CROWN CASTLE

DATA DUCTUS

Learning to Live with Imperfection
(A Service Reconciliation Journey)

Rob Hinst (Crown Castle Fiber)
Scott Barvick (Data Ductus)

NSO Developer Days
JUNE 2020

The Crown Castle Fiber Network

- S&P 500 company
- 80,000 route miles of fiber spread over a dozen legacy networks
- Dark Fiber, IP, Ethernet and Wavelength services
- Numerous device vendors
- Change is constant, fast-paced and not always automated.
- Top-notch engineering and operations teams

Problem and (the start of) a solution

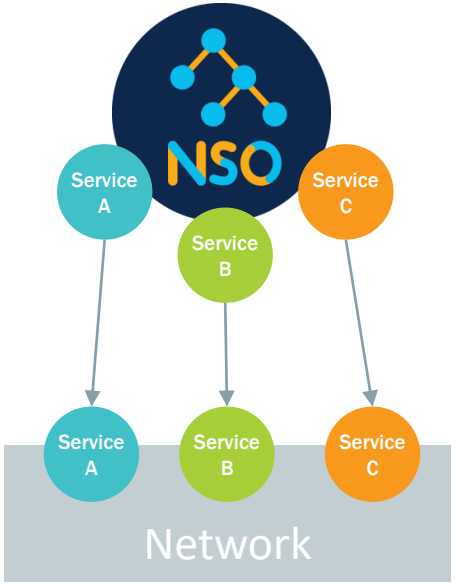
The honeymoon: We develop a bunch of great automation models.

- Everyone loves them
- We're awesome

Challenge: We want to use our new toys on older services configured outside of or before NSO.

Solution: Import services into NSO:

- Determine appropriate service parameters for service creation from OSS database
- Set parameters and commit no-networking
- Away we go!



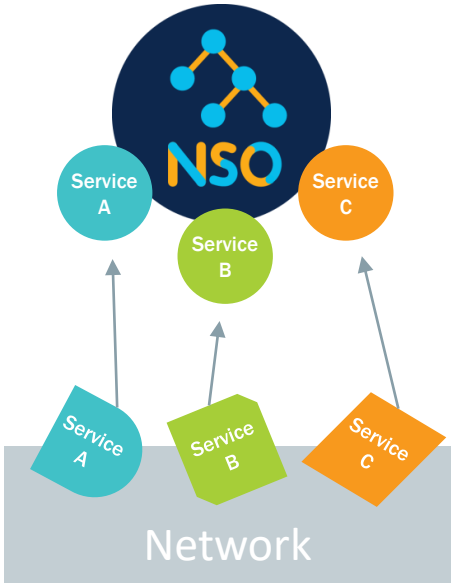
Not so fast!

Many services are out-of-sync.

Can't just re-deploy to network

- Services were imported using service parameters derived from BSS/OSS
- BSS/OSS database might be wrong/out-of-date
- Older records may have important details in free-form notes fields

Figuring out how to make everything "right" is time-consuming and often complicated.



Re-examining our goals

Making everything "right" would be lovely.

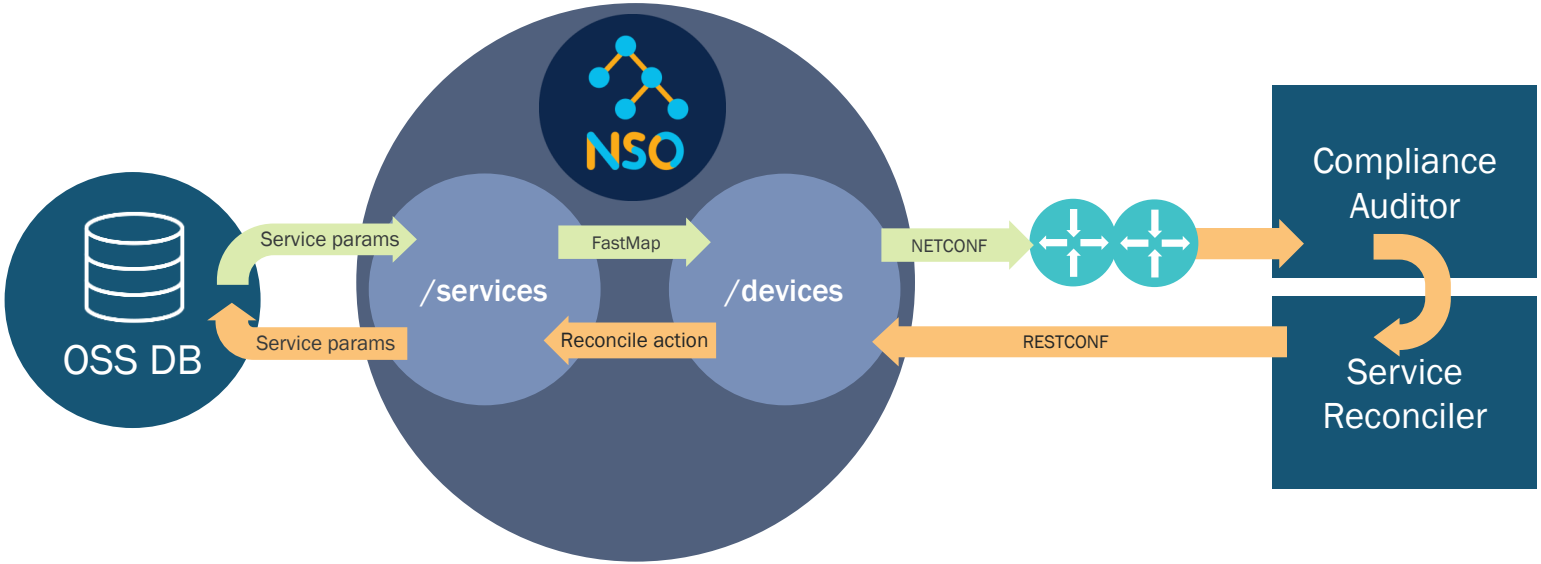
- We'll do that tomorrow.

