Build the Assembly Line, Not the Car

Zero Touch Provisioning as a Foundational Capability for Repeatable Physical Network Deployment

Michael J Martin

September 2026

outlan.net

Abstract

Network automation programs often begin with configuration management or infrastructure-as-code tools aimed at devices that are already reachable, identified, and represented in an authoritative data model. Those tools deliver real value, but they do not by themselves solve the day-0 problem of bringing factory-default physical equipment into a known, authenticated, and governed state. This paper argues that Zero Touch Provisioning (ZTP) should be treated as a foundational bootstrap and lifecycle capability for repeatable physical network deployment—not as the sole foundation of network automation. ZTP must operate with, and remain distinct from, a source of truth, configuration management, orchestration, identity services, assurance, telemetry, compliance, and day-2 lifecycle management. The paper preserves a discovery-first path for brownfield environments, defines a gated management-underlay-overlay deployment sequence, and turns the assembly-line metaphor into an actionable operating model. It also establishes implementation stages, decision rights, failure recovery, and an operational scorecard. The objective is not “zero humans.” It is a controlled production system in which human judgment is captured in approved intent, templates, policy, and exception handling rather than repeatedly improvised at each device.

EXECUTIVE ARGUMENT

The distinction at the center of this paper is simple. Configuration management automates change on a device the organization can already reach. Orchestration coordinates work across systems. A source of truth records approved intent and authoritative attributes. Assurance compares observed state with that intent. ZTP addresses an earlier transition: how a factory-default or reset physical device acquires enough network reachability and trusted identity to enter those governed systems.

That transition makes ZTP foundational for repeatable physical network deployment, but it does not make ZTP the foundation of every automation concern. A reliable network automation capability is a system of capabilities. ZTP is the controlled entry point. The source of truth is the approved bill of materials and production specification. Configuration management applies the detailed build. Orchestration sequences dependencies. PKI and AAA establish and enforce trust. Telemetry and assurance test the output. Compliance determines whether the output may remain in service. Lifecycle management governs software, maintenance, replacement, and retirement.

The assembly line is therefore larger than ZTP, even though ZTP is the station that turns an unknown box into a governed production object. If that first station is absent, engineers must create reachability and identity by hand before higher-order automation can start. If the remaining stations are absent, ZTP can bring a device online quickly but cannot prove that the device is correct, secure, or ready to carry a service. The goal is the line, not a single machine on it.

The executive decision is not whether to buy a “ZTP product.” It is whether to fund and govern a repeatable deployment capability. That decision requires an accountable sponsor, a bounded pilot, explicit dependencies, measurable gates, operating ownership, and a contract for exceptions and recovery. Without those elements, a technically successful demonstration can still become an operational orphan.

THE BOOTSTRAP PROBLEM DEVOPS DOESN'T SOLVE

Ansible and Terraform are both excellent tools within their intended operating models. Ansible commonly applies idempotent configuration to a target it can reach. Terraform commonly declares desired resource state and reconciles it through a provider API. Implementations vary, and either tool can participate in a broader bootstrap workflow, but neither eliminates the underlying requirement for a reachable endpoint, an accepted identity, a usable control path, and authoritative input data.

Cloud platforms abstract much of the initial reachability problem. A provider control plane can create a compute or network resource, assign identifiers, attach it to a virtual network, and expose APIs for subsequent management. That abstraction materially changes the bootstrap experience; it does not make bootstrap, identity, dependencies, or failure disappear. The provider API, tenancy model, identity and access controls, image provenance, service quotas, regional control-plane availability, private endpoints, name resolution, routing, and the automation runner itself can all become preconditions or failure domains. Cloud automation starts at a higher level of abstraction, not outside the laws of dependency and recovery.

Physical network infrastructure exposes more of the first-mile problem. A switch or router may arrive with a factory-default configuration, no organization-assigned management address, and no credential or certificate that the enterprise is prepared to trust. It may be connected at a site where DHCP, DNS, time, routing, or bootstrap services are unavailable or incorrectly scoped. Before ordinary configuration management can act, something must establish a constrained path from that device to trusted bootstrap services and determine what the device is allowed to become.

Historically, an engineer with a laptop and console cable performed this transition. That method remains a valid recovery path, but it does not scale predictably across repeated sites and hardware replacements. It also makes the quality of the result depend heavily on local access, individual memory, and the currency of a runbook.

ZTP formalizes the transition from “present but not yet governed” to “known and eligible for managed configuration.” A sound implementation does not equate network presence with trust. It establishes provisional connectivity, validates device identity and eligibility, retrieves only approved artifacts, records evidence, and admits the device to subsequent stages only after explicit checks pass. Configuration management then takes over the detailed day-1 build, and day-2 systems maintain and verify the result.

TERMS AND TOOL BOUNDARIES

The phrase “zero touch” is often used loosely. Precision matters because ambiguous boundaries create duplicated controls, unowned failure modes, and unrealistic expectations.

Lifecycle terms

Day-0 bootstrap is the constrained process by which a factory-default or reset device obtains provisional reachability, presents or acquires identity, is matched to an approved record, and receives the minimum configuration needed to enter governed management. In this paper, ZTP is the automation capability that controls that process and produces evidence of its result.

Plug-and-play or onboarding is a broader user or vendor term for simplifying device admission. It may include claim codes, controller discovery, inventory registration, licensing, or guided workflows. It can be part of ZTP, but “plug-and-play” does not necessarily imply the identity validation, source-of-truth authority, stage gates, or recovery discipline required here.

