



IXP Automation

Amsterdam, September 2017

Nick Hilliard

Chief Technical Officer

Internet Neutral Exchange Association
Company Limited by Guarantee



Background

- Original purpose of IXP Manager was to support route server config builds
- Designed with a structure capable of storing all participant switch configuration tokens
- Reticent about using database for network configuration
 - Cost / return ratio wasn't right
 - Concerns about how to control configuration deployment
 - Poor tool support for interfacing with network devices

Toolchain Problems

- “Traditional” server automation tools could not interface with network devices
- Tools of the era: RANCID, SSH, bash + perl scripts
- No framework mechanisms available

Roll Forward to 2017

- Multiple automation approaches possible
- Server automation frameworks can interface with network devices
- Network Operating Systems now have APIs and / or API models
- Some NOSs support multiple APIs
- Rationale changes
 - Too much repetitive configuration: “Taking the operator out of operations”
 - Long term cost reduction

Phase 1 Operational Goals

- Configure all IXP participant edge ports
 - Speed, dot1q framing, LAG ports, layer 2 filters
- Configure IXP core
 - Interfaces, BGP, VXLAN configuration
- Ready for service to handle peering LAN forklift upgrade to Arista kit in 2017Q1

Phase 2 Strategic Goals

- Use initial automation process to learn how to do this properly
- Build functionality into IXP Manager: user interface, database, export presentation
- Ensure that abstraction model is usable across different network devices and different organisations
- Release as open source

Approaches

	Openflow	YANG	Vendor API
Abstraction Level	Low	High	Mid Range
Vendor Support	Version Dependent	In Development	Variable
Portability	High	High	Low
Cross-Platform	Low	Currently low	Needs Abstraction
Complexity	High	Mid	Low

Practical Approach

- YANG: only well supported on tiny number of NOSs
- Openflow: too low level
- Decided to use NAPALM
 - Integrates with vendor APIs at the network device interface
 - Integrates with Ansible and SaltStack at control + provisioning DB interface
 - Long term support is likely to be good

Data Presentation

- Most vendor APIs are simply a better-structured CLI
 - mandatory authentication and security
 - XML or JSON formatting
 - commands are issued, replies received
- Drawbacks
 - Some CLIs cannot be automated due to e.g. non-idempotent command support
 - Many APIs / NOSs do not support basic functions like merge / commit / rollback
 - API support is often NOS version specific

IXP AUTOMATION

NAPALM Support

^[2] Hand-crafted by the API as the device doesn't support the feature.

^[3] Not supported but emulated. Check caveats.

^[4] Check the caveats, this is a dangerous operation in this device.

^[5] For merges, the diff is simply the merge config itself. See caveats.

^[6] No for merges. See caveats.

	EOS	JunOS	IOS-XR	FortiOS	NXOS	IOS	Pluribus	PANOS	MikroTik	VyOS
Config Replace	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Config Merge	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Config Compare	Yes	Yes	Yes ^[2]	Yes ^[2]	Yes ^[5]	Yes	No	Yes	No	Yes
Atomic Change	Yes	Yes	Yes	No ^[3]	Yes/No ^[6]	Yes	Yes	Yes/No ^[6]	No	Yes
Rollback	Yes ^[3]	Yes	Yes	Yes	Yes/No ^[6]	Yes	No	Yes	No	Yes

IXP AUTOMATION

INEX Kit Manifest

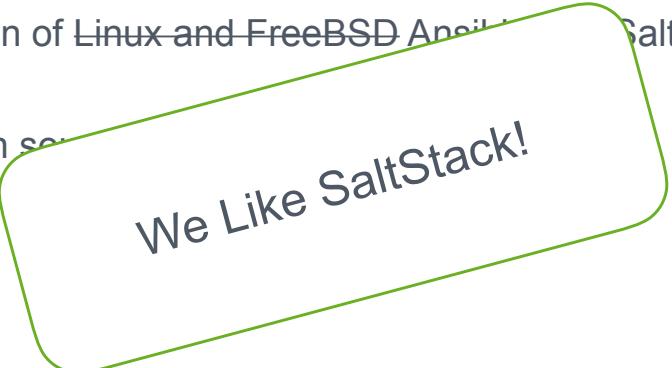
	Brocade FI	Brocade NI	Extreme	Arista EOS	Cumulus
INEX Lifecycle	EOL	EOL	Mid life	Early life	Pre-Deploy
API Support	None	Some YANG	XOS v21+	Excellent	Linux
Openflow	No	v1.3	v1.3	v1.3	No
NAPALM	No	No	*Not yet	Yes	No
Assessment	No plans	No plans	Partial support	Full Support	Full Support

