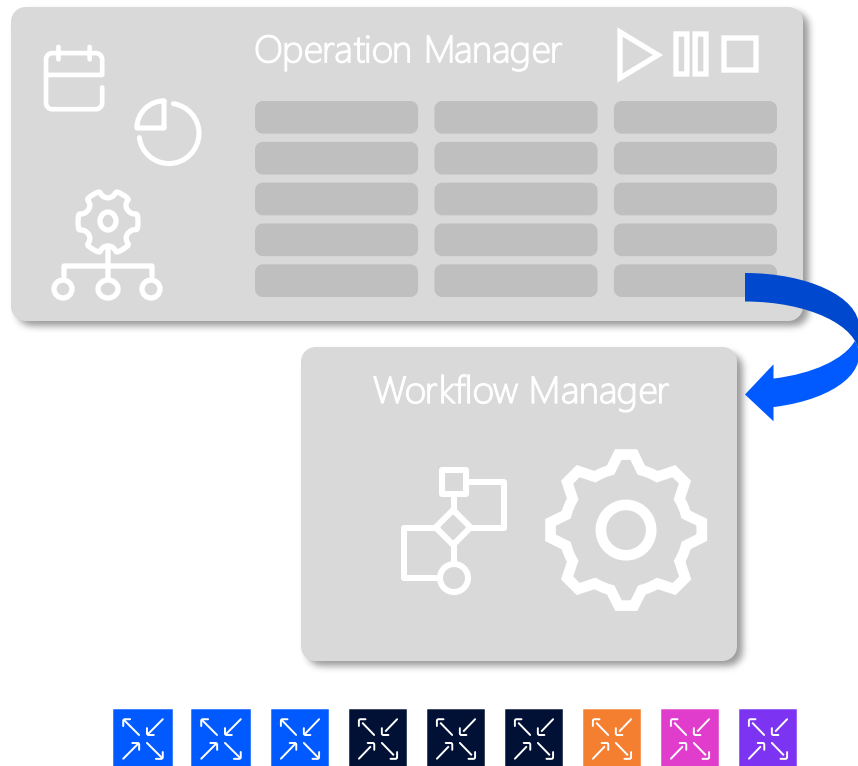
- Parallel execution with concurrency control

- Multi-phase operations for MOP automation

- A la carte execution control:

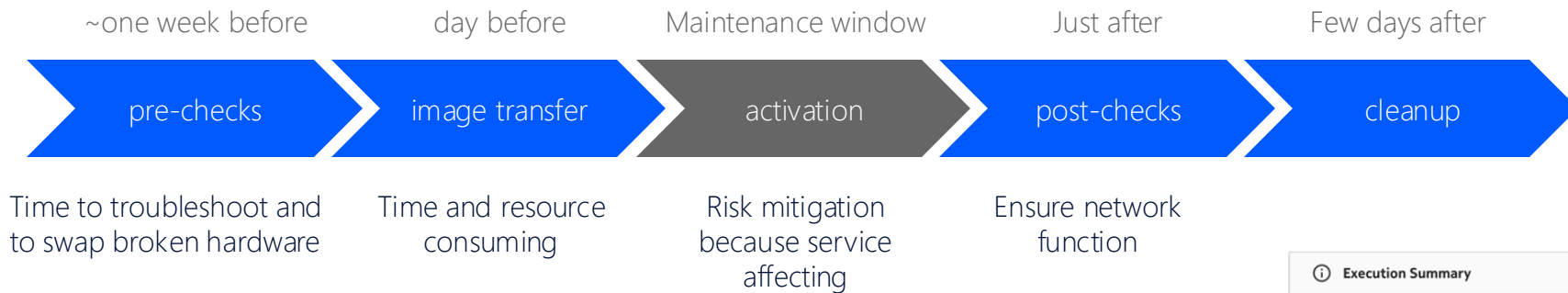
 - operation-level, per phase, per target

- Execution KPIs to trigger automated actions



Device Operations

Phasing Example for NE Software Upgrades



● 1. Pre-Check ● 2. Execute ● 3. Post-Check ● 4. Clean-up

Status	NE Name	Product	Number	
● Success	London	7750 SR		☐
● Error	Auckland	7750 SR		☐
● Success	Berlin	7750 SR		☐
● Success	Madrid	7750 SR		☐

Execution Summary

General

Operation failed for the specific NE ID

Start Time
2022/11/25 16:08

End Time
2022/11/25 16:08

Workflow
nsp-iso-multi-step-workflow-demo

Total Rerun
N/A

Device Operation

Artifact Bundle Anatomy

nsp-ping-v1.0.0

└─ **content**

| └─ **ne-ping**

| | └─ README.md

| | └─ ne-ping.json

| | └─ ne-ping.yaml

Workflow(s)

| └─ **nsp-ping**

| | └─ nsp-ping.yaml

| | └─ nsp-ping.yang

| | └─ operation_types.json

Device Operation

└─ metadata.json

Artifact Bundle META

Device Operation

Learning by examples | Workflow Adjustments

```
--  
version: '2.0'  
  
ne-ping:  
  type: direct  
  
  tags:  
    - LSO  
  
  input:  
    - neld  
    - duration: 5  
  
  output:  
    IsoInfo: <% $.IsoInfo %>  
  
  output-on-error:  
    IsoInfo: <% $.IsoInfo %>  
    IsoStageError: <% $.IsoStageError %>  
  
  tasks:  
    IsoStageMapping:  
      action: std.noop  
      publish:  
        IsoWorkflowStageMapping:  
          ne-ping:  
            - getNodeDetails  
            - pingTargetDevice
```

Adjustments for workflows used for Device Operations

Apply tag "LSO"

Input must contain "neld"

Output contains

IsoInfo – clear-text message about success/failure

IsoStageError – error details in clear-text

IsoWorkflowStageMapping – defines execution steps shown at UI

Device Operation

Learning by examples | Operation Mapping

```
phases:  
  - phase: 'Execute'  
    description: 'Execute ICMP Ping'  
    concurrency_count: 30  
    phase_timeout: 15  
    ne_families:  
      - family_type: 7750 SR, 7950 XRS, 7450 ESS, 7250 IXR, 7220 IXR SRLinux, 7250 IXR SRLinux  
        ne_versions:  
          - version: all  
            workflow_name: ne-ping  
            workflow_inputs:
```

Operation mapping (YAML)

