Multi-domain Service Orchestration

with Cisco Network Service Orchestrator

Gabor Szabo
gabszabo@cisco.com

Version 1.2

Agenda

- Theory & Concepts
- Cisco NSO inroduction
- Demonstration: simple service
- Use-Cases

Statement

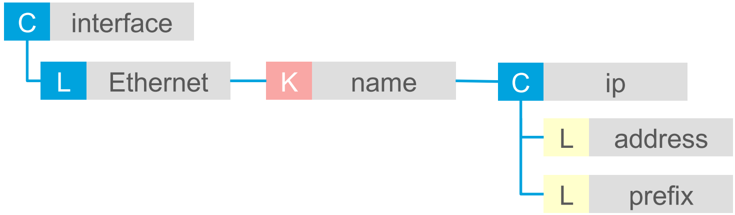
- After a 3-day training
- **Every KIFÜ network engineer can develop and deploy services**
- In multi-vendor environment
- In a multi-domain network
- Within two week (max)

Service Automation: Introduction to Theory

Everything is Model Based

- Network Devices Configuration
  - Routers, Switches, Load-Balancers, etc.
- Services Configuration
  - VPN, Routing, etc.
- System Configuration
  - Users, Groups, Permissions, etc.

```
Router# show running-config
…
…
interface Ethernet1/1
  ip address 192.168.1.1/24
interface Ethernet1/2
  ip address 192.168.2.1/24
```



C interface

L Ethernet

K name

C ip

L address

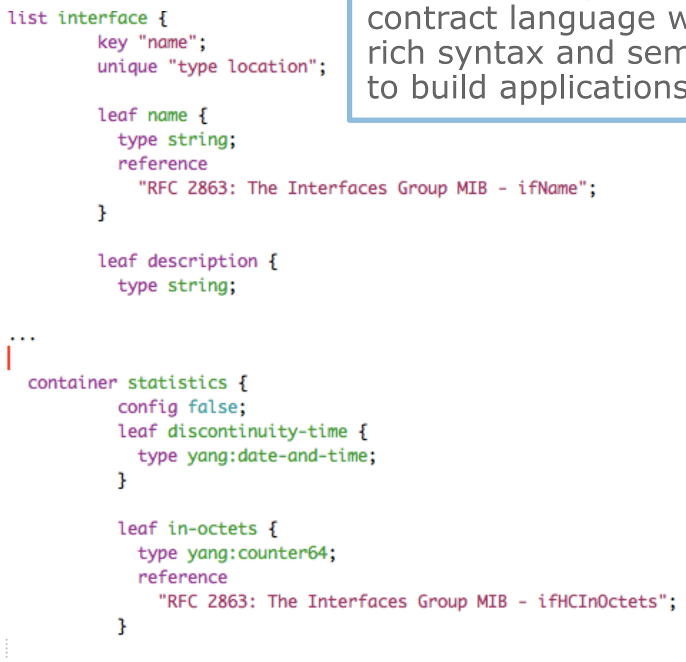
L prefix

YANG – A Data Modeling Language for Networking

- Human readable, and easy to learn representation
- Hierarchical configuration data models
- Reusable types and groupings (structured types)
- Extensibility through augmentation mechanisms
- Supports definition of operations (RPCs)
- Formal constraints for configuration validation
- Data modularity through modules and sub-modules
- Well defined versioning rules

Why you should care:

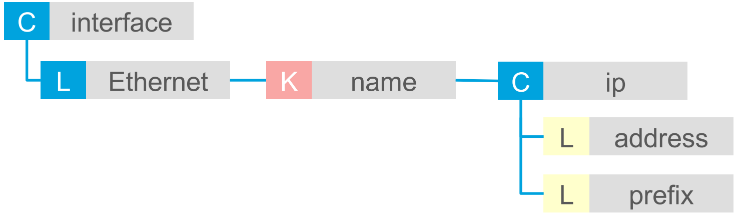
YANG is a full, formal contract language with rich syntax and semantics to build applications on

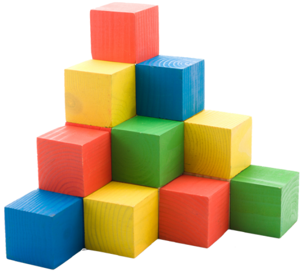


```
list interface {
         key "name";
         unique "type location";

         leaf name {
           type string;
           reference
             "RFC 2863: The Interfaces Group MIB - ifName";
         }

         leaf description {
           type string;

...

  container statistics {
           config false;
           leaf discontinuity-time {
             type yang:date-and-time;
           }

           leaf in-octets {
             type yang:counter64;
             reference
               "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
           }
```

YANG Building Blocks

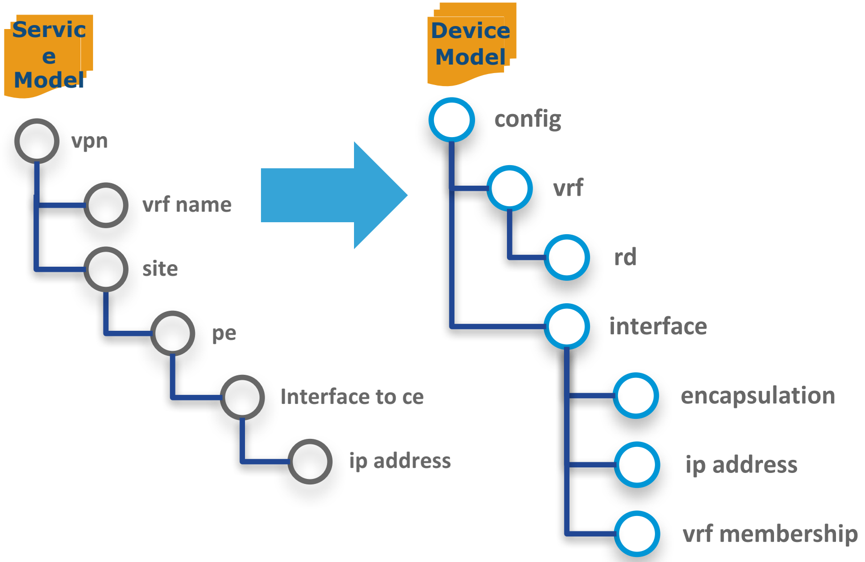
- Leaf L address
  - A node in the data tree
  - Assigned with a datatype and has a value
  - Has no child nodes
- Container C ip
  - Does not have a value
  - Holds one or more child nodes in the data tree
- List L Ethernet — K name
  - Has a key node (leaf node) which serves as a UID.
  - Groups multiple similar elements
  - Each element may consist of multiple nodes of various types.





Service Abstraction and the NSO Magic

- NSO enables creating service-aware applications, e.g. VPN service
- Service attributes stored in service data model and used to configure multi-vendor devices
- Mapping logic is needed to map service models to device models
  - XML template and/or Java/Python code
  - All service- and device-specific information are stored in data models and mapping logic
  - Automation core engine is not aware of technology, vendor or service
- Development needed for service "create" only
  - Modification and decommission created automatically



Device Configuration Consistency

- Configuration is protected by a transaction
- Service instantiation / modification / decommission is treated as an atomic action.
  - All-or-Nothing approach.
  - Implemented all-at-once.

NETCONF – A Protocol to Manipulate Configuration

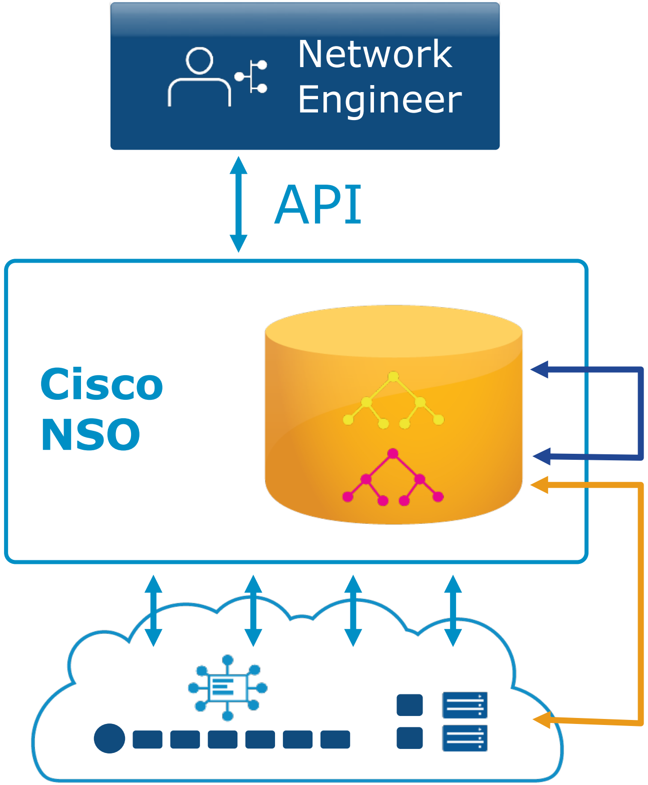
- IETF network management protocol
- Distinction between configuration and state data
- Multiple configuration data stores (candidate, running, startup)
- Configuration change validations
- Configuration change transactions
- Selective data retrieval with filtering
- Streaming and playback of event notifications
- Extensible remote procedure call mechanism

Why you should care:

NETCONF provides the fundamental programming features for comfortable and robust automation of network services

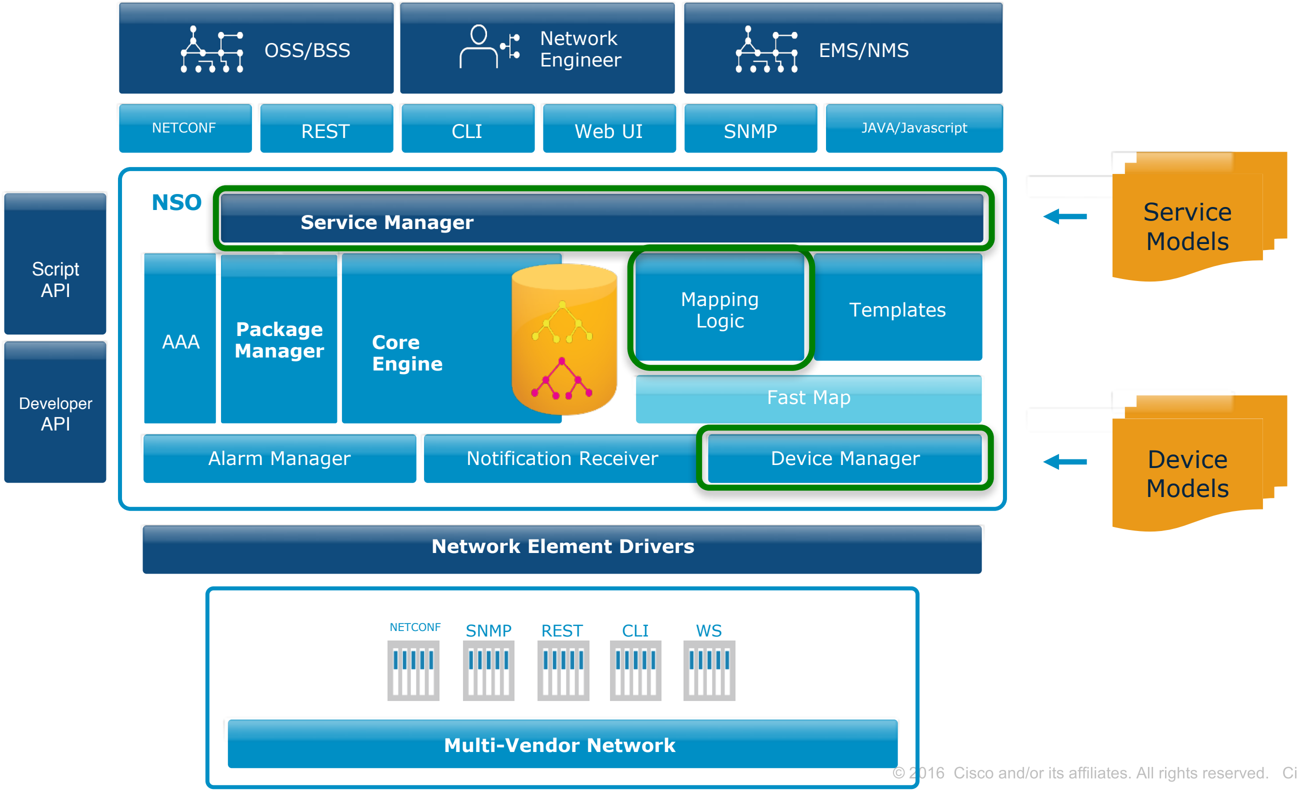
Introducing Network Programmability

- Our challenge?
  - Multi-Vendor Networks
  - Multiple protocols – CLI, Netconf, etc.
- Network Element Drivers handle device communication based on device OS
- Pluggable Custom Service Models
- Instantly available APIs



