CONFIGURATION MANAGEMENT PROCEDURE

Document Reference: QMS-DEV-PRO-006

Version: 1.0

Classification: INTERNAL ENGINEERING ONLY

Role	Name & Signature	Position	Date
Prepared by			
Reviewed by			
Approved by			

TABLE OF CONTENTS

1. AMENDMENT RECORD

Reviewed at minimum annually or following a significant drift incident, tooling change, audit finding, or compliance requirement change. All revisions are subject to the same approval requirements as the original document.

Date	Issue	Rev	Page	Subject	Revised By	Approved By

2. EXECUTIVE SUMMARY

This procedure defines the controls required to identify, baseline, change, track, and audit all configuration items (CIs) across the organisation's software and infrastructure estate. It applies from the point a CI is first registered to the point it is decommissioned.

Teams that maintain an internal developer guide or engineering handbook should treat this procedure as the formal compliance layer governing configuration activity. Teams without such a guide will find the process detail sufficient to operate from directly.

The procedure covers five areas:

- Configuration item identification and baseline establishment
- IaC standards and version control requirements for all configuration types
- Change control for configuration items, including emergency change handling
- Environment consistency enforcement and drift detection
- Audit programme, metrics, and continuous improvement cadences

3. GENERAL

3.1 Scope

This procedure governs all configuration items that affect the build, deployment, operation, or security posture of any shared or production environment. It applies to:

- Application source code, build scripts, dependency manifests, and lock files
- Infrastructure as Code definitions: Terraform, Pulumi, Ansible, Crossplane, Helm charts, and equivalents
- Container definitions: Dockerfiles, base image references, and image build configurations
- CI/CD pipeline definitions and configuration
- Environment configuration: all variables, feature flags, secrets references, and runtime parameters that differ between environments
- Network and security configurations: firewall rules, security group definitions, IAM policies, TLS certificate configurations
- Database schema definitions and migration scripts
- Monitoring and alerting configuration: Prometheus rules, Grafana dashboards, PagerDuty routing
- Disaster recovery and backup configurations
- Documentation that defines or describes the above: architecture decision records, runbooks, and operational playbooks

Configuration items that exist only in a developer's local environment and have not been committed to version control are out of scope. Any CI that is promoted to a shared branch or shared environment immediately falls within scope.

3.2 Purpose

- Ensure every configuration item in a shared or production environment has a corresponding version-controlled definition, a registered baseline, and a change history
- Prevent configuration drift between environments, and detect and remediate drift when it occurs
- Enforce that all configuration changes are authorised, reviewed, tested, and traceable to a change record
- Provide sufficient audit evidence for ISO 9001:2015 clause 7.5 (documented information), clause 8.5 (production and service provision), and ISO 27001:2022 Annex A controls 8.9 (configuration management) and 8.32 (change management)
- Support reproducible builds and deployments: given a commit SHA and an environment name, it must be possible to reconstruct exactly what was deployed

3.3 Guiding Principles

- Traceability: every CI maintains a continuous, auditable record from registration to decommissioning. A CI without a complete change history is non-compliant.

- Immutability: production and staging configuration is driven exclusively from version-controlled state definitions. Manual intervention in a shared environment constitutes a non-conformance regardless of intent.
- Single source of truth: the version control repository is the authoritative definition of environment state. Observed state that diverges from repository state is drift and is treated as a defect, not as a documentation update opportunity.
- Baseline integrity: baselines are tagged, signed snapshots and are never modified in place. An approved change produces a new baseline; it does not overwrite an existing one.
- Continuous detection: drift detection is automated and scheduled. Drift is not a finding discovered at audit time; it is an operational event with defined SLAs for acknowledgement and remediation.
- Least privilege: all access to CI definitions, state backends, and the CMDB is role-based and restricted to the minimum required. Access rights are reviewed on the schedule defined in Section 4.7.

