NAF
NETWORK AUTOMATION FORUM

linx

TEN YEARS OF NETWORK AUTOMATION AT LINX

Riccardo Verzeni

Director of Software Engineering

AUTOCON 5

THE NETWORK AUTOMATION CONFERENCE

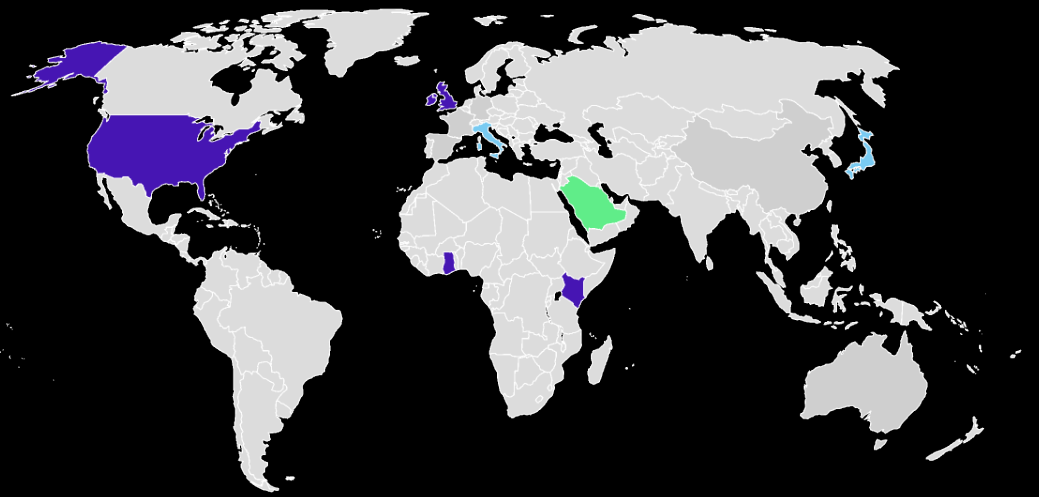
LINX Network Environment

- Large Internet Exchange Point infrastructure
- Multiple switching fabrics and vendors
- Thousands of BGP sessions and member connections
- Continuous operational changes and service provisioning
- Automation required to scale operations safely

LINX IXPs

LINX IXP Partners

IXPs Powered by LINX



What does an IXP actually configure?

Customer Edge Interfaces

- Customer port provisioning
- MAC ACLs for port security
- Rate limiting and traffic controls
- Quarantine / validation testing before activation

Peering Infrastructure

- Thousands of BGP sessions on route servers
- Per-member routing policies

Core Fabric

- Inter-switch links (ISLs) provisioning
- VLAN and fabric configuration
- Network state consistency across multiple switches