Day-1 configuration applies the approved device role, software baseline, interfaces, routing, policy, and service-specific configuration after trusted management is available. Configuration-management tools are well suited to this work.

Day-2 operations include monitoring, assurance, incident response, compliance, drift handling, capacity management, software maintenance, certificate rotation, change, replacement, and retirement. ZTP can support replacement and rebuild, but it does not own this entire lifecycle.

Modeling terms

In this paper, a **device** is a specific managed network hardware instance. **Intent** is the approved outcome or policy the organization expects. A **configuration** is a concrete set of parameters intended to realize that intent on a supported platform. A **template** is a version-controlled transformation that combines approved data and logic to produce configuration or another deployable artifact. **State** is the condition actually observed at a point in time; it may or may not conform to intent.

A **service** is a consumable network outcome delivered to a user, application, device class, or site. A **capability** is the reusable combination of people, process, data, and technology that can deliver or govern one or more services. The **management plane** is the set of paths and controls used to administer devices. The **underlay** is the infrastructure connectivity on which higher-level services depend. An **overlay** is a logical service topology and policy domain built over that underlay. Not every service requires a distinct overlay, and not every platform implements overlays in the same way; separation should follow risk, routing, scale, and operational requirements rather than vocabulary alone.

Capability ownership

Table 1. Capability boundaries across the automation lifecycle

Capability	Primary responsibility	What it does not prove by itself
ZTP / day-0 bootstrap	Establish provisional reachability, validate eligibility and identity, apply the minimum governed bootstrap state, and record the outcome	That full device configuration and services are correct
Source of truth	Hold approved intent, identity, role, addressing, topology, ownership, and lifecycle attributes under change control	That recorded values match the live environment
Configuration management	Render and apply approved day-1 or day-2 configuration to reachable, supported targets	That the target was admitted correctly or that cross-system sequencing is safe
Orchestration	Coordinate dependencies, transactions, approvals, gates, and compensating actions across tools and domains	That each underlying tool or data set is correct
PKI and AAA	Issue, validate, authorize, rotate, revoke, and audit identities and administrative access	That a device is assigned the correct business role or topology
Telemetry and assurance	Observe state, test reachability and service behavior, and compare evidence with intent	That remediation is authorized or safe
Compliance and lifecycle management	Define acceptable baselines and govern maintenance, exceptions, replacement, and retirement	That day-0 bootstrap is available or repeatable

The boundaries are interfaces, not walls. For example, ZTP may write an observed serial number and bootstrap result back to the source of truth, while the source of truth remains authoritative for whether that serial number is approved for a particular site and role. Assurance may trigger orchestration to rebuild a device, while the service owner retains authority to decide whether replace-not-patch is appropriate.

WHERE ORGANIZATIONS ACTUALLY START: DISCOVERY FIRST

A greenfield program can define its data model, approved hardware, address plan, identity dependencies, and templates before the first device is installed. A brownfield environment usually cannot assume that starting point. Its current topology may reflect years of local change, acquisitions, emergency work, inconsistent documentation, and device-specific exceptions. For such an environment, the responsible transition is discovery first.

Discovery is not the same as indiscriminate data collection, and it is not an excuse to postpone decisions indefinitely. It is a bounded reconciliation activity that identifies in-scope devices and services, captures provenance, compares documented and observed state, classifies uncertainty, and produces an accepted baseline. Inventory platforms, IP address management, configuration archives, controller data, flow records, routing tables, neighbor discovery, authentication logs, and field validation can all contribute evidence. No single feed should be treated as authoritative merely because it is machine-readable.

The source-of-truth argument is not “install a database before automating.” It is “establish controlled authority before allowing automation to act.” Each required attribute needs an owner, a schema, an allowed-value policy, a provenance rule, and a change path. A device record should distinguish approved intent from observed facts. If the intended site role says access switch but the observed hardware, uplinks, or configuration indicate something else, the discrepancy must become an exception; it must not be silently overwritten in either direction.

From discovery to an authoritative baseline

A discovery-first transition should produce five concrete outputs:

1. **An in-scope population.** Every device and site is included, explicitly excluded with a reason and owner, or placed in a time-bounded unknown queue.
2. **A minimum authoritative schema.** At a minimum, the program identifies the attributes required to bootstrap safely: device identity, hardware and software support class, site, role, management domain, addressing intent, template version, service owner, lifecycle status, and exception status.
3. **Reconciliation evidence.** Each critical value has a provenance record and a last-validation time. Conflicts are visible and assigned.
4. **Data acceptance tests.** Referential integrity, uniqueness, allowed values, mandatory fields, stale records, address conflicts, and device-to-site consistency are tested automatically where practical.
5. **Operational ownership.** Named roles approve changes, resolve discrepancies, and maintain data after the discovery project ends.

The transition to ZTP should begin with the subset of the environment for which these outputs are trustworthy, not wait for universal perfection. A complete enterprise inventory may take time; a pilot still requires a complete and accepted record for every pilot device, dependency, and site. Scope can be narrow. Authority cannot be ambiguous.

