NSO Service Packages

NSO Service Packages

Mustafa Çağatay Behadır

Module-1

NSO Introduction

Module's Content

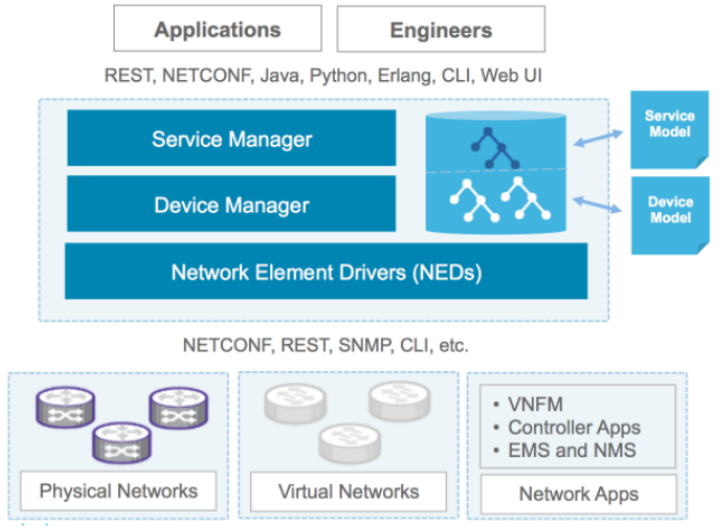
- Module-1: NSO Introduction
  - 1.1 NSO Overview
  - 1.2 NSO Architecture
  - 1.3 NSO Components
  - 1.4 NETCONF and YANG Overview
  - 1.5 Cisco NSO Packages
  - 1.6 Cisco NSO Mapping Logic
  - 1.7 Network Element Drivers

1.1 NSO Overview

- Cisco Network Services Orchestrator (NSO) allows creation, addition, and modification of services faster and more easily through network automation.
- Keep the network in a consistent state.
- Network configuration need to be sync with the NSO.
- Key features of NSO:
  - Multi-vendor nature
  - Configuration Database (CDB)
  - A set of northbound interfaces including human interfaces like web UI and a CLI

1.2 NSO Architecture

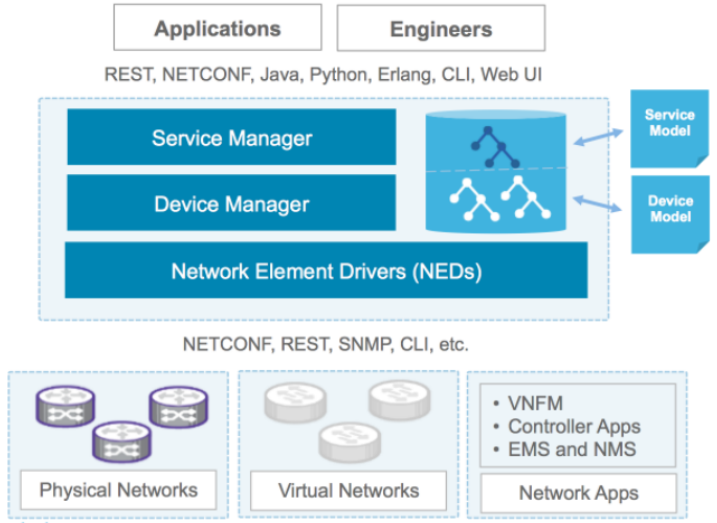
- Modular architecture provides a powerful framework that includes network-wide transactions, abstraction of the network and the ability to orchestrate services end to end.
- Two primary components of Cisco NSO:
  - NETCONF management protocol to enable standard and efficient automation of configuration management on network devices.
  - YANG to enable simple and efficient modeling of services and devices.



https://pubhub.devnetcloud.com/media/nso-doc/docs/5.8/nso_getting_started/pics/nwe_ncs.png#developer.cisco.com

1.3 NSO Components

- The Cisco NSO framework uses a modular architecture and consists of many components.
- The components are as follows:
  - Core Engine
  - Configuration Database (CDB)
  - Service Manager
  - Device Manager
  - Templates
  - Package Manager
  - Alarm Manager
  - Northbound APIs
  - NSO CLI
  - NSO WebUI
  - NSO WebUI One



https://pubhub.devnetcloud.com/media/nso-doc/docs/5.8/nso_getting_started/pics/nwe_ncs.png#developer.cisco.com