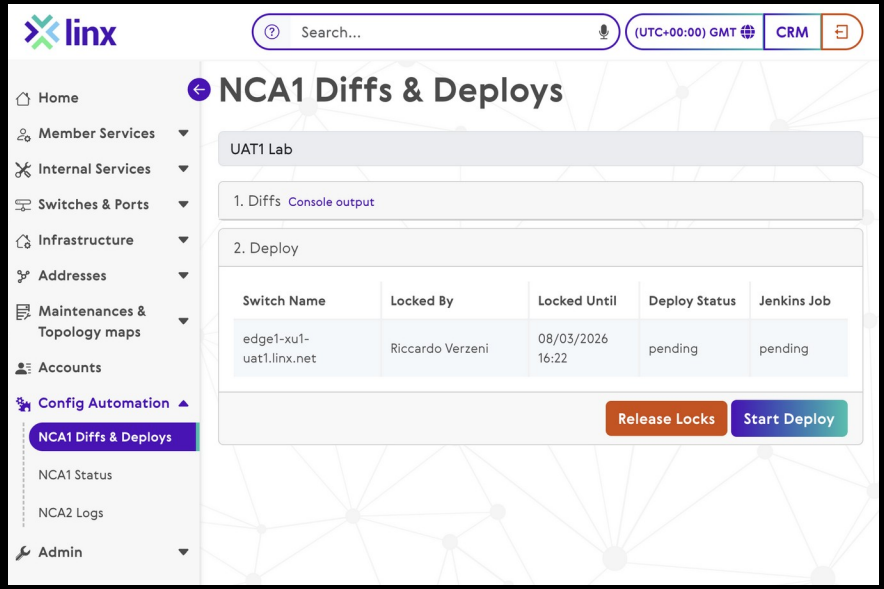
The Starting Point

- Network configuration changes performed manually
- CLI driven operational workflows
- High operational overhead
- Difficult to audit and review changes
- Scaling network growth required automation

```
root@Router7> show configuration
## Last commit: 2019-07-09 20:33:24 UTC by root
version 12.1X47-D15.4;
system {
    host-name Router7;
    root-authentication {
        encrypted-password "$1$1gKc/rPP$YKMuctMztj.XZ9N7vShCD/"; ## SECRET-DATA
    }
    services {
        ssh;
        web-management {
            http {
                interface ge-0/0/0.0;
            }
        }
    }
    syslog {
        user * {
            any emergency;
            match "!(.*Scheduler Oinker*.|.*Frame 0*.|.*ms without yielding*.)";
        }
        file messages {
            any any;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
    license {
        autoupdate {
            url https://ae1.juniper.net/junos/key_retrieval;
        }
    }
    processes {
        jsrp-service disable;
    }
```

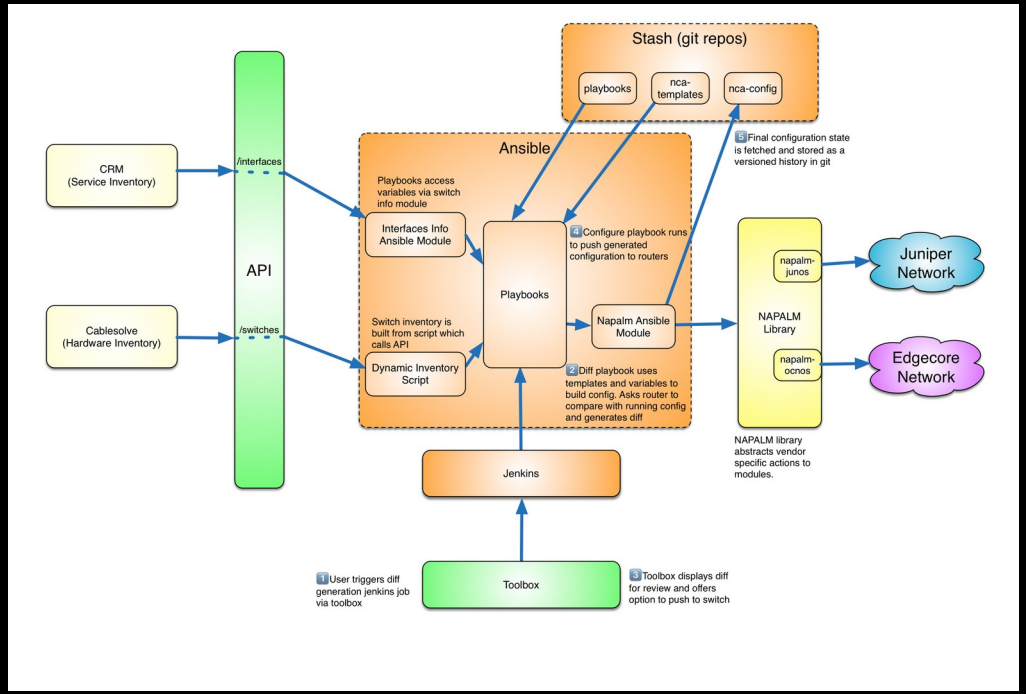
Phase 1 - Network Configuration Automation (NCA)

