

Onderhoudbare detectie- en vergelijkingsketen voor in-house software in EPC en WSIT/CMDB

Realisatiedocument

Quinten De Meyer

Student Bachelor in de Elektronica-ICT — Cloud & Cyber Security

Stagebedrijf: Pfizer Puurs — ICS Infrastructure

Stagecoach Thomas More: Cas Magnus

Stagementor Pfizer: Nikki Lissens

Inhoudsopgave

Lijst van Figuren	3
1. Inleiding.....	5
2. Analyse.....	7
2.1 Uitgangspositie en databronnen.....	7
2.2 Functionele en niet-functionele vereisten	7
2.3 Technologiekeuzes	8
3. Realisatie van de EPC-CMDB Tool	10
3.1 Architectuur en ontwerpprincipes	10
3.2 Fase 1: Bronbestandbeheer	14
<i>YAML-bronbestand.....</i>	<i>14</i>
<i>Software Source Maintainer.....</i>	<i>16</i>
<i>EXE Reviewer</i>	<i>17</i>
<i>Detection Profile Generator.....</i>	<i>17</i>
3.3 Fase 2: Detectie en dataverzameling	18
<i>Detection Engine</i>	<i>18</i>
<i>Endpoint Central Upload Bundle Builder.....</i>	<i>21</i>
<i>Endpoint Central-uitvoering en Dynamic Custom Groups</i>	<i>21</i>
<i>Result Merger / Normalizer</i>	<i>25</i>
<i>WSIT Assets Export Script.....</i>	<i>26</i>
3.4 Fase 3: Vergelijking en CMDB-update	28
<i>Comparison Engine.....</i>	<i>28</i>
<i>WSIT API Import.....</i>	<i>29</i>
3.5 Overkoepelende gebruikerslaag en overdraagbaarheid	32
<i>Launcher-menu</i>	<i>32</i>
<i>Prerequisite check.....</i>	<i>33</i>
<i>Build- en releasevoorbereiding</i>	<i>34</i>
<i>Configuratie, outputrouting en documentatie</i>	<i>35</i>
4. Beperkingen.....	38
5. Besluit.....	40
Literatuurlijst	42
Bijlagen	43
5.1 Bijlage A — Begrippenlijst.....	43
5.2 Bijlage B — Mappenstructuur .OUTPUT	45
5.3 Bijlage C — Launcher-configuratie.....	45

Lijst van Figuren

FIGUUR 1 ARCHITECTUURDIAGRAM VAN DE EPC-CMDB TOOL, OPGEDEELD IN DRIE FASEN: BRONBESTANDBEHEER, DETECTIE EN DATAVERZAMELING, EN VERGELIJKING EN CMDB-UPDATE	11
FIGUUR 2 MAPPENSTRUCTUUR VAN DE GEDEELDE WERKLOCATIE MET BRON-, OUTPUT-, RELEASE- EN ONTWIKKELMAPPEN	12
FIGUUR 3 SNIPPET VAN YAML-BRONBESTAND TOONT IN-HOUSE SOFTWARE BATCH REPORT TOOL DAT 2 VERSIES BEVAT	14
FIGUUR 4 SOFTWARE SOURCE MAINTAINER WAARIN DE SOFTWARE ELOGPROCESSOR V1.0.2 IS GESELECTEERD.	16
FIGUUR 5 DETECTION PROFILE GENERATOR MENU	18
FIGUUR 6 DETECTION ENGINE UPLOAD BUNDLE BUILDER MENU, TOONT NA SUCCES OM DE STAPPEN IN DE GEBRUIKERSAANWIJZINGEN TE VOLGEN	21
FIGUUR 7 CONFIGURATIESCHERM VAN EEN CUSTOM SCRIPT IN ENDPOINT CENTRAL, MET HET LAUNCHERSRIPT EN DE AFHANKELIJKHEIDSBESTANDEN VAN EEN UPLOADPAKKET.....	22
FIGUUR 8 UPLOAD-BUNDLES FOLDER	23
FIGUUR 9 VOORBEELD VAN EEN UPLOAD BUNDEL DEPENDENCY FOLDER RECHTS, SCRIPT FOLDER LINKS, IN DIT GEVAL WINDOWS SERVER 2008 SERVICE PACK 2	23
FIGUUR 10 DETECTION ENGINE OUTPUT, RESULTATEN VAN ALLE ENDPOINTS	25
FIGUUR 11 WSIT ASSETS EXPORT SCRIPT MENU EN OPENT IN NIEUW VENSTER VOOR ACHTERGRONDEXPORT	27
FIGUUR 12 PAT SETUP.....	27
FIGUUR 13 VOORBEELD VAN HET HTML-DISCREPANTIERAPPORT, MET PER ENDPOINT DE GEÏDENTIFICEERDE AFWIJKINGEN TEN OPZICHTE VAN DE CMDB.....	28
FIGUUR 14 DETAILWEERGAVE VAN DE DISCREPANTIE-IMPACTTABEL, MET PER ENDPOINT EEN VERGELIJKING TUSSEN DE HUIDIGE CMDB-STATUS EN DE GEDETECTEERDE REALITEIT, INCLUSIEF DE VOORGESTELDE IMPORTACTIE.	29
FIGUUR 15 VOORVERTONING VAN DE CMDB-IMPORT (PRESERVATIE-TABEL), DIE INZICHTELIJK MAAKT HOE IN-HOUSE SOFTWARE WORDT BIJGEWERKT TERWIJL REGISTRATIES BUITEN DE SCOPE VEILIG BEHOUDEN BLIJVEN.	29

FIGUUR 16 WSIT REST API IMPORT MENU	31
FIGUUR 17 WSIT REST API IMPORT STAP DETAILS NA SUCCESVOLLE IMPORT	31
FIGUUR 18 EPC-CMDB TOOL LAUNCHER MENU	33
FIGUUR 19 PREREQUISITES MENU	34
FIGUUR 20 BUILD / RELEASE PREPERATION MENU	35

1. Inleiding

Dit realisatiedocument beschrijft de technische uitwerking van het stageproject rond de detectie en vergelijking van in-house software bij Pfizer Puurs. Pfizer Global Supply Puurs-Sint-Amands is de grootste Europese productie- en verpakkingsfaciliteit van Pfizer, waar jaarlijks meer dan 400 miljoen units inspuitbare geneesmiddelen en vaccins worden geproduceerd onder strikte kwaliteits- en veiligheidsnormen. Binnen die context is een correcte en actuele software-inventarisatie geen administratieve bijzaak, maar een randvoorwaarde voor betrouwbaar infrastructuurbeheer.

De probleemstelling, de centrale onderzoeksvraag en het evaluatiekader werden vastgelegd in het projectplan (De Meyer, 2026a). Kort samengevat: de standaardinventarisatie in Endpoint Central (EPC), het platform dat binnen Pfizer wordt ingezet voor endpointbeheer en scriptuitvoering, biedt onvoldoende zicht op in-house software. Kennis over die software zit verspreid over installatie-instructiedocumenten (IID's), bronmedia op gedeelde netwerklocaties en ervaring van medewerkers, en er bestond geen reproduceerbare methode om de feitelijke softwaretoestand op endpoints te vergelijken met de administratieve registratie in de Configuration Management Database.

De oplossing die hier wordt beschreven, is een aaneengeschakelde toolketen. Deze vertrekt vanuit een centraal bronbestand met softwarekennis en resulteert in een HTML-discrepantierapport en een importeerbaar bestand waarmee discrepanties automatisch via REST API in de CMDB verwerkt kunnen worden. Het projectplan werd na afloop beperkt geactualiseerd zodat de beschreven deliverables overeenkomen met de uiteindelijke realisatie. In dit realisatiedocument ligt de nadruk daarom op het eindresultaat.

De realisatie sluit grotendeels aan bij de deliverables uit het projectplan, maar bevat enkele afwijkingen. Tijdens de uitwerking zijn er namelijk bijkomende tools ontwikkeld die niet waren voorzien en kregen bepaalde deliverables een andere vorm dan eerder beschreven.

De verdere opbouw van dit realisatiedocument is als volgt. Hoofdstuk 2 bespreekt de analyse van de uitgangspositie, de belangrijkste vereisten en de technologiekeuzes. Hoofdstuk 3 beschrijft de

gerealiseerde EPC-CMDB Tool als eindresultaat, gestructureerd volgens de drie fasen van de workflow: bronbestandbeheer, detectie en dataverzameling, en vergelijking met CMDB-update. Hoofdstuk 4 beschrijft de belangrijkste beperkingen van de oplossing. Hoofdstuk 5 sluit af met een terugkoppeling naar de centrale onderzoeksvraag en het evaluatiekader.

Tot slot is een toelichting van alle gehanteerde begrippen en afkortingen opgenomen in bijlage A.

2. Analyse

De analysefase richtte zich op het begrijpen van de bestaande systemen, het afbakenen van de scope en het bepalen van de technische randvoorwaarden waarbinnen de oplossing moest functioneren. In dit hoofdstuk worden eerst de uitgangspositie en databronnen besproken, daarna de functionele en niet-functionele vereisten die de architectuur stuurden, en tot slot de technologiekeuzes.