1.4 NETCONF and YANG Overview

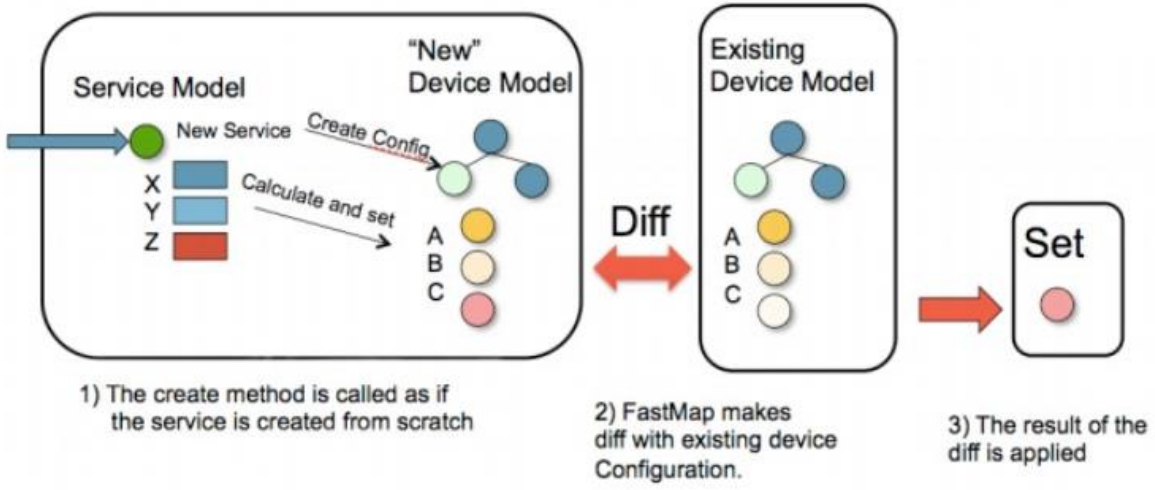
- Key NETCONF Capabilities
  - Distinction between configuration and state data
  - Distinction between data stores (candidate, running, startup)
  - Transaction based
  - Validation of configuration
  - Selective data retrieval with xpath filtering
  - Event based notifications
  - Extensible set of operations
- YANG Capabilities
  - Human-readable
  - Hierarchical data models (Container, List, Leaf etc.)
  - Reusable type definitions and groupings
  - Support augmentation
  - Supports RPCs and actions
  - Supports constraints configuration validation (must etc.)
  - Data modularity is achieved via modules and submodules
  - Versioning rules and development support

1.5 Cisco NSO Packages

- All user code that needs to run in NSO must be part of a package.
- Packages can be added, removed or upgraded.
- A package is basically a directory of files with a fixed file structure.
- Packages are used to extend NSO with custom functionality:
  - Main two components:
    - Data Models (YANG)
    - Code (Java/Python)
  - Various types of packages:
    - NED
    - Service application
    - WebUI

1.6 Cisco NSO Mapping Logic

- You need to express the mapping from a YANG service model to the corresponding device YANG model.
- You can do this by configuration templates that transform service parameters to device configuration parameters.
- You can also perform this task by programmatically (Java/Python)
- Both approaches use Cisco NSO FASTMAP



https://pubhub.devnetcloud.com/media/nso-doc/docs/5.8/nso_development/pics/fastmap_change_service.png#developer.cisco.com

1.7 Network Element Drivers

- Provides connectivity between NSO and southbound devices.
- NEDs are also NSO packages.
- NEDs are composed of two main parts:
  - Data Model
  - Code
- There can be four categories of NEDs depending on the device interface:
  - Netconf NED
  - CLI NED
  - Generic NED
  - SNMP NED

Module-2

Getting and Installing NSO

Module's Content

- Module-2: Getting and Installing NSO
  - 2.1 NSO Installation
  - 2.2 Installing NEDs
  - 2.3 Using Netsim