- First generation automation platform
- Generate configuration from structured data
- Templates used to build device configuration
- Diff generated before deployment
- Controlled and safe deployment pipeline



NCA Architecture

- Inventory and data APIs as source of truth
- Templates generated with Jinja2
- Automation executed via Ansible and NAPALM
- Deployment pipeline orchestrated through CI tooling: Jenkins
- Full logging and configuration history in git



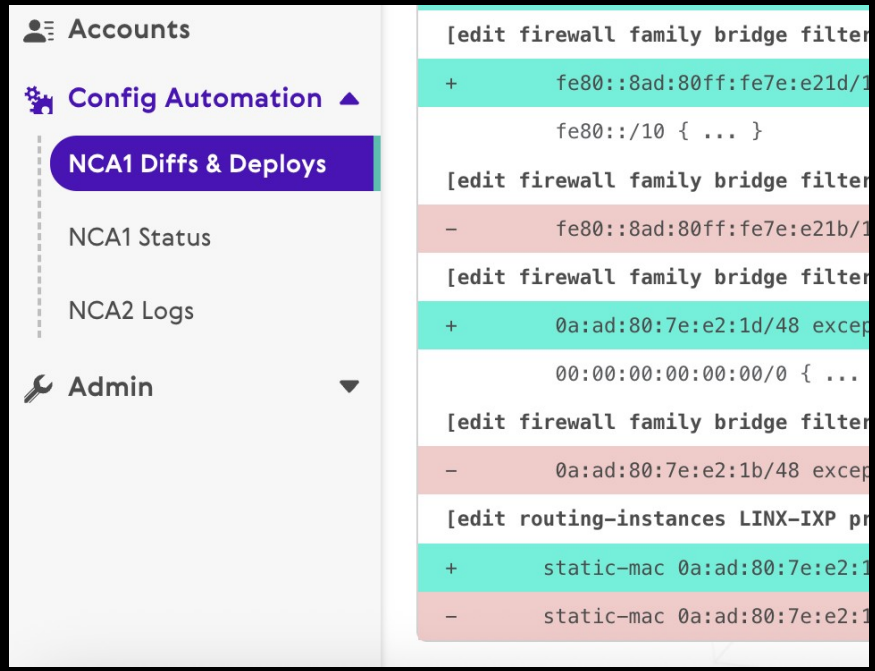
Why NCA Was Important

Configuration as Code

- Deterministic configuration generation from structured data
- Standardised configuration templates for all customer interfaces

Safe Change Management

- Reviewable diffs generated before deployment
- Automated validation and safeguards in the deployment pipeline



