Developer Days
Automation

The bridge to possible

MVR

NSO Service discovery framework using python

Lenin Khaidem
Software Consulting Engineer
10th May 2022

Agenda

- What is Service Discovery?
- Applications of service discovery
- What is MVR?
- MVR Architecture & Features
- Stages of service discovery In NSO
    - Data Extraction
    - Correlation
    - XML template application

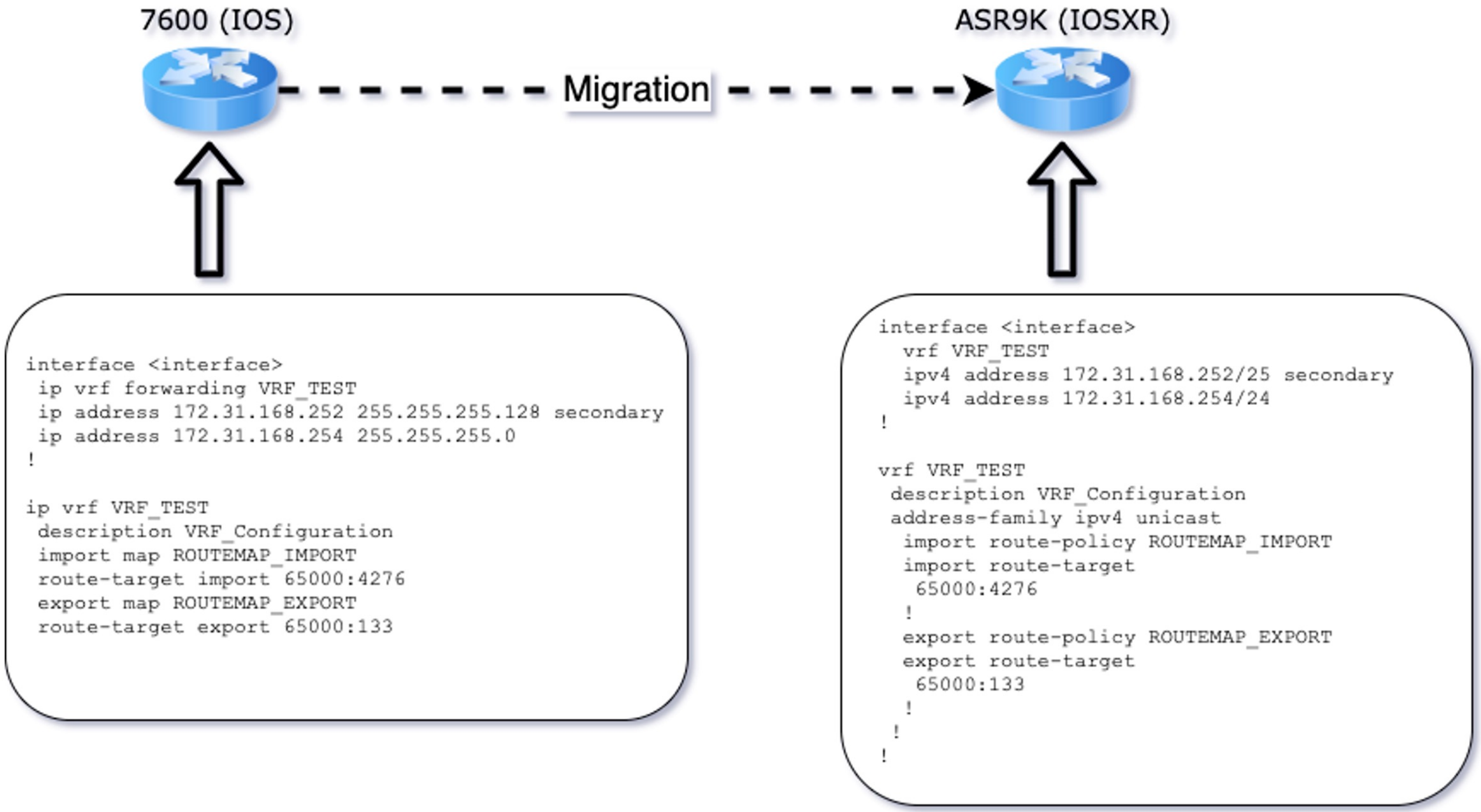
What is Service Discovery in network automation?