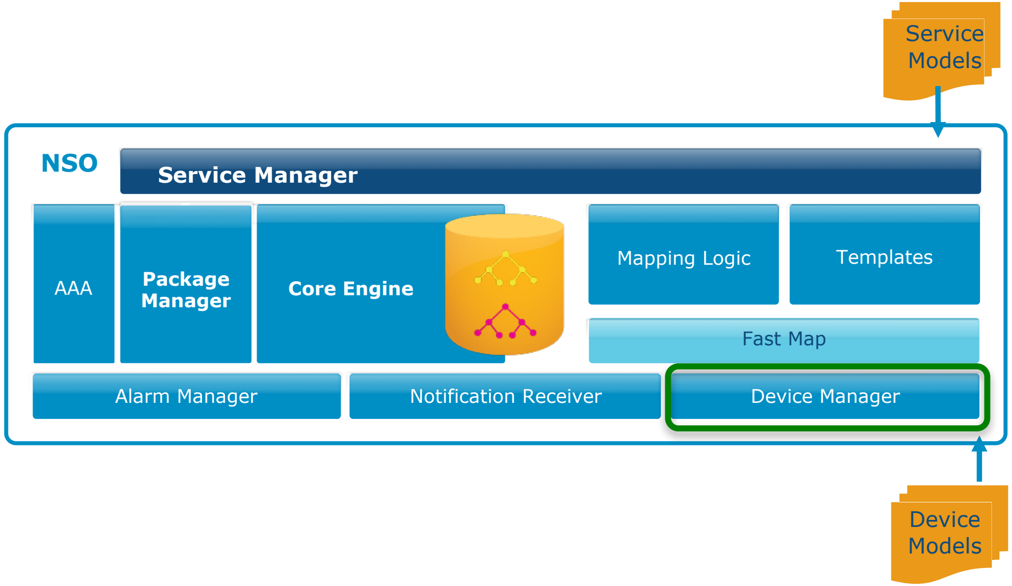
Cisco NSO Introduction

NSO Logical Architecture



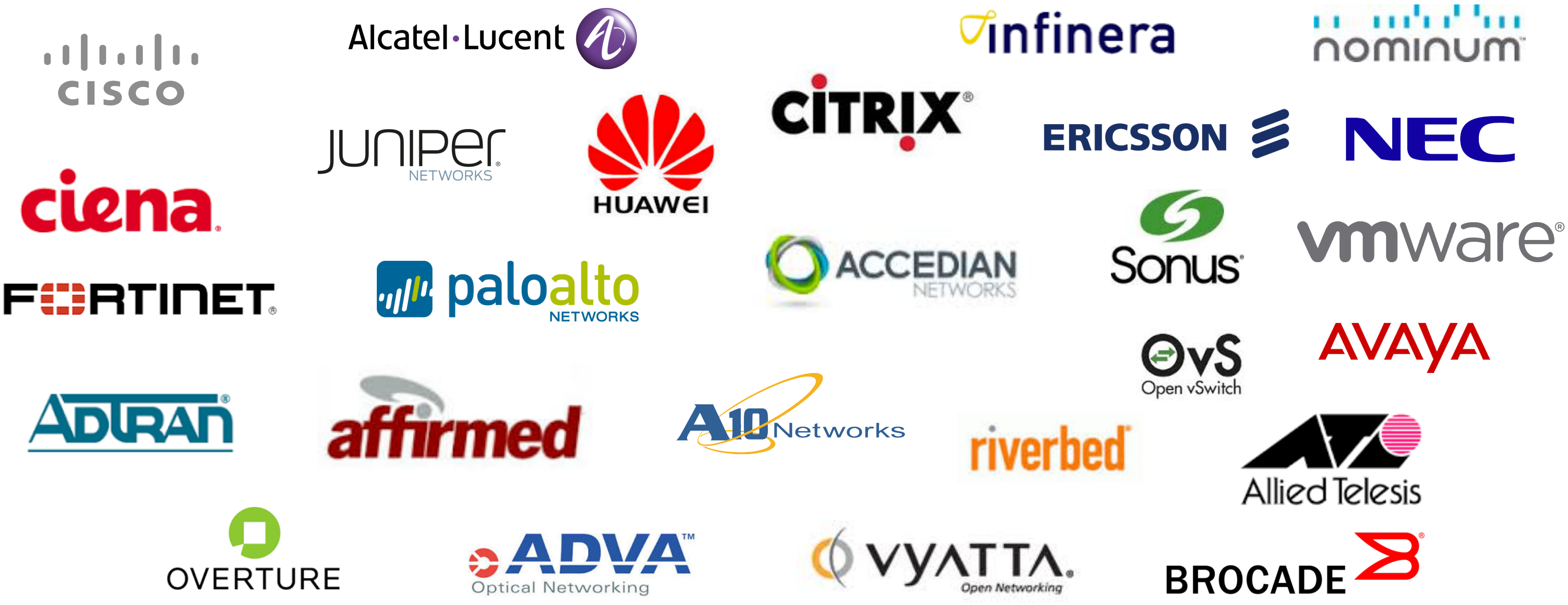
Device Manager

- Transactions and rollbacks
- Configuration synch both-ways
- Configuration validation
- Device Configuration database
  - Stores the configuration model
  - Raw configuration is NOT stored
  - Proprietary, not relational
  - Can be accessed by an API
- Talks to devices via
  Network Element Drivers (NEDs)

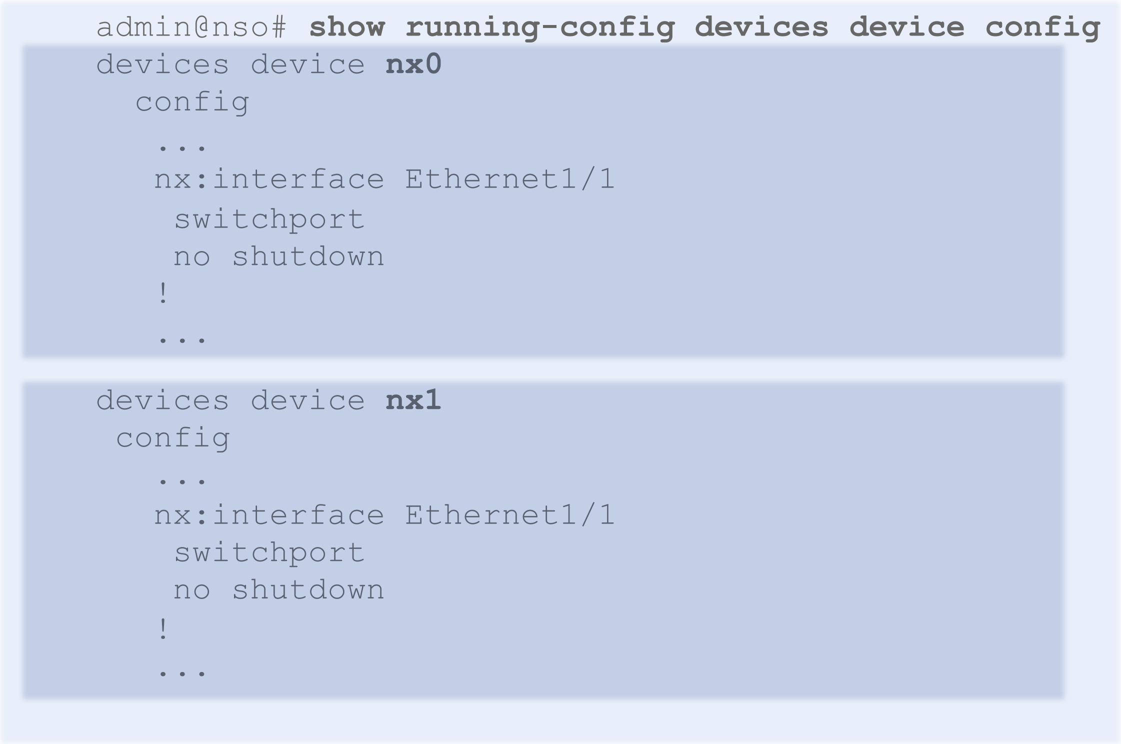


NEDs - Multi-Vendor Support

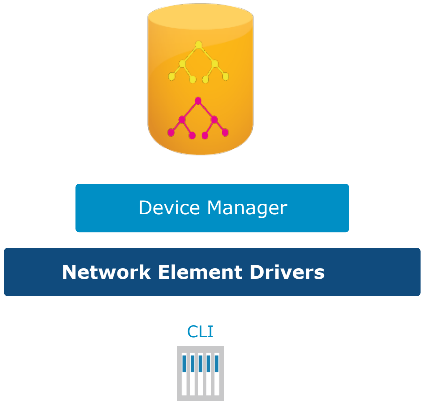
More than 65 and growing fast!



Entire Devices Configuration in a single Show!

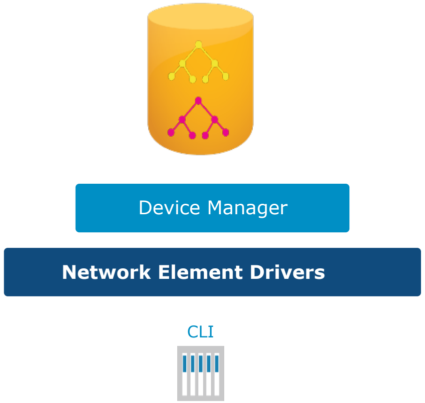


```
admin@nso# show running-config devices device config
devices device nx0
  config
   ...
   nx:interface Ethernet1/1
    switchport
    no shutdown
   !
   ...

devices device nx1
 config
   ...
   nx:interface Ethernet1/1
    switchport
    no shutdown
   !
   ...
```



Example: Verifying Consistent Configuration

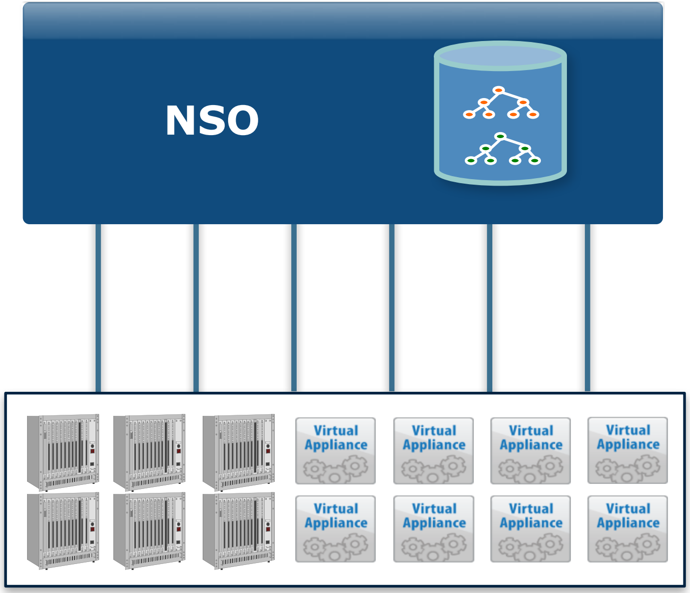
```
gabszabo@ncs# show running-config devices device config ios:vrf definition NAT-VPN rd
devices device 7604-1
 config
  ios:vrf definition NAT-VPN
   rd 10000:201
  !
 !
!
devices device 7604-2
 config
  ios:vrf definition NAT-VPN
   rd 10000:201
  !
 !
!

(...)

devices device budlab-asr1k
 config
  ios:vrf definition NAT-VPN
   rd 10000:201
  !
 !
!
```