~~Vi vs Emacs~~ Ansible vs SaltStack

- Lengthy evaluation process
- Careful consideration of ~~Linux and FreeBSD~~ Ansible and SaltStack pros / cons.
- Rationale resulted in sound engineering decision:

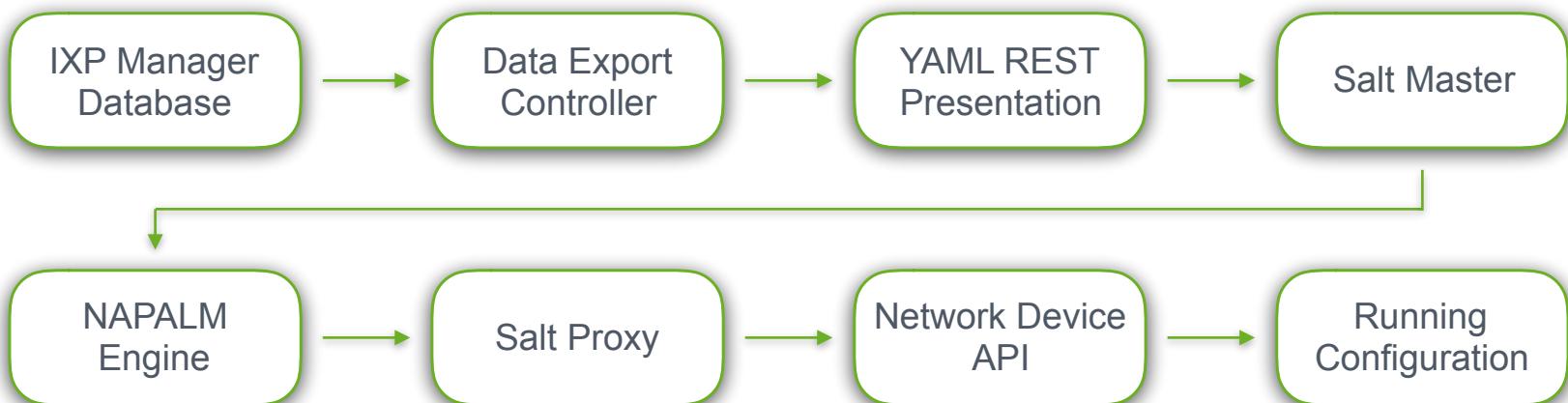
Vi vs Emacs Ansible vs SaltStack

- Lengthy evaluation process
- Careful consideration of ~~Linux and FreeBSD Ansible~~ SaltStack pros / cons.
- Rationale resulted in ~~so~~ SaltStack