- The identification of instances of a specific service type from operational data or configuration data from network elements.
- Pre-defined set of network attributes completes the definition of the service
- Identify & correlate these parameters from the network elements

Applications of service discovery

- Network service migration from one device platform to another
- NSO service reconciliation for service life cycle management – CRUD operations
- Service inventory

Network Service Migration



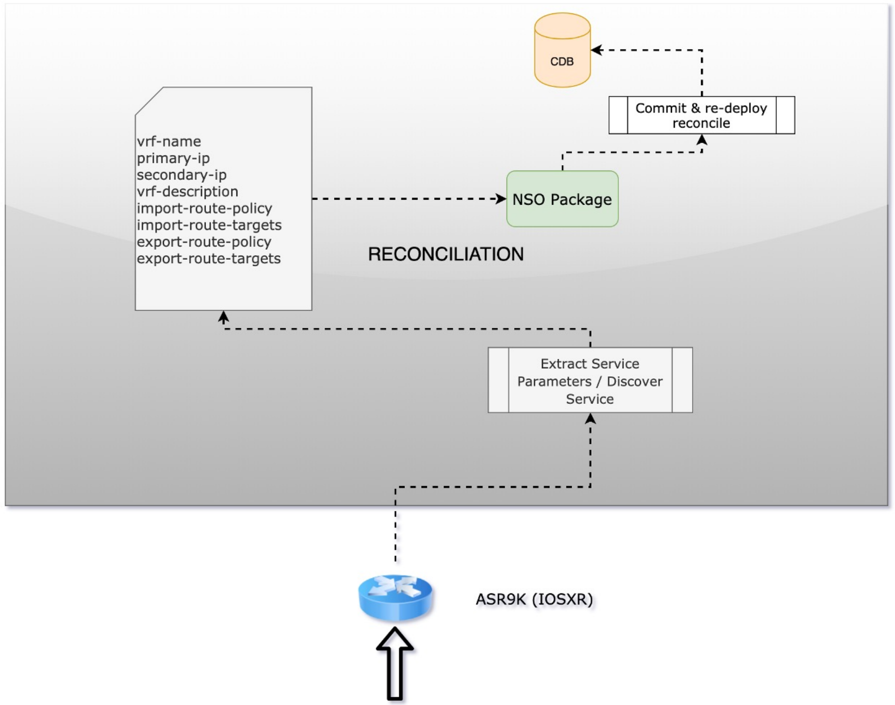
```
interface <interface>
 ip vrf forwarding VRF_TEST
 ip address 172.31.168.252 255.255.255.128 secondary
 ip address 172.31.168.254 255.255.255.0
!

ip vrf VRF_TEST
 description VRF_Configuration
 import map ROUTEMAP_IMPORT
 route-target import 65000:4276
 export map ROUTEMAP_EXPORT
 route-target export 65000:133
```

```
interface <interface>
  vrf VRF_TEST
  ipv4 address 172.31.168.252/25 secondary
  ipv4 address 172.31.168.254/24
!

vrf VRF_TEST
 description VRF_Configuration
 address-family ipv4 unicast
  import route-policy ROUTEMAP_IMPORT
  import route-target
   65000:4276
  !
  export route-policy ROUTEMAP_EXPORT
  export route-target
   65000:133
  !
 !
!
```

Service Reconcliation using NSO



```
interface <interface>
  vrf VRF_TEST
  ipv4 address 172.31.168.252/25 secondary
  ipv4 address 172.31.168.254/24
!

vrf VRF_TEST
 description VRF_Configuration
 address-family ipv4 unicast
  import route-policy ROUTEMAP_IMPORT
  import route-target
   65000:4276
  !
  export route-policy ROUTEMAP_EXPORT
  export route-target
   65000:133
  !
 !
!
```

What are the high-level steps involved in any migration and service reconciliation using NSO ?