3.4 Responsibility for Implementation

The Configuration Manager (or equivalent Engineering Lead or Platform Lead depending on team structure) is accountable for this procedure. Responsibilities are distributed as follows:

- Configuration Manager: CI registry maintenance; baseline governance; change control process; audit programme; drift reporting
- DevOps / Platform Engineer: IaC authoring and maintenance; drift detection tooling; environment provisioning pipelines; secret management tooling
- Engineering Lead: branch protection and merge policy enforcement; CI classification decisions; change approval authority for standard changes
- Security Engineer: security configuration review; IAM and access control CI ownership; compliance configuration audits
- Developers: authoring CIs within defined standards; registering new CIs before promotion to shared branches; updating documentation concurrent with configuration changes

In smaller teams, multiple roles above may be held by the same person. The separation of CI change authorisation from CI change implementation is the one structural requirement that must be maintained regardless of team size.

3.5 References

- ISO 9001:2015, clauses 7.5 (documented information) and 8.5 (production and service provision)
- ISO 27001:2022, Annex A controls 8.9 (configuration management) and 8.32 (change management)
- ITIL 4, Configuration Management and Change Enablement practices
- NIST SP 800-128, Guide for Security-Focused Configuration Management of Information Systems
- CIS Benchmarks, relevant to operating systems, cloud platforms, and container runtimes in use
- SLSA Framework, slsa.dev, for build and artefact provenance requirements
- Internal: SDLC Procedure
- Internal: Code Review Procedure
- Internal: Release Management Procedure

- Internal: CI/CD Procedure
- Internal: Change Management Procedure
- Internal: Incident Response Procedure

3.6 Roles and Responsibilities

Role	Primary Responsibilities	Backup	Notes
Configuration Manager	CI registry; baseline governance; change control process ownership; audit programme; drift metrics reporting	Engineering Lead	In smaller teams this role is absorbed by the Engineering Lead or Platform Lead
DevOps / Platform Engineer	IaC authoring; drift detection tooling; environment provisioning pipelines; secret management; CMDB tooling	Cross-trained team member	Owns the technical implementation of all controls defined in this procedure
Engineering Lead	Branch protection enforcement; merge policy; change approval for standard changes; CI classification decisions	Senior Engineer	Approves all changes to IaC modules and pipeline definitions
Security Engineer	Security CI review; IAM and access control CI ownership; CIS Benchmark compliance; security configuration audit	Security team rotation	Required approver for any change to security-classified CIs
Developers	CI authoring within defined standards; registering new CIs; documentation updates concurrent with configuration changes	Peer developer	Responsible for ensuring their CIs meet naming, structure, and tagging standards before PR

4. PROCESS DETAILS

4.1 Objectives

- Every CI deployed to a shared or production environment has a registered entry in the CI registry, a current baseline, and a complete change history
- Configuration drift in production and staging is detected within 15 minutes and remediated within 1 hour
- All configuration changes are authorised, peer-reviewed, implemented via the pipeline, and produce an updated baseline
- IaC coverage of production and staging infrastructure is maintained at 100%
- Audit evidence is sufficient to support ISO 9001:2015 and ISO 27001:2022 assessments without additional manual documentation

4.2 Key Performance Indicators

All metrics are sourced from CMDB, drift detection tooling, change management system, and CI/CD platform output. Manual collection of any listed metric indicates a tooling gap.

KPI	What it measures	Target	How collected	Breach response
CI Registry Coverage	Percentage of known configuration items that have a registered entry in the CMDB or equivalent registry, with a current baseline reference	100%	CMDB record count vs discovered CI count (automated discovery tool output)	Coverage below 100% means unregistered CIs exist in the environment. Unregistered CIs are unmanaged changes waiting to happen.
Drift Incident Rate	Number of confirmed configuration drift incidents detected per calendar month, across all environments	Zero in production and staging; less than 2 in development integration per month	Drift detection tooling (Terraform plan, Ansible check, Conftest, AWS Config, Azure Policy)	Drift in production or staging is an immediate non-conformance. Drift in development integration is a leading indicator; persistent drift indicates the IaC workflow is not being followed.
Mean Time to Detect Drift (MTTD-D)	Elapsed time from when a configuration diverges from its baseline to when the drift detection system raises an alert	Less than 15 minutes for production; less than 1 hour for staging	Drift detection tooling alert timestamps vs last known-good configuration timestamp	MTTD-D above threshold indicates the drift detection scan frequency is insufficient or alerting is misconfigured.
Mean Time to Remediate Drift (MTTR-D)	Elapsed time from drift alert raised to confirmed remediation, defined as the environment returning to its baseline state	Less than 1 hour for production; less than 4 hours for staging	Drift alert timestamp to CMDB record update timestamp showing remediation	Persistent breach indicates either insufficient on-call coverage or a baseline that is not achievable in practice, which itself indicates a baseline management problem.
Unauthorised Change Rate	Number of configuration changes applied to any shared environment that were not preceded by an approved change record	Zero	Change management system records cross-referenced against environment change event logs and CMDB update history	Any non-zero value is an immediate non-conformance. This metric is also the primary evidence point for ISO 27001:2022 Annex A 8.32 audits.