Incentive alignment and deliverable governance

Discovery and automation engagements contain a familiar principal-agent problem. The client, as principal, wants a durable internal capability and an accurate, transferable result. A supplier, as agent, may be measured by a statement of work, utilization, milestones, or time. The parties can be competent and acting in good faith while still optimizing against different definitions of completion. Information asymmetry makes the problem harder: the supplier sees the technical detail first, while the client may not know whether an apparent delay reflects legitimate complexity, unresolved client dependencies, or an underspecified deliverable.

The answer is not to presume misconduct. It is to make completion verifiable. Before work begins, the contract or internal charter should define:

- the in-scope population and a controlled method for changing it;
- the authoritative schema, required fields, provenance, and accuracy sampling method;
- machine-readable export and API access in formats the client can use without proprietary intervention;
- acceptance tests for completeness, uniqueness, referential integrity, freshness, and reconciliation;
- delivery of source code, templates, pipelines, data dictionaries, credentials under client control, runbooks, and architectural decisions;
- knowledge-transfer outcomes demonstrated by the client team operating, extending, and recovering the capability;
- open-defect thresholds, exception ownership, remediation periods, and warranty or support boundaries;
- milestone payments tied to accepted evidence rather than elapsed effort alone; and
- a clean distinction between completion of the deliverable and any separately purchased managed service.

A source of truth is not finished because it has records in it. It is finished for an agreed scope when the client can query it, test it, change it through governance, trace critical values to evidence, and run the deployment process without the delivery team in the room. That standard protects both parties: it exposes partial completion, but it also prevents the client from withholding acceptance based on requirements that were never defined.

BUILDING THE ASSEMBLY LINE, NOT THE CAR

The governing failure is building the thing once by hand instead of building the process that can build it repeatedly. It is the difference between building a car and building the assembly line that produces cars. When each site is treated as a fresh craft project, local success can conceal system failure. Every new exception adds another version of the product that operations must understand, secure, upgrade, and recover.

ZTP is the first mechanized station on the network deployment line, not the entire factory. It is valuable because it transforms a designed site pattern into a repeatable admission and bootstrap process. The first site still requires judgment: hardware qualification, addressing design, identity architecture, template logic, service policy, failure testing, and operating decisions. The hundredth site should require less reinvention because those decisions have been captured in controlled data and code.

Repeatability does not mean identical output. A production line can support defined product variants. A network deployment capability can support site classes, hardware families, regional requirements, and approved service combinations. The requirement is that variation be represented as governed input or a versioned template—not as an undocumented console change.

The atomic unit of value is therefore not a single device provisioned. It is the reusable capability to take an eligible device from receipt through governed service, with evidence, at a predictable level of risk and effort. Truck rolls avoided and elapsed time matter, but they are consequences. The primary product is the line.

THE LAYERED DEPLOYMENT MODEL

A mature deployment does not configure a device in one undifferentiated pass. It advances through layers whose prerequisites, permissions, validations, and recovery actions are explicit. The exact mechanisms vary by vendor and topology; the control logic should not.

Preconditions: stage zero

Before a device is connected, the program should know the supported hardware and software combinations, approved image and template versions, identity enrollment method, source-of-truth record, site and role, management addressing policy, bootstrap service endpoints, and accountable owner. DHCP, DNS, time, routing, image repositories, PKI, AAA, logging, and orchestration capacity must be tested for the intended site and failure domain. Required artifacts should be signed or otherwise integrity-protected, versioned, and retrievable through an authenticated channel where the platform permits.

The stage-zero gate is eligibility. A device that is absent from the approved population, presents conflicting identity, runs unsupported code, or maps to incomplete intent remains in a restricted state. “Found on the network” is an observation, not approval to build.

Management first, underlay second, services third

Table 2. Gated layered deployment model

Stage	Purpose and prerequisites	Exit gate and evidence	Failure containment and recovery
0. Eligibility and site readiness	Match the physical device to approved identity, hardware/software support, site, role, source-of-truth intent, and available bootstrap dependencies	Identity match; required data complete; site services reachable; artifact versions approved; change or pilot authorization present	Keep the port or bootstrap segment restricted; correct the record, site dependency, image, or hardware; use controlled console recovery only under an exception
1. Governed management plane	Establish a management address and route, trusted time, device and server authentication, AAA, logging, minimum hardening, and reachability to management systems	Mutual trust or approved equivalent; certificate and authorization valid; management reachability tested from approved points; logs received; rescue access proven	Do not enable production-facing services; retry bounded transient failures; revoke questionable identity; fall back to isolated recovery network or console
2. Underlay / infrastructure	Apply supported software and the Layer 2/Layer 3 infrastructure required for stable transport, including uplinks, routing, MTU, and infrastructure policy	Neighbor and routing adjacencies correct; intended paths and failure domains tested; no unexpected reachability; configuration and state conform to approved baseline	Withdraw or suppress incomplete adjacencies; retain management access; roll back to last known-good image/configuration or rebuild; prevent overlays from activating
3. Service overlays	Instantiate only approved service segments, routing domains, policy, quality-of-service treatment, and attachment points over a validated underlay	Segmentation and policy tests pass; service endpoints reachable only as intended; telemetry and ownership registered; capacity within limits	Disable the affected overlay or attachment without dismantling validated management and underlay layers; restore the previous service version; open a time-bounded exception if authorized
4. Assurance and operational handoff	Verify end-to-end service, compliance, monitoring, backup or rebuild evidence, inventory accuracy, and support readiness	Acceptance evidence stored; source of truth updated; alarms and runbooks tested; NOC ownership accepted; exception queue empty or explicitly accepted	Hold the device/site out of production, or limit service to an approved degraded mode; return to the failed stage with preserved evidence and a named incident owner

The gates should be implemented as state transitions, not as narrative advice. An orchestrator or pipeline should record which checks ran, their inputs and versions, the result, the actor or service identity, timestamps, and any override. A retry is not the same as a pass. An override is not the same as compliance.