Our real goal today is to automate more.

- In order to do that, we need to synchronize our data sources:

Lifecycle of a Service

Life is messy. Our lifecycle needs to accept that and work with it.



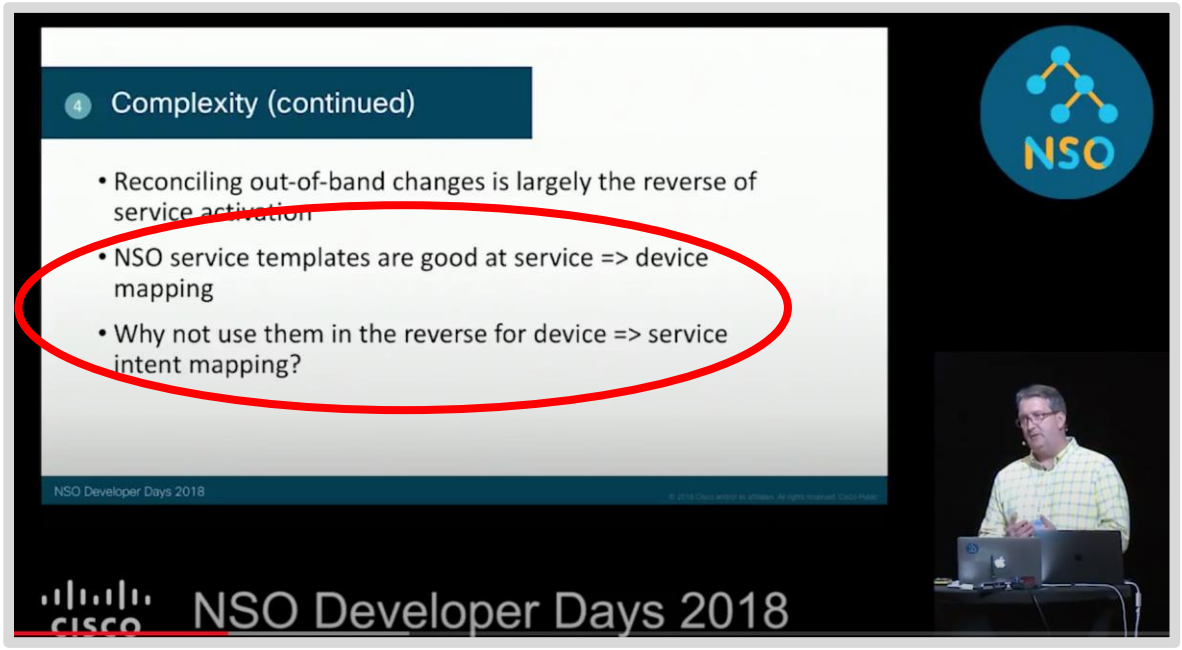
Reconciliation 2017

"The best reconciliation strategy is no reconciliation"