IaC Coverage	Percentage of infrastructure and environment configuration in shared environments that is defined and managed as code in version control	100% for production and staging; greater than 90% for development integration	IaC inventory vs total discovered resource count from cloud provider APIs or infrastructure scanning tools	Resources not defined in IaC cannot be drift-detected, version-controlled, or rolled back. Any resource outside IaC coverage is a compliance gap.
Baseline Age	Number of days since the current baseline for each registered CI was last reviewed and confirmed as current	No baseline older than 90 days without a review record	CMDB baseline review timestamps	Stale baselines become inaccurate references. An approved change that was not followed by a baseline update is a gap in the audit trail.
Secret Rotation Compliance	Percentage of secrets and credentials in the secret management system that have been rotated within their defined rotation SLA	100%	Secret management system rotation timestamp vs defined SLA per secret classification	Secrets that exceed their rotation SLA are a security configuration non-conformance. This metric is tracked here because secrets are configuration items managed under this procedure.
Change Rollback Rate	Percentage of implemented configuration changes that required rollback within 24 hours of implementation	Less than 2%	Change records with rollback events in the change management system	High rollback rate indicates insufficient pre-implementation testing or inadequate staging environment parity.
Audit Finding Closure Rate	Percentage of configuration audit findings closed within their assigned SLA (Critical: 48 hours; High: 5 business days; Medium: 30 days)	100% within SLA	Audit finding records in the audit management system	Open findings beyond SLA are escalated to the Configuration Manager and reported at the monthly KPI review.
Environment Provisioning Time	Elapsed time from provisioning request to a fully configured, health-checked environment ready for use	Less than 30 minutes for ephemeral environments; less than 2 hours for persistent	Pipeline execution timestamps from provisioning request to environment	Provisioning time above threshold is a leading indicator of IaC complexity problems or dependency bottlenecks in the provisioning pipeline.

		environment creation	health check passing	
Policy-as-Code Gate Pass Rate	Percentage of IaC changes that pass all automated policy checks (Checkov, tfsec, OPA/Conftest) on first execution without suppression	Greater than 95%	IaC policy tool output per pipeline run	Low first-run pass rate indicates developers are not running policy checks locally before committing, or policy rules are mis calibrated.

4.3 Configuration Item Classification

All CIs are classified at registration. Classification determines approval requirements, drift detection frequency, audit scope, and retention. Misclassification is a process non-conformance and must be corrected within 5 business days of identification.

CI Type	Description	Classification	Drift Detection Frequency	Change Approval
Application Source Code	Source	High	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Configuration Manager
Infrastructure as Code	Infrastructure	Critical	Every 15 minutes (prod); every 30 minutes (staging)	Engineering Lead + Security Engineer (security CIs) + Configuration Manager
Container Definitions	Infrastructure	Critical	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Security Engineer (security CIs) + Configuration Manager
CI/CD Pipeline Definitions	Infrastructure	High	Every 15 minutes (prod); every 30 minutes (staging)	Engineering Lead + Configuration Manager
Environment Configuration	Environment	High	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Configuration Manager

Secrets and Credentials	Security	Critical	Every 15 minutes (prod); every 30 minutes (staging)	Engineering Lead + Security Engineer (security CIs) + Configuration Manager
Network and Security Configuration	Security	Critical	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Security Engineer (security CIs) + Configuration Manager
Database Schema and Migrations	Application	High	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Configuration Manager
Monitoring and Alerting Configuration	Operations	Medium	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Configuration Manager
Backup and DR Configuration	Operations	High	Every 15 minutes (prod); every 30 minutes (staging)	Engineering Lead + Configuration Manager
Documentation CIs	Documentation	Medium	Every 5 minutes (prod); every 15 minutes (staging)	Engineering Lead + Configuration Manager

4.4 IaC Standards

All infrastructure and environment configuration in shared environments must be expressed as Infrastructure as Code. The following standards apply to all IaC authored under this procedure.