2.1 Uitgangspositie en databronnen

De CMDB, gerealiseerd via Jira Assets binnen WSIT (de overkoepelende Jira Service Management-omgeving van Pfizer), is de database waarin software en endpoints als objecten worden beheerd. Voorafgaand aan de ontwerpfase is een empirische nulmeting uitgevoerd om te onderzoeken waarom de huidige softwaredetectietool onvoldoende is. Hierbij werd vastgesteld dat in-house applicaties die via scripts of handmatige bestandskopieën worden geïnstalleerd, stevast ongedetecteerd blijven door EPC. EPC leunt immers op de aanwezigheid van Windows Installer-transacties (MSI) en aanwezigheid in Programs & Features. Dit bevestigde de noodzaak voor een alternatieve detectiemethode die rechtstreeks controleert of de verwachte bestanden, versies en omgevingsconfiguraties op het endpoint aanwezig zijn.

De CMDB in Jira Assets, de EPC-servers, de beschikbare IID's en de CD-shares werden onderzocht als databronnen. Endpoint Central functioneert via een lokale agent op de endpoints, die standaard inventarisatiedata verzamelt via Windows Installer-registers. Via de Custom Scripts-module kunnen echter ook gerichte scripts met administrator rechten worden uitgevoerd om specifieke bestands- en padcontroles te doen.

2.2 Functionele en niet-functionele vereisten

Uit de analyse volgden zowel functionele als niet-functionele vereisten. De belangrijkste functionele vereisten waren:

- detectie van actieve, vooraf gemodelleerde in-house software;

- automatische generatie van detectieprofielen uit een centraal bronbestand;
- verzameling en normalisatie van endpointresultaten;
- vergelijking van de endpointrealiteit met de administratieve registratie in WSIT/CMDB;
- generatie van een valideerbaar discrepantierapport en een importeerbaar bestand voor semiautomatische opvolging.

De belangrijkste niet-functionele vereisten waren:

- betrouwbaarheid: de detectie moet reproduceerbare resultaten opleveren zonder evidente foutpositieven;
- onderhoudbaarheid: softwaredefinities moeten aangepast kunnen worden zonder de detectie-engine te wijzigen;
- overdraagbaarheid: collega's moeten de workflow kunnen uitvoeren op basis van de launcher en documentatie;
- performantie: de detectie moet uitvoerbaar blijven op grote aantallen endpoints;
- debuggability: logs, tussenresultaten en outputmappen moeten foutanalyse mogelijk maken;
- minimale impact op endpoints: de detectie moet read-only zijn en de systeembelasting beperken.

2.3 Technologiekeuzes

- **YAML als bronformaat:** Het centrale bronbestand gebruikt YAML. Dit is voor mensen leesbaar en bewerkbaar, wat essentieel is voor toekomstig onderhoud. JSON dient als runtimeformaat voor machinale verwerking.
- **Python voor detectie en dataverwerking:** Voor de Detection Engine zelf is weloverwogen gekozen voor Python boven PowerShell of C#. Python biedt sterke

ondersteuning voor objectgeoriënteerde en modulaire code bij complexe geneste logica. Bovendien beschikt Python over embeddable packages (waaronder ingebouwde Universal C Runtime, of UCRT, DLL-bestanden) waarmee de runtime volledig geïsoleerd en zonder dependencies of OS-wijzigingen op legacy systemen uitgevoerd kan worden. Ook de verwerking van omvangrijke JSON-datastructuren verloopt aanzienlijk sneller.

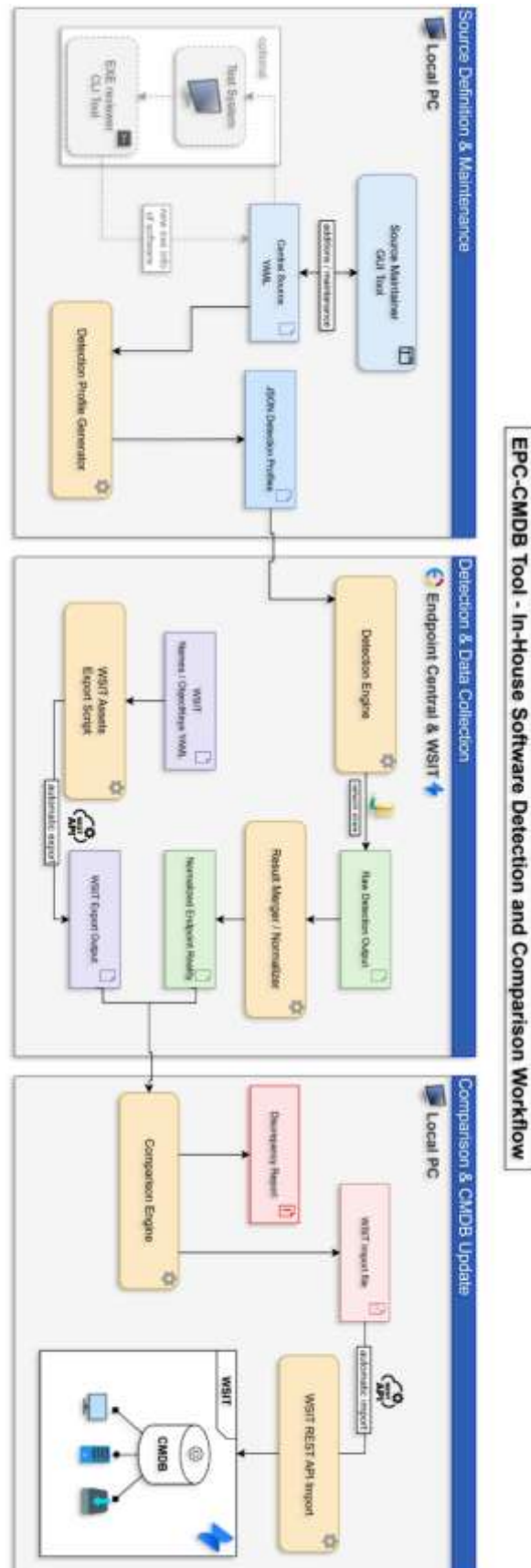
- **PowerShell 7 voor WSIT-export en Launcher:** Voor REST API-interacties en parallelle verwerking is gekozen voor PowerShell 7, inclusief het gebruik van de SecretStore module voor veilig beheer van authenticatietokens.
- **C#/Windows Presentation Foundation (WPF) voor bronbestandbeheer:** De grafische applicatie voor het beheer van YAML is gebouwd in C#/WPF, wat naadloos aansluit bij de Windows-werkomgeving zonder extra runtime-eisen voor eindgebruikers, met uitzondering van .NET 10.

3. Realisatie van de EPC-CMDB Tool

3.1 Architectuur en ontwerpprincipes

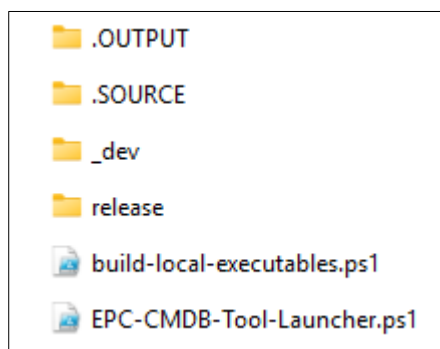
De gerealiseerde oplossing is een keten van tools die samen één workflow vormen. De workflow bestaat uit drie inhoudelijke fasen. In fase 1 wordt softwarekennis beheerd en omgezet naar detectieprofielen. In fase 2 worden die profielen gebruikt voor endpointdetectie en worden de resultaten genormaliseerd. In fase 3 worden de genormaliseerde endpointresultaten vergeleken met WSIT/CMDB en omgezet naar een HTML-discrepantierapport en een importbestand. Daarnaast bevat de oplossing een overkoepelende gebruikerslaag met een centrale launcher waaronder de tools, prerequisite check, buildvoorbereiding, ingebouwde bron- en outputroutering en gebruiksaanwijzingen op Confluence.

Figuur 1 Architectuurdiagram van de EPC-CMDB Tool, opgedeeld in drie fasen: bronbestandbeheer, detectie en dataverzameling, en vergelijking en CMDB-update



De architectuur is gebouwd rond drie centrale principes. Het eerste principe is de scheiding tussen bronkennis en runtime, wat aansluit bij het software engineering-principe *separation of concerns*. In deze oplossing betekent dit dat inhoudelijke softwarekennis gescheiden blijft van de runtimecomponenten die de detectie uitvoeren. Het YAML-bronbestand bevat de volledige kennis over hoe software gemodelleerd is: installatiepaden, versies, detecteerbare executables, services en omgevingskenmerken. De JSON-detectieprofielen die daaruit worden gegenereerd, bevatten enkel wat het detectiescript nodig heeft. Hierdoor is het bronbestand de enige plek waar inhoudelijke wijzigingen worden aangebracht, en hoeven de scripts zelf niet aangepast te worden bij elke softwaretoevoeging. Dit sluit rechtstreeks aan bij de definitie van onderhoudbaarheid in het projectplan (De Meyer, 2026a, sectie 7.3).