Device Manager

Network Element Drivers

NSO Main Features



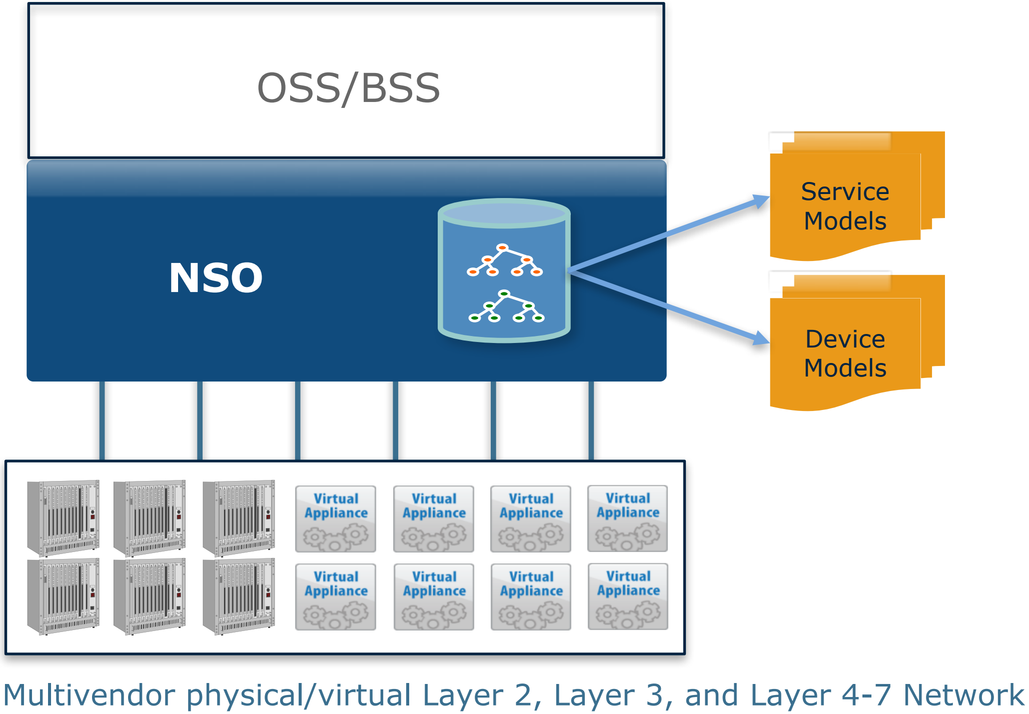
Multivendor physical/virtual Layer 2, Layer 3, and Layer 4-7 Network

NSO Main Features

- Model-based architecture
- Transactional guarantees
- In-memory storage of configuration states for all services and all devices
- FastMap* algorithm for service-layer CRUD operations
- Reactive FastMap*

* Patent No.: US 8,533,303 B2

NSO #1: Model-Based Architecture



Multivendor physical/virtual Layer 2, Layer 3, and Layer 4-7 Network

YANG data models for:

- Network services
- Network topology
- Network devices

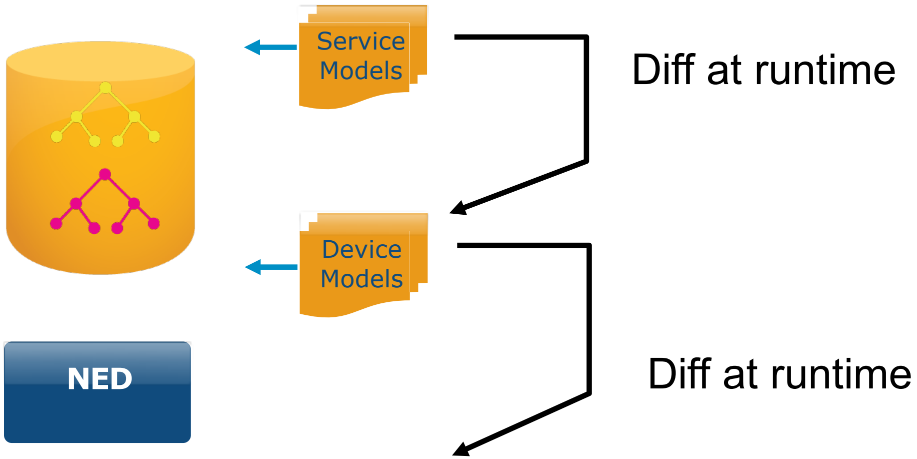
YANG data models drive:

- Northbound APIs
- User interfaces
- Southbound command sequences

Benefits:

- Can be used for all types of services and all types of networks

NSO #1: Model-Based Architecture



In contrast with hard-coded CLI templates

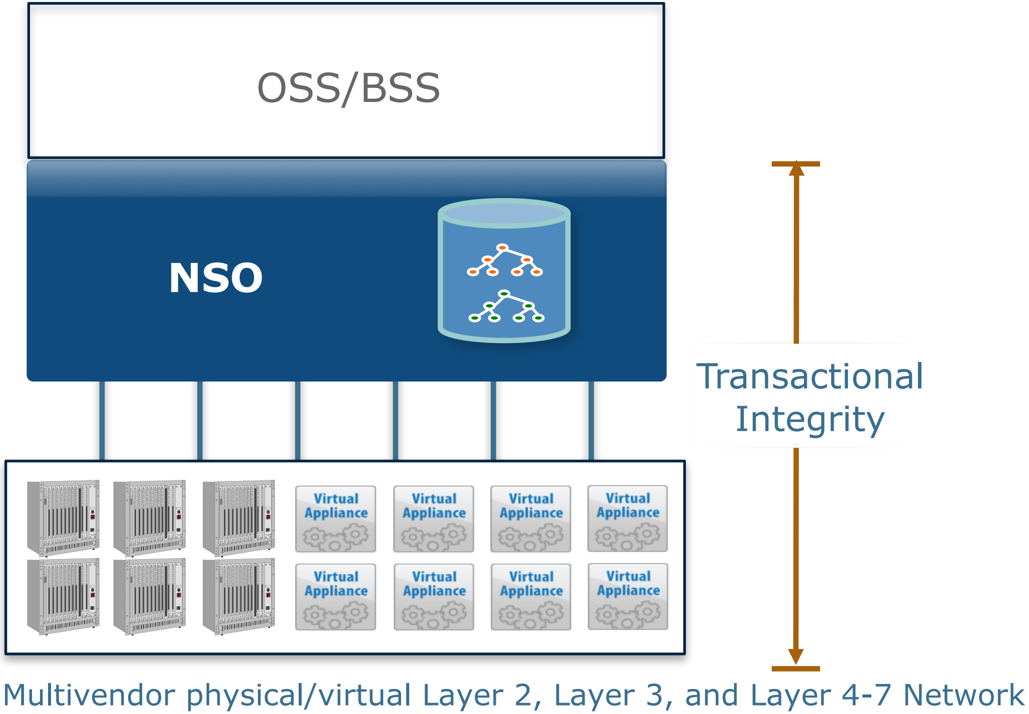
Run-time rendering

No hard-coded templates

NSO knows the actual device configuration

Provision only the difference

NSO #2: Transactional Guarantees


OSS/BSS
NSO
Virtual Appliance
Transactional Integrity
Multivendor physical/virtual Layer 2, Layer 3, and Layer 4-7 Network

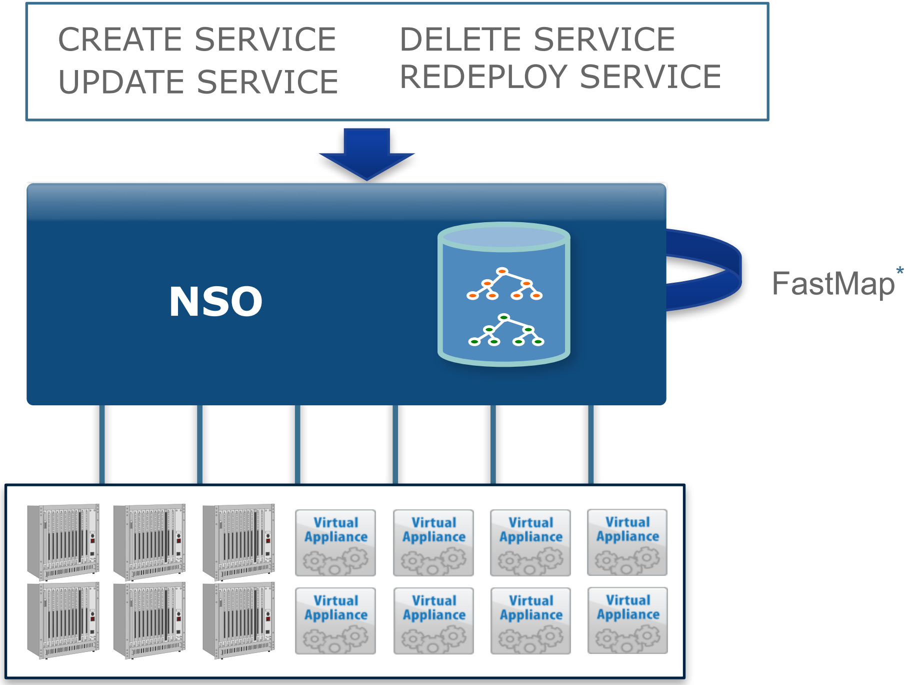
Transactional guarantees:

- Help ensure fail-safe operations (automated handling of exceptions)
- Keep accurate copy of network configuration state in NSO at all times

Benefits:

- Automation can be based on accurate real-time view of service and network state
- Much higher degree of automation possible

NSO #3: FastMap* Algorithm



CREATE SERVICE
UPDATE SERVICE
DELETE SERVICE
REDEPLOY SERVICE

Multivendor physical/virtual Layer 2, Layer 3, and Layer 4-7 Network

FastMap:

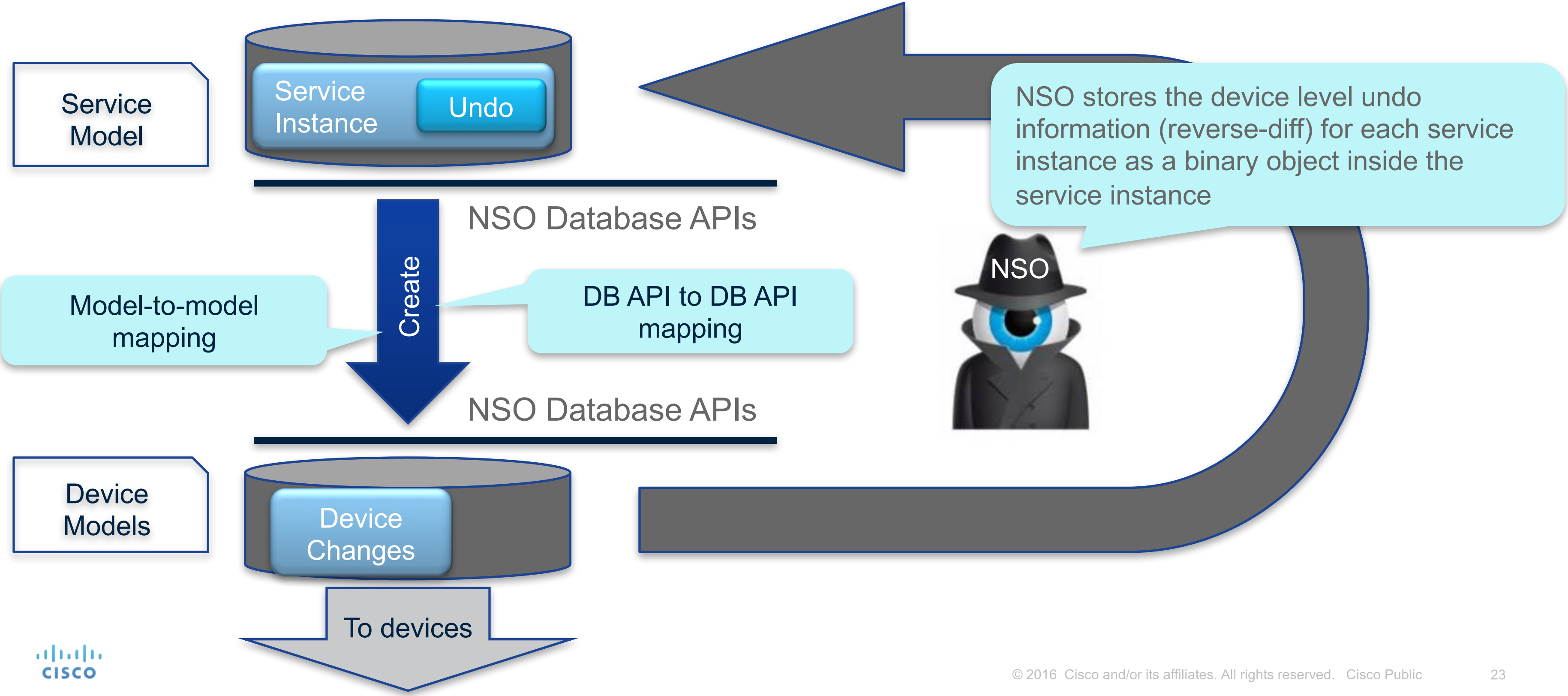
- Only the CREATE operation needs to be specified
- UPDATE, DELETE and REDEPLOY operations are automatically generated and compute minimal change set needed

