Gebruik het programmeerbare installatieprogramma

Inhoud

Inleiding

Dit document beschrijft het Programmable Installer, een specifiek automatiseringsplatform voor het implementeren van Cisco-softwarestacks zoals NSO en CNC.

programmeerbaar installatieprogramma

Veld	Waarde
Product	programmeerbaar installatieprogramma
Documenttype	Technisch witboek — architectuur, implementatie en resultaten
Primair publiek	Oplossingsarchitecten, platformtechnici, DevOps / SRE, leveringsleads
Secundair publiek	Engineering management, Security reviewers, Program managers

samenvatten

Het programmeerbare installatieprogramma is een specifiek automatiseringsplatform voor de implementatie en exploitatie van Cisco-softwarestacks, waaronder Network Services Orchestrator (NSO), Crosswork Network Controller (CNC), Crosswork Data Gateway (CDG) en Business Process Automation (BPA), op Enterprise Linux (RHEL-familie) en bijbehorende infrastructuur waar van toepassing (VMware vCenter, OpenShift, KVM, air-gapped Kubernetes). Het systeem scheidt declaratieve intentie (YAML-specificaties en optionele geleide intentiegeneratie) van uitvoering (Ansible-rollen en playbooks), met een Python-besturingsvlak dat artefacten verpakt, bundels verifieert voordat langlopende installaties worden geïnstalleerd, gecodeerde geheimen voorbereidt en validatiepoorten orkestreert.

In deze whitepaper worden de architectuurlagen, primaire gegevensstromen, implementatiepatronen (waaronder een hybride, gegevensgestuurd verificatiemodel voor artefacten), implementatiemodi (native, gecontaineriseerd, online, air-gapped) en de kaders voor validatie en registratie toegelicht. Voor leverings- en platformorganisaties heeft het platform als doel handmatige arbeid, verkeerde configuratie van oppervlakken en ontbrekende binaries vroegtijdig te verminderen en automatisering voor verschillende producten en topologieën te

standaardiseren met behoud van omgevingsspecifieke parameterisatie.

Trefwoorden: infrastructuurautomatisering, declaratieve implementatie, Ansible, YAML-specificatie, air-gap-verpakking, artefactverificatie, kruiswerk, NSO, CNC, CDG, BPA, validatiebeleid, DevOps.

Hoe dit document te lezen

rol	Aanbevolen focus
Besluitvormers / leads	Abstract; §1 Waardevoorstel; §8 Voordelen en risicohouding; §10 Conclusie
Oplossingsarchitecten	§3–§6 (architectuur, specificatiemodel, implementatie, implementatiepatronen)
DevOps / SRE / delivery engineers	§5–§7; Aanhangsel B; interne bijlagen van het begeleidende witboek
Beveiligingscontroleurs	§7 Beveiliging en compliance houding; vertrouwensgrenzen in §3.2

---

1. Waardevoorstel

1.1 Het leveringsprobleem

De bedrijfsinstallatie van multi-tier netwerkautomatiserings- en orkestratieproducten is van oudsher high-touch: lange runbooks, veel handmatige stappen, versiescheefheid tussen sites en fouten die uren in een proces opduiken (ontbrekende NED's, verkeerde OVA-paden, onvolledige air-gap-beeldsets). Dat patroon verhoogt de kosten, verlengt wijzigingsvensters en maakt audits moeilijker.

1.2 De programmeerbare installateur aanpak

De programmeerbare installateur behandelt een installatie als een programma dat wordt geparametriseerd door een specificatie: topologie, versies, platformkeuze (vCenter vs OpenShift vs vanilla VM's), bestandspaden en rechten. Automatisering is idempotent waar mogelijk, herhaalbaar voor alle klanten en vooraf geladen met controles, zodat 'niet klaar' een snel, expliciet resultaat is voordat cluster of product wordt geïnstalleerd.

1.3 Wat programmeerbaar betekent in deze context

- Declaratief: operators beschrijven wat moet worden geïmplementeerd; playbooks implementeren hoe.
- Gegevensgestuurde verificatie: verwachte artefacten zijn afgeleid van tabellen en regels in plaats van ad-hocscripts per release, wanneer patronen stabiel zijn.
- Kwaliteit op basis van beleidsregels: validatie vóór en na implementatie verloopt volgens hiërarchisch beleid, met gestructureerde rapporten en optionele ticketintegratie.
- Multimodale werking: interactieve menu's voor nieuwe gebruikers; CLI en bevroren binaries voor herhalingen in CI/CD-stijl; Docker-gebaseerde flows voor gestandaardiseerde runtime-images.