2.1 NSO Installation

- You can download NSO free trial version and NEDs from the link below: https://developer.cisco.com/docs/nso/#!getting-and-installing-nso/getting-nso
- You can install NSO on Linux and MAC. For Windows you can install NSO on a Linux virtual machine or in a container.
- Cisco NSO can be installed in two ways:
  - Local installation
  - System installation
- Cisco NSO installation prerequisites:
  - Java
  - Apache Ant
  - Development Tools

2.2 Installing NEDs

- For adding a device to NSO, you must have a NED for this device.
- NED installation process is as below:
  - Copy the demonstration NEDs from the installation directory.
  - Recompile if needed.
  - Reload the packages.
  - Verify NED status.

```
ls -l

# Output
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 a10-acos-cli-3.0/
drwxr-xr-x  12 user  staff   384B Nov 29 05:17 alu-sr-cli-3.4/
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 cisco-asa-cli-6.6/
drwxr-xr-x  12 user  staff   384B Nov 29 05:17 cisco-ios-cli-3.0/
drwxr-xr-x  12 user  staff   384B Nov 29 05:17 cisco-ios-cli-3.8/
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 cisco-iosxr-cli-3.0/
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 cisco-iosxr-cli-3.5/
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 cisco-nx-cli-3.0/
drwxr-xr-x  13 user  staff   416B Nov 29 05:17 dell-ftos-cli-3.0/
drwxr-xr-x  10 user  staff   320B Nov 29 05:17 juniper-junos-nc-3.0/
```

https://developer.cisco.com/docs/nso/#!getting-and-installing-nso/exploring-the-installation

2.3 Using Netsim

- You can simulate the devices by using Netsim.
- Netsim simulates the management plane of devices by using YANG modules provided by NEDs.
- The ncs-netsim simulation tool is used to simulate devices.
- Detailed information can be found on the link: https://developer.cisco.com/docs/nso/guides/#!the-network-simulator

```
admin$ ncs-netsim --help
Usage ncs-netsim  [--dir <NetsimDir>]
            create-network <NcsPackage> <NumDevices> <Prefix> |
            create-device <NcsPackage> <DeviceName> |
            add-to-network <NcsPackage> <NumDevices> <Prefix> |
            add-device <NcsPackage> <DeviceName> |
            delete-network                    |
            [-a | --async]  start [devname]   |
            [-a | --async ] stop [devname]    |
            [-a | --async ] reset [devname]   |
            [-a | --async ] restart [devname] |
            list                      |
            is-alive [devname]        |
            status [devname]          |
            whichdir                  |
            ncs-xml-init [devname]    |
            ncs-xml-init-remote <RemoteNodeName> [devname] |
            [--force-generic]         |
            packages                  |
            netconf-console devname [XpathFilter] |
            [-w | --window] [cli | cli-c | cli-i] devname
```

Thank You !!!

Module-3

Creating YANG Models

Module's Content

- Module-3: Creating YANG Modules
  - 3.1 YANG Module Structure
  - 3.2 YANG Statements
  - 3.3 YANG Data Types
  - 3.4 XPath Overview

3.1 YANG Module Structure

- Every YANG module must include these segments:
  - Module name
  - Header Information
    - Namespace (mandatory)
    - Prefix (mandatory)
    - Organization, contact (optional)
    - Description (optional)
  - Revision information
  - Imports and includes
  - Type definitions
  - Reusable node declarations
  - Configuration and operational data declarations
  - Action (RPC) and notification declarations
- An example module can be found:
  - https://datatracker.ietf.org/doc/html/rfc9182

3.2 YANG Statements

- container: A YANG container can be used when you have a collection of information elements that belong together.
- list: A YANG list works like a container but with a list you can have one or more instances of what is inside in the list.
- leaf: A YANG leaf is single value of a specific type. Each leaf has a type statement inside that defines the type of data that leaf can hold.
- leaf-List: A YANG leaf-list statement represents a list of values (zero or more)

```
list bd-subnet {
    tailf:info "Specify Bridge Domain subnet address & scope";
    key address;

    1 reference
    leaf address {
        tailf:info "A.B.C.D/L;;Prefix or X:X::X/L;;Prefix for L3 Internal Subnet";
        type tailf:ip-address-and-prefix-length;
    }

    leaf preferred {
        tailf:info "Specify preferred value [no]";
        type enumeration {
            enum yes;
            enum no;
        }
        default no;
    }
}
```

```
container l2vni {
    tailf:info "Layer 2 VNI Configuration Parameters";

    leaf vlan-id {
        tailf:info "Vlan id is allocated by resource-manager, use this field to overwrite the default behavior.";
        type vlan;
    }

    leaf vni-id {
        tailf:info "Vni id is allocated by resource-manager, use this field to overwrite the default behavior.";
        type vxlan-id;
    }
}
```

