NETWORK CONTROL SYSTEM:
A POWER TOOL FOR NETWORK ENGINEERS
Advanced networking services are complex to provision, often requiring changes to multiple devices and large numbers of parameters per device. Distributed changes often need to be made in lockstep, because if a single change fails, it can disrupt service and be difficult to diagnose and recover from.

Tail-f’s Network Control System (NCS) addresses this problem. Tail-f NCS acts as the interface to the network for human operators and orchestration and business systems:

Network engineers use Tail-f NCS in two ways:

1. **Using NCS as a central point of management for the network**
   Tail-f NCS provides engineers with a network-wide command line interface (CLI). All devices and services in the network can be accessed and manipulated using this CLI. This makes NCS a powerful tool for network engineers while also defining "guard rails" that can limit what junior engineers can do.

Typical Workflow
   NCS stores the configurations of all devices and all services in its database, enabling the network CLI to support management of devices and services.
Candidate Configuration: Changes are made to a copy of the NCS database, which is referred to as the candidate configuration.

- Validate Command: Ever so often, a validate command is issued to check that any changes made so far are valid with respect to network policies and integrity constraints.

Committed: Finally, the changes are committed. At this point, changes are copied to the NCS database and pushed out to the network. Changes that violate integrity constraints or network policies cannot be committed. Changes to the devices are done in an atomic distributed transaction, which ensures that either all changes take place or nothing is changed. In effect, NCS provides atomic change sets using transactional coherence across devices.

- If later it turns out that the changes had unforeseen consequences, the changes can then be rolled back using a single command.

Device Operations
These examples of device operations can be completed using CLI commands in NCS:

- Copy and paste the configuration of another device or from a template to quickly provision a new device.
- Using distributed transactions, deploy configuration changes to multiple devices in a fail-safe way.
- Validate the integrity of configurations before deploying to the network.
- Apply configuration changes to named device groups.
- Easily roll back changes, if needed.
- Perform configuration audits to check if device configurations are in sync with the NCS database. If they not, what is the diff?
- Synchronize the NCS database and the configurations on devices, in case they are not in sync. This can be done in either direction: import the diff to the NCS database or deploy the diff on devices.

Service Operations
These examples of service operations can be completed using CLI commands in NCS:

- What-if analyses ("dry runs"): What are the effects on device configurations if a service is created or updated?
- Create or update services in a fail-safe way using distributed transactions. The system first automatically calculates what device configuration changes correspond to the service operation and then deploys these configuration changes.
- Easily roll back service updates, if needed. The device configurations associated with the service update are automatically cleaned up. Similarly, retiring a service automatically cleans up the associated device configurations.

Troubleshooting
To help with troubleshooting, NCS keeps track of bidirectional mappings between device configurations and services:

- Straightforward service impact analysis: Given a device configuration element, it is easy to see which services use it. This makes service impact analysis straightforward.
and easy to see services that are affected if a device goes down or if a device configuration is changed.

- **Root cause analysis**: Given a service, it is easy to see which device configuration it uses. This simplifies root cause analysis to see which devices are used by a faulty service and if these devices are potential culprits of the service fault.

### 2. Using NCS for quickly developing automation applications

#### Framework Capabilities

Any framework for developing network automation applications needs to provide the following capabilities:

- Definition of network services (e.g., VPNs, abstract ACLs, BGP peering policies).
- Methods for creating, modifying and deleting services, then mapping these service-level operations to device configuration changes.
- Deployment of updated device configurations.

NCS provides these capabilities as follows:

- Services are defined in YANG (RFC 6020).
- Engineers only have to define a template that defines how the service parameters are mapped to the device configuration parameters. Based on this template, NCS manages the whole life-cycle to create, change and delete services in the network.
- Deployment of updated device configurations is completed in a fail-safe way using distributed transactions. This step requires no programming in the case of Juniper, Cisco and SNMP-based devices, and little programming with many other types of devices.

#### Iterative Development Process

NCS supports an iterative development process, as illustrated in the following figure:
The following are examples of network automation applications that have been developed in NCS:

In typical cases, a first iteration of a complete system is completed in days – not weeks or months.

- VPN provisioning and configuration
- BGP routing policy configuration
- Adding border routers and route reflectors
- Turning up new backbone circuits
- Switch port provisioning