Figuur 2 *Mappenstructuur van de gedeelde werklocatie met bron-, output-, release- en ontwikkelmappen*



Het tweede principe is automatische input- en outputrouting. Elke tool schrijft output automatisch weg naar een eigen submap onder .OUTPUT, georganiseerd per datum en voorzien van een tijdstempel in de mapnaam. Binnen deze submappen worden de relevante outputbestanden opgeslagen, waaronder detectieprofielen, detectielogs en bijbehorende samenvattingen, genormaliseerde inventarissen, CMDB-exports en discrepantierapporten met bijhorende importbestanden. De volgende stappen in de keten selecteren automatisch de meest recente output als input. De gebruiker hoeft daardoor geen paden op te geven, geen bestanden te verplaatsen en

geen tussenresultaten bij te houden. De volledige interne structuur van de outputmap is opgenomen in bijlage B.

Het derde principe is de scheiding tussen ontwikkelomgeving en gebruikersomgeving. De map `_dev` bevat de broncode van alle tools; de map `release` bevat uitsluitend gecompileerde uitvoerbare bestanden, scripts en de Detection Engine. Gebruikers werken enkel met de `release`-map en de launcher; de ontwikkelmap is enkel relevant voor toekomstige aanpassingen aan de broncode.

Figuur 2 toont de mappenstructuur van de gedeelde werklocatie. De hoofdmap bevat een bronmap (`.SOURCE`), een outputmap (`.OUTPUT`), een `release`-map met gebruikersklare bestanden, een ontwikkelmap (`_dev`) met broncode, en de centrale launcher en het buildscript.

3.2 Fase 1: Bronbestandbeheer

De eerste fase van de workflow omvat het beheer van de softwarekennis en de omzetting daarvan naar detectieprofielen. Deze fase bestaat uit het YAML-bronbestand, de Software Source Maintainer, de EXE Reviewer en de Detection Profile Generator.

YAML-bronbestand

Figuur 3 Snippet van YAML-Bronbestand toont in-house software batch report tool dat 2 versies bevat

```
- key: batch-report-tool
  display_name: batch report tool
  install_paths:
    32bit: C:\Program Files\Pfizer\BRT\
    64bit: C:\Program Files (x86)\Pfizer\BRT\
  versions:
    4.3.0:
      software_files:
        - type: install
          path: \\bepuugho\CD2\CD6523\Release
          cd: CD6523
      detection:
        executables:
          - path: ReportGenerator.exe
            role: primary
            file_version: 4.3.0.0
            product_version: 4.3.0.0
          - path: BatchReport.exe
            role: component
            file_version: 4.3.0.0
            product_version: 4.3.0.0
    4.2.0.0:
      software_files:
        - type: install
          path: \\bepuugho\CD\CD5440\Build
          cd: CD5440
      detection:
        executables:
          - path: ReportGenerator.exe
            role: primary
            file_version: 4.2.0.0
            product_version: 4.2.0.0
          - path: BatchReport.exe
            role: component
            file_version: 4.2.0.0
            product version: 4.2.0.0
```

De kern van de oplossing is het YAML-bronbestand. Per softwareproduct worden daarin onder meer de volgende kenmerken vastgelegd:

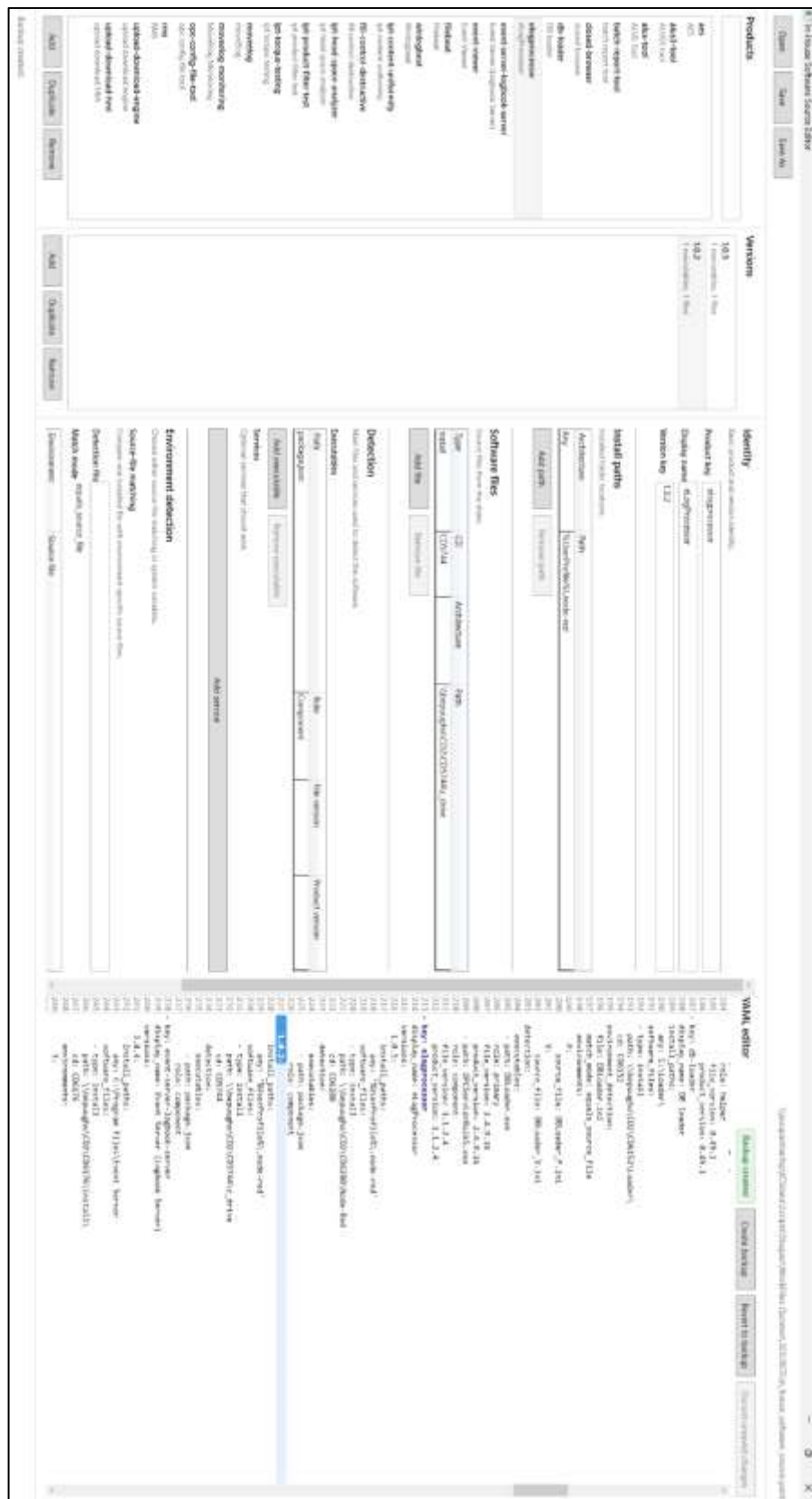
- installatiepaden;
- bronlocaties op CD-shares;
- softwareversies;
- detecteerbare executables met product- en fileversies;
- services op Windows;
- omgevingsspecifieke configuratiebestanden;
- systeemvariabelen;
- negeerpatronen

Dit bestand werd tijdens de stage opgebouwd op basis van IID's, manuele installaties op testsystemen en controles tegen bronmedia. Het projectplan beschrijft dit bestand als de eerste bedrijfsdeliverable (De Meyer, 2026a, sectie 9.1).

Bij het ontwerp van het bronbestand werd bewust gekozen voor duidelijke, herhaalbare structuren per product.

Het beheer en de verrijking van het bronbestand worden ondersteund door twee tools: de Software Source Maintainer en de EXE Reviewer.

Figuur 4 *Software Source Maintainer* waarin de software *eLogProcessor v1.0.2* is geselecteerd.



De *Software Source Maintainer* is een WPF-gebaseerde desktopapplicatie waarmee softwareproducten via een grafische interface kunnen worden toegevoegd, gewijzigd en verwijderd. De applicatie biedt configuratie van alle detectieparameters, waaronder installatiepaden, bronpaden, detectie op basis van *executables*, servicedetectie, omgevingsdetectie en negeerpatronen. Daarnaast bevat de applicatie een live YAML-preview met syntaxkleuring, automatische back-ups bij het openen van een bestand, en een terugzetfunctie waarmee eerdere versies van het bronbestand kunnen worden hersteld. Bij kritieke acties zoals opslaan, herladen of verwijderen wordt om bevestiging gevraagd. Hierdoor kunnen collega's het bronbestand onderhouden zonder kennis van de YAML-syntax en met een beperkt risico op structuurfouten.

EXE Reviewer

De *EXE Reviewer* is een terminaltoepassing die door middel van het YAML-bronbestand de installatiepaden van software scant en per product alle uitvoerbare bestanden presenteert. De gebruiker labelt elk bestand naar functie, bijvoorbeeld als hoofdprogramma, component, helper of ruis. De resultaten worden automatisch opgeslagen in een CSV-bestand. Die informatie wordt vervolgens in het bronbestand verwerkt om de detectie gericht te maken. Tijdens de stage hielp de EXE Reviewer vooral met efficiënter ontwikkelen van het YAML-bronbestand. De gebruikers kunnen dit ook gebruiken maar het is niet noodzakelijk en kan het onderhouden van het YAML-bronbestand minder efficiënt maken, aangezien deze tool ontworpen is met de visie om in bulk software informatie op te halen.