Benefits:

- Reduces service implementation code by two orders of magnitude
- Supports modifications of services at runtime

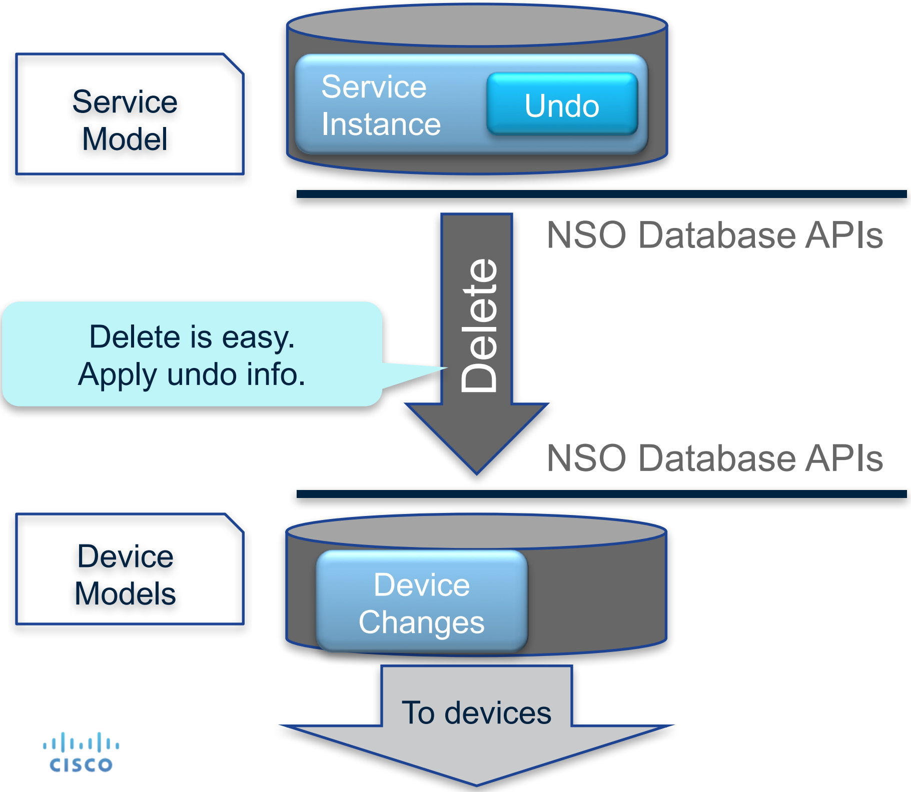
* Patent No.: US 8,533,303 B2

FastMAP: Spying on the CREATE Method



NSO stores the device level undo information (reverse-diff) for each service instance as a binary object inside the service instance

FastMAP: DELETE is Easy



Service Model

Service Instance

Undo

NSO Database APIs

Delete

Delete is easy.
Apply undo info.

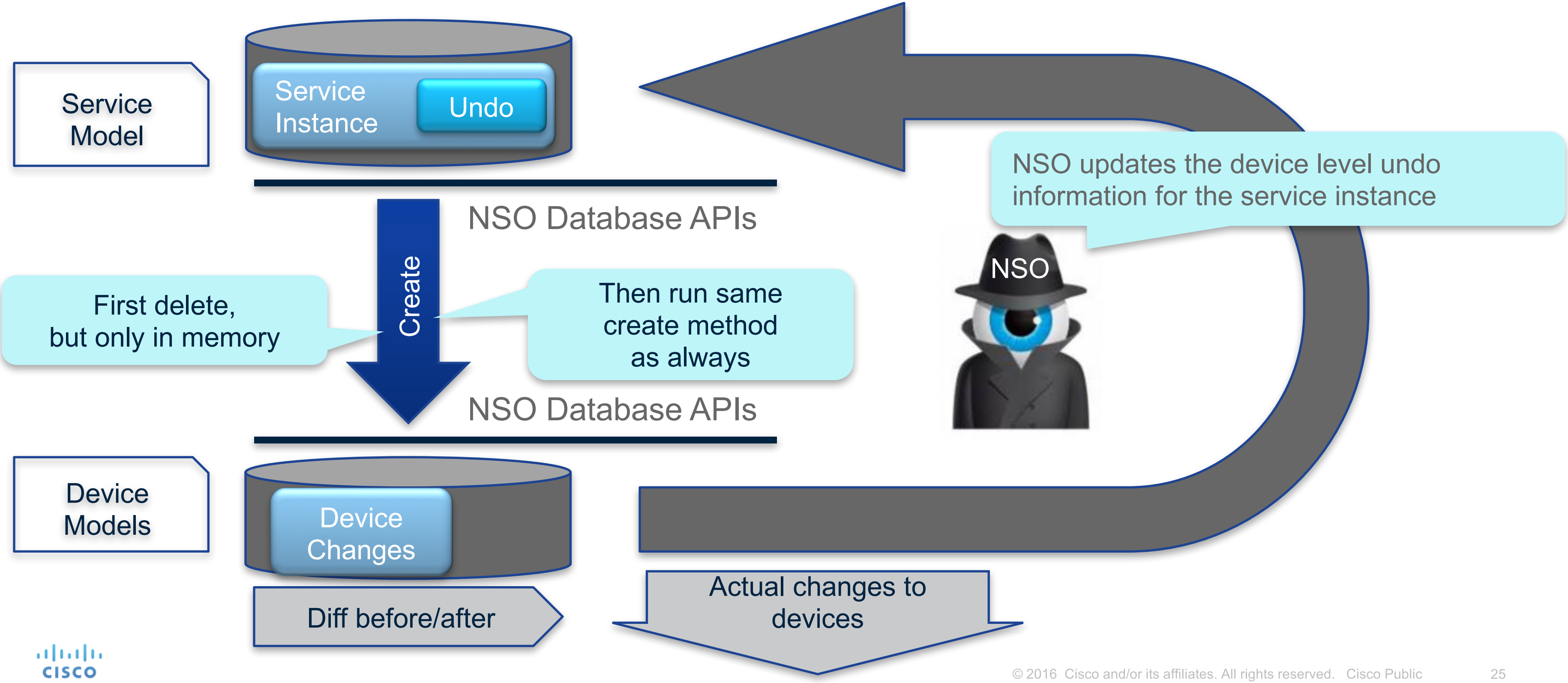
NSO Database APIs

Device Models

Device Changes

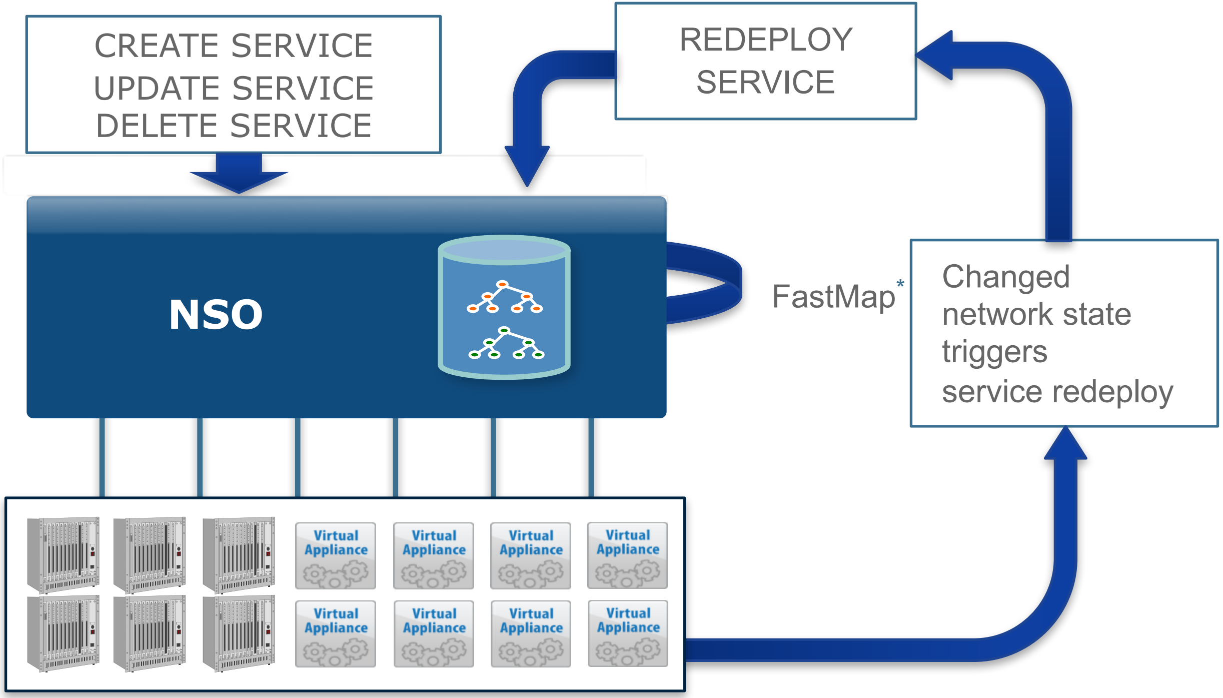
To devices

UPDATE is Delete & Create – In Memory



Service Model
Service Instance
Undo
NSO Database APIs
Create
First delete, but only in memory
Then run same create method as always
NSO Database APIs
Device Models
Device Changes
Diff before/after
Actual changes to devices
NSO
NSO updates the device level undo information for the service instance

NSO Main Feature 4: Reactive FastMap*



CREATE SERVICE
UPDATE SERVICE
DELETE SERVICE

REDEPLOY SERVICE

FastMap*

Changed network state triggers service redeploy

Multivendor physical/virtual Layer 2, Layer 3, and Layer 4-7 Network

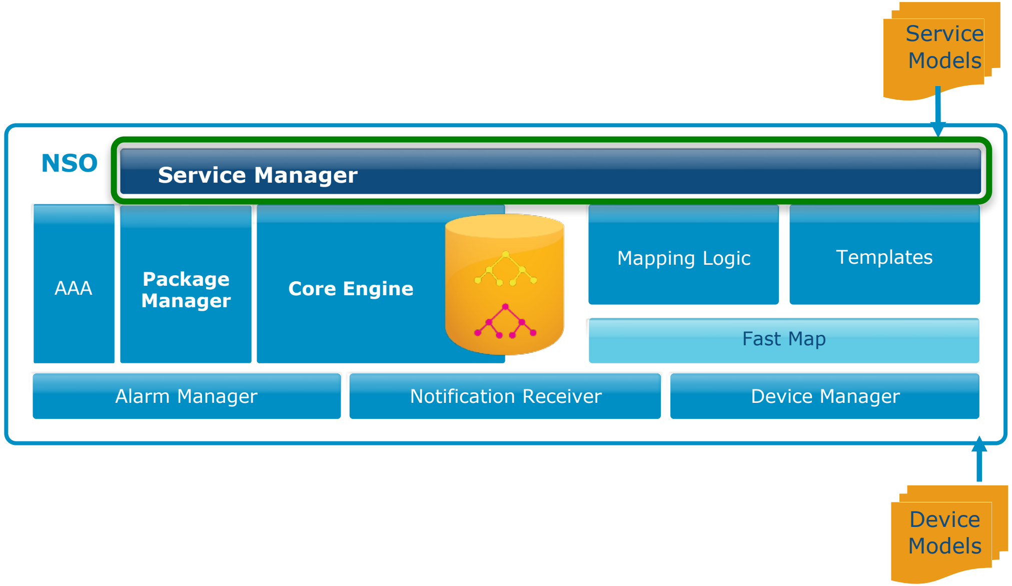
Benefits:
One algorithm supporting:

