OpenFactory

INTEGRATION GUIDE

ServiceNow Integration

GxP Deployment Lifecycle & Incident Management with ServiceNow

OpenFactory Enterprise Integration Series

February 2026

CONFIDENTIAL

Overview

OpenFactory is an open-source, AI-powered platform that manages the complete lifecycle of hardened operating system images in regulated (GxP) environments. Every stage — from recipe creation through continuous runtime monitoring — integrates with ServiceNow to provide enterprise-grade change management, configuration tracking, compliance evidence, and incident response.

This document describes the six-stage deployment lifecycle, the ServiceNow integration points at each stage, and the GxP compliance checkpoints that ensure 21 CFR Part 11, SOC 2, and ISO 27001 auditability throughout.

Design Motivation

The Enterprise Integration Gap

Building a hardened OS image is only the beginning. Regulated industries require that every change to production systems is documented, approved, verified, and traceable. The traditional approach — manual change requests, spreadsheet-based CMDB tracking, and disconnected incident management — creates gaps that auditors exploit and that operators struggle to maintain.

OpenFactory closes this gap by treating ServiceNow as a first-class integration partner, not an afterthought. Every lifecycle event generates a corresponding ServiceNow record automatically. The integration is bidirectional: ServiceNow approvals gate deployments, and ServiceNow incidents trigger automated remediation.