Example above shows a single-phase operation

Mapping of phases, device families and NOS versions to workflows

Execution control can be adjusted: concurrency, timeout

Workflow input can be passed based on phase/family/version

Device Operation

Learning by examples | Artifact Bundle META

```
{
  "meta-data-header": {
    "version": "1.0.0",
    "buildNumber": "1",
    "formatVersion": "1.0",
    "createdBy": "Sven Wisotzky",
    "creationDate": "Fri, 2 May 2025 10:00:00 UTC",
    "title": "Operation for NE Ping",
    "description": "check device reachability with latency"
  },
  "artifact-meta-data": [{
    "targetApplication": "workflow-manager",
    "applicationCompatibility": "24.11+",
    "version": "1.0.0",
    "name": "ne-ping",
    "artifact-content": [{
      "type": "application/octet-stream",
      "path": "content/ne-ping",
      "fileName": "ne-ping.yaml"
    }], {
      "targetApplication": "iso-server-app",
      "applicationCompatibility": "24.11+",
      "version": "1.0.0",
      "name": "nsp-ping",
      "artifact-content": [{
        "type": "application/octet-stream",
        "path": "content/nsp-ping",
        "fileName": "nsp-ping.yaml"
      }], {
        "type": "application/octet-stream",
        "path": "content/nsp-ping",
        "fileName": "nsp-ping.yang"
      }], {
        "type": "application/octet-stream",
        "path": "content/nsp-ping",
        "fileName": "operation_types.json"
      }
    ]
  }
}
```

Artifact Bundle META (JSON)

Used to package everything together to be installed in Artifact Admin

Bundle header provides some general information

List of all artifacts is provided

Artifacts consists of one or multiple files

Artifacts belong to applications: workflow-manager, iso-server-app

Artifacts are versioned

Compatibility is contained

Size/digest to avoid installing a broken/suspicious artifact (optional)

Artifact Bundle META can be protected by a signature



Intent-based Networking

Introduction

Intent-Based Networking

Network Automation Evolution



Configuring myriads of “nerd knobs” on a device-by-device basis is no longer considered an option to manage modern network environments. IBN targets to address following challenges:



I E T F*

- Keep device configuration consistent across the network and with the needs of services
- Streamline operations to lower expenses and to enable rapid (re-)configuration at scale, to ensure networks delivering expected functionality
- Reduce dependencies on human activities (less mistakes)
- Focusing on outcomes to enable operational efficiency and flexibility at scale in time
- Enable autonomous networks, self-management and artificial intelligence techniques

Enabling simplified, outcome-centric interaction that involves higher-layer abstraction.

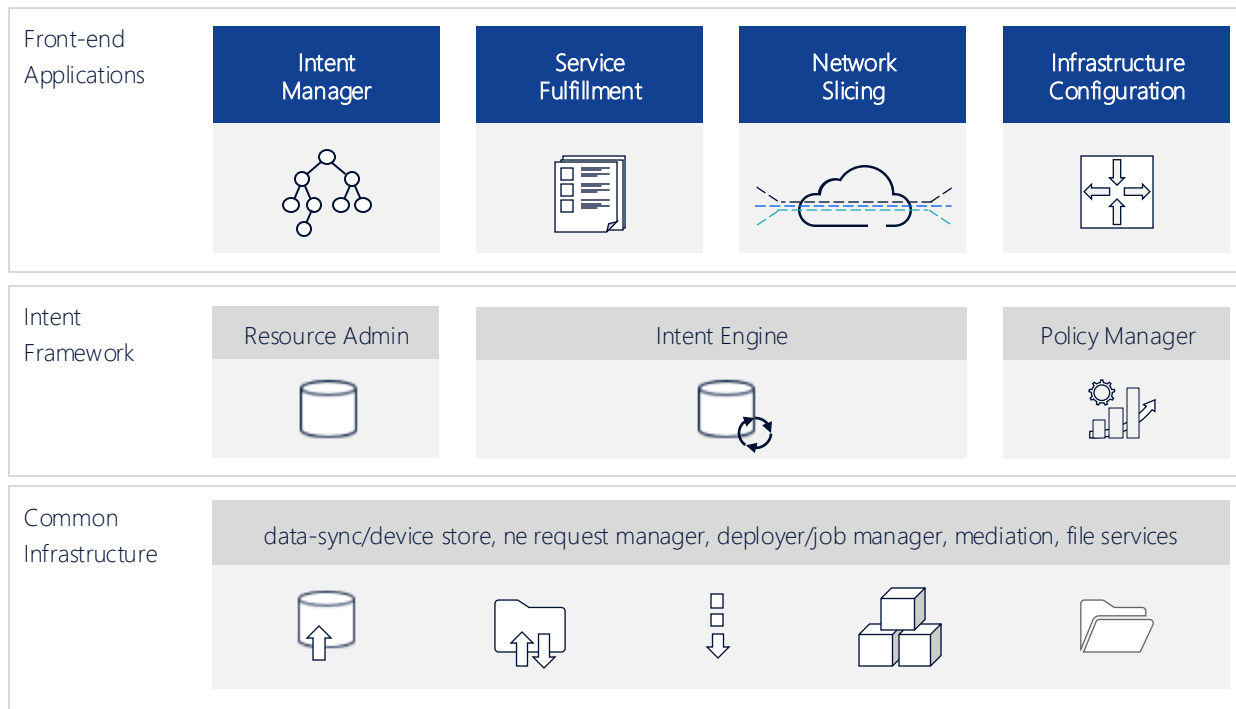
Operators to focus on the intent (desired outcome) by providing a minimal set of high-level parameters.

IBS translates the intent into policies and actions (detailed device-level configuration).

Operational tools help operators to ensure configuration correctness and service behavior.