Security principles embedded in the sequence

Security becomes structural when each stage grants only the access needed to reach the next verified state.

- **Start with minimum trust.** A device on a bootstrap network receives only the connectivity required for identity, time, name resolution, approved artifact retrieval, and logging. It does not receive general management or production access.
- **Authenticate both ends where supported.** The device should validate the bootstrap service, and the service should validate device identity or a controlled enrollment claim. Factory identifiers can contribute evidence, but an identifier alone is not necessarily a secret or proof of possession. Secure Zero Touch Provisioning (SZTP) [1] and Bootstrapping Remote Secure Key Infrastructure (BRSKI) [2], both of which can build on IEEE 802.1AR secure device identifiers [3], define standards-based ways for each side to verify the other using manufacturer-installed credentials.
- **Separate identity, role, and authorization.** A valid device certificate proves an identity within its policy; the source of truth and authorization policy determine whether that device may assume a site and role.
- **Protect artifacts and secrets.** Images, templates, and scripts are versioned and integrity-checked. Long-lived shared secrets should not be embedded in bootstrap files. Enrollment tokens should be scoped, short-lived, single-use where possible, and auditable.
- **Make revocation and rotation operable.** Certificate expiration, compromise, failed renewal, and device replacement are ordinary lifecycle events. The design needs renewal windows, monitoring, revocation behavior, and a recovery identity path.
- **Preserve evidence.** Logs and stage results should leave the device and enter an independently governed system. Local logs alone may disappear during a wipe or failure.
- **Design for unavailable dependencies.** Redundant services, local survivability, bounded caching, and documented degraded modes should reflect business risk. High availability is not the same as silently bypassing a failed security control.
- **Separate duties for high-risk changes.** The people who alter identity policy, approved software, or production templates should not be able to approve and deploy every change without independent review appropriate to the risk.

The layered model does not guarantee security. It creates enforceable control points at which security and operational evidence can be required before the device gains more capability.

WHAT THE MODEL BUYS

The first benefit is controlled repetition. A new site or replacement device follows a known sequence, and variation is visible as data or an exception. This reduces dependence on individual memory and makes elapsed time, failure, and manual effort measurable.

The second benefit is safer segmentation. A validated underlay can carry multiple logical services without placing them into one undifferentiated trust domain. Data, voice, video, storage, guest, operational technology, and IoT traffic may justify separate overlays or other segmentation constructs when their routing, policy, scale, and failure requirements differ. The design decision should be explicit. “Overlay per service” is a useful default pattern, not a rule that every named service must receive a unique control plane regardless of cost.

The third benefit is recoverability. Management connectivity is established before production services, so a failed service deployment can often be withdrawn while preserving the path used to diagnose and repair it. Likewise, an incomplete underlay should not be allowed to advertise or carry service overlays. This reduces the blast radius of partial deployment.

The fourth benefit is local survivability when the architecture requires it. Routing and policy enforcement placed at a site can allow approved local functions to continue during loss of a central management or WAN path. That behavior is not automatic: it depends on where state is held, how long credentials and policy remain valid, which services require a central broker, and whether local dependencies such as DNS or authentication are available. The design should specify what continues, what freezes at last known-good state, and what fails closed.

The fifth benefit is a defensible lifecycle path. A device built from controlled intent is easier to compare, replace, and requalify than one whose production state is the accumulated result of undocumented changes. That does not make every device disposable, but it makes replacement a designed option rather than an improvised event.

WHY CLOUD DOES NOT REMOVE THE DISCIPLINE

Cloud infrastructure is API-driven, which makes many resources easier to create, identify, and connect programmatically. The provider abstracts hardware arrival, console access, and portions of initial network reachability. Those are major advantages. They do not remove the need for authoritative intent, identity, sequencing, assurance, and lifecycle governance.

A cloud resource can be created from the wrong account, region, image, network, security policy, or data record. A control-plane API can be unavailable, rate-limited, misauthorized, or reachable only through another failed dependency. Terraform state can diverge from provider state. A pipeline can consistently deploy the wrong intent. Fast creation can make weak governance more consequential because an error can be repeated quickly.

The relevant comparison is therefore not “cloud has no bootstrap problem while physical networks do.” It is that cloud providers absorb and standardize much of the lowest-level bootstrap, whereas physical network programs must often design and operate more of it themselves. Both environments still need a production line: approved inputs, trusted control paths, versioned artifacts, dependency management, validation, rollback or compensating action, evidence, and ownership.

IMPLEMENTATION ROADMAP AND PILOT MANDATE

A ZTP program should start with a mandate that is narrow enough to finish and important enough to matter. A sensible first scope might be one supported device family, one software train, one repeatable site class, and a limited set of management, underlay, and service patterns. A pilot that accepts every hardware family, every brownfield exception, and every service variant is not representative; it is unbounded.