Me

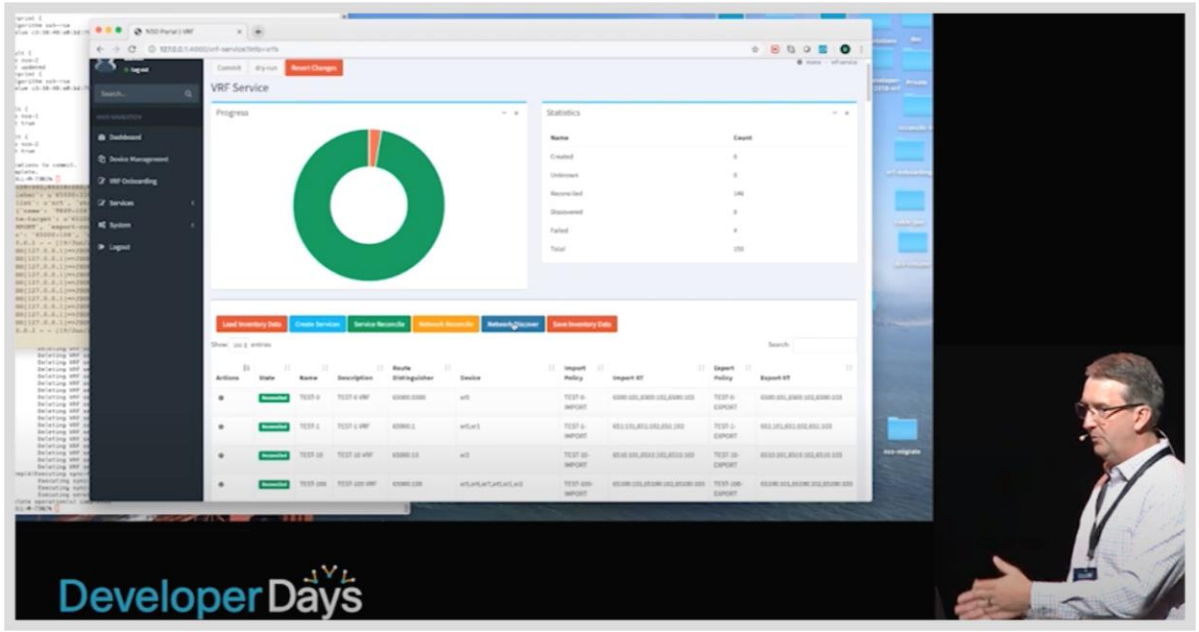
Roque Gagliano: https://www.youtube.com/watch?v=2cD99AJfaVo

Reconciliation 2018



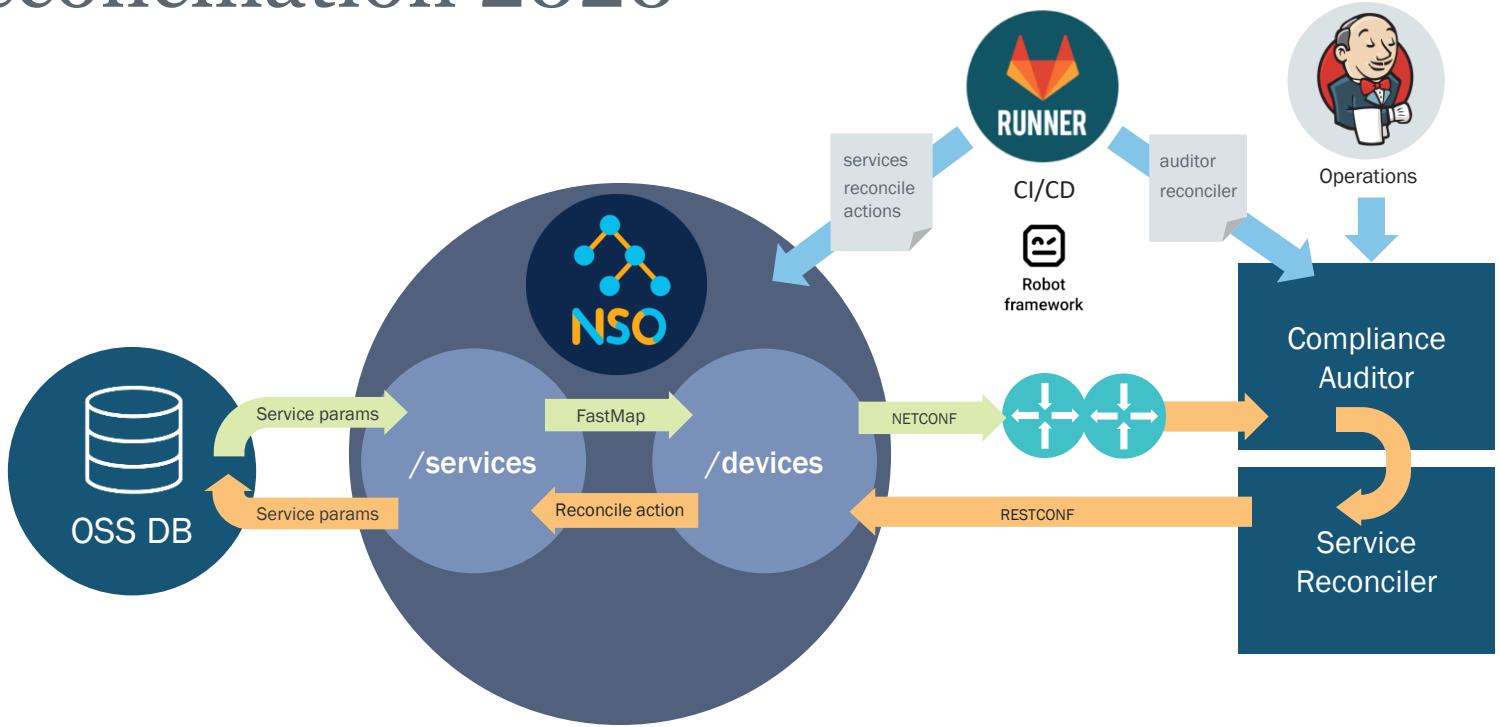
Dan Sullivan: https://www.youtube.com/watch?v=yYzk8aXMCbY