3.3 YANG Data Types

- Data types can be used in YANG are:
  - Built-in data types
  - RFC-defined data types (https://datatracker.ietf.org/doc/html/rfc6020#section-4.2.4)
  - Custom data types
- Derived Types
  - Derived data types can be created with the use of typedef.
  - Derived data types can be restricted range, length, pattern, fraction digits.
- Common YANG Data Types – RFC 6991
  - IETF YANG data types
  - IETF Inet data types

```
import ietf-yang-types {
   prefix yang;
}
```

3.4 XPath Overview

- XPath uses expressions to extract or reference parts of XML documents.
- YANG uses XPath version 1.0.
- XPath is often used with Leafrefs and with when and must conditional statements.
- XPath functions that can be used for example in "must" expressions:
  - **node-set** deref(node-set)
  - **bool** re-match(string, string)
  - **number** string-compare(string, string)
  - **number** min(node-set)
  - **number** max(node-set)
  - **number** avg(node-set)
  - **number** band(number, number)
  - **number** bor(number, number)
  - **number** bnot(number)
  - **node-set** sort-by(node-set, string)

Thank You !!!

Module-4

Services

Module’s Content

- Module-4: Services
  - 4.1 Package Architecture
  - 4.2 Create a Service Package

4.1 Package Architecture

- All user code must be a part of a package. A package is a structured directory with files.
- A package contains code, YANG modules and so on, that are necessary in order to add an application or function to NSO.
- The simplest and fastest way of creating a service package by using templates without any additional Java or Python code.

```
dns-config/
+-- package-meta-data.xml
+-- python
|   '-- dns_config
|       +-- __init__.py
|       '-- main.py
+-- README
+-- src
|   +-- Makefile
|   '-- yang
|       '-- dns-config.yang
+-- templates
'-- test
    +-- < ... output omitted ... >
```

4.2 Creating a Service Package

- The package creation process can be summarized as follows:
  - Create a service skeleton
  - Create a service implementation from NSO CLI
  - Create a service template
  - Create the YANG service data model
  - Compile the service package

```
mustafa@ubuntu:~/Desktop/nso/ncs-run$ ncs-make-package --help
Usage: ncs-make-package [options] package-name

  ncs-make-package --netconf-ned DIR package-name
  ncs-make-package --lsa-netconf-ned DIR package-name
  ncs-make-package --generic-ned-skeleton package-name
  ncs-make-package --snmp-ned DIR package-name
  ncs-make-package --service-skeleton TYPE package-name
  ncs-make-package --data-provider-skeleton package-name
  ncs-make-package --erlang-skeleton package-name

      where TYPE is one of:
          java                  Java based service
          java-and-template     Java service with template
          python                Python based service
          python-and-template   Python service with template
          template              Template service (no code)
```

Thank You !!!