Program mandate

The business sponsor authorizes the outcome, budget, and cross-functional priority. The service owner is accountable for the end-to-end capability and operational acceptance. Network engineering owns network standards and supported designs. The platform/automation team owns pipeline engineering and integration. Cybersecurity/PKI owns identity policy and certificate services. Enterprise architecture arbitrates cross-domain standards and strategic fit. Operations/NOC owns production monitoring and first-line response. Change management governs production authorization and evidence. Site/field services owns physical readiness, cabling, installation, and local recovery. These roles may be combined in a smaller organization, but their decisions should remain explicit.

Table 3. Implementation roadmap and pilot gates

Phase	Required work and dependencies	Exit criteria	Primary evidence
1. Mandate and scope	Name sponsor and service owner; define business outcome, pilot population, supported hardware/software, site class, services, exclusions, risk tolerance, funding, capacity, and timeline	Charter approved; roles staffed; pilot devices and sites enumerated; success, stop, and rollback criteria agreed	Signed charter; population list; responsibility model; risk and dependency register
2. Data and architecture readiness	Define source-of-truth schema and ownership; qualify images/templates; design management, underlay, overlays, PKI/AAA, DHCP/DNS/time/bootstrap, logging, artifact repository, and recovery paths	Critical data complete and tested; dependencies have owners and service expectations; threat model and design review accepted	Data-quality report; architecture decision records; support matrix; security review; dependency test results
3. Lab validation	Exercise clean bootstrap, replacement, retry, identity conflict, expired or unavailable certificate service, bad source data, unsupported software, partial deployment, management-path loss, rollback, and console recovery	All mandatory tests pass; unresolved defects within agreed threshold; runbooks usable by someone other than the builder; capacity model validated	Automated test results; defect log; recovery timings; signed artifacts; runbook exercise records
4. Controlled field pilot	Deploy a small representative wave with enhanced observation, pre-staged recovery, named incident command, and no undocumented scope changes	Each device reaches governed state or a rehearsed recovery path; no unowned exception; metrics captured from system evidence; field and NOC feedback resolved or accepted	Stage records; scorecard; incident and exception log; source-of-truth reconciliation; pilot acceptance

Phase	Required work and dependencies	Exit criteria	Primary evidence
5. Production waves	Roll out by failure domain and site class; limit concurrency to support and rollback capacity; use go/no-go checks before each wave	Wave meets release thresholds; support load and dependencies remain within capacity; next wave explicitly authorized	Wave report; capacity and alert review; change record; rollback readiness; residual-risk acceptance
6. Operational handoff	Transfer monitoring, on-call, runbooks, dashboards, certificate operations, template release process, exception queues, vendor support, and lifecycle backlog	NOC and service owner accept support; ownership survives the project team; recovery drill passes; documentation and access are current	Handoff checklist; access review; training and drill results; service-level objectives; accepted backlog

Stage gates, waves, and capacity

The pilot must test the production control logic, not merely prove that a device can download a configuration in a lab. Each gate should have a named approver, automated checks where practical, a maximum wait time, an escalation path, and a defined rollback or safe-hold state. Go/no-go authority should be independent of schedule pressure.

Rollout waves should be bounded by the team's ability to observe, diagnose, and recover—not by the maximum number of devices the pipeline can start. The capacity plan should include DHCP and DNS load, PKI issuance and validation, artifact distribution, orchestration workers, source-of-truth APIs, telemetry ingestion, NOC alert volume, replacement stock, field access, vendor escalation, and the number of simultaneous rollbacks the organization can safely execute.

The first production waves should vary one dimension at a time where possible: site geography, hardware revision, carrier, software version, or service mix. Canary devices and sites should be selected because they are representative and recoverable, not merely because they are convenient.

Exceptions and rollback

An exception is a governed departure from the standard, not a hidden alternate process. Every exception should identify the device or population, the unmet control, business justification, compensating control, owner, approver, expiration, and plan to remove or incorporate the variation. Exceptions should be visible in the source of truth and scorecard.

Rollback is stage-specific. Before service activation, rollback may mean withdrawing the overlay while retaining management and underlay. During an underlay change, it may mean restoring the last known-good configuration, image, or boot variable. When identity is questionable, rollback may mean revocation and quarantine rather than reinstating an old credential. If safe automated rollback cannot be proven, the plan should specify a controlled hold state and a console or field recovery procedure.

The operational handoff is complete only when the receiving team can detect failure, determine the current stage, access evidence, execute the runbook, escalate dependencies, and restore or isolate the device without the implementation team. A slide presentation is knowledge transfer; a successful receiver-operated drill is capability transfer.

FAILURE MODES AND RECOVERY PATHS

Failure handling should preserve two priorities: do not grant more capability than the device has earned, and do not destroy the evidence needed to recover. The following failure modes belong in design review and lab testing.