Reconciliation 2019



Dan Sullivan: https://www.youtube.com/watch?v=KudcsCAE-Sw

Reconciliation 2020



Base class with (minimal) per service Python code

Base Reconcile Handler

- Handles the transaction management
- Applies the reconcile template
- Validates the successful reconciliation before commit

```python
import ncs.template
from .BaseHandler import ReconcileHandler

class StaticRouteHandler(ReconcileHandler):
    '''This class implements the staticroute-reconcile action'''

    def set_name(self, inputs):
        self.action_name = "staticroute-reconcile[%s]"%(inputs.uid,)
        self.template_name = 'staticroute-reconcile'

    def get_target(self, root, inputs):
        return root.ncs__services.ltf__lightower.service.staticroute__staticroute[inputs.uid]

    def get_variables(self, root, sn, target_serv, inputs, output):
        variables = ncs.template.Variables()
        variables.add('uid', inputs.uid)
        return variables
```

Per service Python:

1. Method to **set** the log name and the action template name
2. Method to **get** the path to the NSO node for the service instance being reconciled (for dry-run)
3. Method to **get** the variables that are passed to reconcile template

Code can be found at: https://bitbucket.org/dataductus/service-reconcile-base/src

Stacked Services Reconciliation Template

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">

<?save-context BASE?>
<services><circuit><eline>
   <circuit-id>{$CIRC_ID}</circuit-id>
            <a-loc tags="merge">     <!-- top level service reconciliation by leaves in a-loc container -->

                  <!-- Handoff subservice leaves -->
                  <?switch-context BASE?>
                  <?if {boolean(../../ncs:services/ccf:service/handoff:handoff[uid=$A_HANDOFF_UID])}?>
                  <?set-context-node {../../ncs:services/ccf:service/handoff:handoff[uid=$A_HANDOFF_UID]}?>
                  <port-description>{port-description}</port-description>
                  <?end?>

                  <!-- Policer subservice leaves -->
                  <?switch-context BASE?>
                  <?if {boolean(../../ncs:services/ccf:service/policer:policer[uid=$A_POL_UID])}?>
                  <?set-context-node {../../ncs:services/ccf:service/policer:policer[uid=$A_POL_UID]} ?>
                  <policer-name>{ports[hostname=$A_POL_HOSTNAME and port=$A_POL_PORT]/policer-name}</policer-name>
                  <ingress-acl-name>{ports[hostname=$A_POL_HOSTNAME and port=$A_POL_PORT]/acl-name}</ingress-acl-name>
                  <policer-rule-number>{ports[hostname=$A_POL_HOSTNAME and port=$A_POL_PORT]/rule-number}</policer-rule-number>
                  <port-description>{port-description}</port-description>
                  <?end?>

                  <!-- ***** Other subservices contributing to the a-loc container ***** -->

            </a-loc>
</eline></circuit></services></config-template>
```

Summary

- **Uncertainty in the network will continue to exist, but it can be managed**
- **The tools and patterns for solutions are maturing to the point where many different use cases can be supported**
- **Reconciliation 2021?**
  - Built-in reconciliation callback function?
  - Finding what's *not* supposed to be there?
  - Auto-discovering contexts based on "anchor" leafs?

Thank you

For further information please contact:

Rob Hinst, robert.hinst@crowncastle.com
Scott Barvick, scott.barvick@dataductus.com