Module-5

Model-to-Model Mapping

Module's Content

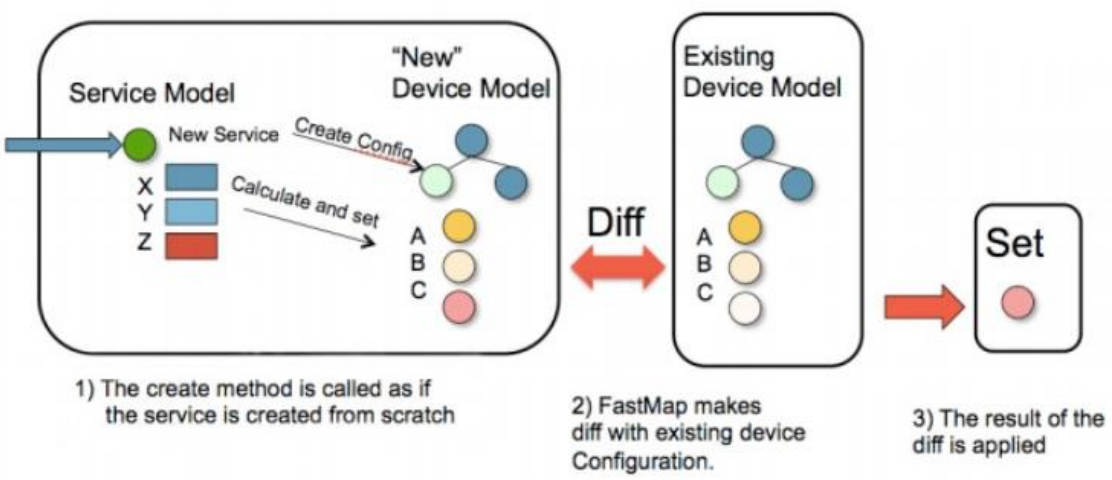
- Module-5: Model-to-Model Mapping
  - 5.1 Service Parameters Mapping
  - 5.2 Fastmap
  - 5.3 Templates

5.1 Service Parameters Mapping

- Cisco NSO uses the IETF standardized YANG (RFC 6020) both for service models and device models.
- The mapping is done for service yang model to device yang model.
- Cisco NSO always calculates the minimum difference toward the network.
- NSO can look at the desired device configuration and calculate the differences versus the current configuration.
- The input parameters from the service instance are glued to configuration templates by mapping logic.
- When the configuration template is ready, it is handed over to the FASTMAP algorithm.

5.2 FASTMAP

- Fastmap provides the complete service life-cycle: create, update and delete of the service.
- Fastmap is based on creation of the service. At the service instance creation time, NSO also stores the reverse of the device configuration with the service instance.
- Anytime when the service is updated, NSO first applies the reverse diff of the service (stores with the service instance). Then it runs the create method again and executes the minimum diff that needs to send the devices.



https://pubhub.devnetcloud.com/media/nso-doc/docs/5.8/nso_development/pics/fastmap_change_service.png#developer.cisco.com

5.3 Templates

- Two types of templates exist, device-templates and config-templates.
- Device template is invoked as an action but config template is invoked with a service instance.
- Config template terminology is as follows:
  - Config template
  - Service template
  - Feature template
- Device templates are created via CLI and stored in are stored in the configuration.
- Config templates are stored in templates directory of packages and are loaded when NSO starts.
- Template processing is done by mapping logic.
- https://developer.cisco.com/docs/nso/guides/#!templates/templates

Thank You !!!

Module-6

Python in Cisco NSO

Module's Content

- Module-6: Python in Cisco NSO
  - 6.1 Cisco NSO Programmability
  - 6.2 Python Scripting
  - 6.3 Python Service Skeleton

6.1 Cisco NSO Programmability

- Programming languages can be used to extend service packages. Python and Java are the most widely used programming languages in Cisco NSO.
- Two options for using Python to interact with NSO:
  - Python scripting
  - Python VM
- We need programming for making complex logic or connecting to external data sources. Python, Java and Erlang API's are provided by Cisco NSO.(https://developer.cisco.com/docs/nso/api/)
- Two options when using programmability:
  - All mappings are done in code.
  - Static mappings are done in XML templates and dynamic mappings are done in code. This is the preferred approach.

6.2 Python Scripting

- Python scripting is useful for one-time automation tasks.
- PYTHONPATH environment varible, can access the NSO Python modules by sourcing the ncsrc file in the installation directory.
- High-level MAAPI API:
  - Provides an easy to use interface for accessing NSO.
- Maagic API:
  - It is used on top of MAAPI high-level API.
  - It helps navigating in the CDB, using the standard Python object dot notation.
  - Any special characters are replaced with underscores.
- https://developer.cisco.com/docs/nso/guides/#!python-api-overview/python-api-overview