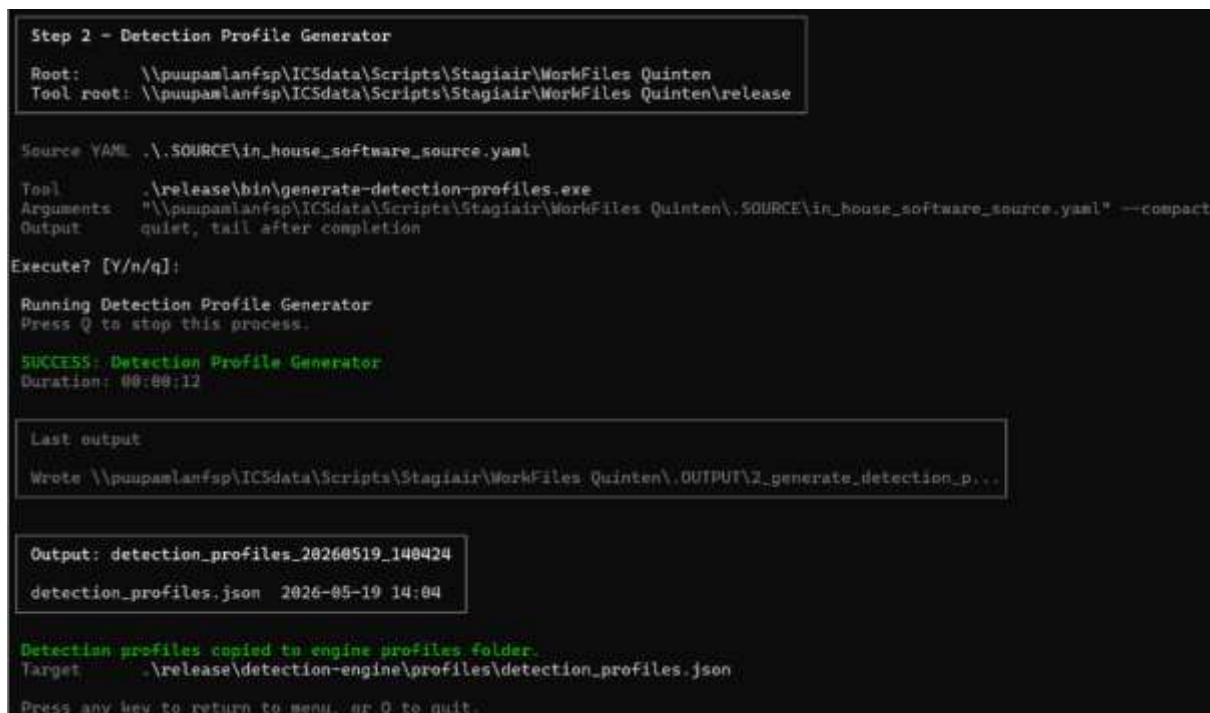
Detection Profile Generator

De Detection Profile Generator is een Python-script dat het centrale YAML-bronbestand omzet naar JSON-detectieprofielen (*detection_profiles.json*). Tijdens deze conversie wordt de geneste YAML-data "platgeslagen" tot afzonderlijke JSON-objecten per softwareversie. Dit verlaagt de complexiteit voor de detectie-engine op de endpoints aanzienlijk, omdat het script daar eenvoudiger over de data kan itereren.

De gegenereerde profielen bevatten uitsluitend de noodzakelijke detectie-indicatoren, zoals paden, executables, services en omgevingskenmerken. Zoals in het YAML-Bronbestand.

De tool wordt rechtstreeks vanuit het launcher-menu uitgevoerd. Na een succesvolle generatie schrijft het script het nieuwe JSON-bestand weg naar de .OUTPUT-map. Tegelijkertijd wordt het profiel automatisch gekopieerd naar de release-omgeving van de detectie-engine stap.

Figuur 5 *Detection Profile Generator menu*



```
Step 2 - Detection Profile Generator
Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool root: \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Source YAML  .\SOURCE\in_house_software_source.yaml
Tool         .\release\bin\generate-detection-profiles.exe
Arguments    "\\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\SOURCE\in_house_software_source.yaml" --compact
Output       quiet, tail after completion

Execute? [Y/n/q]:

Running Detection Profile Generator
Press Q to stop this process.

SUCCESS: Detection Profile Generator
Duration: 00:00:12

Last output
Wrote \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\OUTPUT\2_generate_detection_p...

Output: detection_profiles_20260519_140424
detection_profiles.json  2026-05-19 14:04

Detection profiles copied to engine profiles folder.
Target      .\release\detection-engine\profiles\detection_profiles.json
Press any key to return to menu, or Q to quit.
```

3.3 Fase 2: Detectie en dataverzameling

Detection Engine

De *Detection Engine* is het Python-script dat via Endpoint Central op endpoints wordt uitgevoerd. Het leest de JSON-detectieprofielen en controleert per product of de verwachte kenmerken aanwezig zijn op het endpoint. De detectie combineert meerdere methoden: padcontrole, vergelijking van uitvoerbare bestanden en hun versie-informatie, controle op aanwezigheid van services, herkenning van omgevingsspecifieke configuratiebestanden en

systeemvariabelen. Voor software zonder uitvoerbare bestanden wordt een terugval gebruikt die alle aanwezige bestanden vergelijkt met de bronmedia op de CD-share.

De detectie is met opzet strikt en deterministisch. Er wordt geen gebruikgemaakt van heuristische zoekmethoden (zoals het willekeurig scannen van volledige schijven); de analyse is beperkt tot de expliciet gedefinieerde configuratie uit de Single Source of Truth (SSOT), zijnde het YAML-bronbestand. Die keuze sluit aan bij de omgeving waarin de tool wordt ingezet: software-installaties verlopen via een strikt ticketproces en worden uitgevoerd volgens IID. Het opvangen van willekeurige of foutieve installaties valt buiten de doelstelling van de tool, omdat dergelijke afwijkingen in deze omgeving via het procedureproces worden afgehandeld.

Naast de read-only eis vormde ook de performantie – zoals vastgelegd in de niet-functionele vereisten – een kritiek aandachtspunt voor een brede uitrol. Bij de initiële tests bleek de runtime van de ElogProcessor (een in-house logverwerkingsapplicatie) disproportioneel lang door het ontbreken van een specifieke detectie-indicator, wat leidde tot volledige bestandsvergelijkingen. Door een representatief bestand als indicator in het YAML-bronbestand op te nemen, daalde de gemiddelde runtime van ongeveer 3 minuten naar 18 seconden voor de volledige scope. Op endpoints met een beperkte softwarescope bedraagt de runtime nu slechts 4 tot 6 seconden.

Naast performantie was een ander belangrijk niet-functioneel vereiste dat de detectie-engine volledig read-only moest zijn: er wordt niets gewijzigd op het endpoint. De test- en productieomgevingen hebben geen internettoegang, waardoor de engine ook geen externe afhankelijkheden mag hebben. De Python-runtime wordt bij uitvoering tijdelijk uitgepakt in een tijdelijke map en na afloop opgeruimd. Bestanden, waaronder logbestanden, worden atomair weggeschreven. Hierdoor blijft de data-integriteit te allen tijde gewaarborgd, zelfs in het geval van onverwachte systeemuitval of gelijktijdige schrijfacties. De output wordt weggeschreven naar de gedeelde netwerklocatie. In het projectplan werd dit script beschreven als een centrale deliverable, inclusief gestandaardiseerde output en logging (De Meyer, 2026a, sectie 9.1).

De Detection Engine functioneert als een zelfstandige eenheid. Installatie op het endpoint is overbodig en niet evident. Om de leesbaarheid te verbeteren, is de logische mappenstructuur hieronder weergegeven:

- **\engine:** Bevat het eigenlijke Python-script `detection_engine.py`.
- **\launchers:** Bevat een `.cmd`-script dat de juiste Python-runtime selecteert voor de architectuur van het endpoint.
- **\profiles:** Bevat het gegenereerde configuratiebestand `detection_profiles.json`.
- **\runtimes:** Bevat de 32-bit en 64-bit Python embeddable packages, onderverdeeld per besturingssysteem:
 - Python 3.7.9 (Windows Server 2008 SP2)
 - Python 3.8.10 (Windows 7 SP1)
 - Python 3.11.9 (Windows Server 2012 R2)
 - Python 3.14.3 (Windows NT 10 en nieuwer, inclusief Windows Server 2016+)
- **\tools:** Bevat `7za.exe` (een command-line 7-Zip tool om de gearchiveerde Python-runtime op het endpoint uit te pakken) en het script `build_upload_bundles.ps1`.

Om de compatibiliteit op oudere besturingssystemen te garanderen, is de Python runtime packaging specifiek aangepast. Omdat bepaalde endpoints faalden door ontbrekende Universal C Runtime (UCRT) DLL-bestanden, worden deze nu voor relevante versies (zoals Python 3.8.10) direct naast `python.exe` meegeleverd. Hierdoor functioneert de engine correct op onder meer Windows Server 2012 R2 en Windows 7 Professional x64.