---

2. Systeemcontext

2.1 Actoren en omgevingen

Actor/systeem	rol
Leveringsingenieur	Auteurs of genereert specs, voert verpakkingen, voorbereiding kluis, validatie, orchestrator en Ansible
Installer-host	Linux control node (native of container) met Python, Ansible-configuratie, schijf voor artefacten
Doelinfrastructuur	vCenter-, OpenShift/KubeVirt- of vanilla-VM's per specificatie
Artefactbronnen	Interne spiegels, rechtenlay-outs, softwaredistributie—omgevingsspecifiek
Downstreamsystemen	Monitoring, wijzigingsbeheer, optionele JIRA-workflows

2.2 Vertrouwensgrenzen

1. Specificaties drukken intentie uit; ze kunnen verwijzen naar paden en niet-geheime parameters. Geheimen moeten door Ansible Vault-werkstromen stromen stromen en niet door tekstvelden waar vermijdbaar.
2. Artefact-opslag moet worden beschermd tegen integriteit; verificatie richt zich op aanwezigheid en naamgeving in overeenstemming met de specificatie - uitbreiden met organisatorische controles (cheques, ondertekende bundels) waar beleid vereist.
3. Installer-to-target SSH is een pad met hoge bevoegdheden; het compromitteren van de installatiehost heeft een grote impact. Verharding en toegangscontrole zijn operationele voorwaarden.

---

3. Architectuurbeginselen

1. Declaratief eerst: Gebruikersintentie in YAML; automatisering interpreteert het consistent.
2. Scheiding van planning en uitvoering: Python plant, verifieert en orkestreert poorten; Ansible

voert infrastructuur- en productstappen uit.
3. Composeerbare automatisering: playbooks op siteniveau importeren gerichte playbooks (vooraf installeren, Kubernetes-tracks, productinstallaties).
4. Progressieve openbaarmaking: interactieve installatie voor onboarding; vlaggen en scripts voor geavanceerde automatisering.
5. Dezelfde codebase, meerdere runtimes: Native AlmaLinux/RHEL-paden en op Docker gebaseerde paden delen één repository-lay-out.
6. Expliciete air-gap-ondersteuning: pakket op een verbonden machine; een bundel overdragen; vereisten installeren en implementeren zonder runtime-afhankelijkheid van openbare netwerken.

---

4. Logische architectuur

4.1 Lagen

laag	verantwoordelijkheid
specificatie	Topologie, versies, platforms, paden, rechten
Besturingsplane	Verpakking, bundelverificatie, kluishelpers, validatiestuurprogramma, orchestrator-CLI
Automatiseringsvlak	Host prep, Kubernetes-levenscyclus, productinstallatie en configuratie op dag één
kunstvoorwerpen	Binaries, afbeeldingen, grafieken, OVA's, tarballs

4.2 End-to-end gegevensstromen

1. Pakketstroom: De vereiste tooling downloadt of brengt bestanden in een specifieke locatie, waarbij optioneel een overdraagbare tarball wordt geproduceerd voor offline installaties.
2. Stroom verifiëren: De orchestrator parseert de specificaties, lost verwachte artefacten op (inclusief platformfilters en rechtenlijsten) en rapporteert klaar / ontbrekend voordat de installatie wordt uitgevoerd.
3. Implementatiestroom: Spec plus vault (en optionele pre-validatiedrive) Ansible-playbooks tegen inventarissen die zijn afgeleid van de service.

4.3 Ondersteunde producten (reikwijdte installateur)

Product / bundel
NSO
CNC
CDG
BPA

Product / bundel
CNC + NSO

5. Specificatie en intentiemodel

5.1 Gebruikersspecificaties

Specificaties zijn YAML-documenten die platforms beschrijven (bijvoorbeeld vCenter, OCP, VM, KVM), hosts, toepassingen met versies, topologie (bijvoorbeeld NSO CFS/RFS-lay-outs), rechten (NED's en add-on-pakketten) en bestandspaden voor OVA's, qcow2-afbeeldingen en tarballs op toepassingsniveau.