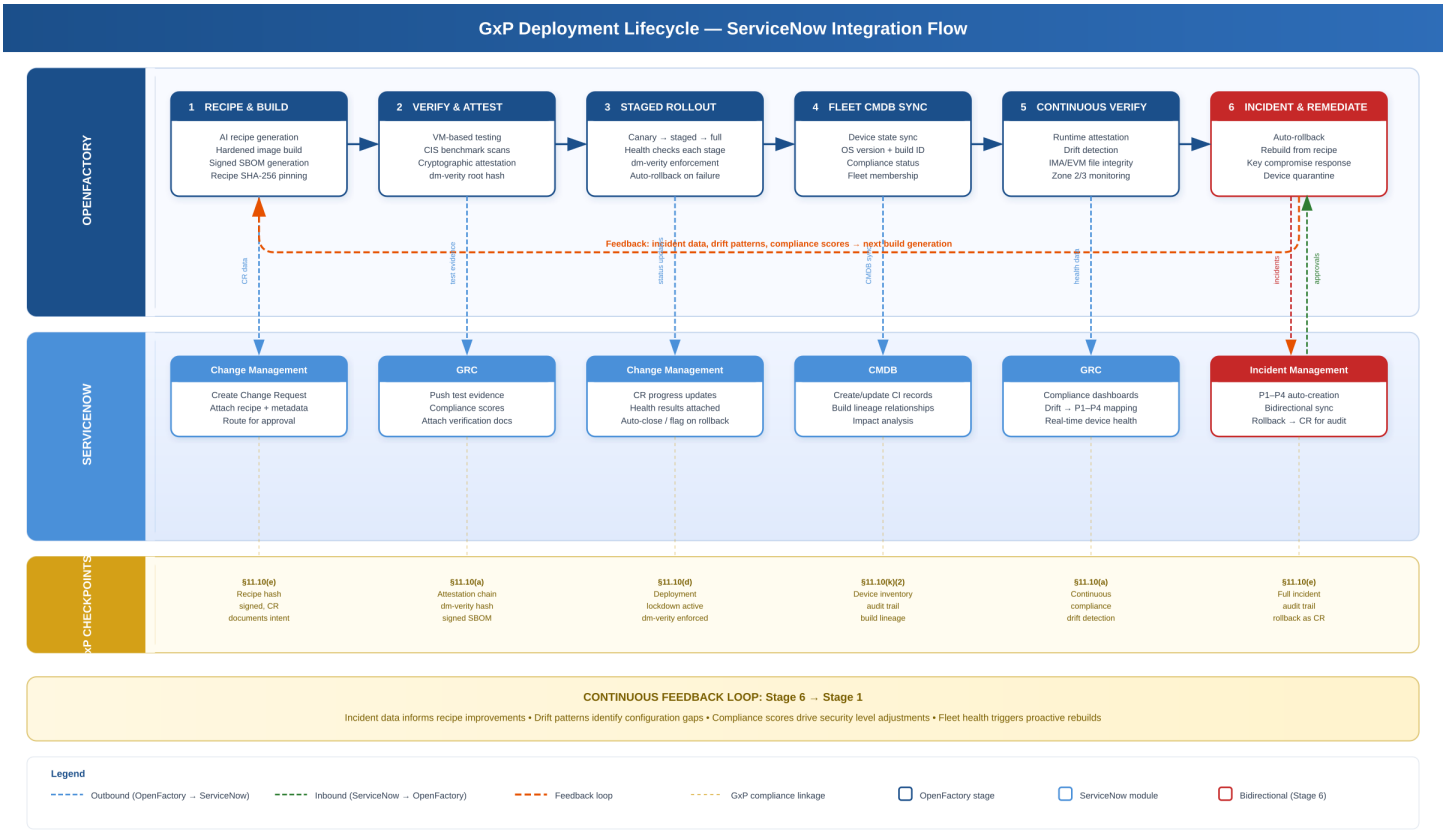
Compliance Drivers

Standard	Requirement	Lifecycle Implication
21 CFR 11 §11.10(a)	Validation of systems to ensure accuracy and reliability	Every build is verified in a VM before deployment
21 CFR 11 §11.10(d)	Limiting system access to authorized individuals	Change Request approval gates deployment
21 CFR 11 §11.10(e)	Secure, computer-generated, time-stamped audit trails	Every lifecycle event logged in both OpenFactory and ServiceNow
21 CFR 11 §11.10(k)(2)	Appropriate controls over systems documentation	SBOM, verification docs, and attestation reports linked to CRs
SOC 2 CC6.1	Logical and physical access controls	Role-based access, OAuth 2.0 scoped credentials
SOC 2 CC7.2	System monitoring for anomalies	Continuous verification with drift detection and alerting
SOC 2 CC8.1	Change management	Automated CR creation, approval tracking, evidence attachment

ISO 27001 A.12.1.2	Change management	Formal change process with ServiceNow workflow integration
ISO 27001 A.14.2.2	System change control procedures	Build-to-deploy audit trail with CMDB traceability

Deployment Lifecycle

OpenFactory manages six lifecycle stages, each with dedicated ServiceNow integration points and GxP compliance checkpoints.



Stage 1: Recipe & Build

OpenFactory Action: AI recipe generation from natural language, hardened image build with deterministic configuration, signed SBOM generation.

ServiceNow Module: Change Management — Change Request auto-created with build metadata including requested features, base image version, requester identity, and risk assessment.

GxP Checkpoint: Recipe hash signed and recorded. Build intent fully documented in the Change Request before any build begins. The recipe SHA-256 pins exactly what was requested; the CR provides the approval chain.

Data Flow: OpenFactory → ServiceNow (outbound)

Change Request Fields

Field	Value
Short description	Custom {base} build: {feature_list}

Category	Operating System
Risk	Based on security level (minimal/standard/strict)
Requested by	Build requester email
Justification	Original natural language request
Attachments	Recipe JSON, SBOM (on completion)

Stage 2: Verify & Attest

OpenFactory Action: VM-based testing (boot, package, service, compliance, performance), SBOM validation, CIS benchmark scans, GxP compliance audit.

ServiceNow Module: Governance, Risk, and Compliance (GRC) — Test evidence pushed including pass/fail results, compliance scores, and verification document with cryptographic attestation chain.

GxP Checkpoint: Cryptographic attestation chain established. dm-verity root hash computed and recorded. Every file in the image hashed (SHA-256) and included in the signed verification document. Build log hash signed. Recipe-to-image binding is cryptographically provable.

Verification Chain

Artifact	Hash / Signature	Purpose
Recipe JSON	SHA-256	Pins build intent
Build log	SHA-256, signed	Proves exact build steps
Squashfs image	dm-verity root hash	Block-level integrity
File manifest	Per-file SHA-256	File-level integrity
SBOM	Signed CycloneDX/SPDX	Package inventory
Verification document	Ed25519 signature	Binds all hashes to trusted build platform

Stage 3: Staged Rollout

OpenFactory Action: Canary deployment → staged rollout → full fleet deploy. Health checks at each stage. Automatic rollback on failure with no manual intervention required.