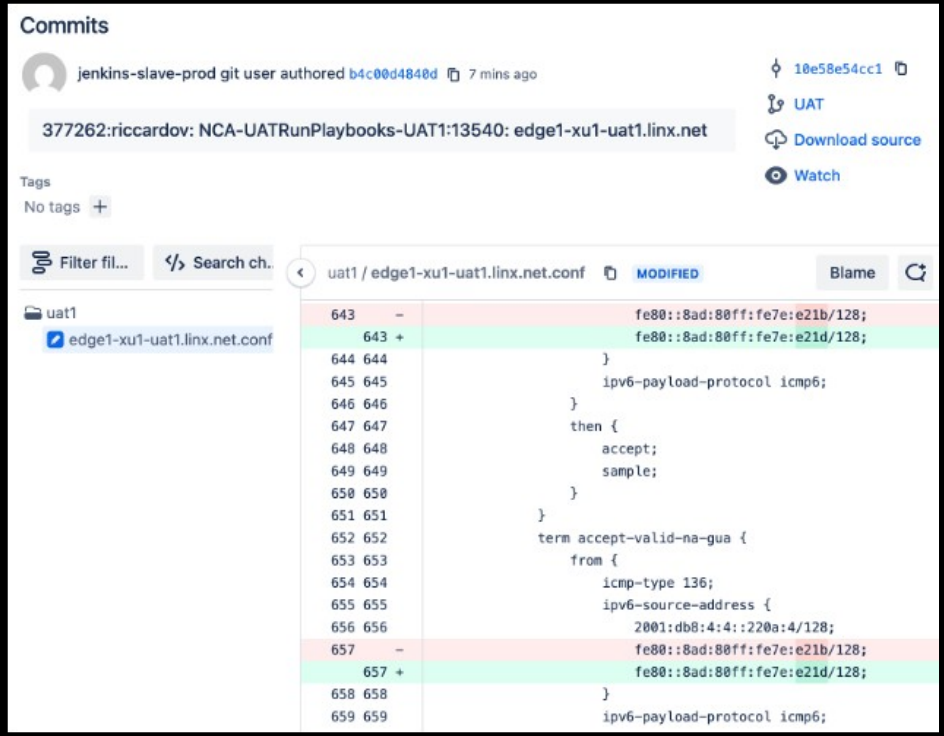
Why NCA Was Important

Operational Visibility

- Full configuration history and auditability via Git
- Traceable change history across the network

Operational Impact

- Significant reduction in manual CLI changes
- Safer, repeatable network operations



Lessons from Phase 1

- Automation requires reliable data models
  - We later created bespoke SoT (TSI) because at the time no suitable solution was found available on the market
- Cross-team collaboration between network and software
- Testing environments are critical
  - Split between DEV/UAT/PROD environment

Lessons from Phase 1

- Safety mechanisms are essential:
  - locking mechanisms
  - serialized deploys
  - validation checks
  - logs and audit trails
- Automation must integrate with operational processes (E.g: Network maintenances)

Our Source of Truth: TSI & ENG-UI

- **TSI** – our bespoke Source of Truth, holding inventory and service data; It represents the desired state of the network.
- **ENG-UI** – the engineering UI on top of TSI, used by NOC and engineers to perform network-state changes before applying them and to provision services manually
- **Why bespoke** – we had a telco's data and technical workflows but not its scale, and nothing on the market fit at the time (today we'd probably use NetBox or similar)

Phase 2 - Next Challenge: Member Driven Automation

- Limitations of the first generation (NCA)
  - Reliable and safe, but optimised for bulk operational changes
  - Execution model too slow for real-time member requests
  - Required operator supervision
- New operational requirement
  - Member self-service through the portal
  - Faster provisioning workflows
  - Unsupervised automation for low-risk changes