5.2 Gemeenschappelijke fragmenten

Een specifieke user_spec-toepassing levert standaardwaarden en padterugvalmogelijkheden voor CNC/CDG. De parser van de orchestrator behandelt de gebruikersspecificatie als bron van waarheid en gebruikt gemeenschappelijke spec-vermeldingen wanneer gebruikerstoetsen afwezig zijn.

5.3 Opzettgeneratie

De intentiegenerator ondersteunt het gericht verzamelen van vereisten via een set vragenlijsten, een regelengine (datumgestuurde logica), en schema-backed mapping naar intent.yaml.

---

6. Implementatie Deep Dive

6.1 Orchestrator voor implementatie (cx_deploy_orchestrator.py)

De orchestrator is de enige ingang voor scripted of interactieve genereren-intentie, verifiëren-bundel, en de coördinatie te installeren. Het ontwerp is expliciet hybride:

- ARTIFACT_DEFS: declareert artefacttypen en naamgevingspatronen per toepassing (NSO-installatieprogramma, NED-ondertekende pakketten, optionele pakketten; CNC OVA/qcow2/tier tarballs; CDG-afbeeldingen; BPA-diagrammen en air-gap-beeldtarballs).
- APP_CONFIG: Kaarten - CLI-app-waarden (nso, crossworksuite, bpa) aan specificatiemapnamen en standaardintentiebestandsnamen.
- Parser / Handlers: CNC / CDG-paden oplossen met behulp van gebruikersspecificaties en

gemeenschappelijke specificaties; aangepaste handlers dekken niet-uniforme naamgeving (bijvoorbeeld TSDN / DLM) en BPA-versieformattering voor grafiek- en tarballepaden.

Gereedheid: AReportaaggregeert ontdekte artefacten; is_ready is true wanneer er geen vereiste bestanden ontbreken na het parseren van de specificaties. De module ondersteunt door PyInstaller bevroren binaries via sys.frozen path resolution.

6.2 Vereiste en verpakkingstools (setup_cxinstaller_prereqs)

Deze component biedt interactieve menu's en CLI-modi voor artefactverpakking en installatie van hostvereisten: online, air-gap of automatisch detecteren; verpakking voor meerdere toepassingen inclusief `gecombineerde CNC_NSO`. Het bevolkt de artefactboom die door Ansible wordt verwacht en door verificatie-bundellogica.

6.3 Ansible Automation Plane

- Samenstelling: Dit illustreert het multi-track karakter van de stack: het importeert verschillende Kubernetes bootstrap paden – wat aangeeft dat verschillende producten zich richten op verschillende Kubernetes bootstrap paden.
- Rollen (representatieve families): voorinstallatie (SELinux, firewall, SSH, registerhelpers), k8s_install / rke2, deploy_nso, deploy_cnc, deploy_cdg, deploy_bpa, postinstall, verwijderen en helpers voor certificaatvernieuwing. Eigendom en testen kunnen het best worden beheerd op rolgrenzen; geneste taakmappen implementeren subworkflows (bijvoorbeeld NSO L3 HA).

6.4 Kader voor validatiecontroles (validatie_controles/)

Het framework biedt hiërarchische beleidscontrole (globale → app → fase → individuele controle), automatische detectie van controles, verbeterde rapportage aan gestructureerde logs en optionele JIRA-integratie. Exploitanten voeren pre- of post-implementatiefasen uit met dezelfde specificatie die wordt gebruikt voor installatie, waarbij automatisering wordt afgestemd op kwaliteitspoorten die geschikt zijn voor bedrijfsveranderingsdiscipline.

Indicatieve schaal op een typische tak: in de orde van dertig controles over BPA en NSO (tellingen worden bevestigd met make-list-validatie-controles bij uw kassa).

6.5 Vault Secrets Manager (scripts/vault_secrets_manager.py)

Hiermee worden de vereiste vaultvariabelen afgeleid van specificaties, worden wachtwoorden

gevraagd of geaccepteerd onder het beleid en wordt gecodeerde group_vars/all_secrets.yaml plus een vaultwachtwoordbestand voor Ansible uitgezonden, waardoor ad-hocgeheime insluiting in afspeelboeken wordt verminderd.