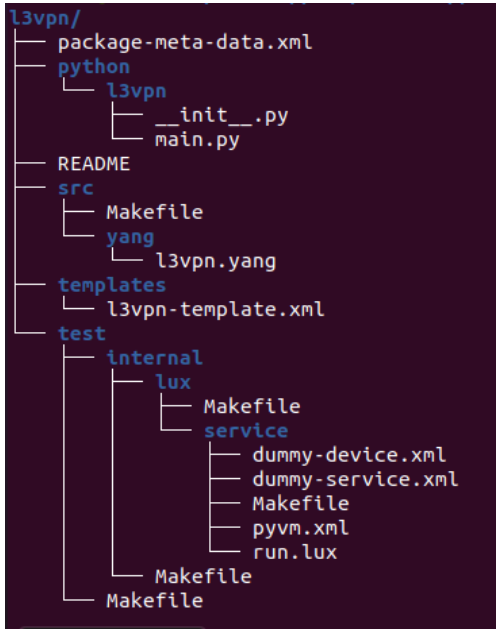
6.3 Python Service Skeleton

- ncs-make-package command line tool is used to create a python and template package.
- Python files are stored in *packages/package-name/python* directory.
- Python versions 3.4 and later are supported python versions.

```
mustafa@ubuntu:~/Desktop/nso/ncs-run$ ncs-make-package --help
Usage: ncs-make-package [options] package-name

  ncs-make-package --netconf-ned DIR package-name
  ncs-make-package --lsa-netconf-ned DIR package-name
  ncs-make-package --generic-ned-skeleton package-name
  ncs-make-package --snmp-ned DIR package-name
  ncs-make-package --service-skeleton TYPE package-name
  ncs-make-package --data-provider-skeleton package-name
  ncs-make-package --erlang-skeleton package-name

      where TYPE is one of:
          java                  Java based service
          java-and-template     Java service with template
          python                Python based service
          python-and-template   Python service with template
          template              Template service (no code)
```



Thank You !!!

Module-7

Cisco NSO Restconf API

Module's Content

- Module-7: Cisco NSO Restconf API
  - 7.1 Restconf API
  - 7.2 Methods & Response Codes
  - 7.3 Query Parameters

7.1 The RESTCONF API

- Cisco NSO supports RESTCONF API over HTTP or HTTPS. The REST interface is deprecated and removed in the recently versions.
- Request and response data can be in XML or JSON format. It can be negotiated in the HTTP header. (application/yang-data+xml or application/yang-data+json)
- To work with RESTCONF API in NSO, RESTCONF must be enabled in the **ncs.conf** configuration file.
- Methods are used with RESTCONF:
  - Use GET to retrieve a resource.
  - Use POST to create a resource.
  - Use PUT to replace a resource.
  - Use PATCH to merge a resource.
  - Use DELETE to delete a resource.
- https://developer.cisco.com/docs/nso/guides/#!the-restconf-api

```
<restconf>
  <enabled>true</enabled>
</restconf>


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

7.2 Methods & Response Codes

- **POST** method is sent by the client to create a resource or invoke an action. (Create)
- **GET** method is sent by the client to retrieve data resource. It is not supported for operation resources. (Read)
- **PUT** method is sent by the client to create or replace a data resource. (Create, Update)
- **PATCH** method is used to create or update a subresource in the target resource. But if the target resource does not exist, it does not create it. (Update)
- **DELETE** method is used to delete the target resource. (Delete)
- Status Codes:
  - 1xx - Informational
  - 2xx – Success (200 OK, 201 Created, 204 No Content)
  - 3xx - Redirection
  - 4xx - Client Error (400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 405 Method Not Allowed)
  - 5xx - Server Error (500 Internal Server Error, 501 Not Implemented, 503 Service Unavailable)

7.3 Query Parameters

- Zero or more query parameters are allowed to be present in the request URI.
- Query parameters can be given in any order but can appear at most once.
- The *content* Query Parameter:
  - Controls if configuration, non-configuration or both types of data should be returned.
- The *depth* Query Parameter:
  - Limit the depth of subtrees returned by the server.
- The *fields* Query Parameter:
  - Retrieve a subset of all nodes in a resource.
- Additional Query Parameters:
  - dry-run, no-networking, reconcile, rollback-id

Thank You !!!