4.4.1 Repository Structure

- IaC is maintained in a dedicated repository or in a clearly delineated directory within the application repository. Mixed IaC and application code without clear separation is a non-conformance.
- Standard directory structure:

```
/modules -- reusable IaC modules, versioned independently
/environments -- per-environment variable definitions and backend configuration
/policies -- OPA/Conftest policy files and CIS Benchmark mappings
/tests -- IaC unit and integration tests (Terratest, Kitchen-Terraform, or equivalent)
/docs -- architecture decision records for infrastructure decisions
```

- Module versions are pinned in all environment configurations. Floating version constraints are a non-conformance.

4.4.2 Naming and Tagging Standards

- All cloud resources created by IaC carry a defined minimum tag set: environment, service, owner, cost-centre, and managed-by (value: terraform or equivalent tool name).
- Resources without the minimum tag set are flagged by automated policy checks and blocked from deployment.
- IaC module and resource naming follows the convention: {environment}-{service}-{resource-type}-{sequence}. Deviations require a documented exception in the CMDB.

4.4.3 State Management

- IaC state is stored in a remote, access-controlled backend. Local state storage in any shared or production pipeline path is a non-conformance.
- IaC execution environments must enforce state locking to prevent concurrent execution conflicts and state corruption. Execution without enforced locking on any shared environment backend is a non-conformance.
- State storage backends must enforce encryption at rest. The backend configuration, including encryption settings, is itself a registered CI under this procedure.
- State files must not be committed to version control. All IaC repositories must include state file exclusion patterns in their version control ignore configuration.

4.4.4 Policy as Code

- All IaC changes run through automated policy checks (Checkov, tfsec, Terrascan, OPA/Conftest, or equivalent) in the CI pipeline before any plan or apply operation.
- Policy rules are mapped to the relevant CIS Benchmark control and ISO 27001:2022 Annex A control where applicable. The mapping is maintained in the /policies directory.
- Policy suppressions require a justification comment referencing the change record that authorised the exception. Unsupported suppressions are a non-conformance.
- The policy-as-code gate pass rate is tracked per Section 4.2. Suppression rate is audited monthly.

4.5 Drift Detection

Configuration drift is defined as any divergence between the observed state of an environment and the current baseline for that environment's CIs. Drift detection is automated and continuous.

4.5.1 Detection Tooling and Schedule

Environment	Tooling	Scan Frequency	Alert Channel	SLA
Production	Terraform plan (drift mode), AWS Config / Azure Policy / GCP	Every 5 minutes	PagerDuty P1 alert to on-call	MTTD-D: 15 min; MTTR-D: 1 hour

	Config, Ansible --check, Conftest			
Staging	Terraform plan (drift mode), cloud provider config service, Conftest	Every 15 minutes	Slack alert to DevOps channel + Configuration Manager	MTTD-D: 30 min; MTTR-D: 4 hours
Development Integration	Terraform plan (drift mode), Conftest	Every 30 minutes	Slack alert to engineering channel	MTTD-D: 1 hour; MTTR-D: next business day
Ephemeral Environments	Destroyed on TTL expiry; no persistent drift detection	TTL enforcement on creation	Automated TTL alert if environment exceeds defined lifetime	N/A; TTL breach is a non-conformance

4.5.2 Drift Classification and Response

- Expected drift: a deployment is actively in progress. The CI/CD pipeline deployment record is confirmed before the drift alert is acknowledged. No non-conformance raised.
- Configuration drift: the environment state diverges from baseline with no corresponding in-progress deployment. Immediate non-conformance raised. Environment remediated by re-applying the baseline via the pipeline. The source of the drift is investigated and a root cause documented within 24 hours.
- Infrastructure drift: a cloud provider-initiated change (provider-managed patching, availability zone rebalancing) has altered a resource attribute outside IaC control. Investigated within 4 hours. If the change is valid, the IaC definition is updated via the standard change process. If invalid, the resource is remediated to the baseline.
- Persistent drift: a CI that has drifted from baseline more than twice in a 30-day period for non-expected reasons. Triggers a mandatory root cause review and an assessment of whether the baseline is achievable in practice.