ServiceNow Module: Change Management — Change Request updated with rollout progress at each stage. Health check results attached. CR auto-closes on successful full deployment; CR flagged for review on rollback.

GxP Checkpoint: Deployment lockdown activated after verification passes. dm-verity enforcement enabled — base image is read-only and block-level verified. Overlay policy engine activated.

Rollout Stages

Stage	Target	Pass Criteria	Failure Action
Canary	1 device	Boot + health check + 15min soak	Rollback canary, halt
Staged	10% fleet	All health checks pass, no drift	Rollback staged, halt
Full	100% fleet	All health checks pass within SLA	Rollback to previous image

Stage 4: Fleet CMDB Sync

OpenFactory Action: Device state synchronization — OS version, build ID, fleet membership, compliance status, last attestation timestamp, overlay state.

ServiceNow Module: Configuration Management Database (CMDB) — Each deployed device registered as a Configuration Item (CI). Relationships established: base image → custom build → deployed device. Enables impact analysis, dependency tracking, and lifecycle management.

GxP Checkpoint: Device inventory audit trail. Every device's build lineage is traceable — from the recipe that defined it, through the build that produced it, to the deployment that installed it. CMDB provides the authoritative source of truth for "what is running where."

CMDB Configuration Item Mapping

OpenFactory Field	ServiceNow Field	Notes
device.id	asset_tag / serial_number	Unique device identifier
device.os_version	os_version	Current OS build version
device.fleet	assignment_group	Maps fleet to ServiceNow group
device.build_recipe	short_description / custom	Reference to build configuration
device.compliance_status	operational_status	Compliant / Drifted / Unknown
device.last_verified	last_discovered	Timestamp of last compliance check
device.distribution	os (operating system)	Ubuntu, Debian, Fedora, Elster OS
build.features	custom attributes	docker, ssh, hardening, auditd,

		etc.
build.security_level	custom attributes	minimal / standard / strict

Stage 5: Continuous Verification

OpenFactory Action: Runtime attestation at configurable intervals. Drift detection against build-time baseline. Health checks covering service state, file integrity (IMA/EVM), overlay mutations, and resource utilization.

ServiceNow Module: Governance, Risk, and Compliance (GRC) — Compliance dashboards populated with real-time device health. Drift events mapped to ServiceNow priority levels (P1–P4) based on severity and blast radius.

GxP Checkpoint: 21 CFR Part 11 continuous compliance. The system continuously proves that deployed devices match their approved build configuration. Any deviation is detected, classified, and reported — maintaining the validated state that regulators require.

Drift Severity Mapping

Priority	Drift Type	Example
P1	Security-critical file modified	/etc/ssh/sshd_config changed, PAM module replaced
P2	Service state deviation	Required service stopped, unauthorized service started
P3	Configuration drift	NTP server changed, DNS resolver modified
P4	Non-critical overlay write	Log rotation config changed, user preference file modified

Stage 6: Incident & Remediation

OpenFactory Action: Automated rollback to last known-good image on critical incidents. Rebuild from original recipe on confirmed compromise. Quarantine affected devices during investigation.

ServiceNow Module: Incident Management — Incidents auto-created with severity P1–P4 based on drift classification. Bidirectional sync: ServiceNow incident updates trigger OpenFactory remediation actions. Rollback operations recorded as Change Requests for audit trail.

GxP Checkpoint: Full incident audit trail from detection through resolution. Every rollback is recorded as a formal change. The remediation action, its justification, affected devices, and outcome are all documented in ServiceNow — providing the complete audit trail regulators require for deviation management.

Incident Priority Matrix

Priority	Criteria	Respond	Resolve	Automation
P1 — Critical	Security file tampered, key compromise, dm-verity failure	15 min	4 hours	Auto-rollback, auto-quarantine, auto-incident
P2 — High	Required service stopped, unauthorized	30 min	8 hours	Auto-incident, alert escalation

	service, attestation failure			
P3 — Medium	Configuration drift, non-critical compliance deviation	4 hours	48 hours	Auto-incident creation
P4 — Low	Non-critical overlay change, informational drift	24 hours	1 week	Logged, batched into daily report

Automated Remediation Actions

Trigger	OpenFactory Action	ServiceNow Record
dm-verity failure	Quarantine device, rebuild from recipe	P1 incident + CR for rebuild
Critical file tampered	Rollback to last known-good image	P1 incident + CR for rollback
Required service down	Restart service, escalate if restart fails	P2 incident
Configuration drift	Log drift, apply remediation policy	P3 incident
Key compromise detected	Revoke key, quarantine builds, rebuild	P1 incident + CRs for affected builds
Attestation timeout	Re-probe device, escalate if unreachable	P2 incident

Bidirectional Sync Detail

OpenFactory → ServiceNow: Incident creation with severity, affected device, build ID, and evidence. Work notes updated with remediation progress. Resolution recorded with rollback/rebuild details.