---

7. Implementatiepatronen en runtimes

Patroon	Samenvatting
Standaard (AlmaLinux / RHEL)	PYTHONPATH en ANSIBLE_CONFIG instellen; verpakking, kluis, validatie, orkestrator en afspeelboeken uitvoeren per productgids
Docker-gebaseerd installatieprogramma	scripts/setup_installer.sh en scripts/start_installer.sh met hostbevestigingen voor grote artefacten;
Luchtspleet	Pakket op een verbonden machine; transferbundel; uitpakken op doel; installeren met-airgap
macOS-bundelcreatie	Gebruik sepython3 ./setup_cxinstaller_prereqs.py op Mac om bundels voor te bereiden; doelimplementatie blijft Linux-georiënteerd per project docs

---

8. Veiligheid, validatie en observeerbaarheid

8.1 Beveiligingshouding (ontwerpintentie)

- Geheimen: Geef de voorkeur aan gecodeerde groepsvariabelen van Ansible Vault; gebruik waar nodig strikte modi voor vaultbeheer.
- Installer host: behandelen als een controlevlak met een hoog vertrouwen; toegang en monitor beperken.
- Artefacten: Verkoopkanalen beschermen; organisatorische processen kunnen verificatie-bundel uitbreiden met cryptografische verificatie.
- Logboekregistratie: onderimplementatie/logs van toepassingen en Ansible-logs/

8.2 Validatie als een operationele gate

Voorbeeld aanroeping vóór implementatie:

```
cd /opt/cx-installer
python3 validation_checks/run_validation_checks.py -t pre -s specification/your_spec.yaml
```

Met optionele vlaggen:

- -p <policy_file>— Gebruik een aangepast validatiebeleid (standaard op validation_checks/validation_policies/default.yaml)
- -a <app> — beperk controles tot een specifieke app (kleine letters, zoals CNC, NSO, BPA, CDG)
- --report-file <path> — schrijf een zelfstandig JSON-voorcontroleverslag

8.3 Discipline bij vrijgave

Dit is een model voor inner sourcing waarbij voor meer CISCO applicatie-installatie hetzelfde principe kan worden gevolgd door het forken van de repository.

---

9. Voordelen en resultaten

thema	uitkomst
Tijd en zwoegen	Minder handmatige stappen; fouten gedetecteerd in verificatie-bundel- en validatiefasen in plaats van te laat in Ansible of product installateurs
consistentie	Gedeelde schema's, rollen en artefactlay-out voor alle opdrachten verminderen verschillen in "sneeuwvlok"
Niet-verbonden bewerkingen	Gedocumenteerde bundeloverdracht ondersteunt gereguleerde netwerken zonder runtime-downloads
bestuur	Gestructureerde validatierapporten en optionele JIRA-aansluitingen ondersteunen wijzigingsrecords en follow-up
uitbreidbaarheid	Uitbreidingspunten wissen: ARTIFACT_DEFS, handlers, nieuwe rollen/playbooks, intentieschema's

Kwantitatieve statistieken (installatieduur, defectpercentages) zijn organisatiespecifiek; teams moeten baselinewaarden vergelijken met bestaande runbooks over vergelijkbare topologieën.

---

10. Uitbreidbaarheid en onderhoud

10.1 Typen artefacten toevoegen

1. Verleng ARTEFACT_DEFS (en labels indien nodig) incx_deploy_orchestrator.py.
2. Voeg aangepaste handlers toe wanneer naamgeving niet alleen door patronen kan worden vastgelegd.
3. Verpakkingslogica bijwerken insetup_cxinstaller_preqswanneer downloads worden geautomatiseerd.

10.2 Ansible Behavior toevoegen

Geef de voorkeur aan nieuwe taken binnen samenhangende rollen; introduceer nieuwe rollen wanneer grenzen duidelijk zijn. Draad playbooks via import_playbook of gedocumenteerde entry playbooks. Bewaar veilige standaardinstellingen in group_vars / vars.

10.3 Evolutie van de intentiegenerator

Update YAML schema's underintent-generator / schema / en chatbot-ingangen; zorg ervoor dat gegenereerde bestanden overeenkomen met bestandsnamen die worden verwacht door APP_CONFIG.

10.4 Overwegingen van de routekaart (illustratief)

- Diepere integratie van SBOM of verificatie van beeldhandtekeningen.
- Uitgebreide validatiedekking voor CNC/CDG-scenario's.
- CI referenties voor spec linting en Ansible syntax checks.