Table 4. Representative failure modes and recovery ownership

Failure mode	Detection and containment	Recovery path and owner
Bootstrap services unreachable	Device remains on restricted bootstrap network; health checks distinguish local link/DHCP/DNS/time/routing failures from service failure; bounded retries prevent storms	Restore site or central dependency; use approved redundant endpoint or isolated recovery path; site/field services owns local readiness, platform owns bootstrap service, NOC coordinates incident
Incorrect, duplicate, or untrusted identity	Serial, certificate, claim, site, or role mismatch blocks eligibility; suspect credentials are not reused; event is logged outside the device	Reconcile physical evidence and source-of-truth record; revoke or reissue credentials as appropriate; cybersecurity/PKI decides identity validity, network engineering confirms device placement
Unsupported hardware or software	Support-matrix check fails before production template application; device remains restricted	Replace hardware, load an approved intermediate image through a controlled path, or approve a time-bounded exception with a dedicated template; network engineering is accountable
Brownfield inconsistency	Observed topology, addressing, configuration, or services conflict with intended model; automation does not overwrite ambiguity	Quarantine the conflicting object from automated change; reconcile with service owner and operations; migrate through a documented intermediate state or exclude with expiration
Partial deployment	Stage evidence shows some but not all intended changes; overlays remain disabled until underlay and policy checks pass	Apply compensating action for the failed stage, restore last known-good artifact, or rebuild while retaining management if trusted; platform and network engineering own technical recovery
Certificate enrollment, renewal, or validation failure	No production authorization is granted; impending expiry is alerted before service impact; validation errors are separated from reachability errors	Correct time, chain, policy, enrollment authorization, or revocation status; use a tightly controlled recovery identity path; cybersecurity/PKI owns issuance, NOC owns response

Failure mode	Detection and containment	Recovery path and owner
Source-of-truth error	Schema, integrity, range, uniqueness, topology, or reconciliation checks fail; critical attributes cannot be overridden silently	Correct through governed data workflow, re-render artifacts, re-run prechecks, and retain the rejected input/version for audit; data owner approves, service owner accepts impact
Loss of management connectivity	Local forwarding behavior follows an explicit survivability policy; configuration freezes at last known-good state unless local automation is authorized	Restore management path through redundant routing or out-of-band access; do not make blind changes; NOC coordinates, network engineering repairs path, field services performs local recovery
Rollback fails or device will not boot	Device is isolated from production; boot state, console output, and artifact versions are captured; wave is stopped when threshold is crossed	Invoke rescue image, alternate boot bank, hardware replacement, or vendor escalation; incident commander controls further rollout and service owner decides resumption

Recovery paths should be rehearsed against real hardware and realistic latency. A runbook that depends on the same unavailable management path as the failed process is not a recovery path. Neither is “send an engineer” unless the program has named dispatch ownership, access requirements, spares, console methods, and an expected response time.

DECISION RIGHTS AND OPERATING MODEL

Cross-functional participation is not sufficient if no one knows who can decide. The following RACI-style model is a starting point. “A” is the single role accountable for the outcome and final decision, “R” performs the work, “C” is consulted before the decision, and “I” is informed. Multiple responsible roles are acceptable; multiple accountable roles usually signal an unresolved boundary.

Role abbreviations: **BS** business sponsor; **SO** service owner; **NE** network engineering; **PA** platform/automation; **CP** cybersecurity/PKI; **EA** enterprise architecture; **NO** operations/NOC; **CM** change management; **FS** site/field services.

Table 5. RACI-style decision rights

Decision or work product	A	R	C	I
Business outcome, funding, and risk appetite	BS	SO	EA, NE, PA, CP, NO	CM, FS
End-to-end service design and production acceptance	SO	NE, PA, NO	CP, EA, CM, FS	BS
Network standards and supported hardware/software catalog	NE	NE	PA, CP, EA, NO, FS	SO, CM

Decision or work product	**A**	**R**	**C**	**I**
Source-of-truth schema, data quality, and authority	SO	NE, PA	EA, CP, NO, FS	BS, CM
ZTP pipeline, integrations, templates, and release engineering	PA	PA, NE	CP, EA, NO, CM	SO, FS
Device identity, PKI, AAA, secrets, and cryptographic policy	CP	CP	NE, PA, EA, NO	SO, CM, FS
Cross-domain architecture and standards exceptions	EA	NE, PA, CP	SO, NO	BS, CM, FS
Lab validation and recovery qualification	SO	NE, PA, CP, NO	EA, CM, FS	BS
Site readiness, physical installation, and local recovery	FS	FS	NE, PA, NO, CP	SO, CM
Production change authorization and wave control	CM	SO, NE, PA, NO	CP, FS	BS, EA
Production monitoring, incident command, and first response	NO	NO	NE, PA, CP, FS	SO, CM, BS
Exception acceptance, rollback, and rollout resumption	SO	NE, PA, NO	CP, CM, EA, FS	BS

The matrix should be adjusted to local governance, but several decision rights should remain clear. Cybersecurity/PKI decides whether an identity and cryptographic method are acceptable. Network engineering decides whether hardware, software, and network design are supported. Platform/automation decides whether pipeline code meets release and reliability standards. Change management decides whether production change evidence meets the process, while the service owner decides whether the service risk is acceptable within sponsor-approved appetite. Operations/NOC can stop a wave when operational thresholds or recovery capacity are exceeded. Site/field services can reject a site that is not physically ready. The business sponsor resolves priority and funding conflicts; the sponsor should not override a failed technical security gate without formal risk acceptance.

An operating forum should review the scorecard, exceptions, expiring certificates, unsupported platforms, capacity, failed stages, and upcoming waves on a defined cadence. It should make decisions and assign owners, not merely receive status.

OPERATIONAL SCORECARD

The scorecard should measure whether the assembly line produces governed outcomes, not whether it stays busy. Metrics must be calculated from pipeline, source-of-truth, assurance, PKI, change, and incident evidence rather than from self-reported completion alone.