Intent-Based Networking

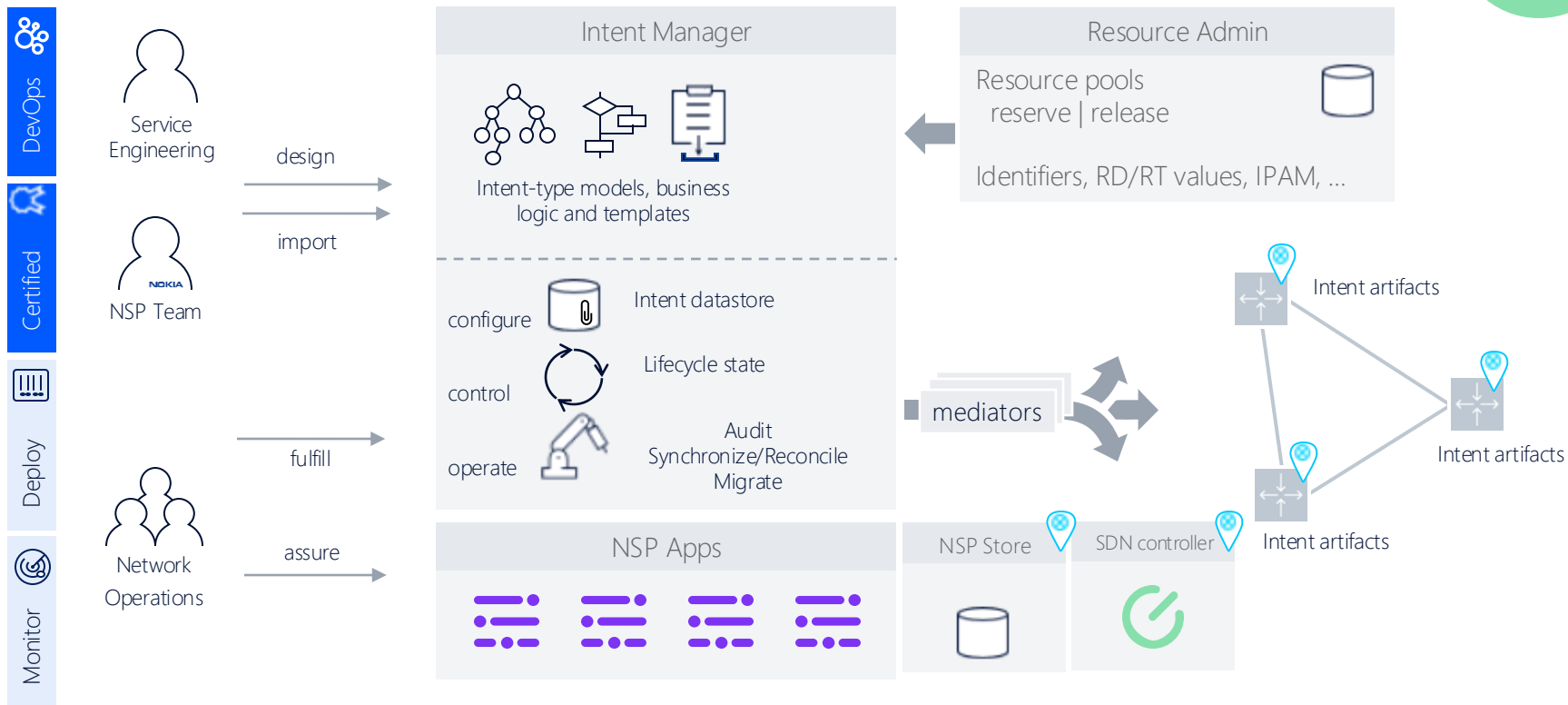
NSP Architecture, Offered Solutions



Intent-Based Networking

NSP Solution Details

Solution
Overview



Intent-Based Networking

Operational flexibility enabled by NSP



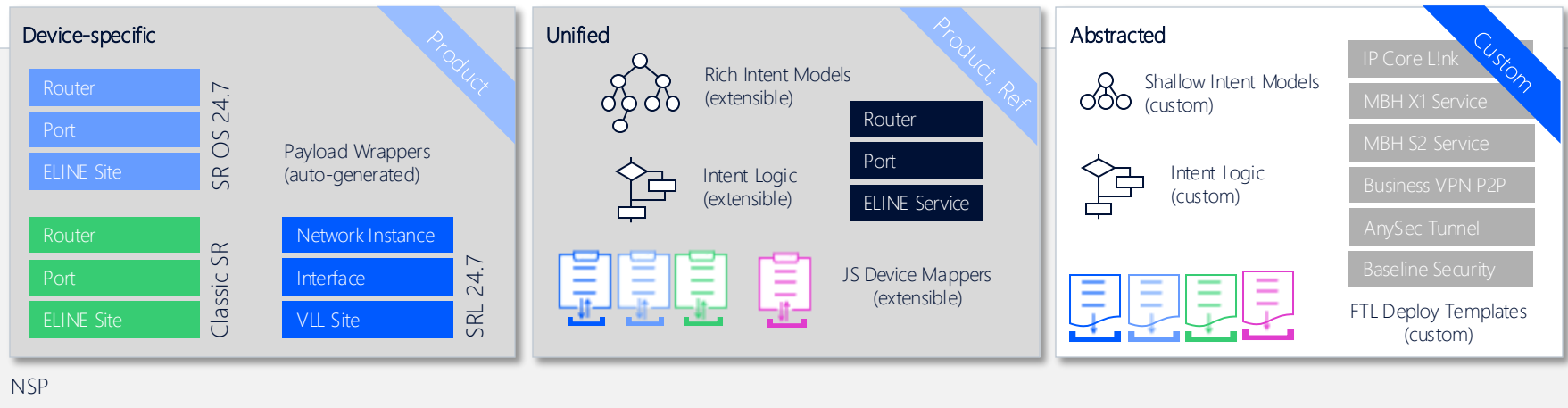
WebUI centric (EMS style)
100% attribute coverage



WebUI/OSS centric (NMS style)
common attribute coverage (normalized)

