NSO Python Overview & Labs

Jan Lindblad, Tail-f Business Development Solutions Architect

NSO Developer Days, June 2017

Language Bindings per API

	Java	Python	Erlang
Service	MAAPI or NAVU	MAAPI or MAAGIC	MAAPI
NED	NED	-	-
Subscriber	CDBAPI	CDBAPI	CDBAPI
Oper Data Writer	MAAPI or CDBAPI	MAAPI or CDBAPI	MAAPI or CDBAPI
Data Provider/Transform	DPAPI	DPAPI	DPAPI
Action Provider	DPAPI	DPAPI	DPAPI
Notifications	NOTIF	EVENT	EVENT
Schema Introspection	CS	CS or MAAGIC	CS
HAFW	HAAPI	HAAPI	HAAPI

Language Implementation Details

	Java	Python	Erlang
Source+binary directory	java/jar	src/ncs/pyapi	src/ncs/econfd
Doc directory	doc/api/java	doc/api/python	doc/api/econfd
Log files	logs/ncs-java-vm.log	logs/ncs-python-vm.log logs/ncs-python-vm-*.log	Up to app
VM	Single external JVM for all apps	Multiple external PyVMs, one per app	Internal ErlVM: same External ErlVM: up to app
Required symbol prefix	-	-	"ec_" if intenal ErlVM
Sweet spot	NED	Service	Transform

Can you mix packages written in different languages? Of course!
YANG defines the interface

Package Creation by Language

```
ncs-make-package --erlang-skeleton package-name

ncs-make-package --service-skeleton TYPE 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)
```

Some Python Application Types

```
from ncs.application import Application,
  Service, NanoService
from ncs.dp import Action

class FooService(Service):
  @Service.create
  def cb_create(self, tctx, root,
    service, proplist):

    # service code here

class FooNanoService(NanoService):
  @NanoService.create
  def cb_nano_create(self, tctx, root,
    service, component, state, proplist):

    # service code here

class FooAction(Action):
  @Action.action
  def cb_action(self, uinfo, name, kp,
    input, output):

    # action code here
```

```
class MyApp(Application):
  def setup(self):
    self.log.debug('MyApp start')

    self.register_service(
      'myservice-1', FooService)

    self.register_service(
      'myservice-2', FooService,
      'init_arg')

    self.register_nano_service(
      'nano-1', 'myserv:router',
      'myserv:ntp-initialized',
      FooNanoService)

    self.register_action(
      'action-1', FooAction)

  def teardown(self):
    self.log.debug('MyApp finish')
```

Imports and _ _init_ _.py

```
import ncs

import ncs.maapi as maapi

import ncs.maagic as maagic

import resource_manager.\
  ipaddress_allocator as ip_allocator

from ncs.application import Service
```

- In a deeper structure, you need __init__.py files
  - Could be empty
  - or have init code, e.g.:

```
__all__ = ['action', 'namespaces', 'op']

from action import Action
```

MAAGIC

- API for manipulating data according to YANG schema
- Use . dot-notation to navigate YANG model
- When crossing namespace boundaries, use *namespace-name double- underscore symbol-name*, e.g.
  - root.myns__top.val

mymod.yang

```
module mymod {
  prefix myns;
  container top {
    leaf val { type string; }
  }
}
```

MAAGIC

- V = root.myns__top.val
- root.myns__top.val = V

NAVU

- V = ncsRoot.container(" myns","top").leaf(" val").valueAsString()
- ncsRoot.container(" myns","top").leaf(" val").sharedSet(V)

MAAPI

- V = get_elem(Sock, Trans, "/myns:top/myns:val")
- shared_set_elem(Sock, Trans, "/myns:top/myns:val", V)

ipython-superuser

```
JLINDBLA-M-J8L9# ipython
Python 2.7.11 (default, Jan 22 2016, 08:29:18)
Type "copyright", "credits" or "license" for
more    IPython 5.1.0 -- An enhanced
Interactive Python.    ?         ->
Introduction and overview of %quickref ->
Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use
'object??' for extra details.
In [1]: for dev in root.devices.device:
        ...: print dev.name
        ...:
ce0
ce1
ce2
ce3
```

- ncs, ncs_trans_id, ncs_sess_id, maapi, trans, root=ncs.maagic.get_root(trans)

```
In [32]: trans.query_start(
  expr="/devices/device[port='8300']",
  context_node='/',
  chunk_size=10,
  initial_offset=0,
  result_as=1,
  select=['name'],
  sort=['name'])
Out[32]: 3957
In [33]: for res in maapi.query_result(Out[32]):
    ...:     print res
    ...:
['/ncs:devices/device{p3}/name']
['/ncs:devices/device{pe2}/name']
['/ncs:devices/device{xr-local}/name']
```

Exceptions

- Simply define your own exceptions, inherit from Exception
- raise to abort the transaction/action
- Easy to deliver a message

```
class PoolNotFound(Exception):
  pass
```

```
if not service.pool:
  raise PoolNotFound(
    "No valid pool selected in
     subnet %s" %
    str(service._path))
```

Lab 1: Write Python Service

Create a Python service that pushes qos-classification settings to each IOS device in a specified device group

The qos classification holds a configurable list of queues that application flows can be assigned to

The qos classification defines application flows based on protocol (tcp|udp), source port or port range, destination port or port range and optionally a dscp value. The operator assigns one of the queues mentioned above to each application flow

- The YANG model for the service, XML templates and some Python starting point code is given

Lab 2: Write Python RFM Service

Create a Reactive-Fastmap Python service that assigns a unique IPv4 address on the same subnet to each interface listed in the service

Allocate the subnet using the resource-manager

- Always allocate as small a subnet as possible
- You need to configure some address pools in the resource-manager

Seven interface types across three device types

- The YANG model for the service, XML templates and some Python starting point code is given
- Use the resource-manager package for the IP address allocation
- Optional: Fill in the operational data leaf in the model with the allocated address.