Endpoint Central Upload Bundle Builder

Figuur 6 Detection Engine Upload Bundle Builder menu, toont na succes om de stappen in de gebruikersaanwijzingen te volgen

```
Step 3 - Detection Engine Upload Bundle Builder

Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool_root: \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Tool      C:\Program Files\WindowsApps\Microsoft.PowerShell_7.6.1.0_x64__8wekyb3d8bbwe\pwsh.exe
Arguments -NoProfile -ExecutionPolicy Bypass -File "\\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release\UploadBundleBuilder.ps1"
Output     quiet, tail after completion

Execute? [Y/n/q]:

Running Upload Bundle Builder
Press Q to stop this process.

SUCCESS: Upload Bundle Builder
Duration: 00:00:12

Last output

Refreshed upload bundles:
cfg_detect_engine_server2008sp2
cfg_detect_engine_server2012r2
cfg_detect_engine_win10_server2016plus
cfg_detect_engine_win7sp1

Next step
Follow the guide on Confluence before going to step 4.

Press any key to return to menu, or Q to quit.
```

Dit script genereert kant-en-klare uploadpakketten voor gebruik in Endpoint Central.

Per ondersteund besturingssysteem wordt een bundel aangemaakt in de map \upload-bundles, met daarin een submap \script (het launcherscript) en een submap \dependency (de detectie-engine, de detectieprofielen, 7za.exe en de twee gearchiveerde Python-runtimes voor dat besturingssysteem).

Endpoint Central-uitvoering en Dynamic Custom Groups

De gebruiker kan bij het aanmaken van een *Custom Script*-configuratie in Endpoint Central zoals hieronder aangetoond in figuur 7 de bestanden uit \dependency rechtstreeks toevoegen als afhankelijkheidsbestanden, terwijl het launcherscript al beschikbaar is in de *scriptrepository* van Endpoint Central.

de launcher werd uitgevoerd met een /belownormal procespriority, waardoor de impact op de systeembronnen van actieve endpoints minimaal bleef, zelfs bij systemen onder hoge belasting

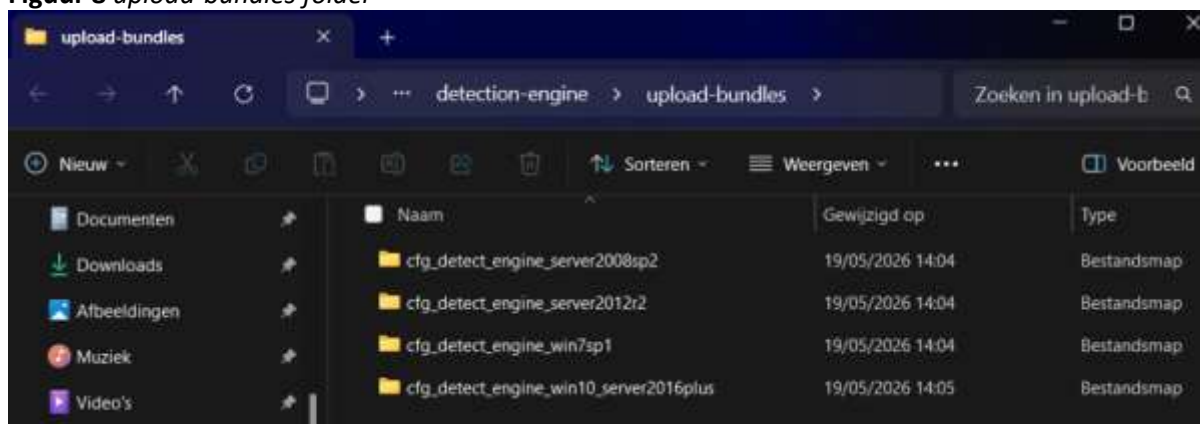
(zoals WinCC-rendering). De finale productiescan is succesvol uitgevoerd op duizenden Windows 10-machines.

Figuur 7 Configuratiescherm van een Custom Script in Endpoint Central, met het launcherscript en de

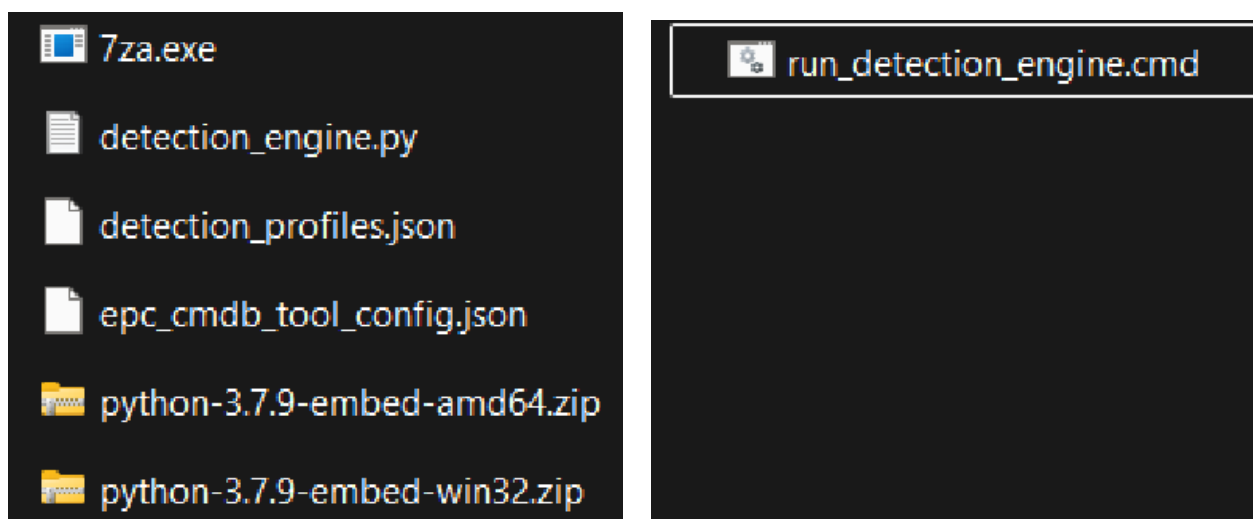
The screenshot displays the 'Configure Custom Script' interface in Endpoint Central. The 'Name and Description' section at the top has a 'Name' field containing 'in-house software detection' and an 'Add Description' link. Below this, the 'Configure Custom Script' section includes radio buttons for 'Execute Script from / Run', with 'Repository' selected over 'Command Line'. The 'Script Name' field is 'run_detection_engine.cmd', with a 'Create/Modify Script' link. The 'Script Arguments(s)' field is empty. The 'Dependency Files(s)' section shows a list of five uploaded files: 'python-3.14.3-embed-win32.zip', '7za.exe', 'detection_engine.py', and 'detection_profiles.json', with a confirmation that all five files are uploaded. The 'Specify the exit code(s)' field contains '0', with a note that multiple codes should be separated by commas. The 'Frequency' section has 'Once' selected, and the 'Enable Logging for Troubleshooting' checkbox is checked. A 'Run As' button is located below the frequency settings. The 'Define Target' section at the bottom shows 'Target 1' with a dropdown set to 'Remote Office' and a 'Local Office' button labeled 'Select'. Below this, the 'Filter Computers based on' dropdown is set to 'Computer', with a specific filter 'I-EPCWST02' selected and a 'Select' button.

afhankelijkheidsbestanden van een uploadpakket

Figuur 8 *upload-bundles folder*



Figuur 9 *Voorbeeld van een upload bundel dependency folder rechts, script folder links, in dit geval Windows Server 2008 Service Pack 2*



Om de juiste bundel op de juiste endpoints uit te voeren, werden in Endpoint Central vier *Dynamic Custom Groups* aangemaakt, elk corresponderend met een ondersteund besturingssysteem. Een *Dynamic Custom Group* is een endpointgroep waarvan het lidmaatschap automatisch wordt bepaald op basis van filtercriteria; endpoints die aan de criteria voldoen, worden automatisch toegevoegd, en endpoints die er niet langer aan voldoen, worden automatisch verwijderd (ManageEngine, z.d.-a).

De keuze van het filtercriterium werd bepaald door middel van een voorafgaande validatie. Op elk van de drie Endpoint Central-servers werd een testscript uitgevoerd dat enkel een bevestigingsbericht terugschreef, om te verifiëren welke endpoints scriptuitvoering ondersteunden.

Uit die validatie bleek dat filteren op Windows-versie, meer specifiek het geïnstalleerde service pack (een grote, identificeerbare updatebundel van Windows), het meest betrouwbare en fijnmazige criterium was om endpoints per besturingssysteemversie te groeperen. Omdat *Dynamic Custom Groups* hun lidmaatschap automatisch bijwerken wanneer nieuwe endpoints aan de omgeving worden toegevoegd of bestaande endpoints van besturingssysteem veranderen, is deze aanpak minder afhankelijk van manuele opvolging. De groepsindeling blijft daardoor beter onderhoudbaar dan een statische indeling.

Omdat de beoogde endpoints variëren van Windows Server 2008 SP2 tot en met Windows 10 en nieuwere Windows-versies op basis van de NT 10-kernel (inclusief Windows Server 2016 en hoger), werden meerdere Python *embeddable packages* meegeleverd, afgestemd op de officiële ondersteuningsmatrix per Windows-versie. Tijdens het testen bleek dat de detectie-engine correct functioneerde op 64-bit endpoints met Windows 7, Windows Server 2008, Windows Server 2012 en Windows 10. Na testen bij 32-bit endpoints deed zich echter een fundamentele beperking voor: deze systemen konden geen scripts uitvoeren met de vereiste administratieve gebruiker in Endpoint Central. Dat probleem werd niet veroorzaakt door de detectie-engine zelf, maar door de manier waarop Endpoint Central scripts uitrolt naar 32-bit systemen.