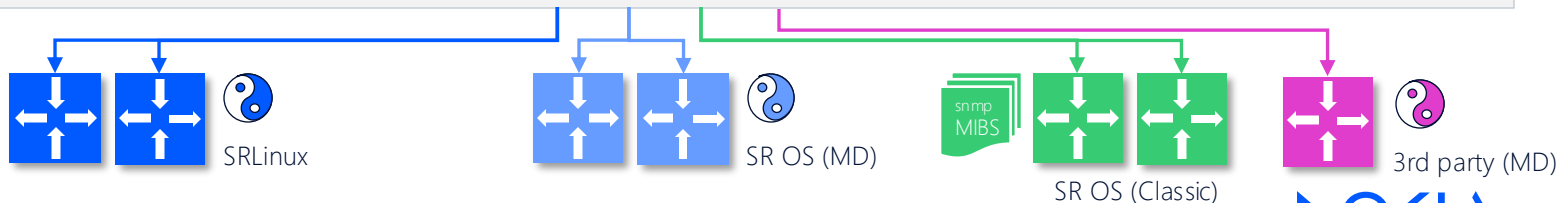


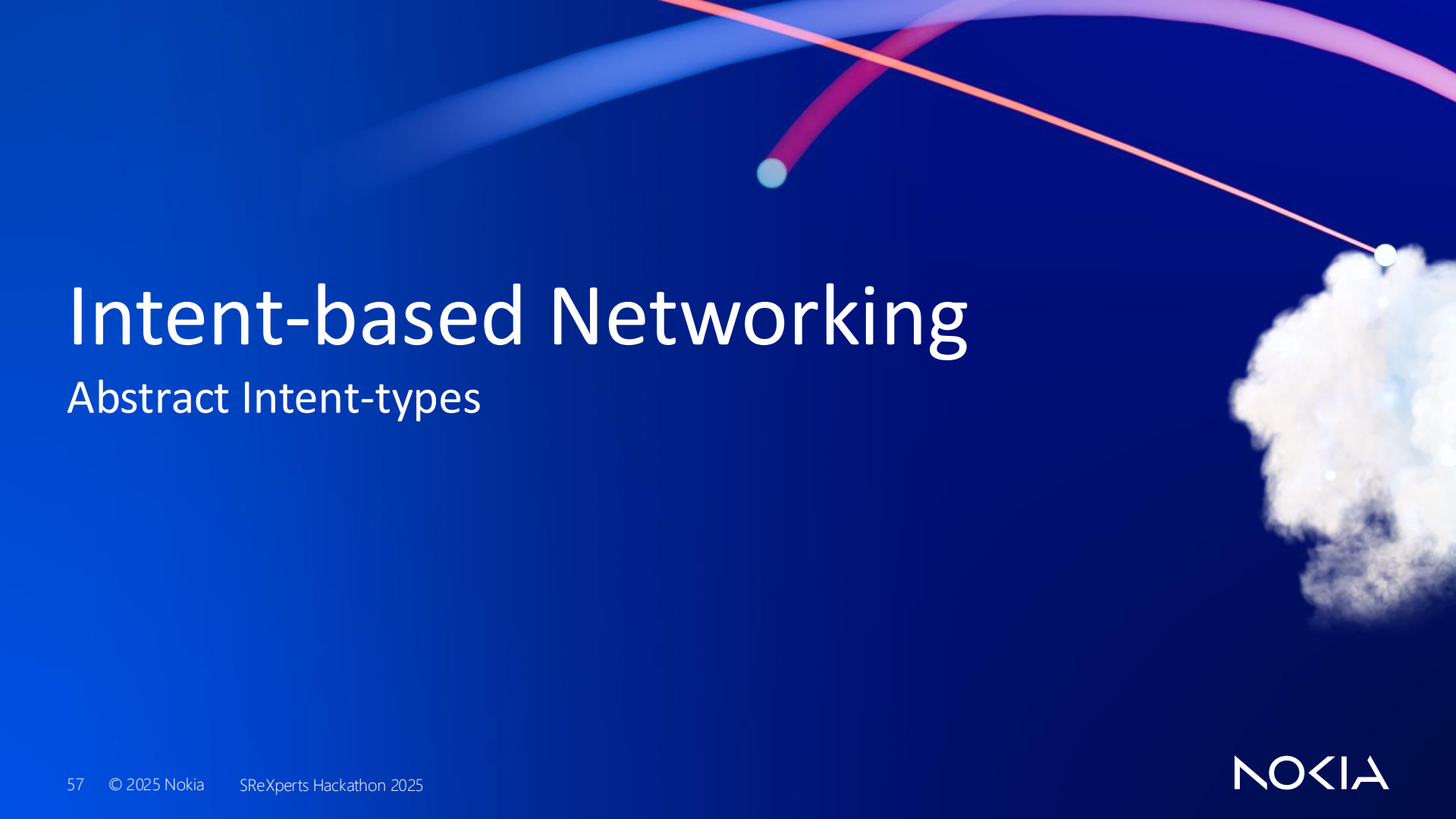
Network Design centric (OSS/WebUI)
Abstraction: higher-level attributes only



NSP

heterogeneous network





Intent-based Networking

Abstract Intent-types

IBN-enabled abstraction

DESIGN TARGETS

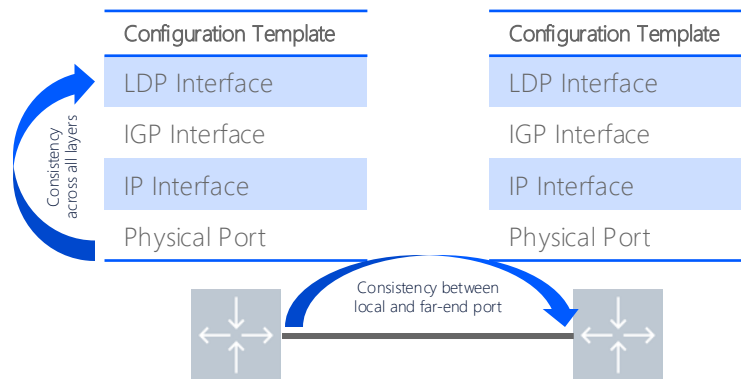
Enabling network engineers to automate
network design configuration (targeting SMEs, not programmers)

Positioned as evolution of XMLAPI scripts and templates, but...

Simpler! and More Powerful!

IBN-enabled abstraction | A practical example

Concept study for an inter-router link (IRL or IPL!nk)



Provisioning principles

Configure all layers on both peers to build an "IP link"

IP subnet (/31) from pool (Resource Admin) shares life-cycle

Only minimum input required: nodeA/portA and nodeB/portB

Detailed settings accordance to network design are hard-coded into the FTL
Examples: MTU, QoS, LLDP, EFM OAM, Accounting, ETH mode/encapsulation

Solution benefits

All layers are configured consistently in accordance to network design (LLD)

Configuration across all layers avoids misconfiguration and provisioning errors

Deleting an IP-Link intent will clean-up the entire config and resources

Simplification for the user (or OSS) to create a working IP link

Minimized input, No enforced imperative order, Light-weight inventory

Better performance/scale (committed to nodes involved as single transaction)

Easy to support new product families and vendors

No extra/custom auditing tools for link endpoint consistency

Abstract Intent-type Anatomy

iplink_v1

└─ intent-type-resources

└─ common

- └─ IntentHandler.mjs
- └─ NSP.mjs
- └─ ResourceAdmin.mjs
- └─ WebUI.mjs
- └─ patch.ftl



Common library for
abstract intent-types

Never touch(!)

└─ mappers

- └─ OpenConfig.ftl
- └─ SR OS.ftl
- └─ SRLinux.ftl

Device Mappers
(NetworkOS specific)