Before the pilot, define an **eligible provisioning attempt** as one approved device entering stage zero with complete required intent and a unique attempt identifier. Retries within that workflow remain part of the same attempt unless a formal restart or rollback creates a new attempt. Define **governed state** as completion of all mandatory management, underlay, service, assurance, and handoff gates for the approved scope, with no expired blocking exception.

Table 6. Operational scorecard

Metric and definition	Owner and system of record	Cadence	Baseline and target guidance	Anti-gaming control
Provisioning success rate = eligible attempts reaching governed state within the agreed window ÷ all eligible attempts started in the period	SO accountable; PA calculates from orchestrator stage records; NO validates production outcome	Per wave; weekly during rollout; monthly steady state	Establish by device family and site class in lab and first controlled wave; set a production release floor and improve without relaxing gates	Do not remove failed attempts after work starts; report first-pass and eventual success separately; segment planned test failures
Time to governed state = elapsed time from accepted stage-zero entry to final governed-state timestamp; report median and 95th percentile	PA owns event data; SO owns definition; change system supplies approved pauses	Per attempt and wave; monthly trend	Baseline by site class and dependency conditions; target lower tail latency while preserving validation	Show active processing and approved wait time separately; never stop the clock for unapproved manual work or failed retries
Exception rate = eligible attempts invoking any nonstandard path, manual waiver, or expired prerequisite ÷ eligible attempts	SO owns exception register; CM validates approvals; source of truth records affected objects	Per wave; biweekly review; monthly trend	Baseline by reason and age; target downward for repeat causes, with explicit caps on expired and security exceptions	Count each affected attempt and unique exception; do not rename repeat exceptions as “standard” without design review
Drift / compliance rate = compliant in-scope checks ÷ all valid in-scope checks, accompanied by count and age of devices with critical drift	NO and CP own assurance/compliance platforms; NE owns network baseline	Daily collection; weekly operations review; monthly governance	Baseline after pilot stabilization; critical controls require an explicit release threshold; improve by severity and age, not only aggregate percentage	Preserve failed and unavailable checks; report unknown/not-observed separately; prohibit excluding devices after a failure is detected
Rollback rate = production attempts requiring automated or manual reversion to a prior approved state ÷ production attempts	PA records pipeline rollback; CM and NO validate change/incident outcome	Per wave; monthly trend	Baseline by change type; seek stable or declining rate, but do not set zero as an incentive to hide prudent rollback	Distinguish planned rollback tests, safety rollback, and failed rollback; link each event to cause and corrective action

Metric and definition	Owner and system of record	Cadence	Baseline and target guidance	Anti-gaming control
Manual-touch rate = eligible attempts requiring a human configuration, console, or data intervention beyond approved physical installation and authorization ÷ eligible attempts	PA captures workflow interventions; FS records console/field work; SO owns classification	Per wave; monthly trend	Baseline by device/site class; reduce repeatable touches while retaining authorized safety decisions	Count emergency and "quick fix" actions; publish allowed-touch taxonomy; do not classify remote CLI work as automation
Source-of-truth accuracy = sampled required field values matching validated physical or observed evidence ÷ required field values sampled, with critical-field defects reported separately	SO accountable; named data stewards execute sampling; NE/FS provide evidence	Before each wave; monthly sample; quarterly deeper audit	Critical identity, site, role, serial, address, and template fields should be defect-free at the entry gate; set broader target after a statistically and operationally representative baseline	Use independent samples, preserve provenance, count missing/stale values as defects, and prevent data authors from selecting only their own records

Targets should be approved before the measured wave. They should be segmented where populations have materially different complexity, but segmentation must not become a way to isolate every failure into a unique category. A baseline describes current performance; it does not define acceptable risk. Security gates, identity conflicts, unsupported software, and inability to recover can remain release blockers even if the aggregate success rate is high.

The scorecard also needs a denominator review. At each governance meeting, the service owner and an independent reviewer should reconcile devices physically received, devices entered in the source of truth, eligible attempts, completed attempts, exclusions, and exceptions. Without that control, the easiest way to improve a percentage is to keep difficult devices out of the denominator.

EPHEMERAL DEVICES, CONTROLLED REMEDIATION, AND LIFECYCLE

Treating a device as replaceable changes engineering behavior. If approved intent, software, and configuration exist outside the device, a failed or corrupted unit can be reset or replaced and rebuilt through the same governed process. That is a powerful property. It reduces the device's role as the only copy of its own history and makes disaster recovery testable.

Replace-not-patch is practical when the network has sufficient redundancy, the device role is reproducible, bootstrap dependencies are healthy, images and templates are qualified, configuration is derived from authoritative data, session or local state can be drained or reconstructed, spare capacity exists, and the replacement time meets the service objective. It is especially useful for standardized access devices, leaf/spine nodes with adequate path diversity, lab fleets, and routine hardware replacement—subject to the actual platform and service design.

Physical network infrastructure also has cases where controlled in-place remediation remains necessary. A nonredundant site edge, a large chassis with long convergence or reboot time, hardware with scarce spares, platforms carrying stateful functions, operational technology with strict outage coordination, remote locations with limited field access, and emergency containment during an incident may not support immediate replacement. An in-place software upgrade may itself be the vendor-supported and lower-risk path.