Parameter Extraction

- Extraction of parameters from the source, which is expected to reside in NSO CDB

Correlation

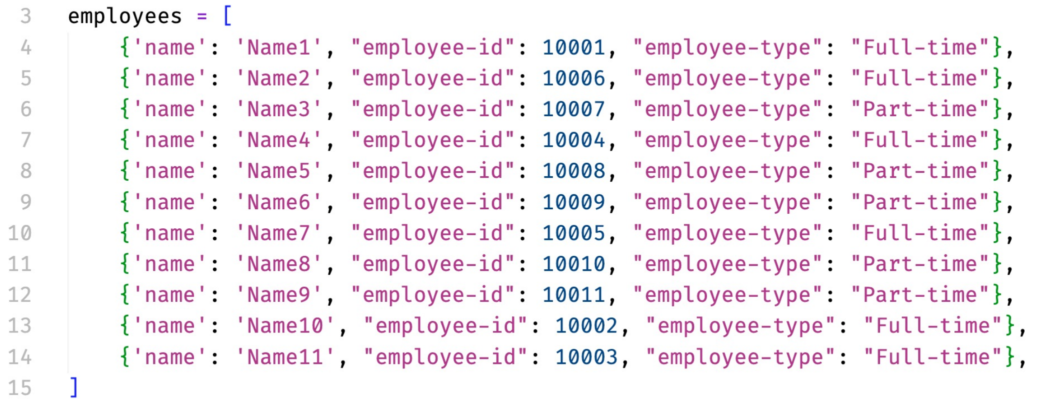
- Correlation of extracted parameters to build payloads

Service inventory

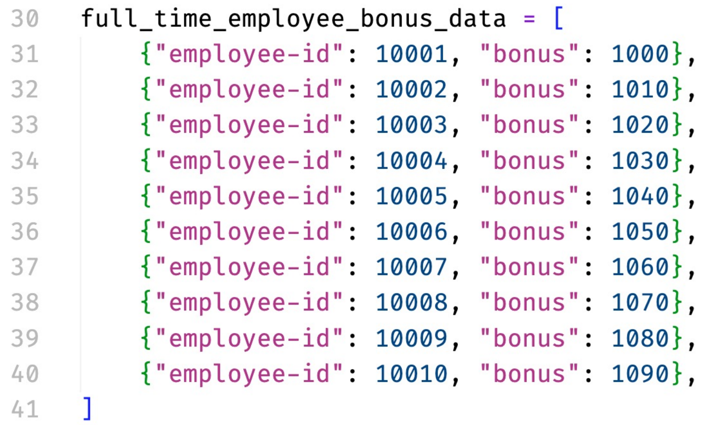
- Target service inventory / model could have NSO service point

What do they actually mean ?