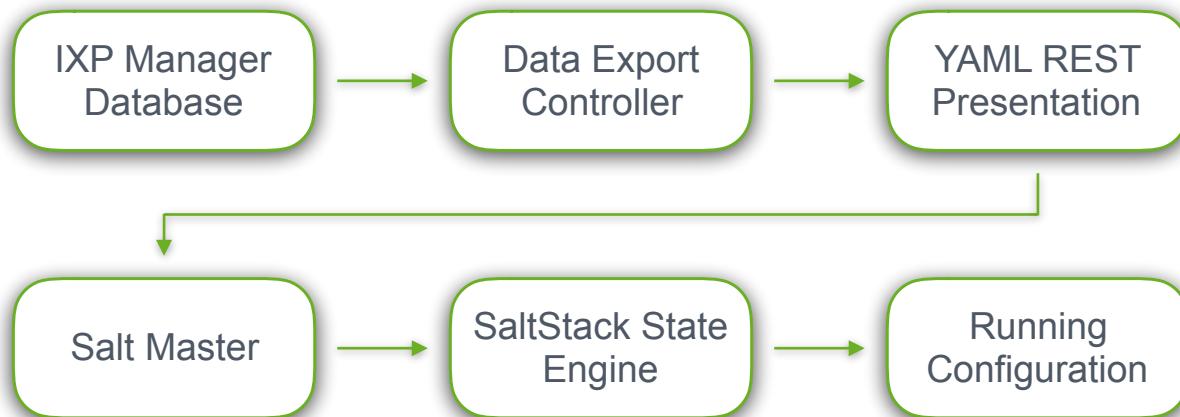


We Like SaltStack!

Data Flow - Traditional NOS



Data Flow - Cumulus Linux



IXP Manager Data Presentation

- API version 4 exports YAML via REST calls
- Exported data roughly breaks down as:
 - Vlans
 - Layer 2 interface information
 - Layer 3 interface information
 - Information required for routed core (bgp + vxlan)

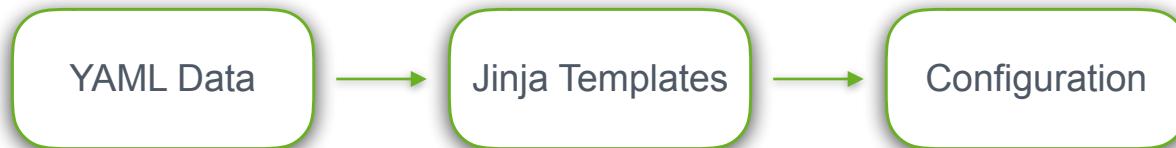
Sample YAML

```
- name: swp2
  type: edge
  description: "Packetloss Services Ltd"
  dot1q: yes
  shutdown: yes
  autoneg: yes
  speed: 10000
  lagindex: 1
  lagmaster: no
  fastlacp: yes
  virtualinterfaceid: 334
  vlans:
    - number: 12
      macaddress:
        - "54:1e:56:35:77:d0"
```

Sample YAML

```
- name: swp49
  type: core
  description: "edge1-edge2"
  dot1q: yes
  shutdown: no
  stp: yes
  cost: 100
  autoneg: yes
  speed: 40000
  lagindex: 1010
  lagmaster: no
  virtualinterfaceid: 342
  vlans:
    - number: 12
    - number: 32
```

Data Templating



Sample YAML

```
{% if pillar.get('interfacescust', {}) is iterable %}  
  {% for iface in pillar.get('interfacescust', {}) %}  
  
    default interface {{ iface.name }}  
    interface {{ iface.name }}  
      load-interval 30  
  
    {% if iface.description|length > 0 %}  
      description {{ iface.description }}  
    {% else %}  
      no description  
    {% endif %}  
  
    [...]  
  
  {% endfor %}  
  {% endif %}
```

Sample YAML

```
{% if iface.speed == 100 %}  
    speed forced 100full  
{% elif iface.speed == 1000 %}  
    {% if iface.autoneg is defined and iface.autoneg == False %}  
        speed forced 1000full  
    {% else %}  
        speed auto  
    {% endif %}  
{% elif iface.speed == 10000 %}  
    {# speed auto #}  
{% elif iface.speed == 40000 %}  
    speed forced 40gfull  
{% elif iface.speed == 100000 %}  
    speed forced 100gfull  
{% endif %}
```

Modelling Problems

- Different switches use different data models for configuration
- For example, Link Aggregation
 - Brocade uses physical interfaces only
 - Extreme has a separate configuration item: “enable sharing XX”
 - Others devices use a virtual interface (Port-ChannelX, bondY, etc)
 - But not all the semantics are the same (channel-group vs bond-slaves)
- Lessons learned: ensure your data model is flexible enough to support substantial semantic differences between device config models, and that it can be extended easily