- Provisioning
- Orchestration
- Elasticity
- Virtual machine and VNF mobility
- Self-healing network

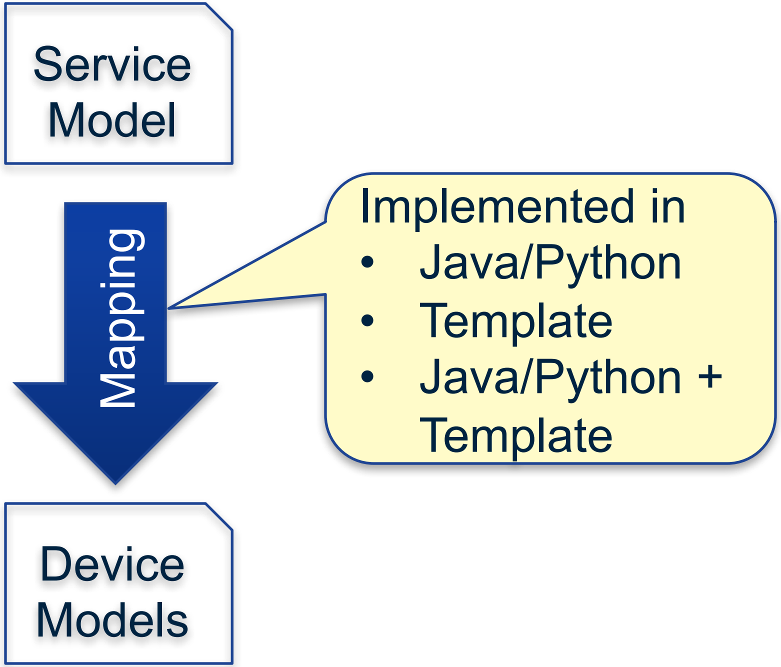
* Patent No.: US 8,533,303 B2

Service Manager

- Service modeling
- Mapping to device model
- Service activation
- Service modification
- Service decommissioning



Alternate Mapping Approaches



Implemented in

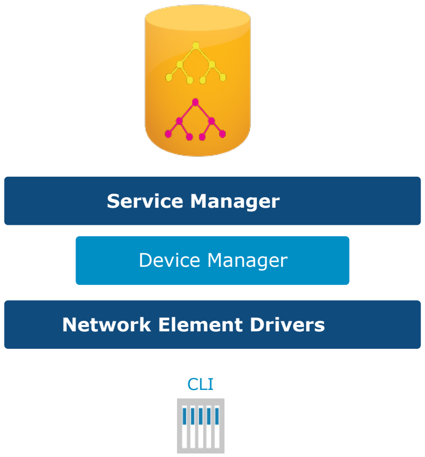
- Java/Python
- Template
- Java/Python + Template

Implementation alternatives

- Java/Python only
  Most expressive power, but also most work
  Make calls to external applications
  Execute complex algorithms
- Template only
  Only simple mappings
  Implemented in minutes (e.g. in CLI)
- Java/Python instantiating template
  Do the complex computations in Java/Python
  Apply the bulk of the settings in template
  Java/Python exports variables to the template

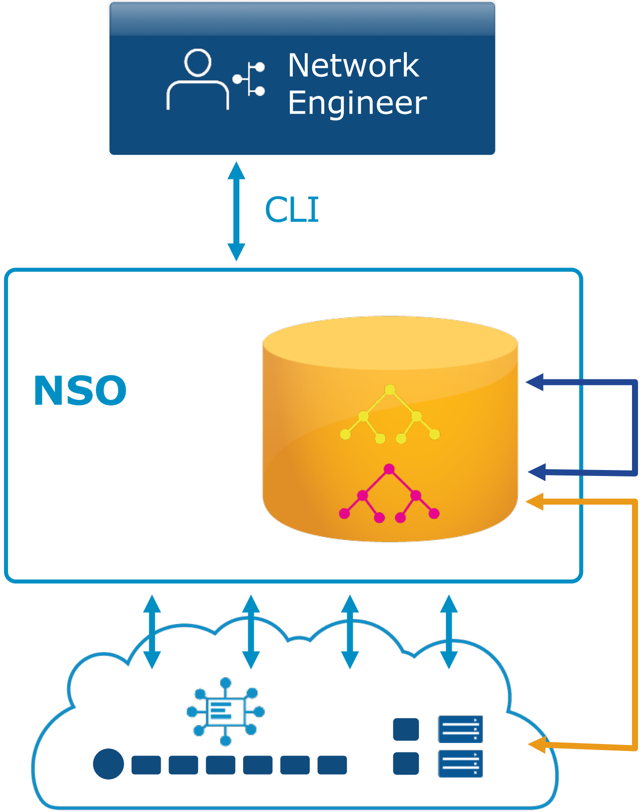
Entire Services Configuration in a single Show!

```
admin@nso# show running-config services
services fabricpath DC01
  spine dc01spine1
   switch-id 105
   ...
  spine dc01spine2
   switch-id 106
   ...

services vpc nx1-nx2
  devices nx1 nx2
   peer-gateway
   ...
services vpc nx3-nx4
  devices nx3 nx4
   peer-gateway
   ...
```



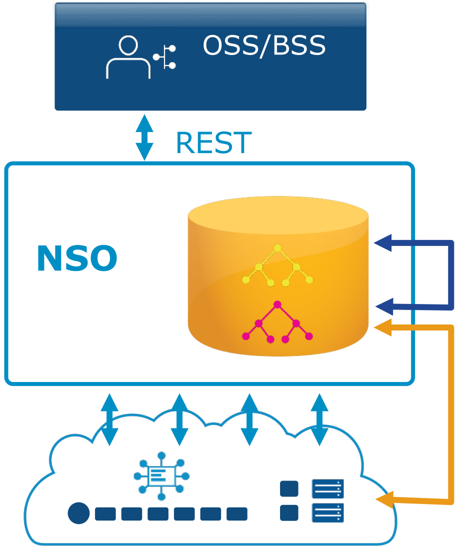
NSO CLI

- Service-aware
- Network-wide
- Juniper / Cisco XR style
- Powertool
- Helps keep the current domain experts
- Rich editing with tab-completion for commands, static elements and dynamic instances
- History, hints, help
- Extensible with custom/external commands, wizards



NSO REST

- Relies on verbs of transport layer:
- HTTP 1.1
  - GET : get resources
  - PUT : replace existing resource
  - POST : create resource
  - DELETE : delete resource
  - PATCH (RFC5789) : modify existing resource
  - HEAD, OPTIONS
- Stateless, client-server
- Hyperlinked, just like the web
- XML or JSON as data containers
- Links to available data-stores and operations



```
$curl –u admin:admin –s http://localhost:8008/api
```

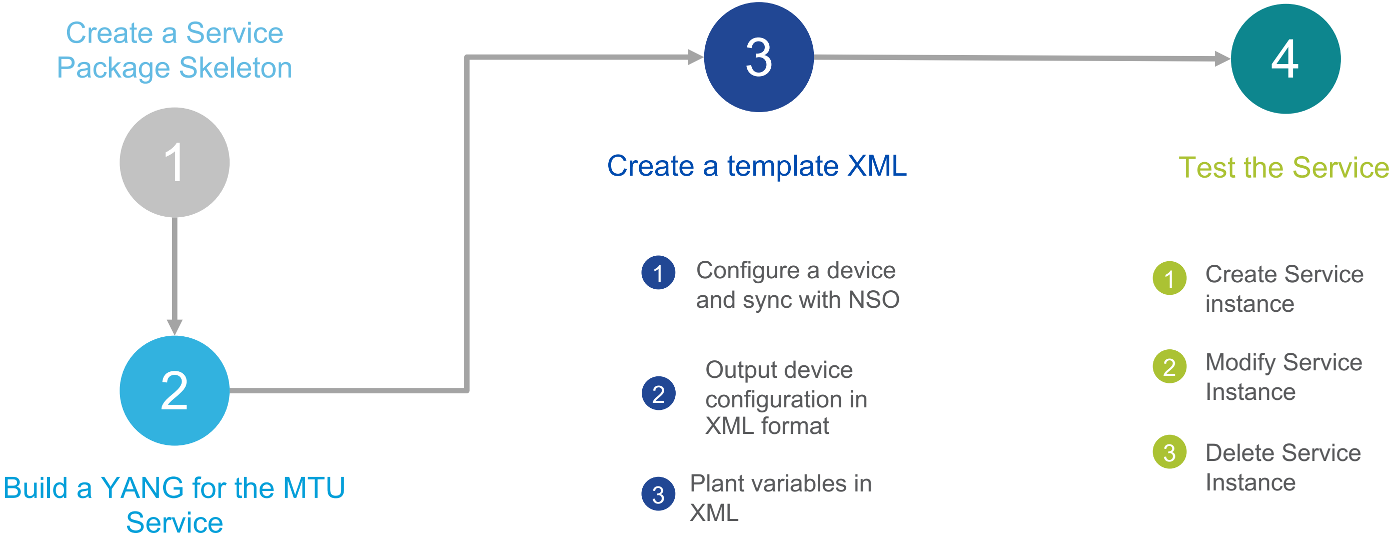
- /api/running
- /api/candidate
- /api/operations
- /api/operational
- /api/rollback

Demonstration:
Interface MTU Service

Contents of a Service Package

- Service Model
  - YANG!
- Mapping Logic
  - Java, Python, XML
  - How service parameters map to device configuration

Creating a Service Package



Create a Service Package Skeleton

1

2

Build a YANG for the MTU Service

3

Create a template XML

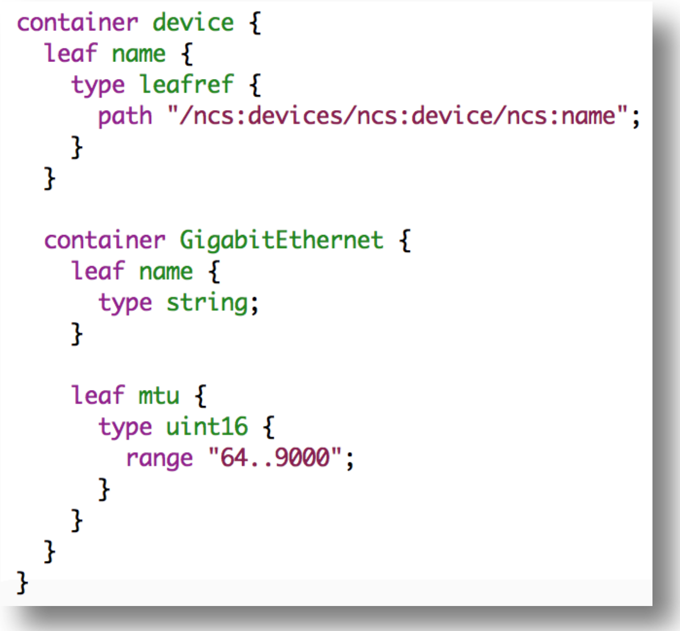
1. Configure a device and sync with NSO
2. Output device configuration in XML format
3. Plant variables in XML