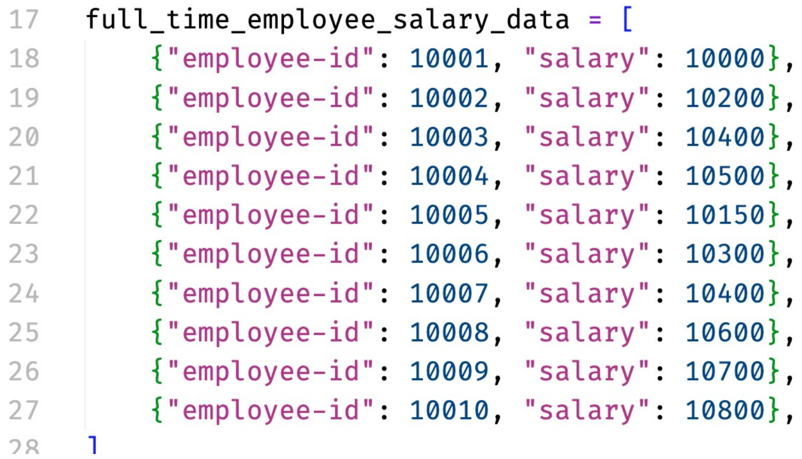
Let’s look at the employee & pay data



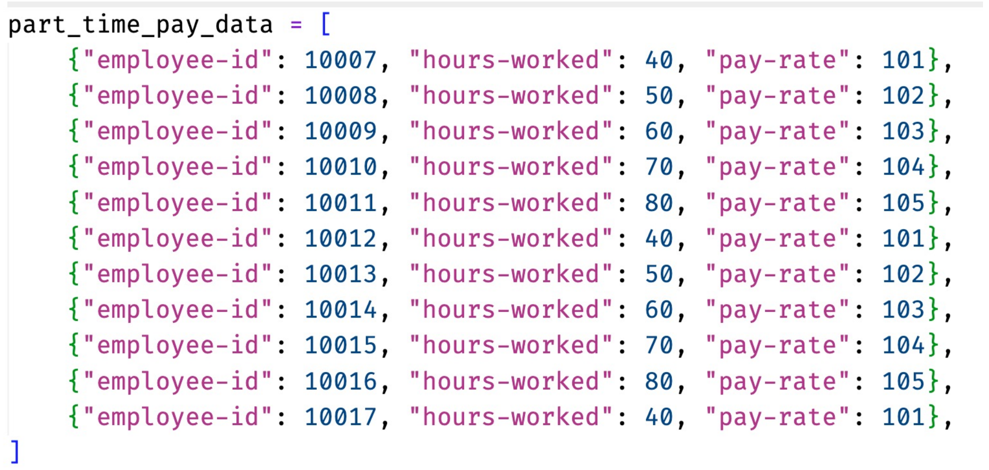
```
employees = [
    {'name': 'Name1', "employee-id": 10001, "employee-type": "Full-time"},
    {'name': 'Name2', "employee-id": 10006, "employee-type": "Full-time"},
    {'name': 'Name3', "employee-id": 10007, "employee-type": "Part-time"},
    {'name': 'Name4', "employee-id": 10004, "employee-type": "Full-time"},
    {'name': 'Name5', "employee-id": 10008, "employee-type": "Part-time"},
    {'name': 'Name6', "employee-id": 10009, "employee-type": "Part-time"},
    {'name': 'Name7', "employee-id": 10005, "employee-type": "Full-time"},
    {'name': 'Name8', "employee-id": 10010, "employee-type": "Part-time"},
    {'name': 'Name9', "employee-id": 10011, "employee-type": "Part-time"},
    {'name': 'Name10', "employee-id": 10002, "employee-type": "Full-time"},
    {'name': 'Name11', "employee-id": 10003, "employee-type": "Full-time"},
]
```



```
full_time_employee_bonus_data = [
    {"employee-id": 10001, "bonus": 1000},
    {"employee-id": 10002, "bonus": 1010},
    {"employee-id": 10003, "bonus": 1020},
    {"employee-id": 10004, "bonus": 1030},
    {"employee-id": 10005, "bonus": 1040},
    {"employee-id": 10006, "bonus": 1050},
    {"employee-id": 10007, "bonus": 1060},
    {"employee-id": 10008, "bonus": 1070},
    {"employee-id": 10009, "bonus": 1080},
    {"employee-id": 10010, "bonus": 1090},
]
```