ServiceNow → OpenFactory (via webhook): CR approval unblocks gated deployments. Incident assignment notifies responsible engineers. Remediation instructions trigger specific actions (rollback, rebuild, quarantine). Incident closure updates device status in fleet manager.

Key Compromise Response

Key compromise is a distinct class of failure from attestation drift. While drift detects when a deployed system's runtime state diverges from its build-time configuration, a compromised signing key means the build-time configuration itself may be fraudulent. A compromised build signing key means the verification document — the root of the trust chain — cannot be trusted. Attestation that cross-references a fraudulent verification document produces confident-looking results for a tampered system.

OpenFactory's key compromise response operates through four phases, each generating specific ServiceNow records to maintain the complete audit trail required for GxP compliance.

Phase 1: Contain (≤ 1 Hour)

Objective: Stop the compromised key from being used for new operations.

- Revoke the compromised key in the vault — immediately prevents new signatures
- Publish revocation to fleet; rotate secrets for short-lived keys; publish CRL/OCSP for long-lived keys
- Halt all builds signed with the compromised key and mark as quarantined
- Isolate the compromised component — take build worker offline or revoke device certificate

ServiceNow Records Created

Record Type	Content	Purpose
P1 Incident	Key compromise detected: key ID, key type, suspected compromise window, initial blast radius estimate	Triggers incident response workflow, pages on-call team
Change Request (Emergency)	Emergency key revocation: key ID, revocation timestamp, affected services	Documents the revocation action for audit trail; bypasses standard CR approval for emergency
CMDB CI Update	All devices with builds signed by the compromised key marked as "Quarantined"	Operational status change visible fleet-wide for impact analysis

Phase 2: Assess (1–4 Hours)

Objective: Determine the blast radius — which artifacts, devices, and time window are affected.

- Identify the compromise window via vault audit logs
- Enumerate all artifacts (ISOs, verification documents) signed during the window
- Enumerate all devices running builds from the compromised window
- Cross-reference with runtime attestation for IMA mismatches and unexpected services

Severity Classification by Key Type

Compromised Key	Severity	Affected Scope

Root signing key	Critical	All subordinate keys must be rotated. Full re-build required.
Build platform signing key	High	All verification documents in the compromise window
Release signing key	High	All ISOs distributed during the compromise window
Device attestation key	Medium	Single device. Re-provision the device.
JWT session secret	High	All active sessions. Force re-authentication.
Fernet encryption key	High	ServiceNow credentials decryptable — rotate all SN OAuth tokens immediately
SSH host key	Low	Single device. Regenerate host keys.
LUKS passphrase	High	All devices sharing the passphrase. Full disk decryption risk.
Git deploy key	High	Attacker may have modified build-iso repos

ServiceNow Records Updated

Record Type	Content	Purpose
P1 Incident (update)	Blast radius enumeration: count of affected artifacts, count of affected devices, compromise window timestamps	Work notes capture assessment findings as they develop
GRC Compliance Item	Affected compliance frameworks: 21 CFR 11 §11.10(d) authority controls, SOC 2 CC7.2 anomaly monitoring	Documents which compliance controls are impacted
CMDB Relationship Query	Impact analysis: “Runs on” relationships from compromised build CIs to all deployed devices	Determines which devices need remediation

Phase 3: Remediate (4–48 Hours)

Objective: Replace the compromised key, re-sign or re-build affected artifacts, and update the fleet.