4

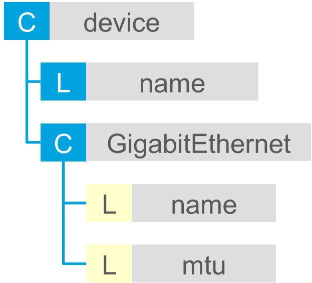
Test the Service

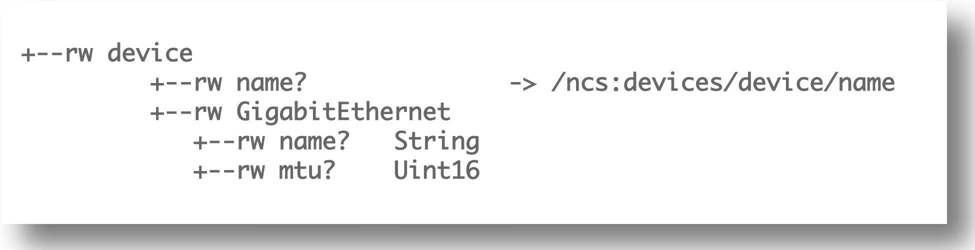
1. Create Service instance
2. Modify Service Instance
3. Delete Service Instance

Interface MTU Service Model



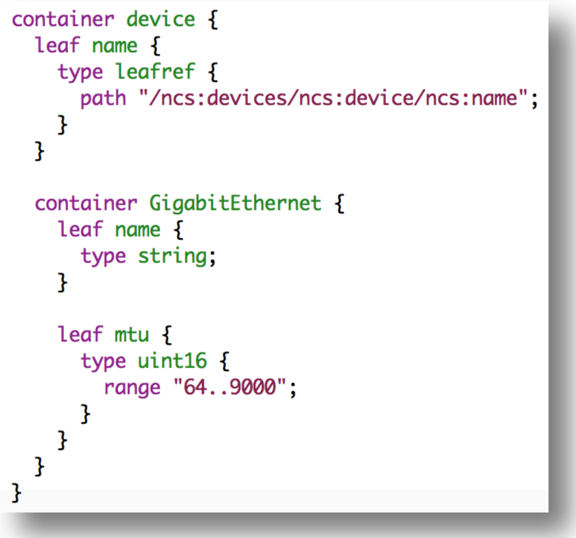
```
container device {
  leaf name {
    type leafref {
      path "/ncs:devices/ncs:device/ncs:name";
    }
  }

  container GigabitEthernet {
    leaf name {
      type string;
    }

    leaf mtu {
      type uint16 {
        range "64..9000";
      }
    }
  }
}
```



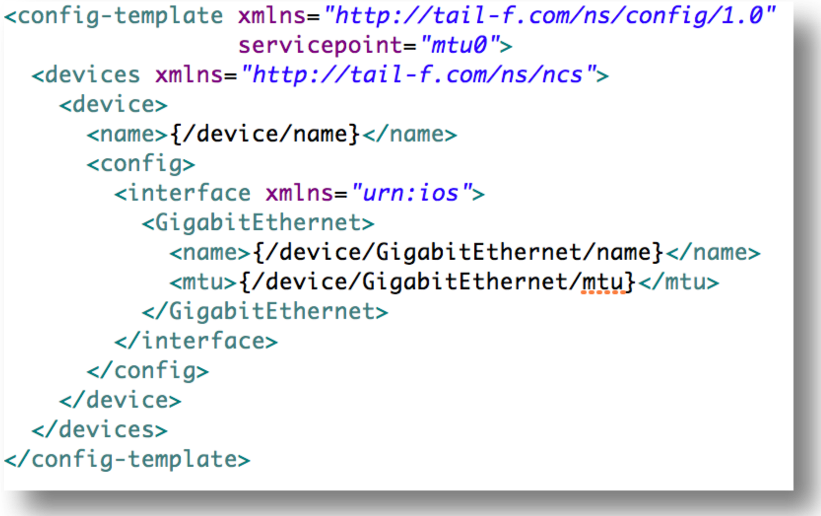


```
+--rw device
         +--rw name?                 -> /ncs:devices/device/name
         +--rw GigabitEthernet
            +--rw name?   String
            +--rw mtu?    Uint16
```

Mapping YANG model to XML Template



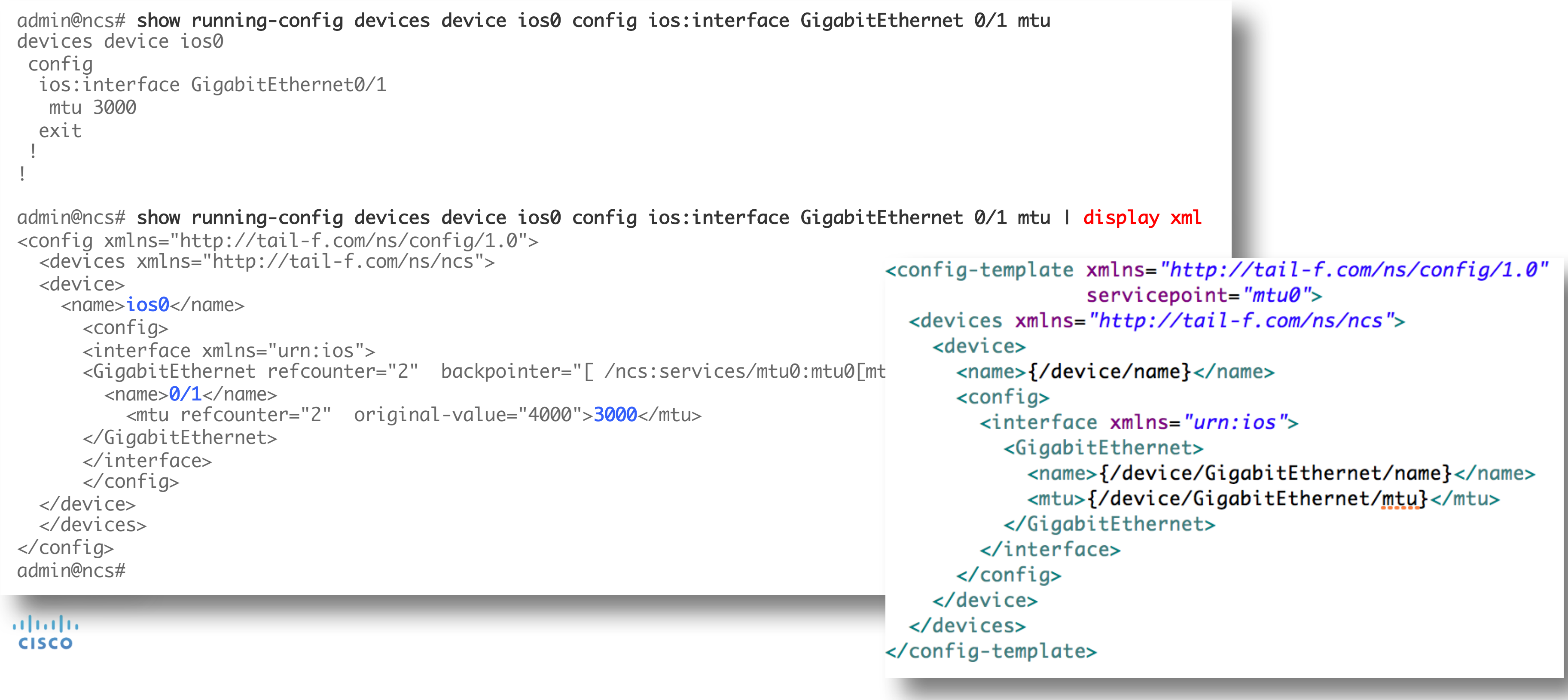
```
container device {
  leaf name {
    type leafref {
      path "/ncs:devices/ncs:device/ncs:name";
    }
  }

  container GigabitEthernet {
    leaf name {
      type string;
    }

    leaf mtu {
      type uint16 {
        range "64..9000";
      }
    }
  }
}
```



```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="mtu0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device/name}</name>
      <config>
        <interface xmlns="urn:ios">
          <GigabitEthernet>
            <name>{/device/GigabitEthernet/name}</name>
            <mtu>{/device/GigabitEthernet/mtu}</mtu>
          </GigabitEthernet>
        </interface>
      </config>
    </device>
  </devices>
</config-template>
```

How To Get the Initial Template?

Ask NSO ☺

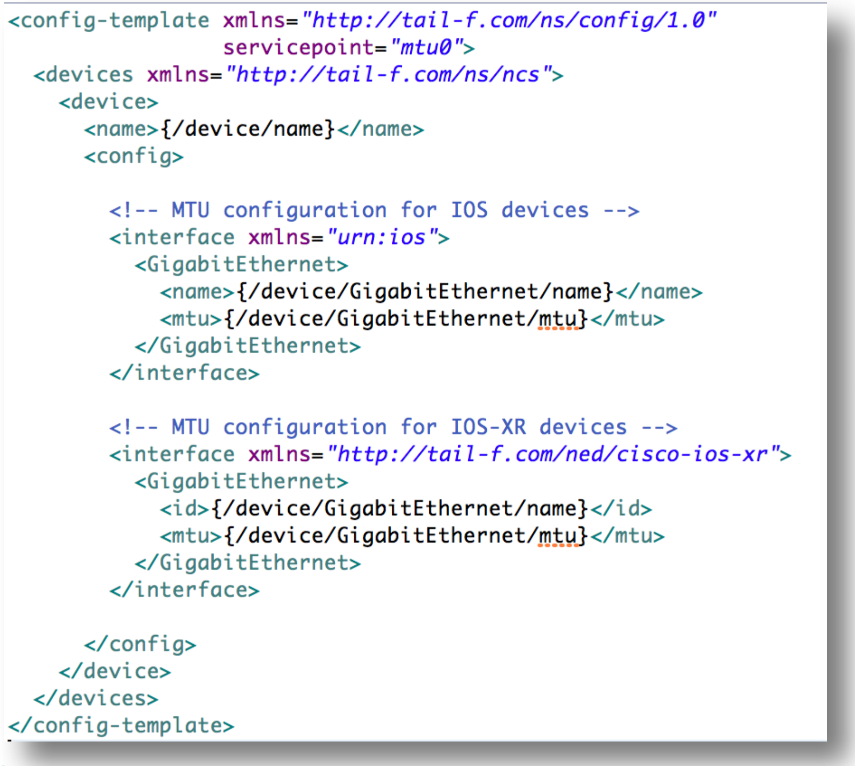


```
admin@ncs# show running-config devices device ios0 config ios:interface GigabitEthernet 0/1 mtu
devices device ios0
 config
  ios:interface GigabitEthernet0/1
   mtu 3000
  exit
 !
!

admin@ncs# show running-config devices device ios0 config ios:interface GigabitEthernet 0/1 mtu | display xml
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
  <device>
    <name>ios0</name>
      <config>
      <interface xmlns="urn:ios">
      <GigabitEthernet refcounter="2"  backpointer="[ /ncs:services/mtu0:mtu0[mt
        <name>0/1</name>
          <mtu refcounter="2"  original-value="4000">3000</mtu>
      </GigabitEthernet>
      </interface>
      </config>
  </device>
  </devices>
</config>
admin@ncs#
```