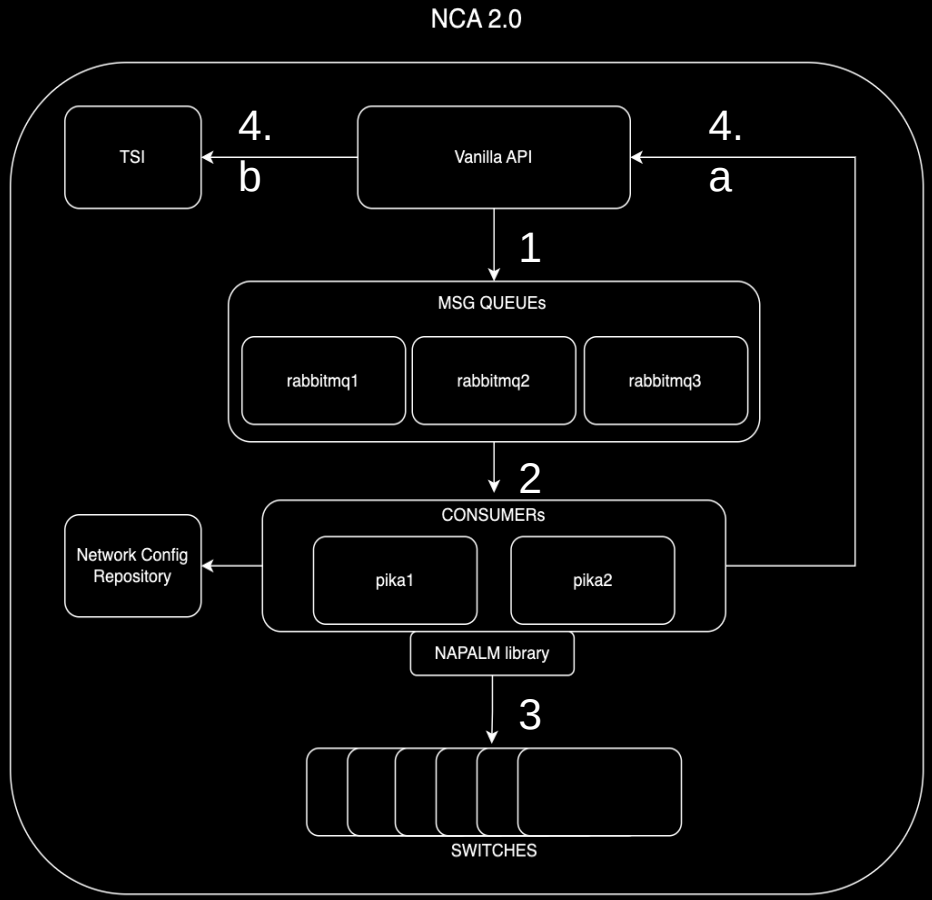
NCA 2.0

NCA	NCA 2.0
supervised	unsupervised
synchronous	asynchronous
internal ops	member facing

NCA 2.0 was designed specifically for low-risk automated configuration merges triggered by customer portal.