- └─ default.viewConfig

WebUI customization

- └─ meta-info.json

Intent-type META

- └─ script-content.mjs

Intent-specific logic

└─ yang-modules

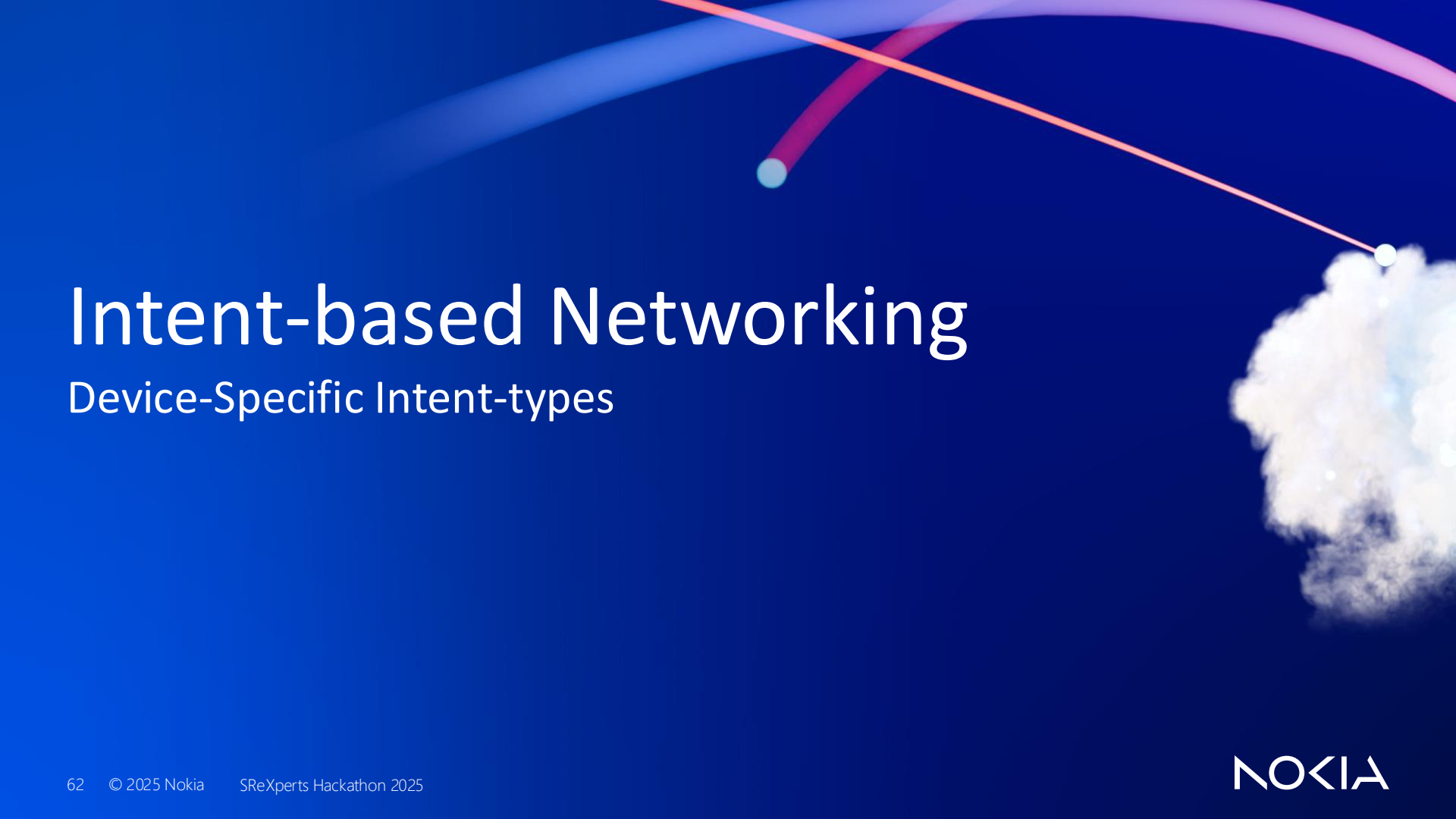
- └─ iplink.yang

Abstract Intent Model

Extract device config in JSON format:

SR OS info json; pwc model-path

SRLinux info | as json



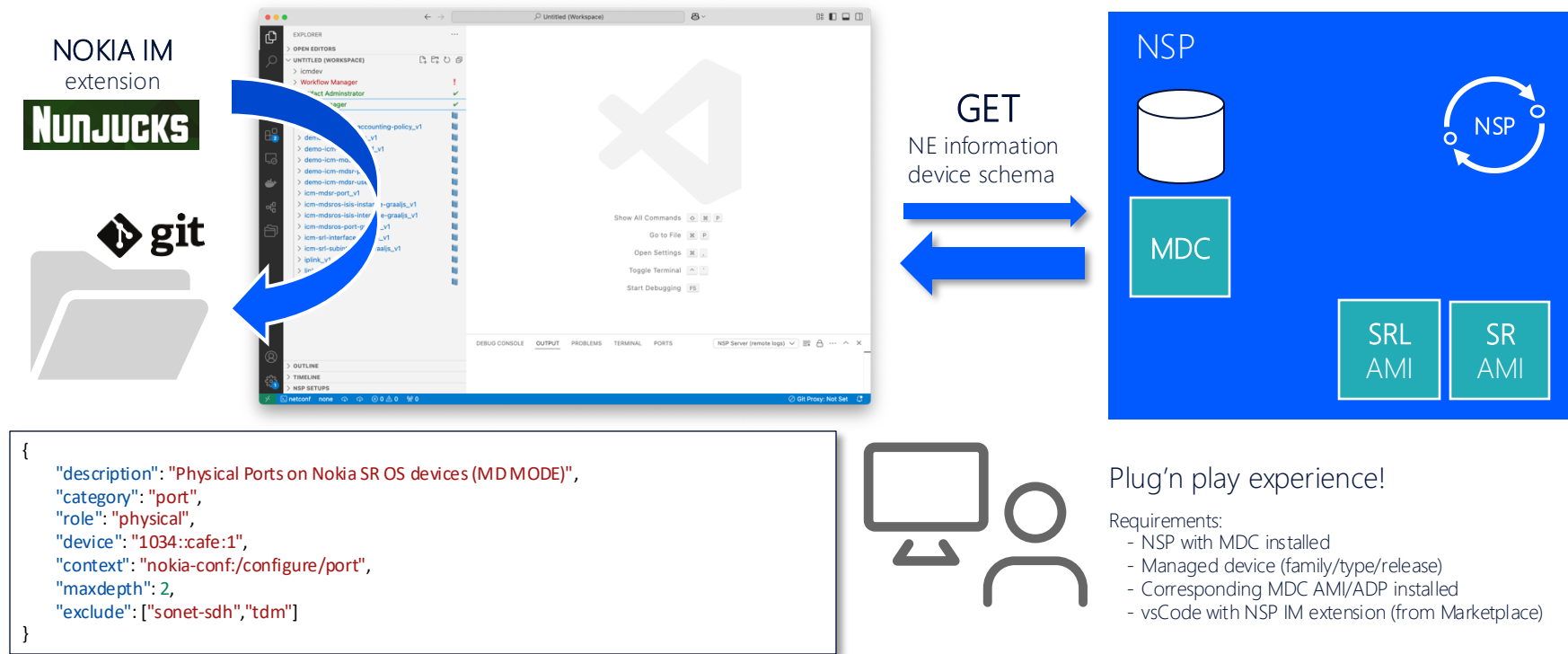
Intent-based Networking

Device-Specific Intent-types

Designing *device-specific* intent-types

- Support all device configuration (100% objects/attributes)
- Simplified Design (utilize device model, minimize custom code)
- DX-centric approach using Visual Studio Code as one-stop-shop
- Following best-practices to ensure supportability
- Decomposed design to simplify maintainability
- No-code principles: avoid human error and manual testing
- Easy regeneration of the entire library
- Hooks to adjust behavior (if needed) all maintained in a single file