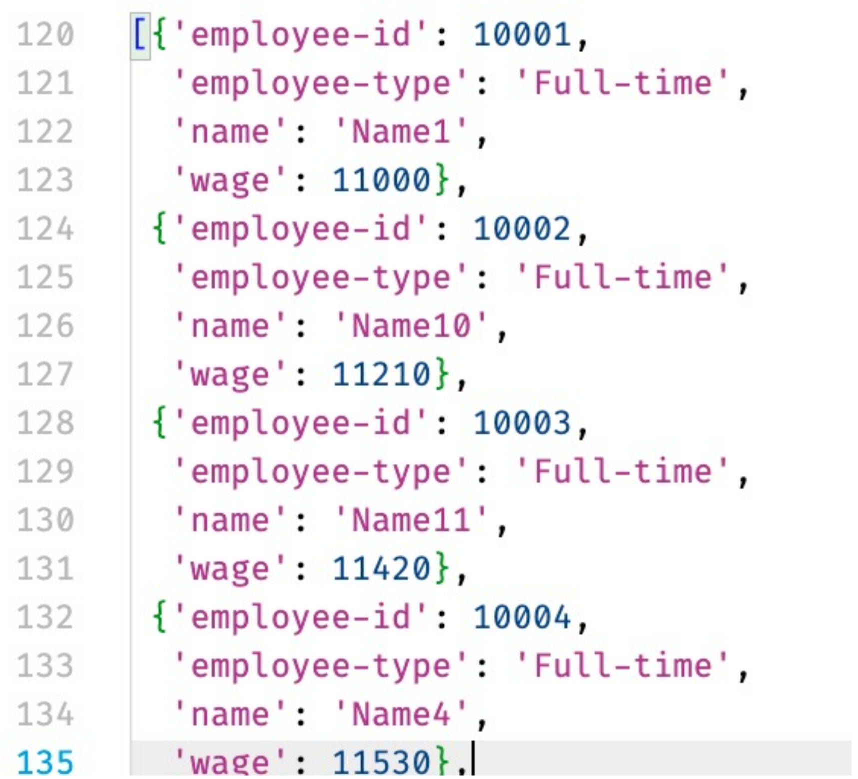


```
full_time_employee_salary_data = [
    {"employee-id": 10001, "salary": 10000},
    {"employee-id": 10002, "salary": 10200},
    {"employee-id": 10003, "salary": 10400},
    {"employee-id": 10004, "salary": 10500},
    {"employee-id": 10005, "salary": 10150},
    {"employee-id": 10006, "salary": 10300},
    {"employee-id": 10007, "salary": 10400},
    {"employee-id": 10008, "salary": 10600},
    {"employee-id": 10009, "salary": 10700},
    {"employee-id": 10010, "salary": 10800},
]
```



```
part_time_pay_data = [
    {"employee-id": 10007, "hours-worked": 40, "pay-rate": 101},
    {"employee-id": 10008, "hours-worked": 50, "pay-rate": 102},
    {"employee-id": 10009, "hours-worked": 60, "pay-rate": 103},
    {"employee-id": 10010, "hours-worked": 70, "pay-rate": 104},
    {"employee-id": 10011, "hours-worked": 80, "pay-rate": 105},
    {"employee-id": 10012, "hours-worked": 40, "pay-rate": 101},
    {"employee-id": 10013, "hours-worked": 50, "pay-rate": 102},
    {"employee-id": 10014, "hours-worked": 60, "pay-rate": 103},
    {"employee-id": 10015, "hours-worked": 70, "pay-rate": 104},
    {"employee-id": 10016, "hours-worked": 80, "pay-rate": 105},
    {"employee-id": 10017, "hours-worked": 40, "pay-rate": 101},
]
```

Correlator computes the wage for all the employees



```
[{'employee-id': 10001,
  'employee-type': 'Full-time',
  'name': 'Name1',
  'wage': 11000},
 {'employee-id': 10002,
  'employee-type': 'Full-time',
  'name': 'Name10',
  'wage': 11210},
 {'employee-id': 10003,
  'employee-type': 'Full-time',
  'name': 'Name11',
  'wage': 11420},
 {'employee-id': 10004,
  'employee-type': 'Full-time',
  'name': 'Name4',
  'wage': 11530},
```

What is MVR ? – Stands for migration, validation & reconciliation

NSO Action Package

Discovery logic is defined in callback functions in the MVR implemented python classes

Technology & Vendor agnostic tool

Service discovery is based on NSO CDB data

Service Discovery Framework

Written in python3.