1. Generate a new key with metadata linking to the incident ID
2. Re-sign affected artifacts if confirmed untampered — re-hash from stored ISO, compare, re-sign with new key
3. Re-build tampered artifacts — retrieve original recipe, execute new build, sign new verification document
4. Push fleet updates — updated verification documents for good builds; OTA updates for tampered builds
5. Rotate all dependent keys if root signing key was compromised

ServiceNow Records Created

Record Type	Content	Purpose
Change Request (per	Rebuild triggered by incident INC#:	Links each remediation rebuild to the

rebuild)	original recipe, new build ID, new signing key version	originating incident for audit trail
Change Request (key rotation)	Key rotation: old key ID revoked, new key ID generated, rotation scope	Documents cryptographic key lifecycle change
CMDB CI Updates	Affected device CIs updated with new build ID, new verification document hash, status → "Remediated"	Tracks fleet convergence to known-good state
GRC Evidence	Re-verification results: new attestation chain for rebuilt artifacts	Compliance evidence that remediation restored validated state

Phase 4: Recover (24–72 Hours)

Objective: Verify fleet convergence to known-good state and complete the incident.

1. Trigger fleet-wide attestation — 99% convergence within 24 hours, 100% within 72 hours
2. Confirm CRL/OCSP includes the revoked key
3. Post-incident review: root cause analysis, detection timeline, response timeline
4. Update key policies based on lessons learned

ServiceNow Records Closed

Record Type	Content	Purpose
P1 Incident (resolve)	Resolution: fleet convergence confirmed, root cause identified, policy changes enacted	Closes the incident with full remediation evidence
All associated CRs	Close notes: linked rebuilds completed, key rotations verified, fleet attestation passing	Closes the audit trail for each remediation action
GRC Post-Incident Report	Lessons learned, updated key management policies, revised rotation schedules	Feeds back into compliance framework for continuous improvement
CMDB CI Updates	All affected devices: status → "Compliant", new verification document linked	Fleet returned to known-good baseline

Zone 2 Overlay Change Response

OpenFactory's Persistent Overlay architecture separates the immutable base image (Zone 1) from the device's persistent configuration (Zone 2) and ephemeral runtime state (Zone 3). Zone 2 contains the device's accumulated identity: network configuration, TLS certificates, SSH host keys, calibration data, and audit logs. Because Zone 2 persists across base image updates and is protected by deployment lockdown, any unauthorized change to Zone 2 is a significant security event that generates specific ServiceNow records.

Zone Architecture Summary

Zone	Storage	Lifecycle	Contents
Zone 1: Immutable Base	SquashFS (read-only)	Replaced atomically on update	Kernel, libs, binaries, default configs
Zone 2: Persistent Overlay	ext4 (read-write)	Preserved across base updates	Network config, certs, calibration, logs
Zone 3: Ephemeral Overlay	tmpfs or ext4	Wiped on base update	Caches, temp files, PID files, runtime state

Unauthorized Zone 2 Modification

When a device is in the Locked state, Zone 2 is mounted read-only and monitored by a daemon that validates integrity against a sealed lockdown reference. Any modification attempt — whether from a compromised process, an operator bypassing fleet authorization, or a direct block device write — is detected by both file-level monitoring (fanotify) and periodic hash-level verification.

ServiceNow Integration

Trigger	ServiceNow Action	Priority
Zone 2 hash mismatch (daemon detects modification)	P1 Incident created: device ID, sealed hash vs. current hash, timestamp, process ID of writer if available	P1 — Critical: locked overlay tampered
Write attempt to read-only Zone 2 in deny mode	P2 Incident created: device ID, blocked path, process attempting write, operation type	P2 — High: attempted unauthorized modification
New executable detected in Zone 2	P1 Incident created: device ID, file path, file hash, Zone 2 manifest violation (no binaries permitted)	P1 — Critical: Zone 2 governance prohibits executables
Zone 2 file outside manifest	P3 Incident created: device ID, unexpected path, anomaly classification	P3 — Medium: unrecognized configuration file

Zone 2 Constraint Violations

Beyond path-level governance, Zone 2 values are validated against a constraint manifest. An IP address in the wrong subnet, a certificate that has expired, or GPS coordinates outside the expected deployment region are all policy violations that flow into ServiceNow.