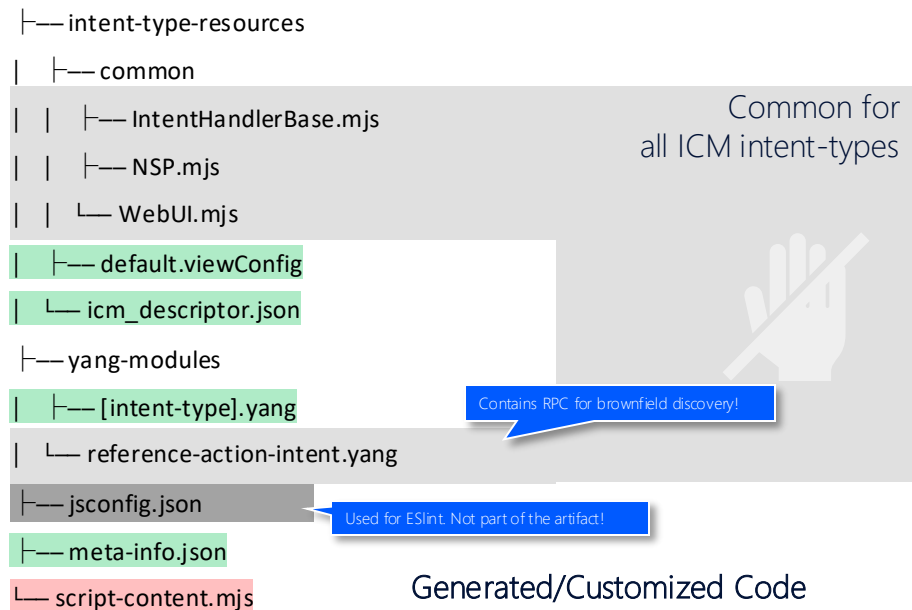
Intent-type generator using Visual Studio Code



Optimized input: quicker turn-around

Generated Intent-types

Less Code, Clear Structure



Common Code base

JavaScript | ~2.000 lines of code, 3 files

Generated/Customized Code

JavaScript | <100 lines of code, 1 file

Generated Intent-types

What's inside? It's all auto-generated! It's easy!

```
/*  
 * DEVICE-SPECIFIC INTENT-TYPE  
 * Vendor: Nokia  
 * Device families: 7750 SR  
 * Device version: 24.10.R1  
 *  
 * (c) 2025 by Nokia  
 */  
***** Family / Type / Release
```

```
import { IntentHandlerBase } from 'common/IntentHandlerBase.mjs';  
import { NSP } from 'common/NSP.mjs';
```

```
class IntentHandler extends (IntentHandlerBase) {  
  constructor() {  
    super();  
    NSP.checkRelease(24, 11);  
    this.ignoreChildren = ['sonet-sdh', 'tdm', ...];  
  }  
}
```

NSP compatibility check

Exclusions (pre-approved misalignments)

ROOT XPATH

```
getDeviceModelPath(target) {  
  const items = target.split('#');  
  return `nokia-conf:/configure/port=${encodeURIComponent(items[2])}`;  
}
```

```
getDesiredConfig(target, intentConfigJSON) {  
  const items = target.split('#');  
  const config = Object.values(JSON.parse(intentConfigJSON)[0])[0];  
  
  config['port-id'] = items[2];  
  return {'nokia-conf:port': [this.cleanupConfig(config)]};  
}
```

Rewrapping: intent config to desired config

Hooks, e.g. masking passwords

```
preAuditHook(neld, path, aConfig, iConfig) {  
  }  
  
suggestPortId(context) {  
  const target = this.getTarget(context);  
  return this.getDeviceModelObjects(context, `nokia-conf:/configure/port`);  
}  
  
suggestServiceOperGroupName(context) {  
  const target = this.getTarget(context);  
  return this.getDeviceModelObjects(context, `nokia-conf:/configure/service/oper-group`);  
}  
  
suggestDistCpuProtectionPolicyName(context) {  
  const target = this.getTarget(context);  
  return this.getDeviceModelObjects(context, `nokia-conf:/configure/system/security/dist-cpu-protection/policy`);  
}  
  
suggestLogAccountingPolicyId(context) {  
  const target = this.getTarget(context);  
  return this.getDeviceModelObjects(context, `nokia-conf:/configure/log/accounting-policy`);  
}  
}  
  
new IntentHandler();
```

Suggest callouts

Your journey starts with an *.igen file in your VS Code workspace...

demo-icm-mdsr-ospf-area.igen

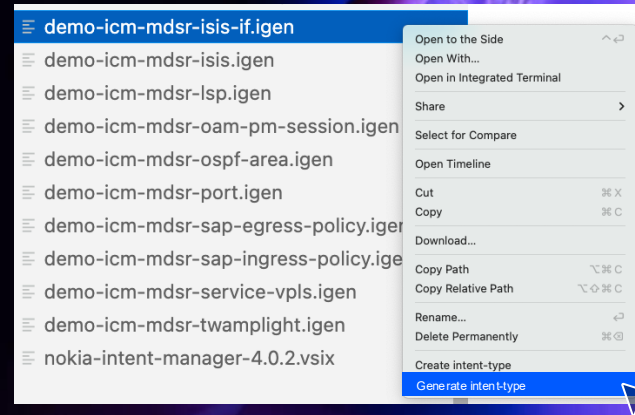
```
{
  "description": "OSPF area on Nokia SR OS devices (MD MODE)",
  "category": "router",
  "role": "logical",

  "device": "1034::cafe:1",

  "labels": ["ApprovedMisalignments"],
  "author": "NOKIA",
  "intent_type": "demo-icm-mdsr-ospf-area",
  "date": "2025-05-01",

  "context": "nokia-conf:/configure/router=Base/ospf/area",
  "exclude": ["interface", "virtual-link"],
  "maxdepth": 2,

  "withdefaults": true,
  "constraints": true,
  "applygroups": true,
  "icmstyle": true
}
```



ISIS example

icm-mdsr-isis.igen

```
{
  "description": "ISIS Interfaces on Nokia SR OS devices (MD MODE)",
  "category": "router",
  "role": "logical",
  "device": "1034::cafe:1",
  "context": "nokia-conf:/configure/router=Base/isis",
  "exclude": ["interface"],
  "icmstyle": true
}
```