4.6 Change Control for Configuration Items

All changes to registered CIs follow the change control process defined in the Change Management Procedure. This section defines the classification and specific requirements for configuration item changes.

Change Type	Definition	Min Lead Time	Approval and Process
Standard	Non-emergency change to a registered CI. Planned, tested, and scheduled in advance.	2 business days	Change record raised; technical review by CI owner; security review for Critical-classified CIs; Engineering Lead approval; implementation via

			pipeline; post-implementation drift scan; CMDB updated
Emergency	Change required to resolve an active production incident or mitigate an imminent critical risk. Abbreviated review.	Immediate	Engineering Lead and Configuration Manager synchronous approval; CI/CD pipeline Stages 1-4 not bypassed; full documentation within 24 hours; unplanned production emergency event record raised automatically; review within 5 business days
Pre-approved (Standard Change)	Repeatable, low-risk changes with a pre-defined procedure and no need for individual approval each time. Examples: scheduled dependency updates within defined version ranges, certificate rotation.	As defined in the pre-approved change procedure	Pre-approved change procedure referenced in change record; implementation via pipeline; CMDB updated post-implementation
IaC Module Update	Change to a shared IaC module used by multiple services. Requires impact assessment across all consumers.	3 business days	Module owner review; impact assessment of all consuming environments; Engineering Lead approval; staged rollout: development first, then staging, then production; drift scan at each stage before promotion

4.7 Configuration Audits

Audits verify that the configuration management process is operating as defined and that the deployed state of CIs matches their registered baselines. Audits are planned events with defined scope, not ad-hoc inspections.

Audit Type	Scope	Frequency	Owner	Output
Functional Configuration Audit (FCA)	Verifies CIs perform the function defined in their specification and that as-deployed state matches the version-controlled definition	Quarterly for Critical CIs; annually for High CIs	Configuration Manager + CI Owner	FCA report; finding records; updated CMDB where discrepancies found
Physical Configuration Audit (PCA)	Verifies that CI attributes in the CMDB match the actual deployed state across all registered environments	Quarterly	Configuration Manager	PCA report; CMDB update records; non-conformances for unregistered CIs

Security Configuration Audit	Verifies security-classified CIs conform to the applicable CIS Benchmark profile and to the organisation's security policy	Monthly for Critical security CIs; quarterly for High	Security Engineer	Security audit report; finding records with CIS Benchmark reference; corrective action plan
IaC Compliance Audit	Verifies IaC coverage, naming and tagging compliance, state management configuration, and policy suppression records	Quarterly	DevOps / Platform Engineer + Configuration Manager	IaC compliance report; suppression audit; gaps in IaC coverage
Process Compliance Audit	Verifies that the configuration management process itself conforms to this procedure and to the referenced ISO standards	Annually; also triggered by any Critical non-conformance	Quality Manager or equivalent	Process audit report; non-conformances against procedure; procedure amendment where required

Audit findings are classified as Critical (exploitable security risk or active non-conformance), High (significant process or compliance gap), or Medium (improvement opportunity or minor deviation). Closure SLAs: Critical 48 hours; High 5 business days; Medium 30 days.

4.8 Process Steps

Five phases govern the lifecycle of every configuration item from registration to decommission.

Phase 1: CI Identification and Registration

Steps

- Identify the configuration item and determine whether it already exists in the CI registry. Duplicate registration is not permitted.
- Classify the CI using the classification table in Section 4.3. Classification determines the approval requirements, drift detection frequency, and audit scope.
- Define the CI's relationships: what it depends on, what depends on it, and which environment tiers it applies to.
- Author the CI definition in the appropriate version-controlled location per the IaC standards in Section 4.4.
- Register the CI in the CMDB or equivalent registry with the required metadata: ID, classification, owner, repository location, environment scope, and baseline reference.
- Submit a PR for the CI definition. The PR must include documentation of the CI's purpose, its relationship dependencies, and its baseline values.
- Once the PR is approved and merged, tag the repository at the CI version. The tag constitutes the initial baseline.

Gate	CI registered in CMDB; definition committed to version control; initial baseline tag created; owner assigned.
Outputs	CMDB record; version-controlled CI definition; baseline tag; relationship map entry.