ServiceNow Integration

Constraint Violation	ServiceNow Action	Priority
Network address outside permitted subnet	P2 Incident: device may be on wrong network segment; assignment_group set to network ops	P2 — High
TLS certificate expired or invalid issuer	P2 Incident: device identity compromised; CMDB CI updated with “Cert Expired” flag	P2 — High
GPS coordinates outside installation zone	P1 Incident: device may have been physically relocated without authorization	P1 — Critical
Hostname format violation	P3 Incident: non-conforming hostname; flags potential misconfiguration	P3 — Medium
Department code not in allowed set	P3 Incident: device assigned to unknown department	P3 — Medium
Certificate key usage flags mismatch	P2 Incident: certificate does not meet key usage requirements	P2 — High

Zone 2 Migration Failures

When a base image update triggers the overlay migration service, Zone 2 is prepared for compatibility with the new base through five phases: whiteout resolution, schema migration, path relocation, anomaly detection, and integrity check. If any phase fails, Zone 2 is restored from its pre-migration snapshot and the failure is reported.

ServiceNow Integration

Migration Event	ServiceNow Action	Details
Whiteout conflict: stale whiteout hides new base content	P2 Incident + CR update: migration halted, snapshot restored, CR flagged for review	Whiteout from previous admin action hides required config in new base version
Schema migration transform failure	P2 Incident: config file in Zone 2 cannot be migrated to new schema; device held on old base version	Migration manifest declared a transform, but the existing config cannot be converted
Anomaly detection: unexpected binary in Zone 2	P1 Incident: binary found in persistent overlay violates Zone 2 governance; migration halted, device quarantined	Zone 2 should never contain executables; their presence suggests tampering
Integrity check failure:	P2 Incident: device identity may	New base version introduced a

certificate chain broken after migration	be invalid post-migration; re-provisioning required	new CA requirement that existing device certs do not chain to
Post-boot validation failure after migration	P2 Incident + automatic rollback: Zone 2 restored from snapshot, previous base image reactivated, CR updated with rollback details	Device booted on new base but constraint validation failed; automatic recovery triggered
Successful migration	CR updated: migration phases completed, constraint validation passed, new lockdown seal written	Normal path — no incident; CR progresses through rollout stages

Lockdown State Transitions

Zone 2 lockdown state changes are tracked in ServiceNow to maintain the audit trail. Each transition is logged to ensure regulators can verify that devices spent the vast majority of their operational life in the Locked state.

State Transition	Zone 2 Access	Monitoring Mode	ServiceNow Record
Locked → Migration	Read-write	Audit mode	CR updated: migration window opened, snapshot taken
Migration → Locked	Read-only	Deny mode	CR updated: migration complete, new seal written, lockdown restored
Locked → Unlocked	Read-write	Audit mode	Emergency CR: unlock authorized by fleet command, justification required, operator logged
Unlocked → Locked	Read-only	Deny mode	CR closed: re-provisioning complete, new combined SBOM sealed, lockdown restored

Continuous Lifecycle

The lifecycle is not linear — it is a continuous loop. Stage 6 feeds back into Stage 1:

- **Incident data** informs recipe improvements for the next build generation
- **Drift patterns** identify configuration gaps that should be addressed in build-time hooks
- **Compliance scores** drive security level adjustments in future recipes
- **Fleet health trends** trigger proactive rebuild cycles before degradation reaches incident threshold

Integration Architecture

Three-Component Design

The OpenFactory–ServiceNow integration uses a three-component architecture for durable, auditable data exchange:

Component	Responsibility
Event Router	Captures build, deploy, and compliance events from OpenFactory and routes them to the appropriate ServiceNow API endpoint (Change, CMDB, GRC, or Incident). Event classification determines the target module.
Sync Queue	Durable message queue with retry logic and dead-letter handling. Ensures no event is lost even during ServiceNow maintenance windows. Failed deliveries retried with exponential backoff. Dead-lettered events trigger operator alerts.
Webhook Receiver	Bidirectional endpoint receiving ServiceNow callbacks — CR approvals, incident updates, remediation instructions — and triggers corresponding OpenFactory actions (gate builds on approval, execute rollback on incident).