icm-mdsr-isis-if.igen

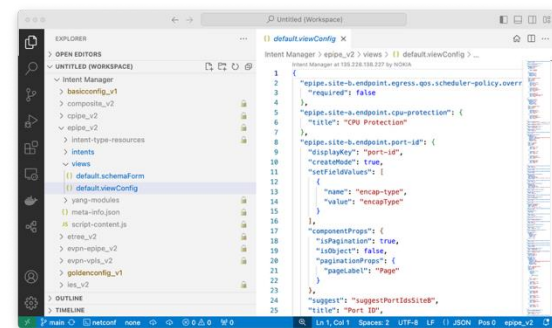
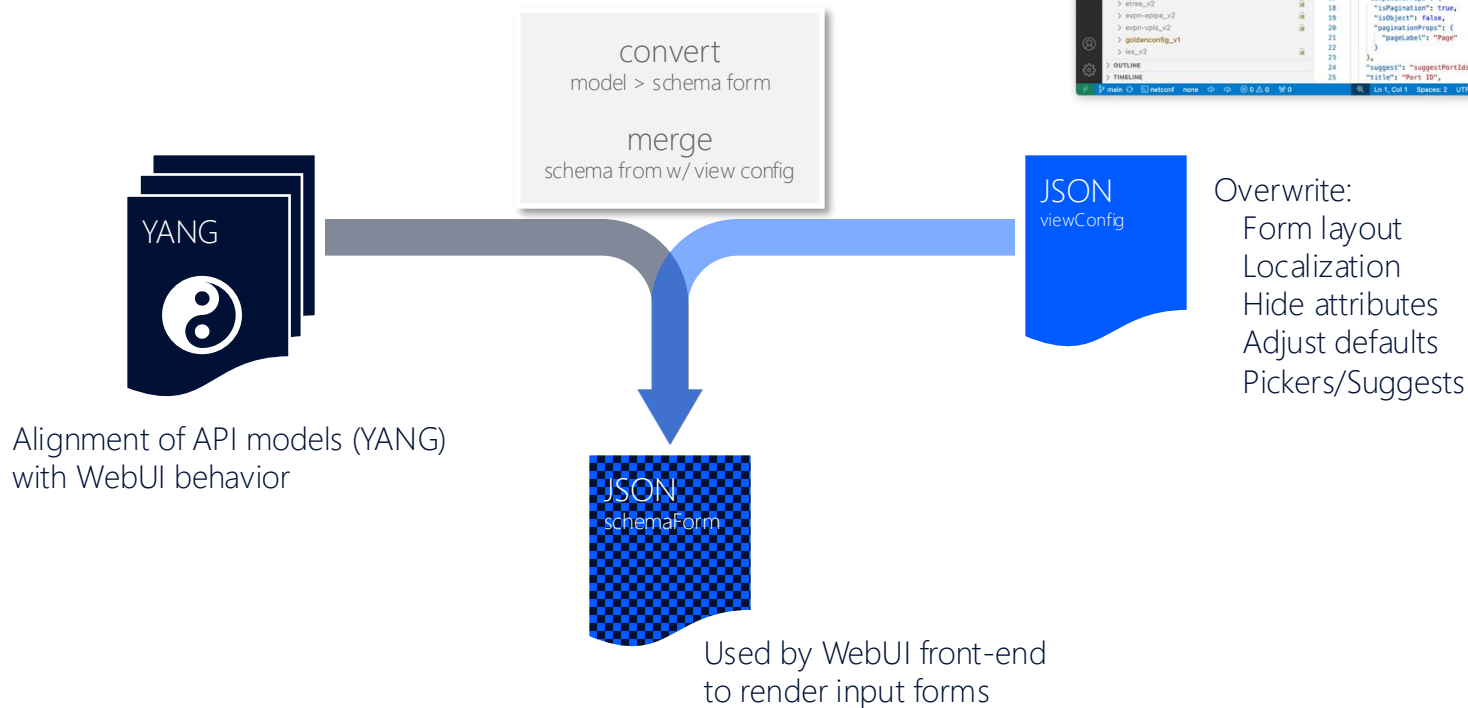
```
{
  "description": "ISIS Interfaces on Nokia SR OS devices (MD MODE)",
  "category": "router",
  "role": "logical",
  "device": "1034::cafe:1",
  "context": "nokia-conf:/configure/router=Base/isis=0/interface"
}
```



Getting it out of the doors

WebUI Customization

SchemaForm & ViewConfig



WebUI Customization

View Config Builder | since NSP 24.11

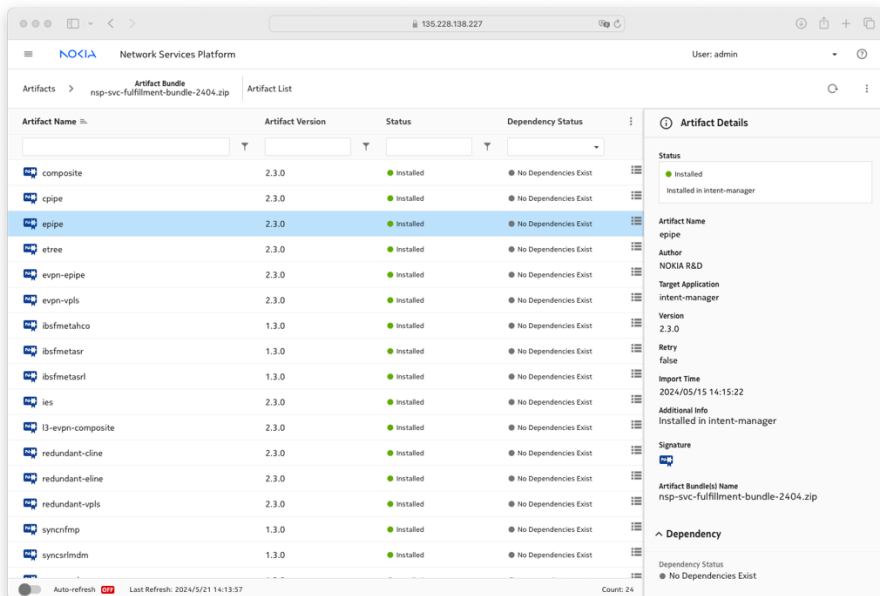
The screenshot shows a web browser window with the URL `nsp.srexperts.net`. The page title is "Nokia Network Services Platform" and the user is logged in as "admin". A modal dialog titled "Adjust Attribute - port-id - string" is open. The dialog has a sidebar on the left with the following menu items: "General Attributes" (selected), "Field Removal", "Field Specific Attributes", "Cross Launch Attributes", and "Statements". The main content area is titled "General Attributes" and contains the following fields:

- Name**: port-id
- View Config Path**: epipe.site-a.endpoint.port-id
- Title**: Port ID
- Description**
- Type***: Leaf Ref
- Column Span**: 2
- Default Value**
- ☐ Read Only
- ☐ Disabled

At the bottom right of the dialog, there are two buttons: "CLOSE" and "UPDATE". A mouse cursor is pointing at the "UPDATE" button. Below the dialog, at the bottom of the browser window, there are two more buttons: "CLOSE" and "SAVE VIEWCONFIG".