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="mtu0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device/name}</name>
      <config>
        <interface xmlns="urn:ios">
          <GigabitEthernet>
            <name>{/device/GigabitEthernet/name}</name>
            <mtu>{/device/GigabitEthernet/mtu}</mtu>
          </GigabitEthernet>
        </interface>
      </config>
    </device>
  </devices>
</config-template>
```

How to handle multivendor networks?

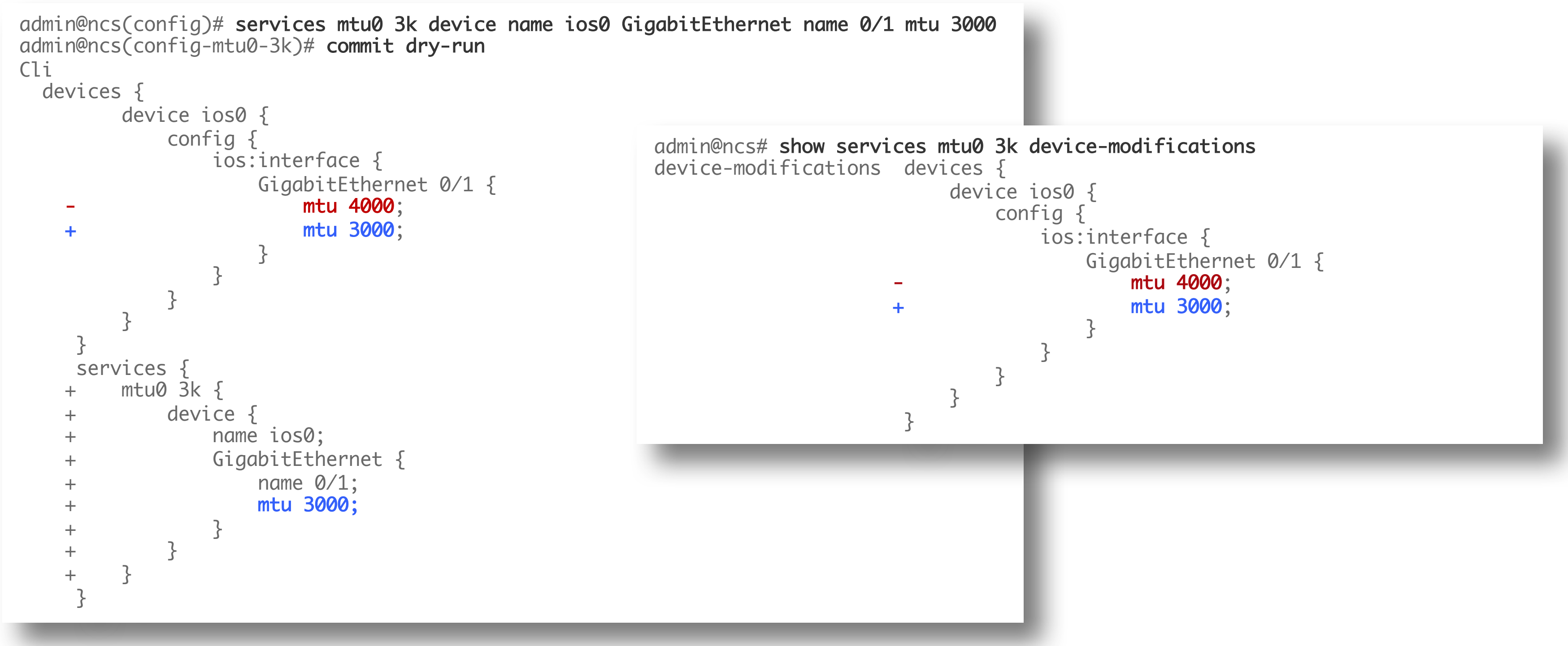
XML namespaces



```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="mtu0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device/name}</name>
      <config>

        <!-- MTU configuration for IOS devices -->
        <interface xmlns="urn:ios">
          <GigabitEthernet>
            <name>{/device/GigabitEthernet/name}</name>
            <mtu>{/device/GigabitEthernet/mtu}</mtu>
          </GigabitEthernet>
        </interface>

        <!-- MTU configuration for IOS-XR devices -->
        <interface xmlns="http://tail-f.com/ned/cisco-ios-xr">
          <GigabitEthernet>
            <id>{/device/GigabitEthernet/name}</id>
            <mtu>{/device/GigabitEthernet/mtu}</mtu>
          </GigabitEthernet>
        </interface>

      </config>
    </device>
  </devices>
</config-template>
```

Platform-specific mapping for the same function

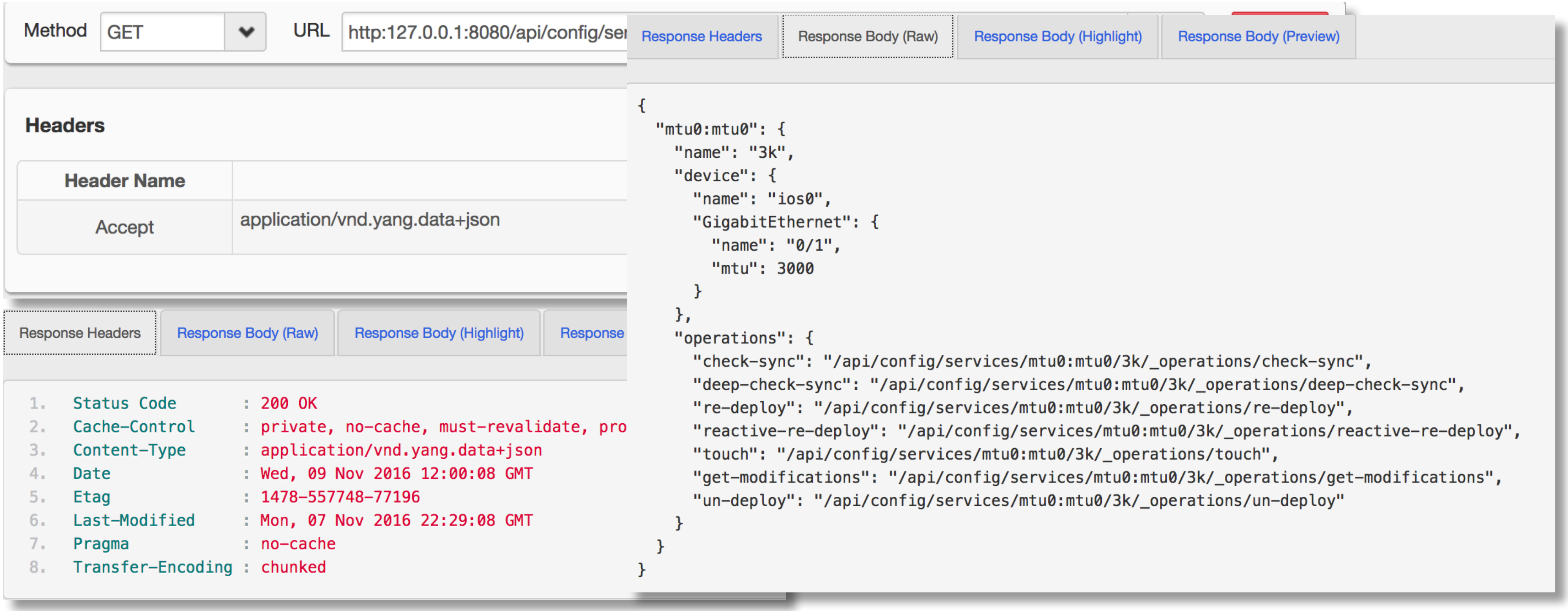
Services: Recording of Modification



```
admin@ncs(config)# services mtu0 3k device name ios0 GigabitEthernet name 0/1 mtu 3000
admin@ncs(config-mtu0-3k)# commit dry-run
Cli
  devices {
         device ios0 {
             config {
                 ios:interface {
                     GigabitEthernet 0/1 {
    -                    mtu 4000;
    +                    mtu 3000;
                     }
                 }
             }
         }
     }
     services {
    +    mtu0 3k {
    +        device {
    +            name ios0;
    +            GigabitEthernet {
    +                name 0/1;
    +                mtu 3000;
    +            }
    +        }
    +    }
     }
```

```
admin@ncs# show services mtu0 3k device-modifications
device-modifications  devices {
                          device ios0 {
                              config {
                                  ios:interface {
                                      GigabitEthernet 0/1 {
                     -                    mtu 4000;
                     +                    mtu 3000;
                                      }
                                  }
                              }
                          }
                      }
```

Service Query via REST API



Method GET

URL http:127.0.0.1:8080/api/config/ser

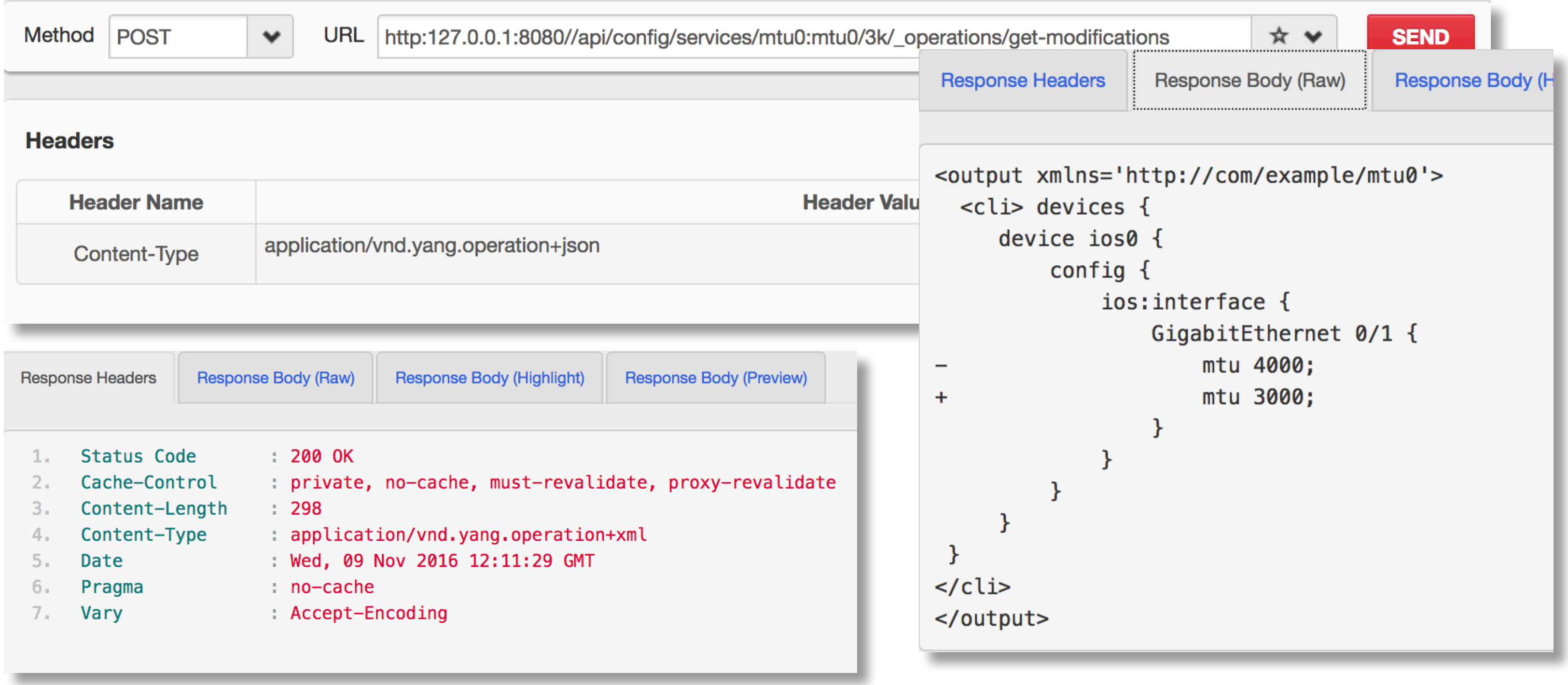
Headers

Header Name	
Accept	application/vnd.yang.data+json

```
Status Code       : 200 OK
Cache-Control     : private, no-cache, must-revalidate, pro
Content-Type      : application/vnd.yang.data+json
Date              : Wed, 09 Nov 2016 12:00:08 GMT
Etag              : 1478-557748-77196
Last-Modified     : Mon, 07 Nov 2016 22:29:08 GMT
Pragma            : no-cache
Transfer-Encoding : chunked
```