NCA 2.0 Architecture

1. API Action
↓
2. Message Queue
↓
3. Worker deploys change
↓
4. Inventory updated

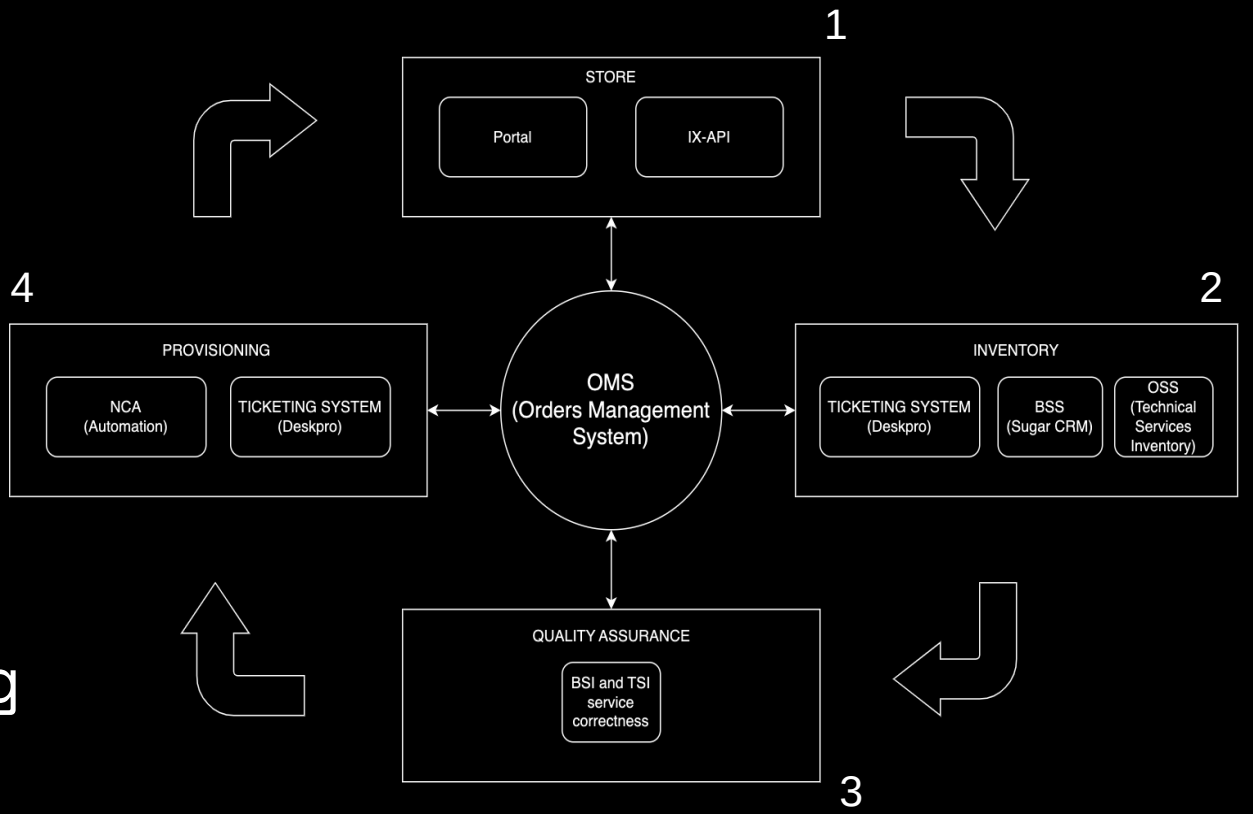


Phase 3 - Service Automation

- Configuration automation alone was **not sufficient**
- Service provisioning required coordination across multiple systems
- Systems involved:
  - Member portal (and IX-API)
  - BSS inventory (CRM)
  - OSS inventory (SoT) (TSI / Cablesolve)
  - Ticketing systems

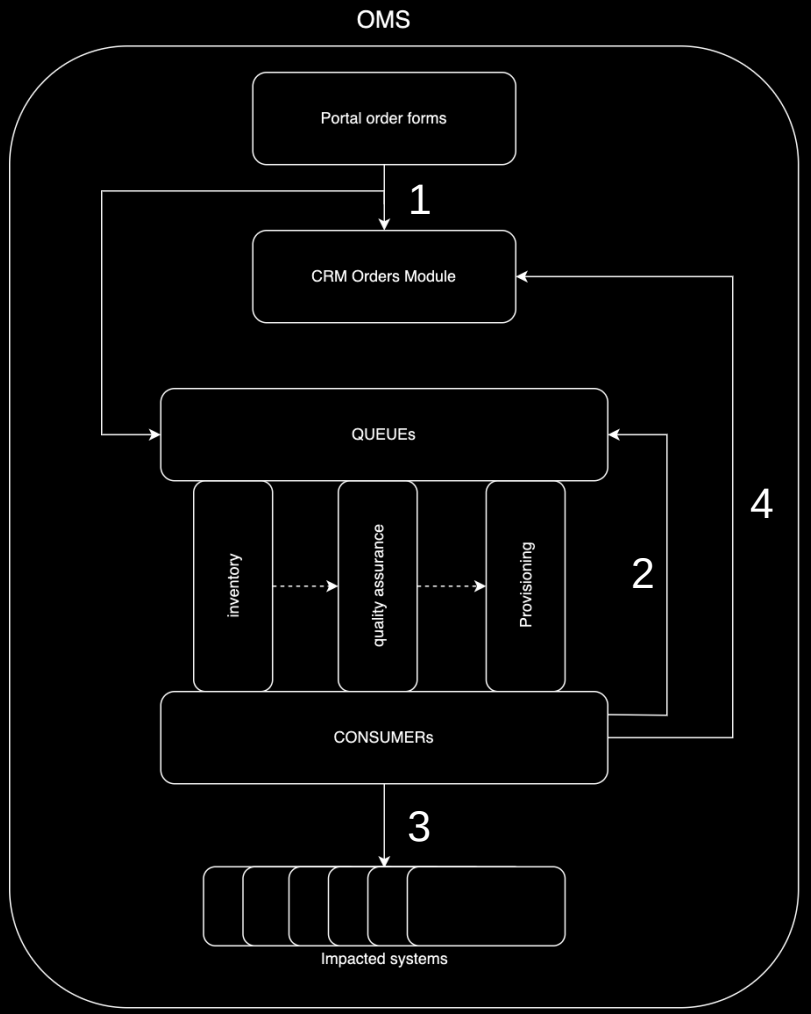
Order Management System (OMS)