Operational Problems

- Beware of upgrades!
- O/S package management vs pip install
- Jinja 2.7 -> 2.9 broke lots of templates
 - undefined variables are no longer evaluated as `False`
 - iterating over an undefined variable returns an error rather than skipping eval
- Need to be careful with SaltStack upgrades

Jinja 2.7

```
{% for iface in pillar.get('interfescust', {}) %}

    interface {{ iface.name }}

    {% if iface.shutdown %}
        shutdown
    {% endif %}

{% endfor %}
```

Jinja 2.9

```
{% if pillar.get('interfacescust', {}) is iterable %}  
  {% for iface in pillar.get('interfacescust', {}) %}  
  
    interface {{ iface.name }}  
  
    {% if iface.shutdown is defined and iface.shutdown %}  
      shutdown  
    {% endif %}  
  
  {% endfor %}  
  {% endif %}
```

IXP AUTOMATION

```
2017-09-04 13:42:01,662 [salt.loader][CRITICAL][794] Failed to load grains defined in grain file napalm.model in
function <function model at 0x80d0fec08>, error:
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/site-packages/salt/loader.py", line 722, in grains
    else:
  File "/usr/local/lib/python2.7/site-packages/salt/grains/napalm.py", line 174, in model
    return {'model': _get_grain('model', proxy=proxy)}
  File "/usr/local/lib/python2.7/site-packages/salt/grains/napalm.py", line 102, in _get_grain
    grains = _retrieve_grains_cache(proxy=proxy)
  File "/usr/local/lib/python2.7/site-packages/salt/grains/napalm.py", line 71, in _retrieve_grains_cache
    GRAINS_CACHE = proxy['napalm.get_grains']()
  File "/usr/local/lib/python2.7/site-packages/salt/loader.py", line 1088, in __getitem__
    self.missing_modules = {} # mapping of name -> error
  File "/usr/local/lib/python2.7/site-packages/salt/utils/lazy.py", line 101, in __getitem__
    raise KeyError(key)
KeyError: 'napalm.get_grains'
```

IXP AUTOMATION

```
2017-09-04 13:48:19,254 [salt.minion][CRITICAL][85409] Unexpected error while connecting to localhost
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/site-packages/salt/minion.py", line 864, in _connect_minion
    yield minion.connect_master(failed=failed)
  File "/usr/local/lib/python2.7/site-packages/tornado/gen.py", line 1055, in run
    value = future.result()
  File "/usr/local/lib/python2.7/site-packages/tornado/concurrent.py", line 238, in result
    raise_exc_info(self._exc_info)
  File "/usr/local/lib/python2.7/site-packages/tornado/gen.py", line 1063, in run
    yielded = self.gen.throw(*exc_info)
  File "/usr/local/lib/python2.7/site-packages/salt/minion.py", line 1042, in connect_master
    yield self._post_master_init(master)
  File "/usr/local/lib/python2.7/site-packages/tornado/gen.py", line 1055, in run
    value = future.result()
  File "/usr/local/lib/python2.7/site-packages/tornado/concurrent.py", line 238, in result
    raise_exc_info(self._exc_info)
  File "/usr/local/lib/python2.7/site-packages/tornado/gen.py", line 1069, in run
    yielded = self.gen.send(value)
  File "/usr/local/lib/python2.7/site-packages/salt/minion.py", line 3124, in _post_master_init
    self.functions['saltutil.sync_all'](saltenv=self.opts['environment'])
  File "/usr/local/lib/python2.7/site-packages/salt/modules/saltutil.py", line 850, in sync_all
    ret['clouds'] = sync_clouds(saltenv, False, extmod_whitelist, extmod_blacklist)
  File "/usr/local/lib/python2.7/site-packages/salt/modules/saltutil.py", line 652, in sync_clouds
    ret = _sync('clouds', saltenv, extmod_whitelist, extmod_blacklist)
  File "/usr/local/lib/python2.7/site-packages/salt/modules/saltutil.py", line 99, in _sync
    saltenv = _get_top_file_envs()
  File "/usr/local/lib/python2.7/site-packages/salt/modules/saltutil.py", line 81, in _get_top_file_envs
    top = st_.get_top()
  File "/usr/local/lib/python2.7/site-packages/salt/state.py", line 3089, in get_top
    tops = self.get_tops()
  File "/usr/local/lib/python2.7/site-packages/salt/state.py", line 2787, in get_tops
    saltenv
  File "/usr/local/lib/python2.7/site-packages/salt/fileclient.py", line 189, in cache_file
    return self.get_url(path, '', True, saltenv, cachedir=cachedir)
  File "/usr/local/lib/python2.7/site-packages/salt/fileclient.py", line 495, in get_url
    result = self.get_file(url, dest, makedirs, saltenv, cachedir=cachedir)
  File "/usr/local/lib/python2.7/site-packages/salt/fileclient.py", line 1044, in get_file
    hash_server, stat_server = self.hash_and_stat_file(path, saltenv)
TypeError: 'bool' object is not iterable
```