Authentication

- OAuth 2.0 Client Credentials flow for all outbound API calls
- Scoped credentials — each integration point uses the minimum ServiceNow role required
- Access tokens are short-lived (30 minutes) and automatically refreshed
- Client secrets encrypted at rest (Fernet AES-128-CBC + HMAC-SHA256)
- All integration events logged to an immutable audit trail

ServiceNow API Endpoints

Lifecycle Stage	Operation	Method	Table / Resource
1. Recipe & Build	Create CR	POST	/api/now/table/change_request
2. Verify & Attest	Attach evidence	POST	/api/now/table/change_request/{sys_id}
3. Staged Rollout	Update CR	PATCH	/api/now/table/change_request/{sys_id}

4. Fleet CMDB Sync	Create/update CI	POST/ PATCH	/api/now/table/cmdb_ci_os_image
4. Fleet CMDB Sync	Create relationship	POST	/api/now/table/cmdb_rel_ci
5. Continuous Verify	Push compliance	POST	/api/now/table/sn_grc_item
6. Incident	Create incident	POST	/api/now/table/incident
6. Incident	Webhook callback	POST	OpenFactory /api/webhooks/servicenow

GxP Compliance Summary

Compliance Checkpoints by Stage

Stage	Checkpoint	Regulation
1. Recipe & Build	Recipe hash signed, build intent documented in CR	21 CFR 11 §11.10(e)
2. Verify & Attest	Cryptographic attestation chain, dm-verity hash, signed SBOM	21 CFR 11 §11.10(a)
3. Staged Rollout	Deployment lockdown, dm-verity enforced, overlay policy active	21 CFR 11 §11.10(d)
4. Fleet CMDB Sync	Device inventory audit trail, build lineage traceable	21 CFR 11 §11.10(k)(2)
5. Continuous Verify	Continuous compliance, drift detection	21 CFR 11 §11.10(a)
6. Incident & Remed.	Full incident audit trail, rollback as change records	21 CFR 11 §11.10(e)

Integrity Maturity Model

The lifecycle supports six levels of increasing integrity assurance:

Level	Name	What It Proves
L1	Package audit	SBOM generation and package verification — you know what is installed
L2	Image signing	Cryptographic signatures on build artifacts — you know who built it
L3	Runtime verify	dm-verity base layer verification at boot — you know it has not been tampered with
L4	Overlay governance	Policy-enforced overlay writes with audit trail — you know what changed and why
L5	Attestation	Continuous runtime integrity reporting — you know the current state
L6	Hardware root	TPM-backed measured boot and sealed secrets — hardware-rooted proof

Implementation Timeline

A typical OpenFactory–ServiceNow integration deployment follows a phased approach spanning 4–8 weeks.

Phase	Duration	Activities
Discovery	Week 1	Review ServiceNow instance config, define use cases (which events create incidents vs. CRs), identify CMDB schema customizations, confirm compliance standards in scope
Configuration	Weeks 2–3	OAuth app creation, field mapping setup, business rule creation for inbound webhooks, fleet-to-assignment-group mapping, GRC module configuration
Testing	Weeks 4–5	End-to-end testing with ServiceNow developer instance: simulate deployments, drift detection, rollbacks, bidirectional status sync, and GxP audit trail validation

Go-Live	Week 6	Production deployment, monitoring dashboard config, runbook handoff, initial CMDB sync of all managed devices, GRC baseline population
Optimization	Weeks 7–8	Performance tuning, additional field mappings, compliance report customization, incident response drill, feedback incorporation

Support & SLAs

Integration support is included with the OpenFactory Enterprise tier:

- **P1 (integration down):** 1-hour response, 4-hour resolution target
- **P2 (degraded sync):** 4-hour response, 1-business-day resolution target
- **P3 (configuration help):** 1-business-day response
- Dedicated integration support channel via Slack or email
- Quarterly integration health reviews and optimization recommendations
- Access to OpenFactory's open-source integration codebase for self-service customization

Next Steps

To begin the OpenFactory–ServiceNow integration process:

6. Schedule a discovery call with the OpenFactory integration team to review your ServiceNow environment and fleet management requirements
7. Provision a free ServiceNow Personal Developer Instance at developer.servicenow.com for testing
8. Identify stakeholders from your ServiceNow administration team and your fleet operations team
9. Prepare a list of device fleets, compliance standards in scope (GxP, SOC 2, ISO 27001), and desired field mappings
10. Review your ServiceNow instance's API rate limits and adjust if necessary for fleet-scale sync volumes

For questions or to get started, contact your OpenFactory account manager or book a demo at openfactory.tech/book-demo