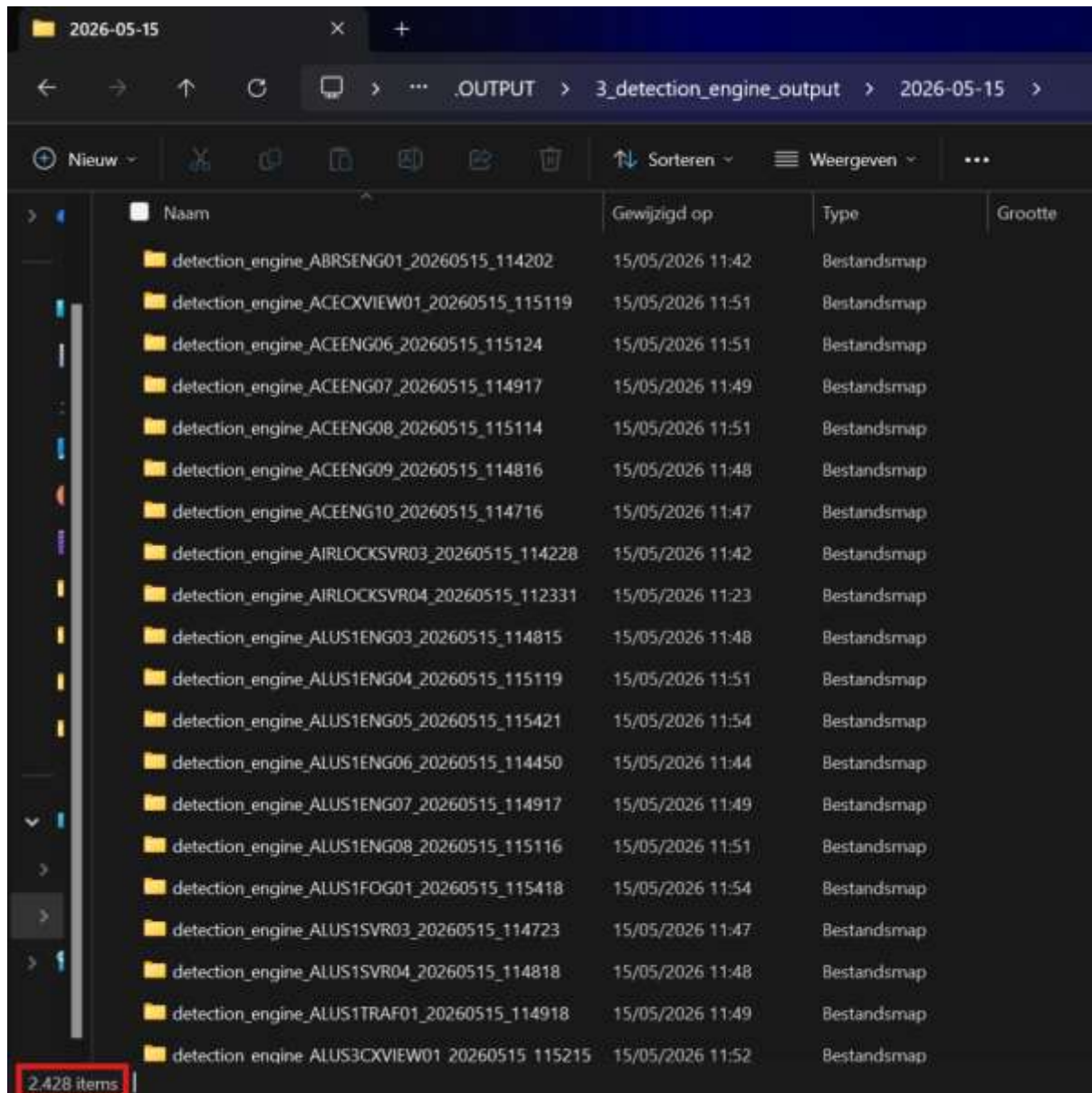
Naast OS-beperkingen is er een infrastructurele beperking vastgesteld op Endpoint Central server 02 (domain: EMEA). Hier bleek custom script execution niet mogelijk onder het system-account, persoonlijke EMEA-accounts of admin-accounts. Aangezien dit buiten het beheer van het team valt en de betreffende EBR-machines (Electronic Batch Record) een zeer statische softwarebezetting kennen, is de impact op de totale scanvolledigheid beperkt gebleven.

Daarnaast moest bij het uitvoeren via Endpoint Central rekening worden gehouden met het feit dat het detectiescript draait onder een administratief serviceaccount en niet onder een gebruikersaccount. Dit was noodzakelijk, aangezien het script verhoogde rechten vereiste om toegang te verkrijgen tot de CD-shares. Deze uitvoeringscontext had gevolgen voor de detectie van software die in gebruikersprofielen wordt geïnstalleerd: de oorspronkelijke detectie via

%UserProfile% werkte niet meer. Dit werd opgelost door de gebruikersprofielen programmatisch op te halen via het Windows-register en elk profiel individueel te controleren.

Result Merger / Normalizer

Figuur 10 *detection engine output, resultaten van alle endpoints*



De Result Merger / Normalizer consolideert alle ruwe detectieresultaten uit de .OUTPUT-share. De individuele endpoints plaatsen hun data daar in afzonderlijke mappen. De component selecteert per uniek endpoint uitsluitend het meest recente bestand.

Deze resultaten worden samengevoegd in één centraal JSON-bestand. Dit bestand dient als de gestructureerde input voor de Comparison Engine. Daarnaast genereert de tool een CSV-bestand voor debugging en controle.

WSIT Assets Export Script

Naast de genormaliseerde endpointrealiteit is ook de administratieve registratie uit de CMDB nodig als tweede invoerbron. Het WSIT Assets Export Script haalt deze software-endpointrelaties op via de REST API van Jira Assets. Omdat WSIT classificeert onder verschillende objecttypes met afwijkende datastructuren, is één globale bevraging onmogelijk. Het script gebruikt daarom een lus van meerdere specifieke AQL-queries (Assets Query Language; zie bijlage A). Dit waarborgt een volledige export waarin in-house en buiten-scope software correct worden onderscheiden, in plaats van dat alle applicaties onder een te algemene noemer zoals 'software' worden gecategoriseerd.

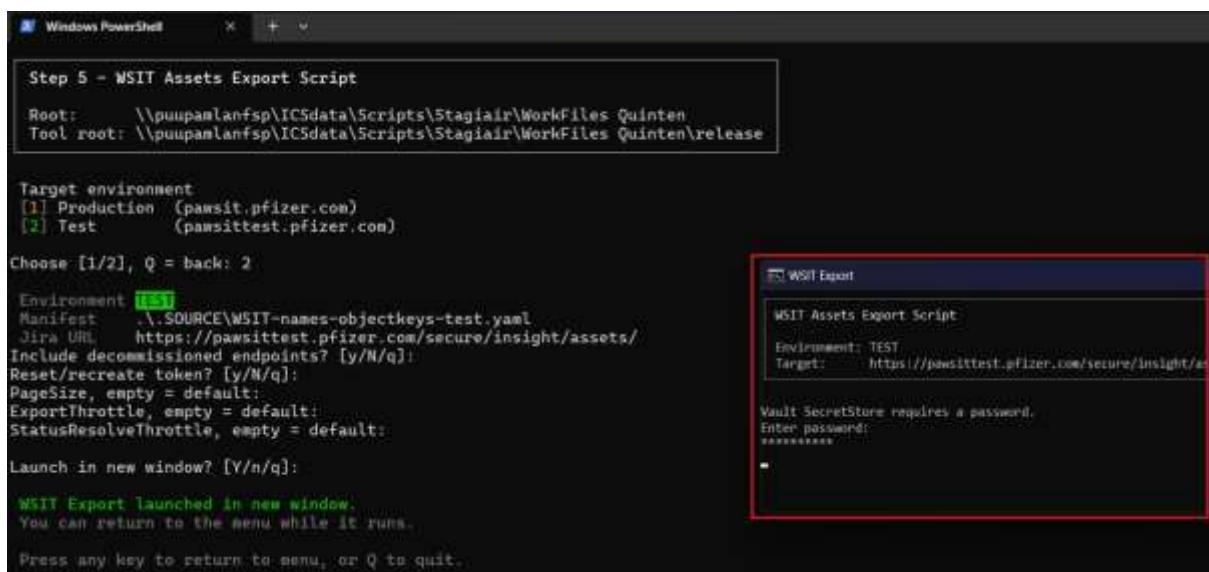
Het script exporteert bewust alle aanwezige software per endpoint, niet uitsluitend de in-house software binnen scope. Deze logica vloeit voort uit de initiële ontwerpfase. Oorspronkelijk was een import via CSV voorzien, waarbij het systeem bestaande koppelingen automatisch zou verwijderen als deze ontbraken in de payload. Om onbedoeld dataverlies te voorkomen, was een volledige export toen een absolute vereiste. In de finale realisatie is deze methode vervangen door een meer geavanceerde en gerichte REST API-import, die gerichte updates uitvoert en dit overschrijvingsrisico elimineert. Toch is de brede exportlogica bewust behouden. De buiten-scope software reist mee in de dataketen en wordt tijdens de importfase simpelweg genegeerd. Dit waarborgt een compleet en feitelijk endpointoverzicht en sluit elk theoretisch risico op beschadiging van de CMDB uit.