Phase 2: Baseline Establishment and Control

Steps	• A baseline is established at the point of initial CI registration (see Phase 1) and at each subsequent approved change that alters the CI's defined state. • Baselines are created by tagging the version control repository at the commit that represents the approved state. Tags are signed and immutable. • The CMDB baseline record is updated to reference the new tag, the change record that authorised the change, and the date of the update. • Baselines are reviewed on the schedule defined in Section 4.7. A baseline that has not been reviewed within 90 days is flagged in the monthly KPI report. • Environment state is compared against the current baseline on the drift detection schedule defined in Section 4.5. Deviation triggers an alert per the MTTR-D SLA in Section 4.2. • Baselines are never modified in place. If the current state of an environment is accepted as correct but diverges from the baseline, a change record is raised, the change is approved, and a new baseline is created.
Gate	Baseline tag created and signed; CMDB updated with baseline reference and change record link; drift detection configured against new baseline.
Outputs	Signed baseline tag; updated CMDB record; drift detection rule update.

Phase 3: Change Control

Steps	• All changes to registered CIs are initiated via a change request per the Change Management Procedure (QMS-CHG-PRO-001). Changes applied directly to a shared environment without a change record are a non-conformance. • Changes are classified as Standard, Emergency, or Infrastructure per the change types defined in Section 4.6. • Standard changes: technical review by the CI owner; security review for security-classified CIs; impact assessment covering dependent CIs; approval by the Engineering Lead. • Emergency changes: abbreviated review on an incident bridge; minimum approval by the Engineering Lead and Configuration Manager; CI/CD gates for Stages 1 through 4 are never bypassed; full documentation within 24 hours. • All approved changes are implemented via the CI/CD pipeline from the version-controlled definition. Manual changes to shared environments are prohibited. • Post-implementation: pipeline smoke tests and drift detection scan confirm the environment matches the new intended state. CMDB updated with new baseline reference. • Change record closed with implementation evidence: pipeline run ID, deployment log reference, and updated CMDB baseline reference.

Gate	Change record approved; implementation via pipeline; post-implementation drift scan confirms no residual drift; CMDB updated; change record closed with evidence.
Outputs	Closed change record; new baseline tag; updated CMDB; implementation evidence.

Phase 4: Status Accounting and Drift Detection

Steps	• Drift detection runs on an automated schedule per the frequency defined in Section 4.5. Tooling compares observed environment state against the current CMDB baseline. • Any detected drift produces an alert to the on-call engineer and the Configuration Manager within the MTTD-D SLA defined in Section 4.2. • Drift is classified on receipt: expected drift (a deployment is in progress and the state will converge), configuration drift (unauthorised or untracked change), or infrastructure drift (cloud provider change outside IaC control). • Configuration drift raises an immediate non-conformance record. The CI is remediated by re-applying the baseline via the pipeline, not by updating the baseline to match the drift. • Infrastructure drift (cloud provider-initiated change) is investigated. If the change is valid (e.g., a provider-managed security patch), the IaC definition is updated via the standard change process and a new baseline is created. • Status reports covering all CIs, their current baseline compliance status, and open drift incidents are produced on the schedule defined in Section 6.
Gate	Drift detection running at defined frequency; all drift incidents acknowledged within MTTD-D SLA; all confirmed drift remediated within MTTR-D SLA; status reports generated on schedule.
Outputs	Drift detection alerts; non-conformance records for unauthorised changes; status reports; CMDB compliance status.

Phase 5: Verification and Audit

Steps	• Functional Configuration Audit (FCA): verifies that CIs perform the function defined in their specification and that the as-deployed state matches the version-controlled definition. Conducted quarterly for Critical CIs; annually for High CIs. • Physical Configuration Audit (PCA): verifies that CI attributes in the CMDB match the actual deployed state. Conducted quarterly. • Security Configuration Audit: verifies that security-classified CIs conform to the applicable CIS Benchmark profile and the organisation's security policy. Conducted monthly for Critical security CIs. • Compliance Audit: verifies that the configuration management process itself conforms to this procedure and to the referenced ISO standards. Conducted annually; also triggered by any significant non-conformance. • Audit findings are classified as Critical, High, or Medium per the severity definitions in Section 4.7. Finding closure SLAs are: Critical 48 hours; High 5 business days; Medium 30 days. • All audit records, findings, and closure evidence are retained per Section 4.9.