Pluggable Automation

Artifact Manager



Artifact Name	Artifact Version	Status	Dependency Status
composite	2.3.0	Installed	No Dependencies Exist
cpipe	2.3.0	Installed	No Dependencies Exist
epipe	2.3.0	Installed	No Dependencies Exist
etree	2.3.0	Installed	No Dependencies Exist
evpn-epipe	2.3.0	Installed	No Dependencies Exist
evpn-vpls	2.3.0	Installed	No Dependencies Exist
ibsfmetahco	1.3.0	Installed	No Dependencies Exist
ibsfmetasr	1.3.0	Installed	No Dependencies Exist
ibsfmetasrl	1.3.0	Installed	No Dependencies Exist
ies	2.3.0	Installed	No Dependencies Exist
l3-evpn-composite	2.3.0	Installed	No Dependencies Exist
redundant-cline	2.3.0	Installed	No Dependencies Exist
redundant-eline	2.3.0	Installed	No Dependencies Exist
redundant-vpls	2.3.0	Installed	No Dependencies Exist
syncnfp	1.3.0	Installed	No Dependencies Exist
syncrindm	1.3.0	Installed	No Dependencies Exist

Artifact Details

Status
● Installed
Installed in intent-manager

Artifact Name
epipe

Author
NOKIA R&D

Target Application
intent-manager

Version
2.3.0

Retry
false

Import Time
2024/05/15 14:15:22

Additional Info
Installed in intent-manager

Signature
[Signature]

Artifact Bundle(s) Name
nsp-svc-fulfillment-bundle-2404.zip

Dependency

Dependency Status
● No Dependencies Exist



One-stop-shop for
Artifact Lifecycle Management

Intuitive to use
User Interface



Track all artifacts/bundles
installed in NSP

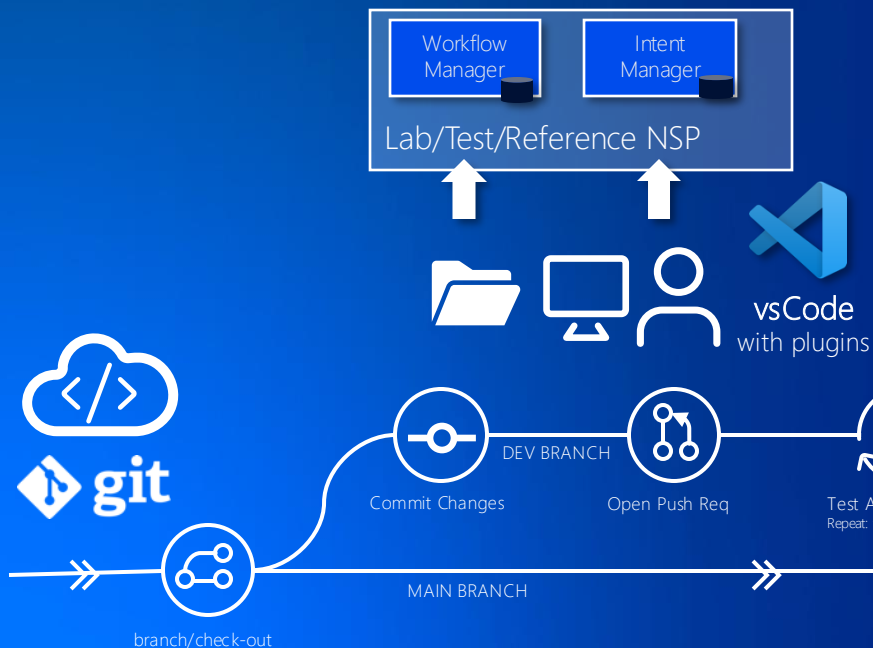
Validate artifact/bundle
installation w/ dependencies



Alerts installation issues and
unresolved dependencies

Artifact CI/CD

Enabled by NSP vsCode extensions



GitHub, GitLab

Version control system for source code.

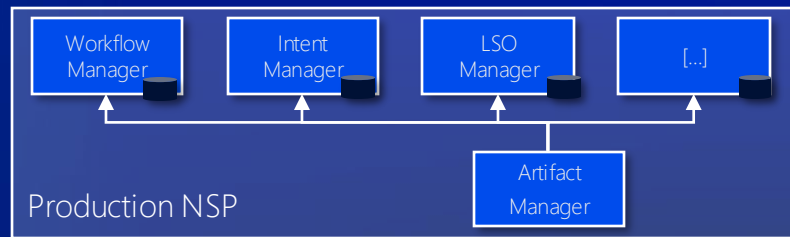
Visual Studio Code

Improved coding experience with native git integration.

NSP extensions to connect to lab system and to enable IntelliSense.

Artifact Bundles

Artifact Administrator provides a simplified/centralized way to install artifacts to NSP to enable new use-case and to add support for new vendors, device families and software releases.





Visual Studio Code Extensions by Nokia

Available since mid 2023

NETCONF extension

3200 users

Enable self-learning, testing, reproducing of behaviour, and integration (templating).

Works with Nokia NI networking products and 3rd party vendors.

NSP Workflow Manager extension

710 users

Provide better offline and online editing experience.

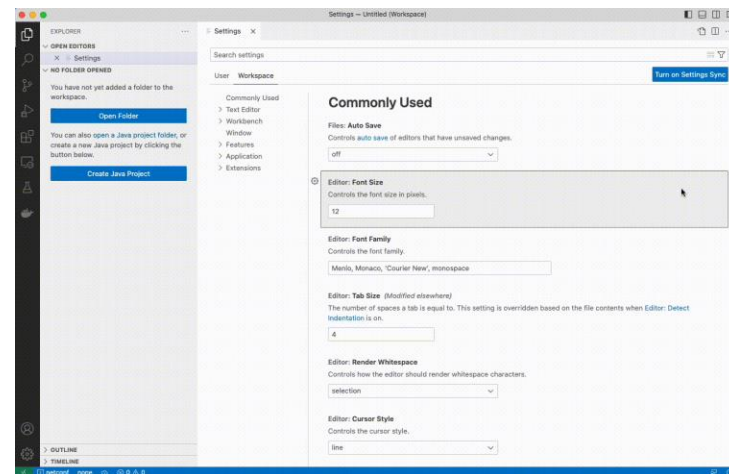
Support download/upload of workflows and execution.

NSP Intent Manager extension

560 users

Functionality aligned with Workflow Manager extension.

Supports CRUD operations and synchronization of intents.



NSP Programmability

Resources

Network Developer Portal

https://network.developer.nokia.com/learn/24_11/programming

Visual Studio Code

<https://code.visualstudio.com>

Visual Studio Marketplace

<https://marketplace.visualstudio.com/items?itemName=Nokia.nokia-intent-manager>

<https://marketplace.visualstudio.com/items?itemName=Nokia.nokia-wfm>

<https://marketplace.visualstudio.com/items?itemName=Nokia.netconf-client>

Mistral DSL

https://docs.openstack.org/mistral/latest/user/wf_lang_v2.html

YAML

<https://yaml.org>

Jinja2

<http://jinja.pocoo.org>

Markdown

<https://www.markdownguide.org>

YAQL

<https://yaql.readthedocs.io/en/latest>

<http://yaqluator.com>

Regular Expression

<https://regex101.com>

JSON ↔ YAML

<https://www.json2yaml.com>

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.