Om de API-belasting en uitvoeringstijd te optimaliseren, ondersteunt het script instelbare paginering (*PageSize*) en parallelle verwerking (*Throttling*). Daarnaast kan de gebruiker via de launcher bepalen of gedecommissioneerde endpoints in de export moeten worden opgenomen.

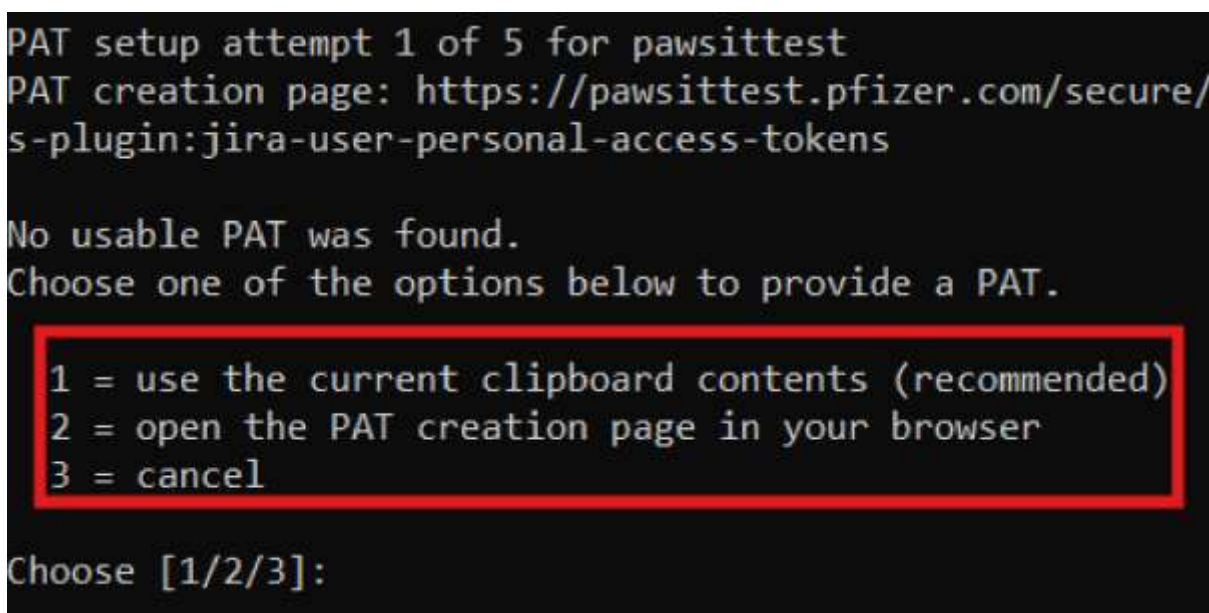
Voor de authenticatie wordt per omgeving een Personal Access Token (PAT) lokaal en versleuteld opgeslagen via de PowerShell SecretStore-module. De launcher bevat een ingebouwde,

interactieve prompt. Wanneer een token ontbreekt of is vervallen, helpt de interface de beheerder om via het klembord snel een nieuwe PAT in te stellen. Tot slot genereert het script tijdens de uitvoering een configuratie-artefact (tekstbestand) in de outputmap. Dit bestand registreert de actieve doelomgeving (TEST of PROD), waardoor de daaropvolgende vergelijkings- en importstappen automatisch de juiste omgevingscontext selecteren zonder extra gebruikersinput.

Figuur 11 *WSIT Assets Export Script menu en open in nieuw venster voor achtergrondexport*



Figuur 12 *PAT setup*

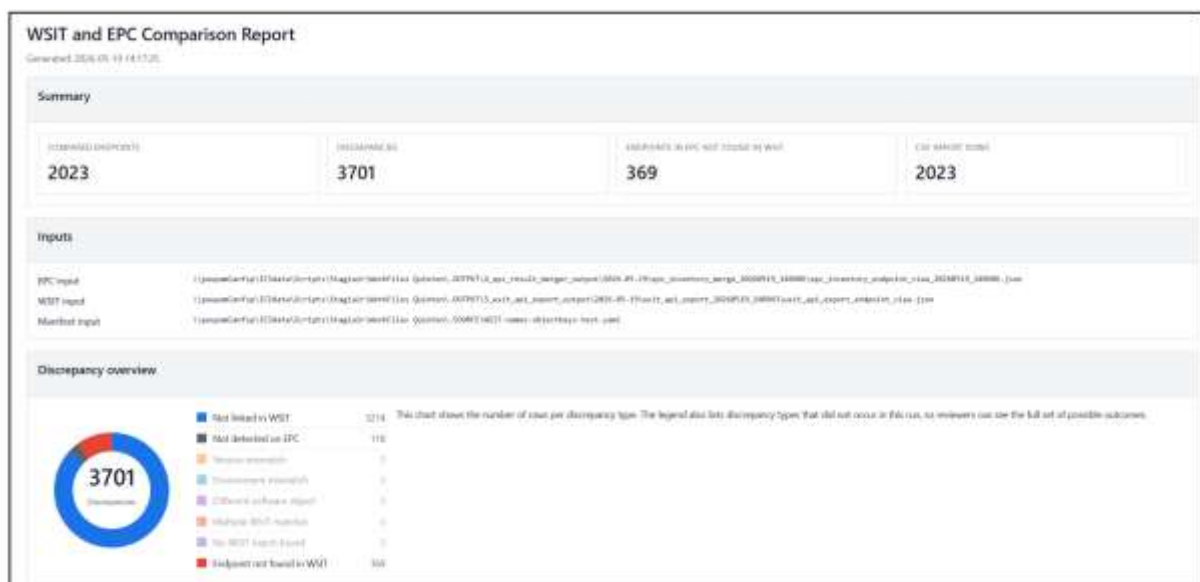


3.4 Fase 3: Vergelijking en CMDB-update

Comparison Engine

De *Comparison Engine* vergelijkt de genormaliseerde endpointrealiteit met de CMDB-export en identificeert discrepanties: ontbrekende koppelingen, overbodige koppelingen, versieverschillen en omgevingsafwijkingen. De output bestaat uit een discrepantierapport in HTML-formaat en een importeerbaar CSV-bestand voor de CMDB. Het discrepantierapport is bedoeld voor menselijke validatie; Figuur 13 toont een voorbeeld. Het importbestand is het artefact waarmee de CMDB semiautomatisch kan worden bijgewerkt.

Figuur 13 Voorbeeld van het HTML-discrepantierapport, met per endpoint de geïdentificeerde afwijkingen ten opzichte van de CMDB



Figuur 14 Detailweergave van de discrepantie-impacttabel, met per endpoint een vergelijking tussen de huidige CMDB-status en de gedetecteerde realiteit, inclusief de voorgestelde importactie.

Discrepancy impact table						
Search endpoint, endpoint key, discrepancy, WSIT state, EPC, reality, target or act						
Discrepancy per page: 25						
First Previous 2221 rows, 1778 endpoints, page 121 of 185 Next Last						
Endpoint	Discrepancy	Current WSIT state	Endpoint reality from EPC	Target WSIT state after import	Import action	Reason
STMAU005A 4128-413001	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005A 4128-413002	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413003	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413004	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005A 4128-413005	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005A 4128-413006	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413007	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413008	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005A 4128-413009	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005A 4128-413010	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413011	Not linked in WSIT	Not linked in WSIT	Event Viewer Name: 6.7.2.1 Product key: none value	Event Viewer 4.7.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.
STMAU005B 4128-413012	Not linked in WSIT	Not linked in WSIT	Upload download HMI Name: 7.3.1 Product key: upload-download key	Upload Download HMI T.2 (F) 4128-413001	412_114 4128-413001	Software detected in EPC is not linked on the WSIT endpoint.

Figuur 15 Voorvertoning van de CMDB-import (preservatietabel), die inzichtelijk maakt hoe in-house software wordt bijgewerkt terwijl registraties buiten de scope veilig behouden blijven.