Gate	Audit executed on schedule; all findings recorded with severity and assigned owner; closure SLAs tracked; audit report issued within 5 business days of audit completion.
Outputs	Audit report; finding records; corrective action records; updated CMDB where applicable.

4.9 Process Map

The end-to-end configuration management lifecycle is provided as a separate artefact in the link below. The map shows all five phases, decision points for CI classification and change type, drift detection feedback loops, and integration points with the Change Management, CI/CD, and Incident Response procedures.

Configuration management process map

4.10 Retained Documentation

Document Type	Content	Retention	Location	Access
CI Registry / CMDB	Complete inventory of all registered CIs with metadata, baseline references, and relationship map	Product lifetime	CMDB tooling / version control	Team
Baseline Records	Signed baseline tags; CMDB baseline references; change records authorising each baseline	Product lifetime	Version control system + CMDB	Team + Management
Change Records	Change requests; approvals; implementation evidence; post-implementation baseline updates	3 years	Change management system	Team + Management
Drift Detection Logs	Automated scan results; drift alerts; drift classification records; remediation confirmation	2 years	Monitoring platform / SIEM	DevOps + Security
Audit Reports	Audit scope; findings with severity; corrective actions; closure evidence	5 years	QMS repository	Management
Non-Conformance Records	Unauthorised change records; process deviation records; corrective action plans	5 years	QMS non-conformance register	Management

Secret Rotation Records	Rotation timestamps; owner; SLA compliance status; rotation authorisation records	3 years	Secret management system audit log	Security only
IaC Policy Scan Records	Per-run policy check results; suppression records with justifications; suppression audit log	2 years	CI/CD platform + security tooling	Security + DevOps
Environment Provisioning Records	Provisioning request; pipeline run log; health check results; provisioning time	1 year	CI/CD platform	DevOps

5. RISKS AND OPPORTUNITIES

5.1 Risk Register

Reviewed quarterly. The Configuration Manager owns the risk register for this procedure. Risk status is a standing item at the quarterly engineering review.

Category	Risk	Mitigation
Process	Configuration drift in production not detected within SLA due to drift detection tooling failure	Drift detection tooling health monitored as a service; alerting on scan failure as well as on drift; fallback manual scan procedure documented; pipeline blocks deployments if drift detection reports unhealthy
Process	Baseline records not updated after an approved change, creating a gap between CMDB and actual deployed state	Post-implementation baseline update is a mandatory step in the change control process; pipeline gate confirms CMDB update before change record is closed; CMDB vs pipeline deployment record reconciliation runs weekly
Process	IaC coverage gaps leaving resources outside configuration management controls	Automated resource discovery (cloud provider APIs) run weekly and compared against IaC inventory; unregistered resources flagged as non-conformances; resource tagging enforcement blocks provisioning of untagged resources
Technical	IaC state file corruption or loss causing inability to manage existing infrastructure	Remote state backend with versioning enabled; state file backups on a defined schedule; state recovery procedure documented and tested annually
Technical	Shared IaC module change breaking multiple downstream environments	Module versioning enforced; consuming environments pin module versions; staged rollout (dev, then staging, then production) with drift scan at each stage; rollback procedure per module version
Technical	Secret rotation failure leaving expired credentials in use	Automated rotation enforcement via secret management system; expiry alert at 80% of rotation SLA; rotation failure

		triggers on-call alert; expired credentials are treated as a security incident
Security	Unauthorised change to a security-classified CI going undetected	Security CI drift detection frequency is highest tier; all changes to security CIs require Security Engineer approval; CMDB changes to security CIs generate SIEM events; monthly security configuration audit
Security	Unmanaged Policy Suppression: undetected or unapproved circumvention of automated policy-as-code guardrails	Suppression rate tracked as KPI; all suppressions require justification referencing a change record; suppression audit monthly; suppression without justification is an immediate non-conformance
Compliance	Audit finding closure SLA breach causing a finding to remain open beyond its deadline	Finding closure tracked in the audit management system; automated reminder at 50% and 80% of SLA; breach escalated to Configuration Manager and reported at monthly KPI review
Operational	Key person dependency on the Configuration Manager role	Deputy Configuration Manager designated and trained; procedure documented sufficiently for any trained engineer to execute; CMDB and audit records accessible to the backup role