---

11. Conclusie

De CX Programmable Installer combineert declaratieve specificaties, een Python-besturingsvlak voor verpakking en verificatie en een Ansible-automatiseringsvlak voor schaalbare, herhaalbare implementaties van Crosswork-gerelateerde producten in verschillende infrastructuur- en connectiviteitsmodellen. De architectuur scheidt opzettelijk intentie van uitvoering, past datagestuurde artefactverwachtingen toe waar praktisch en integreert validatie- en vaultworkflows die geschikt zijn voor zakelijke levering. Zie de bijbehorende interne whitepaper voor volledige operationele bijlagen (afspeelboektabellen, matrices voor probleemoplossing, connectiviteitsmatrices en uitgebreide opdrachtverwijzingen).

Referenties en documentatiekaart

document	Pad
Handleiding voor de operator	README.md
Afspeelboek vrijgeven	RELEASE_GUIDE.md
Interne architectuur bijlagen	docs/CX_INSTALLER_TECHNICAL_WHITE_PAPER_INTERNAL.md

document	Pad
Docker (online / air-gap / gebruik)	SETUP_ONLINE_DOCKER.md, SETUP_AIRGAPPED_DOCKER.md, USAGE_DOCKER.md
Valideringskader	docs/validation_checks/README.md
Vaultbeheer	docs/scripts/VAULT_SECRETS_MANAGER.md
Productgidsen	docs/nso.md, docs/bpa.md, docs/CNC_VCENTER_DEPLOYMENT_GUIDE.md, docs/CNC_OCP_DEPLOYMENT_GUIDE.md, docs/CNC_NSO_DEPLOYMENT_GUIDE.md
Intentiegenerator	intent-generator/README.md
Chatbot / regelstroomoverzicht	docs/HowItWorks.md

Bijlage A — Opmaak van opslagplaats (samenvatting)

```
cx-installer/
├── ansible_playbooks/     # ansible.cfg, files/, group_vars/, playbooks/, roles/, vars/
├── apps/                  # App-specific supporting content
├── deploy/                # Python deploy helpers, logging utilities
├── docs/                  # Technical documentation
├── intent-generator/      # Chatbot, rule engine, schemas, output/
├── scripts/               # Docker setup/start, vault_secrets_manager.py, ...
├── specification/         # User specs, samples, common fragments
├── validation_checks/     # Policies, runners, reports
├── cx_deploy_orchestrator.py
├── setup_cxinstaller_prereqs*
├── requirements.txt
└── README.md
```

Post-setup artefact focal points (typisch): ansible_playbooks/files/artefacts/, files/bin/, files/charts/, files/images/.

Aanhangsel B — Orchestrator CLI (samenvatting)

Script:cx_deploy_orchestrator.py

woordenstrijd	Beschrijving
--app / -a	NSO | CrossWorkSuite | BPA
--spec / -s	Pad naar YAML-specificatie
stap-voor-stap	Generatie-intentie | Verify-Bundle | Installeren
--Alleen verifiëren	Bundel controleren; niet-nul afsluiten als u niet klaar bent

woordenstrijd	Beschrijving
--dry-run	Testrun waar ondersteund
--list-specs	Bekende specificaties opnoemen

Omgeving (typische sessie):

```
export PYTHONPATH=$(pwd)
export ANSIBLE_CONFIG=$(pwd)/ansible_playbooks/ansible.cfg
```

Aanhangsel C — Woordenlijst

Begrip	Definitie
specificatie	YAML-gebruikersspecificatie: platforms, apps, topologie, paden, rechten
bedoeling	Genormaliseerde YAML van de intentiegenerator of het met de hand geschreven equivalent
bundel	Verpakte installateur boom (vaak tarball) voor air-gap overdracht
orkestrator	cx_deploy_orchestrator.py — Coördinatie controleren / plannen / installeren
Artefactverificatie	Bestandssysteem controleert of de vereiste binaries/images bestaan per specificatie
gewelf	Ansible Vault-gecodeerd variabel bestand voor geheimen
NED	Network Element Driver Package (NSO)
CFS / RFS	NSO-clusterforwarder/redundante forwarder-topologieconcepten
Luchtspleet	Omgeving zonder installateur-time toegang tot eindpunten voor pakketdownload

---

Documentrevisiegeschiedenis

Versie	datum	Opmerkingen
1.0	2026-03-27	Initiële publicatie-ready technische whitepaper (programmeerbare framing van installateurs)

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.