1. Store
   ↓
2. Inventory
   ↓
3. Quality Assurance
   ↓
4. Provisioning



OMS Architecture

1. Order submission
↓
2. OMS processes order through lifecycle queues
↓
3. Execution on impacted systems
↓
4. Status updates



From OMS to Self-Service

With OMS in place, we could finally open provisioning up to members

- **In practice, provisioning ran through ENG-UI** — NOC carried out the provisioning workflows by hand and deployed the changes via NCA
- **With OMS we started cannibalising the manual steps** — automating them away one by one, moving NOC from doing the work to handling exceptions
- **Then we moved to the harder cases** — some workflows can't be fully automated — which is where hybrid automation comes in

Hybrid Automation: The Pause-Resume Model

- Some service workflows require external actions before automation can proceed.
- Pause-Resume automation model
  - OMS pauses the workflow when external action is required
  - Automation resumes automatically once the condition is satisfied
- Actors involved
  - Members via Portal
  - Operations via ENG-UI – the engineering UI on top of TSI

Hybrid Automation: Use Cases

- **Quarantine testing**
  - Members must verify configuration on their router before activation
- **Private VLAN provisioning**
  - Automation pauses until the invited member accepts the connection
- **Physical port provisioning**
  - NOC performs patching or ODF configuration before deployment

Example: Automated Quarantine Testing

- Before activating a new peering service — or when modifying critical parameters (e.g. MAC address) — the port is placed in a **quarantine VLAN**.

- Automated workflow
  - Port provisioned in **quarantine mode**
  - OMS **pauses provisioning workflow**
  - Member validates router configuration
  - Member resumes provisioning via **Portal**
  - Automation completes service activation

Route Server Automation (RSA): Scope & Architecture

- **Automation beyond switching**
  - Management of thousands of BGP sessions
  - Automated route server configuration
- **Shared automation architecture**
  - Built on the same **event-driven messaging system**
  - Asynchronous execution model

Route Server Automation (RSA): Config Model & Impact

- **Different configuration model**
  - Configuration **compiled dynamically**
  - Combines internal inventory with public routing datasets
- **Operational impact**
  - Reduced manual management of BGP sessions
  - Consistent route server configuration across fabrics

LINX Network Automation Ecosystem

SERVICE INTERFACES
Member Portal · IX-API · ENG-UI

↓

ORCHESTRATION
OMS — service lifecycle management

↓

AUTOMATION ENGINES
NCA · NCA 2.0 (switching) · RSA (route servers)