Object based architecture

Abstraction of repetitive code

Developer focuses solely on the service discovery logic

Features & benefits of the framework

- Easy to use, plug and play
- Discovery focused
- Service discovery and service creation code is decoupled
- Easily extendible
- Reusability
- Package generator script(mvr-make-package) included

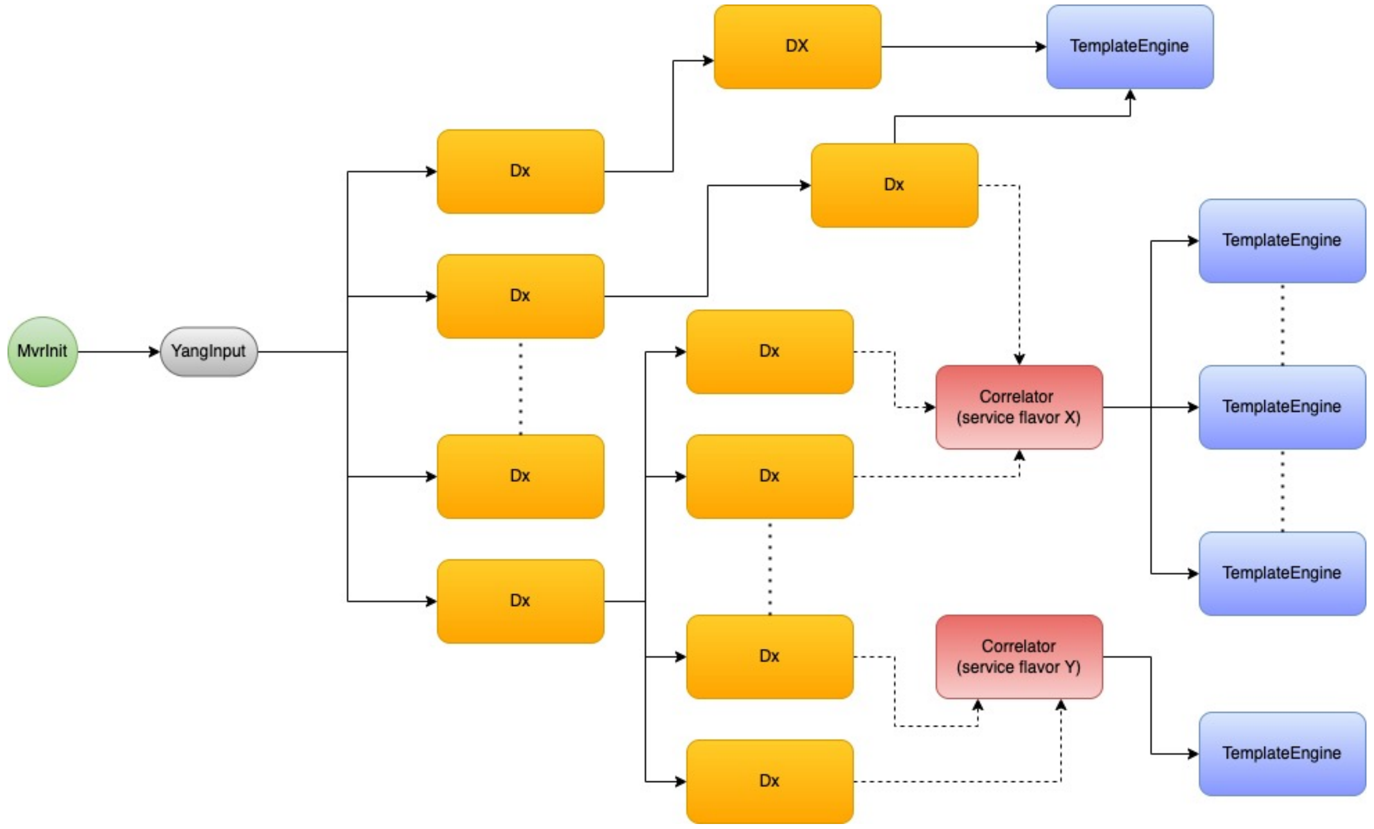
Skills required to start developing service discovery code

- Understanding of the service discovery workflow
- NSO, XPATH filters
- Python, list / set comprehensions, filter

Architecture is python object based

MvrInit	Stores execution meta data and other internal objects
YangInput	Extract user action input and store them
Dx	Extracts service parameters from CDB. Accepts user defined input arguments
Correlator	Correlate all extracted data by Dx
TemplateEngine	Responsible for applying NSO XML template. Can accept input from a Dx or a Correlator Object

Putting them all together



Demo

The bridge to possible