The governing principle is not “always wipe.” It is “make the desired end state reproducible, and choose the recovery method through policy.” Both paths need prechecks, change authorization, backup or reconstruction evidence, staged validation, rollback or rescue, telemetry, and post-change reconciliation. An emergency manual fix should be captured immediately as observed state, linked to an incident, reconciled against intent, and either encoded into the standard or removed within an approved time.

ZTP contributes to lifecycle management by making reset, replacement, and re-entry repeatable. Certificate rotation, vulnerability remediation, software maintenance, capacity, spares, end-of-support, and retirement remain broader lifecycle responsibilities. At retirement, the process should revoke identity, remove secrets, update authoritative status, withdraw services, preserve required evidence, and dispose of hardware under policy.

REDUNDANCY BEYOND THE FAILOVER PAIR

An ephemeral model changes the set of available recovery patterns; it does not invalidate stateful failover. VRRP, HSRP, chassis clustering, or other paired mechanisms can remain appropriate where clients require a stable gateway, platform state must be synchronized, or convergence objectives favor the design. These patterns require their own controls for version compatibility, split-brain behavior, state synchronization, and correlated failure.

Where the service and topology permit, multiple active routed paths can reduce dependence on a specific pair and allow traffic to converge around a failed or rebuilding node. Equal-cost multipath, anycast, redundant route reflectors, distributed gateways, and identity-brokered service access solve different availability problems at different layers. None is universally superior. The right pattern depends on the protected service, failure domain, state location, convergence requirement, operational competence, and recovery evidence.

The useful change in mindset is to stop treating “redundancy” as a single product pattern. Availability is an end-to-end property. ZTP improves the speed and consistency with which capacity can be restored, but routing design, service architecture, dependency resilience, and operations determine whether that restoration is fast enough.

WHERE THIS MODEL DOES NOT FIT

This paper does not claim that every network needs the same ZTP investment or an overlay for every service. A small, stable environment with few deployments and reliable local expertise may not justify a complex orchestration platform. A one-off lab or proof of concept may reasonably use a simpler, explicitly temporary process. A brownfield site with unsupported equipment may need staged modernization before it can enter the line.

There is also a threshold below which the governance machinery costs more than the repetition it removes. That threshold should be assessed against device volume, site repetition, replacement frequency, compliance burden, outage cost, field access, software cadence, and the number of variants—not against device count alone.

Even when full ZTP is not justified, its design disciplines remain useful: authoritative inputs, supported combinations, minimum-trust onboarding, versioned artifacts, validation, evidence, named ownership, and a recovery path. The assembly line may be small. It should still have quality control.

WHAT THIS REQUIRES ORGANIZATIONALLY

The architecture will not survive an organization that rewards only visible device activity. If leadership measures tickets closed and devices installed without measuring repeatability, governed outcomes, exceptions, and data quality, teams will rationally optimize for local completion. Building the line can initially look slower because the team is converting judgment into standards, tests, code, and runbooks. The investment case should compare the capability over its useful population and lifecycle, not against the fastest one-time manual install.

The talent model must value both network judgment and production engineering. Network engineers define safe intent and failure behavior. Platform engineers make workflows observable, testable, and recoverable. Security engineers make identity and artifact trust operable. Operations engineers determine whether the system can be supported at 2 a.m. Field teams expose physical realities that a lab can miss. Architecture aligns the capability with enterprise standards. Change management ensures that speed does not outrun evidence. The service owner integrates those perspectives into one accountable product.

Automation does not eliminate human judgment. It moves judgment earlier, captures it in approved definitions, and reserves live intervention for conditions the system cannot safely resolve. That is a better use of expertise than requiring an expert to reconstruct the same decision at every console.

CONCLUSION

Zero Touch Provisioning is not the sole foundation of network automation. It is a foundational bootstrap and lifecycle capability for repeatable physical network deployment: the controlled transition from an unknown or reset device to a reachable, authenticated, governed participant. Configuration management, orchestration, a source of truth, PKI and AAA, telemetry, assurance, compliance, and lifecycle management remain distinct capabilities and are all required for a trustworthy production system.

The practical sequence is clear. For brownfield environments, discover and reconcile enough reality to establish authoritative pilot data. Define completion contractually and operationally. Qualify hardware, software, templates, and dependencies. Deploy management first, underlay second, and service overlays third. Put gates, evidence, security, and recovery at every transition. Pilot a narrow representative pattern. Measure governed outcomes, not activity. Transfer the capability through receiver-operated drills. Expand in waves no faster than the organization can observe and recover.

The executive call to action is equally clear: appoint a sponsor and service owner, select one repeatable site and device pattern, fund the dependencies, and authorize a pilot with explicit entry, exit, stop, exception, and rollback criteria. Do not begin by promising zero touch everywhere. Begin by proving one production line that the organization can run, measure, secure, and repair.

Build the assembly line. The cars will follow.

REFERENCES

[1] K. Watsen, I. Farrer, and M. Abrahamsson, “Secure Zero Touch Provisioning (SZTP),” RFC 8572, IETF, April 2019. https://datatracker.ietf.org/doc/html/rfc8572

[2] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, “Bootstrapping Remote Secure Key Infrastructure (BRSKI),” RFC 8995, IETF, May 2021. https://www.rfc-editor.org/rfc/rfc8995

[3] IEEE, “IEEE Standard for Local and Metropolitan Area Networks—Secure Device Identity,” IEEE Std 802.1AR-2018. https://standards.ieee.org/ieee/802.1AR/6995/