Idempotent Configuration Merge

```
{% if bgp.local_as is number %}

    no router bgp {{ bgp.local_as }}
    router bgp {{ bgp.local_as }}
        no bgp default ipv4-unicast
        bgp always-compare-med
    [...]
    {% endif %}

    {% for iface in pillar.get('interfacescust', {}) %}

        default interface {{ iface.name }}
        interface {{ iface.name }}
            load-interval 30
    [...]
    {% endfor %}
```

Deployment Workflow

- NAPALM supports config test, config load, commit and rollback
- SaltStack and Ansible support different deployment environments
 - e.g. lab / production, etc
- Good idea to use these mechanisms on production systems

IXP AUTOMATION

```
root@saltmaster:~ # salt swi1-pwt1-1 saltutil.refresh_pillar
[...]
root@saltmaster:~ # salt swi1-pwt1-1 net.load_template \
    template_path=/srv/napalm/templates/eos/templates configure_cust_interfaces.j2 test=true
swi1-pwt1-1:
-----
already_configured:
    False
comment:
    Configuration discarded.
diff:
    @@ -500,7 +500,7 @@
        10 permit 30:b6:4f:e4:f8:f6 00:00:00:00:00:00 any
    !
    mac access-list l2acl-ixp-viid325
    - 10 deny any any
    + 10 permit 01:23:45:67:89:ab 00:00:00:00:00:00 any
    !
    mac access-list l2acl-ixp-viid326
        10 deny any any
loaded_config:
result:
    True
root@saltmaster:~ #
```

IXP AUTOMATION

```
root@saltmaster:~ # salt swi1-pwt1-1 net.load_template \
    template_path=/srv/napalm/templates/eos/templates configure_cust_interfaces.j2 commit=false
swi1-pwt1-1:
-----
already_configured:
    False
comment:
diff:
    @@ -500,7 +500,7 @@
        10 permit 30:b6:4f:e4:f8:f6 00:00:00:00:00:00 any
    !
        mac access-list l2acl-ixp-viid325
    -    10 deny any any
    +    10 permit 01:23:45:67:89:ab 00:00:00:00:00:00 any
    !
        mac access-list l2acl-ixp-viid326
            10 deny any any
loaded_config:
result:
    True
root@saltmaster:~ #
```

IXP AUTOMATION

```
root@saltmaster:~ # salt swi1-pwt1-1 net.load_template \
    template_path=/srv/napalm/templates/eos/templates configure_cust_interfaces.j2 commit=true
swi1-pwt1-1:
-----
already_configured:
    False
comment:
diff:
    @@ -500,7 +500,7 @@
        10 permit 30:b6:4f:e4:f8:f6 00:00:00:00:00:00 any
    !
        mac access-list l2acl-ixp-viid325
    -    10 deny any any
    +    10 permit 01:23:45:67:89:ab 00:00:00:00:00:00 any
    !
        mac access-list l2acl-ixp-viid326
            10 deny any any
loaded_config:
result:
    True
root@saltmaster:~ #
```