5.2 Opportunities

Area	Opportunity	Action
Drift Detection Maturity	Real-time drift detection using cloud provider native services (AWS Config Rules, Azure Policy, GCP Organisation Policy) reduces MTTD-D below the current scan-based threshold	Evaluate cloud-native configuration service adoption per cloud platform in use; assess against scan-based tooling on cost and coverage
GitOps Adoption	Full GitOps implementation (ArgoCD, Flux) for Kubernetes workloads eliminates the category of drift caused by manual kubectl or Helm operations; the cluster state is continuously reconciled to the repository	Evaluate GitOps tooling for Kubernetes environments; define reconciliation policies per environment tier
Automated CMDB Population	Automated CI discovery and CMDB population from IaC state and cloud provider APIs reduces	Evaluate Backstage, ServiceNow ITOM, or equivalent for automated CMDB population from Terraform state and cloud APIs

	manual registry maintenance and eliminates registration gaps	
IaC Testing Maturity	Expanding IaC test coverage (Terratest, Kitchen-Terraform, Checkov unit tests) provides pre-deployment verification of IaC behaviour, reducing the rollback rate for IaC changes	Establish IaC test coverage metric; target greater than 60% of IaC modules with automated tests within one roadmap cycle
Configuration Intelligence	Correlating drift events with deployment records and incident data identifies which CI types and change patterns produce the most drift and incidents, enabling targeted process improvement	Implement a configuration event data pipeline feeding a reporting dashboard; review at quarterly engineering review

6. CONTINUOUS IMPROVEMENT

6.1 Improvement Cadences

Cadence	Trigger / Frequency	Scope	Required Output
Weekly KPI Review	Weekly	CI registry coverage; drift incident rate and MTTD-D/MTTR-D compliance; unauthorised change count; policy gate pass rate	Status report; action item for any metric outside target with a named owner
Monthly KPI and Audit Review	Monthly	Full KPI set in Section 4.2; audit finding closure status; secret rotation compliance; suppression rate audit	KPI report; audit finding status report; escalation of any SLA breaches
Quarterly Engineering Review	Quarterly	Full KPI trend; risk register update; IaC coverage report; drift pattern analysis; process improvement backlog; opportunities assessment	Updated risk register; process change decisions; procedure amendment where required
Post-Drift Root Cause Review	After any confirmed configuration drift in production or staging	Drift timeline; root cause; whether drift detection SLAs were met; whether the	Root cause record; corrective action; procedure

		baseline is achievable in practice	or baseline update where applicable
Annual Procedure Review	Annually (minimum)	Full document review; ISO 9001:2015 and ISO 27001:2022 alignment; tooling landscape; regulatory changes	Updated procedure version; amendment record; re-approval cycle

6.2 Non-Conformance Handling

The following conditions are non-conformances under this procedure and require a documented corrective action within 5 business days:

- A configuration change applied to a shared environment without a corresponding approved Change Request (CR) record in the change management system
- Configuration drift in production or staging not acknowledged within the MTTD-D SLA defined in Section 4.2
- Configuration drift in production or staging not remediated within the MTTR-D SLA defined in Section 4.2
- A CI operating in a shared environment without an associated, validated CMDB asset ID
- IaC coverage below 100% for production or staging without an approved Change Request authorising a remediation plan and timeline
- A policy suppression within an IaC definition lacking a metadata reference to an approved Change Request token
- A secret or credential that has exceeded its rotation SLA without an approved Change Request authorising an extension
- An audit finding that has exceeded its closure SLA without an escalation record and a named remediation owner
- A CMDB baseline record that has not been reviewed within 90 days without a Configuration Manager-approved extension

Non-conformances are logged in the QMS non-conformance register and reviewed at the quarterly engineering review. Repeat non-conformances on the same item trigger a formal corrective action process.

6.3 Document Control

This procedure is reviewed whenever: a post-drift review identifies a process gap; an audit finding reveals a procedural deficiency; tooling changes affect a defined control; or the annual review cycle is due.