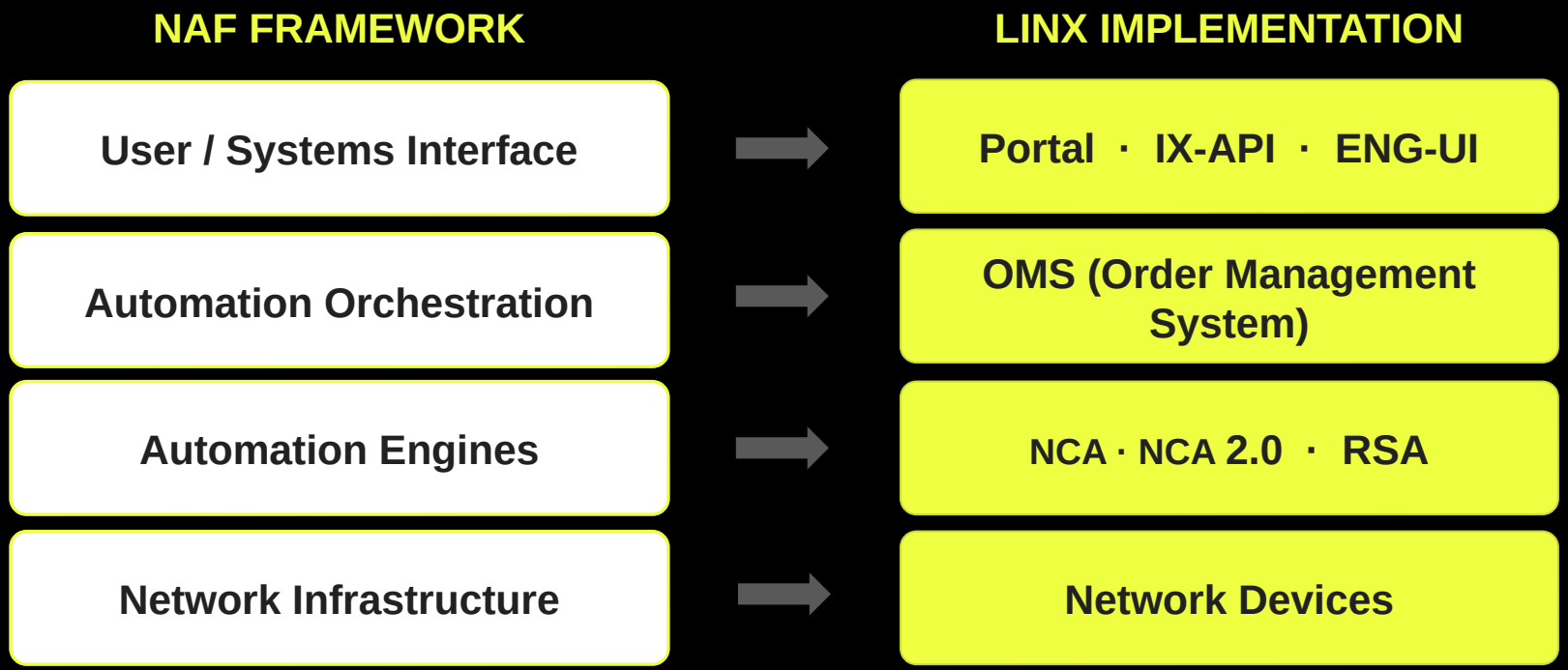
↓

NETWORK INFRASTRUCTURE
Switching fabric · Route servers

SUPPORTING SYSTEMS

CRM / BSS
OSS inventory
Ticketing

Mapping Our Architecture to the NAF Reference Framework



Lessons from 10 Years of Automation

1. **Data is the foundation** — reliable inventory and service data are what everything else is built on
2. **Automation evolves incrementally** — capability compounds; start small and earn trust in each next phase
3. **Think beyond the network** — real service delivery needs orchestration across systems
4. **Hybrid wins** — manual and automated workflows coexisting is what builds resilience

DO

From principles to practice — how to put the lessons to work

- **Separate commercial & technical data / systems** — but keep them loosely coupled so you can audit and reconcile across both
- **Automate with software best practices** — version control, DEV/UAT/PROD, CI/CD, automated testing; no unmaintained scripts running on someone's laptop
- **Start from the most common use case** — leave the rest manual, then expand progressively through trial and error
- **Match architecture to the operation** — synchronous for sporadic ops, asynchronous (queues + consumers) for frequent, multi-user ones like self-service

DON'T

From principles to practice — what to avoid

- **Don't build bespoke by default** — build only what you need; if an open-source tool fits or is built to extend, use it
- **Don't work in isolation** — bring every stakeholder in from day one and keep regular show-and-tells as you build
- **Don't over-engineer error handling** — defaulting to a human for the corner cases is usually the most cost-effective option
- **Don't underestimate guardrails** — for catastrophic-risk actions, hash-verify pushed files and cut impact radius, or keep a human in the loop


NAF
NETWORK AUTOMATION FORUM
Questions?