IXP AUTOMATION

```
root@saltmaster:~ # salt swt-cwt1-edge1 state.apply cumulus.configure_bgp saltenv=lab test=true
swt-cwt1-edge1:
-----
          ID: /etc/frr/frr.conf
Function: file.managed
    Result: None
Comment: The file /etc/frr/frr.conf is set to be changed
 Started: 08:51:09.367960
Duration: 89.872 ms
Changes:
-----
diff:
---
+++
@@ -16,12 +16,6 @@
    neighbor pg-ebgp-ipv4-ixp description eBGP IXP session policy
    neighbor pg-ebgp-ipv4-ixp timers 3 10
    neighbor pg-ebgp-ipv4-ixp capability extended-nexthop
-   neighbor 10.37.4.1 remote-as 65302
-   neighbor 10.37.4.1 peer-group pg-ebgp-ipv4-ixp
-   neighbor 10.37.4.1 description swt-cwt1-edge2
-   neighbor 10.37.4.3 remote-as 65302
-   neighbor 10.37.4.3 peer-group pg-ebgp-ipv4-ixp
-   neighbor 10.37.4.3 description swt-cwt1-edge2
    neighbor 10.37.2.2 remote-as 65311
    neighbor 10.37.2.2 peer-group pg-ebgp-ipv4-ixp
    neighbor 10.37.2.2 description swt-cwt1-mlnx1
-----
```

IXP AUTOMATION

```
[...]
```

```
-----  
      ID: /etc/frr/frr.conf  
Function: service.running  
    Name: frr  
  Result: None  
Comment: Service is set to be reloaded  
Started: 08:51:09.803190  
Duration: 314.587 ms  
Changes:
```

```
Summary for swt-cwt1-edge1
```

```
-----  
Succeeded: 5 (unchanged=2, changed=1)
```

```
Failed: 0
```

```
-----  
Total states run:      5
```

```
Total run time:   1.872 s
```

```
root@saltmaster:~ #
```

```
root@saltmaster:~ # salt swt-cwt1-edge1 state.apply cumulus.configure_bgp saltenv=lab test=false
```

```
[...]
```

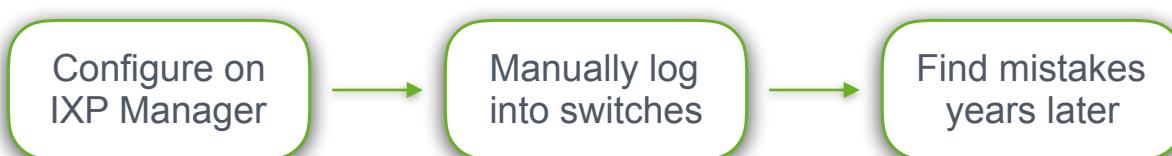
Phase 1 Results

- Configure all IXP participant edge ports [in service using production DB]
- Configure IXP core [in service using pilot model data source]
- Handled INEX LAN1 forklift upgrade successfully
- Operations workflow changed to be safer, simpler and more reliable
- Single source of authoritative data about network configuration

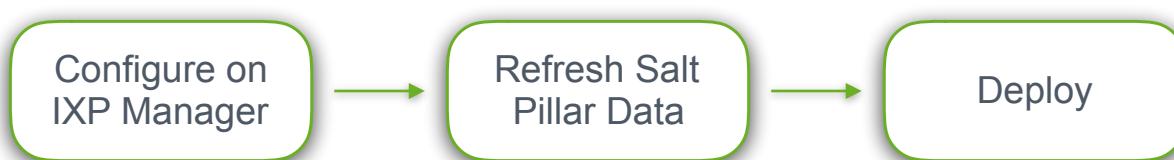
Phase 2 Progress

- Data abstraction model is complete, needs refactoring
- IXP Manager: coding almost complete, needs refactoring
- Templating for NAPALM: Arista: 100%, Cumulus: 95%
- Release as open source: planned in 2017Q4
- Creation of operational workflow procedures

Operations Then



Operations Now



THANK YOU

Any Questions?

visit us online at inex.ie