```
{
  "mtu0:mtu0": {
    "name": "3k",
    "device": {
      "name": "ios0",
      "GigabitEthernet": {
        "name": "0/1",
        "mtu": 3000
      }
    },
    "operations": {
      "check-sync": "/api/config/services/mtu0:mtu0/3k/_operations/check-sync",
      "deep-check-sync": "/api/config/services/mtu0:mtu0/3k/_operations/deep-check-sync",
      "re-deploy": "/api/config/services/mtu0:mtu0/3k/_operations/re-deploy",
      "reactive-re-deploy": "/api/config/services/mtu0:mtu0/3k/_operations/reactive-re-deploy",
      "touch": "/api/config/services/mtu0:mtu0/3k/_operations/touch",
      "get-modifications": "/api/config/services/mtu0:mtu0/3k/_operations/get-modifications",
      "un-deploy": "/api/config/services/mtu0:mtu0/3k/_operations/un-deploy"
    }
  }
}
```

Get Device Modifications via REST API



Method: POST

URL: http:127.0.0.1:8080//api/config/services/mtu0:mtu0/3k/_operations/get-modifications

Headers

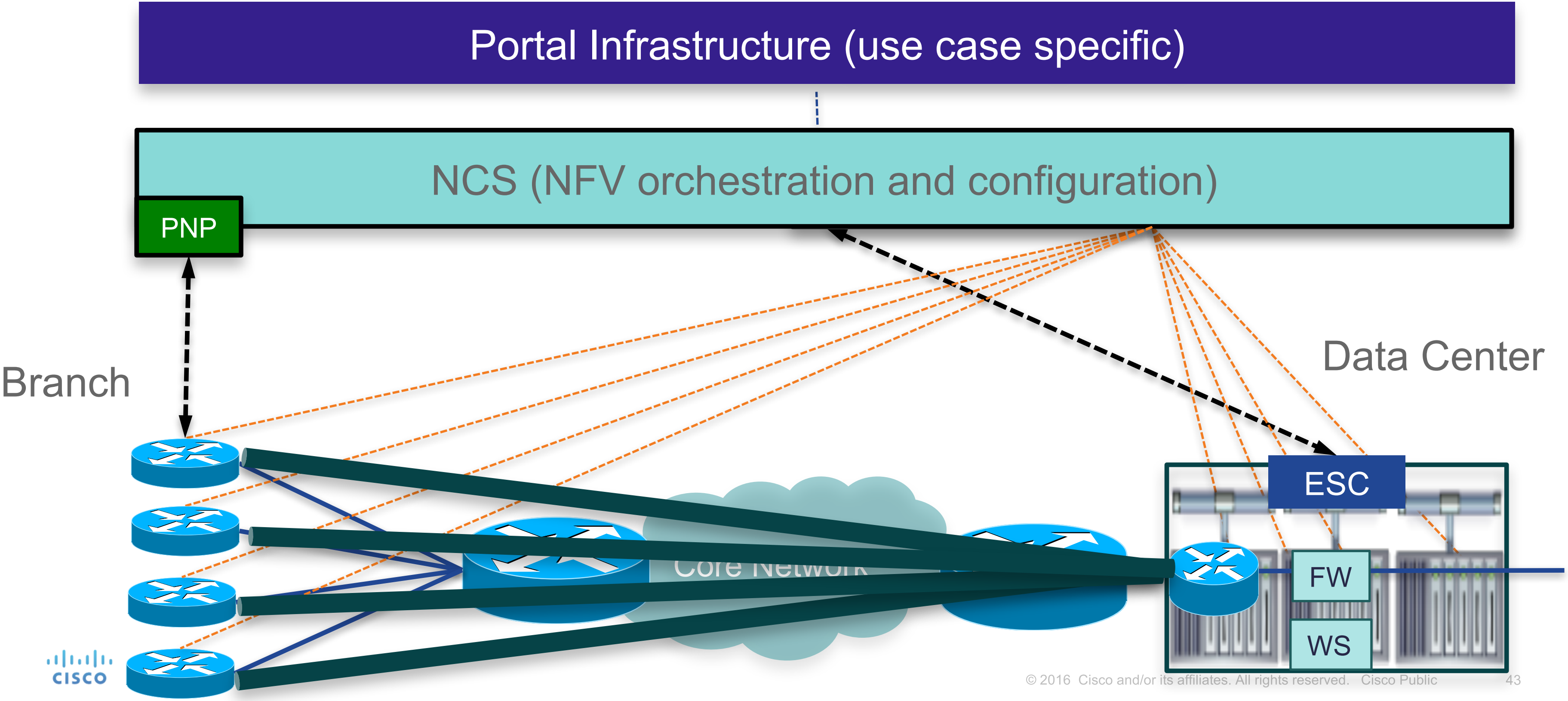
Header Name	Header Valu
Content-Type	application/vnd.yang.operation+json

```
Status Code          : 200 OK
Cache-Control        : private, no-cache, must-revalidate, proxy-revalidate
Content-Length       : 298
Content-Type         : application/vnd.yang.operation+xml
Date                 : Wed, 09 Nov 2016 12:11:29 GMT
Pragma               : no-cache
Vary                 : Accept-Encoding
```

```
<output xmlns='http://com/example/mtu0'>
  <cli> devices {
     device ios0 {
         config {
             ios:interface {
                 GigabitEthernet 0/1 {
-                    mtu 4000;
+                    mtu 3000;
                 }
             }
         }
     }
 }
</cli>
</output>
```

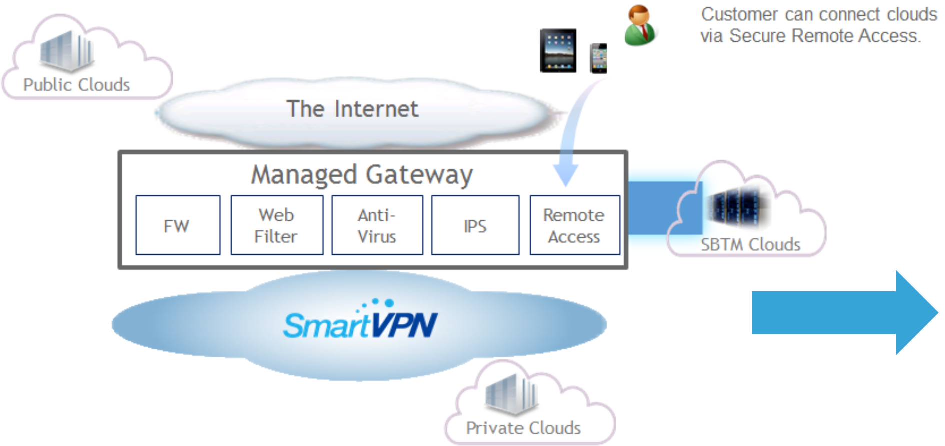
Real-world Use-Cases

CloudVPN


Portal Infrastructure (use case specific)
NCS (NFV orchestration and configuration)
PNP
Branch
Data Center
ESC
Core Network
FW
WS

SoftBank "Virtual Gateway" Project Goal

SOFTBANK TELECOM
Business: Telecommunications
Headquarters: Tokyo
Chairman & CEO: Masayoshi Son
Founded: October, 1984
Annual Revenue: 372,900 million yen
Number of Employees: 4,400 (Approx.)



RFI Goal: Virtualizing Managed Gateway

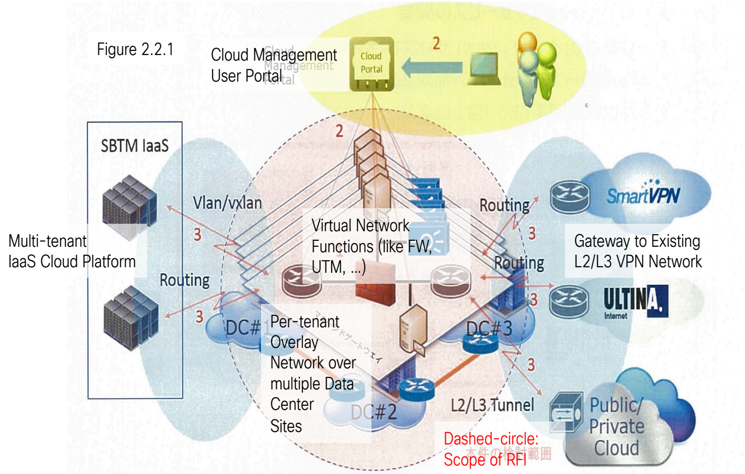


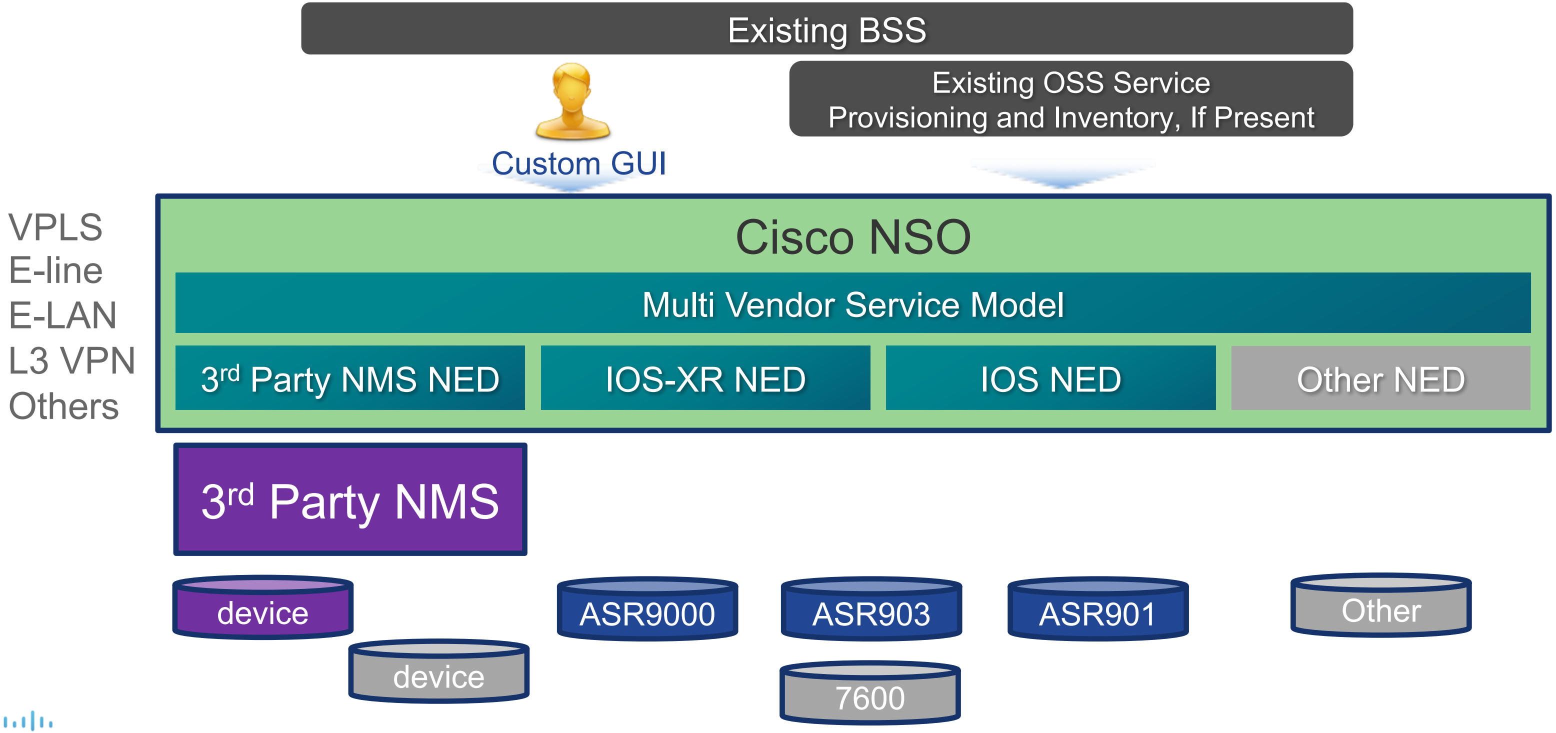
Figure 2.2.1
Cloud Management User Portal
Cloud Portal
2
SBTM IaaS
Multi-tenant IaaS Cloud Platform
Vlan/vxlan
3
Routing
Virtual Network Functions (like FW, UTM, ...)
Routing
SmartVPN
Gateway to Existing L2/L3 VPN Network
ULTINA Internet
DC#1
Per-tenant Overlay Network over multiple Data Center Sites
DC#2
DC#3
L2/L3 Tunnel
Public/ Private Cloud
Dashed-circle: Scope of RFI

→ Huge OPEX on service delivery, especially in human costs

- **Manual provisioning of devices**
- **Various options offered to end users to allow flexible customization**

→ Increased pressure from enterprise customers, with keeping current service levels/menus

NSO for Cisco and 3rd Party Vendor's NMS



Statement

- After a 3-day training

- **Every KIFÜ network engineer can develop and deploy services**

- In multi-vendor environment
- In a multi-domain network
- Within two week (max)

Are you believing now?

TOMORROW starts here.