WSIT import preview and preservation table							
This table shows the before/after result of the proposed CSV import in-house software but not the actual import. This CSV import software is shown as reference to see why the existing WSIT data not proposed.							
Search endpoint, software, endpoint key, scope, before/after state or import effect							
Discrepancy per page: 25							
First Previous 2221 rows, 1778 endpoints, page 8 of 179 Next Last							
Endpoint	Software	Type	Scope	Before import	EPC comparison result	After import	Import effect
BAGC1703 4128-413001	Google Chrome v 66.0.3448.04 - 64bit 4128-413001	System Component	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BAGC1703 4128-413002	CrowdStrike 6.18.112.0 4128-413002	System Component	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BAGC1703 4128-413003	Johnson Controls SCT M.2 4128-413003	Application	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BEPU00001 0 4128-413004	Mosquitto Application Monitoring v0.8 4128-413004	Application	In scope / in-house software	Not linked in WSIT	Detected by EPC and expected in WSIT	Linked after import	Added by import
BEPU00001 0 4128-413005	SIEM - SCADA FDI-11-12 - Filling Freeze Dry - sensor B (F) 4128-413005	Application	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BEPU00001 4128-413006	Mosquitto V0.8.2.6 (F) 4128-413006	Application	In scope / in-house software	Linked in WSIT	Not detected by EPC for this endpoint, so replaced by expected	Not linked after import	Removed by import
BEPU00001 4128-413007	Oracle Client 12.2 (x64) 4128-413007	System Component	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BEPU00001 4128-413008	Allen Client (F) 4128-413008	Application	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BEPU00001 4128-413009	Mosquitto Application Monitoring v0.8 4128-413009	Application	In scope / in-house software	Linked in WSIT	Detected by EPC and already linked correctly	Linked after import	Keep
BEPU00001 4128-413010	System software IMATIC NET V8 4128-413010	Application	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved
BEPU00001 4128-413011	Siemens WinCC V7.5 SP2 Update 13 - RUG 4128-413011	Application	Out of scope / preserved	Linked in WSIT	Outside comparison scope	Linked after import	Preserved

WSIT API Import

De laatste fase van de workflow betreft het daadwerkelijk doorvoeren van de berekende doelstaat in de CMDB. Dit gebeurt via het WSIT REST API Import-script, dat het gegenereerde CSV-importbestand als invoer gebruikt. Endpoints worden hierbij aan de correcte software gekoppeld op basis van hun unieke *object keys*. Het proces ondersteunt proactief zowel het aanmaken van nieuwe

softwarerelaties als het verwijderen van koppelingen die volgens de vergelijkingsfase niet langer actueel zijn.

Vanuit een *secure by design*-principe wordt de import standaard uitgevoerd in een veilige simulatiemodus (*dry-run*). In deze modus valideert het script de wijzigingen tegen de API zonder deze effectief naar de database weg te schrijven. Pas na expliciete goedkeuring via de launcher worden de aanpassingen daadwerkelijk doorgevoerd in WSIT. Om menselijke fouten te minimaliseren, leest de launcher automatisch het configuratie-artefact uit de eerdere exportstap (stap 5, WSIT Assets Export Script) uit, waardoor direct de juiste doelomgeving (Productie of Test) wordt geselecteerd.

Voor gerichte probleemoplossing of gefaseerde uitrol biedt de tool granulaire controle over de uitvoering. Beheerders kunnen de import filteren op één specifieke endpoint *object key* of het aantal te verwerken CSV-rijen limiteren. De interactieve terminalweergave genereert bovendien klikbare *object keys* die direct doorlinken naar de desbetreffende asset-pagina in de Jira webinterface. Om volledige traceerbaarheid te garanderen, produceert de importstap uitgebreide outputbestanden, waaronder een plannings-CSV, een resultaat-CSV, een JSON-samenvatting en een gedetailleerd logbestand.

Figuur 16 WSIT REST API Import menu

```

Windows PowerShell

Step I - WSIT REST API Import

Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool root: \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Last Step 5 run used: TEST

Target environment
[1] Production (pawsit.pfizer.com)
[2] Test       (pawsittest.pfizer.com)

Choose [1/2], default=2, Q = back:

Environment TEST
Jira API    https://pawsittest.pfizer.com

WSIT login reminder

Before continuing, make sure you are logged in to:
- WSIT
- WSIT TEST

```

Figuur 17 WSIT REST API Import stap details na succesvolle import

```

Summary
Mode          : DRY_RUN
Selected rows : 100
Changed endpoints : 34
Identical/skipped endpoints : 66
Updated endpoints : 0
Failed endpoints : 0
Plan CSV      : \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\OUTPUT\I_wsit_rest_apply_output\2026-
Result CSV    : \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\OUTPUT\I_wsit_rest_apply_output\2026-
Summary JSON  : \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\OUTPUT\I_wsit_rest_apply_output\2026-
Log           : \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\OUTPUT\I_wsit_rest_apply_output\2026-

Changed WSIT references preview
Showing 5 of 34 (default preview limit = 5). Full details and URLs are in wsit_rest_apply_result.csv.

Endpoint      Key          Attribute  Status          Delta  Changes
-----
BEP0USC74B    AS20-634480 Software   DRY_RUN_WOULD_UPDATE +2/-0 + AS20-623187, AS20-693757
BEP0USC74A    AS20-634481 Software   DRY_RUN_WOULD_UPDATE +2/-0 + AS20-623187, AS20-693757
BEP0USC76     AS20-634482 Software   DRY_RUN_WOULD_UPDATE +2/-0 + AS20-623187, AS20-623276, AS20-693757
BEP0USC75     AS20-634483 Software   DRY_RUN_WOULD_UPDATE +4/-0 + AS20-623187, AS20-624219, AS20-693757 (+1 more)
BEP0USC78     AS20-634484 Software   DRY_RUN_WOULD_UPDATE +2/-0 + AS20-623107, AS20-693757

... 29 more changed row(s). Full list: wsit_rest_apply_result.csv

SUCCESS: WSIT REST Apply
Duration: 00:00:26

Output: wsit_rest_apply_20260519_142742

wsit_rest_apply_plan.csv 2026-05-19 14:28
wsit_rest_apply_result.csv 2026-05-19 14:28
wsit_rest_apply_summary.json 2026-05-19 14:28
wsit_rest_apply.log 2026-05-19 14:28

```

3.5 Overkoepelende gebruikerslaag en overdraagbaarheid

Launcher-menu

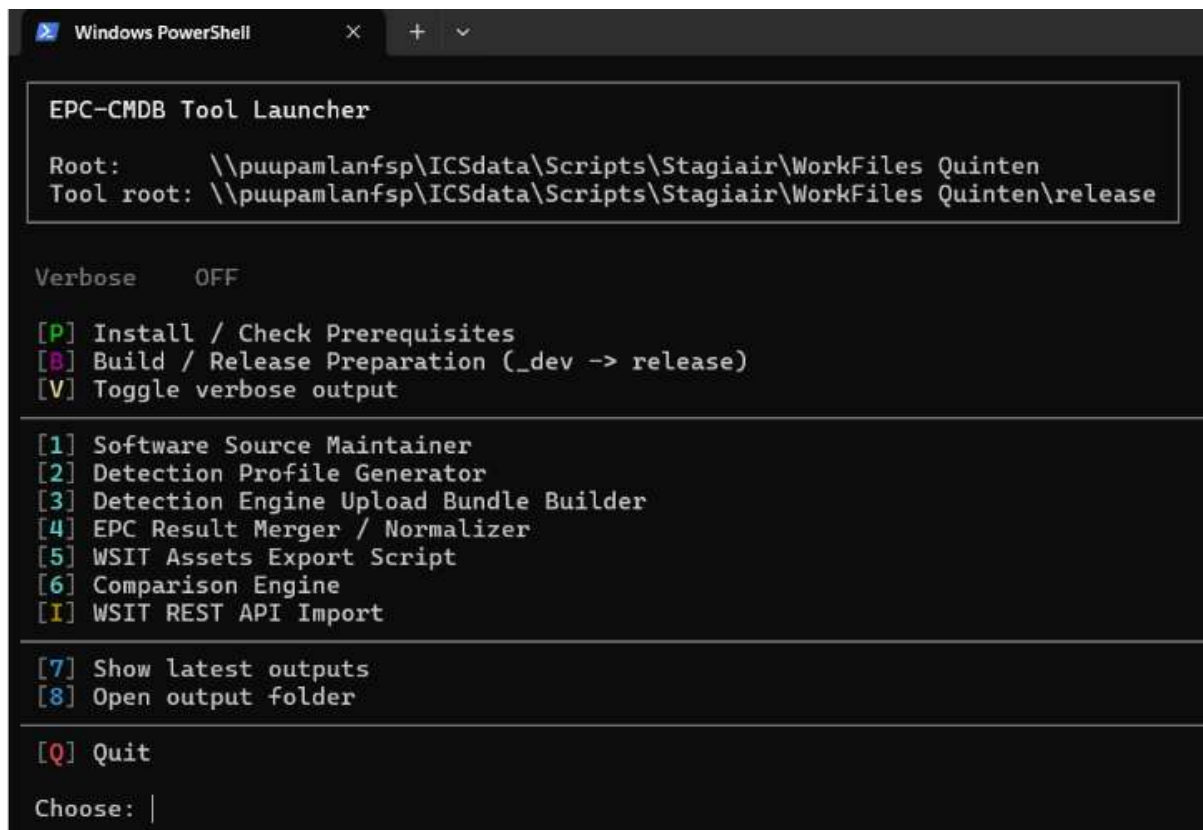
Om de volledige toolketen toegankelijk te maken voor toekomstige beheerders en collega's zonder kennis van de onderliggende scripts, werd een centrale launcher ontwikkeld in PowerShell. Deze launcher vormt de gebruikerslaag boven op de verschillende componenten van de EPC-CMDB Tool en fungeert als centraal toegangspunt voor de volledige workflow.

Bij het opstarten toont de launcher automatisch de geconfigureerde rootlocatie, die is ingesteld in het configuratie bestand in de root, en release-omgeving. Vanuit één menu kunnen gebruikers alle stappen van de workflow uitvoeren, gaande van het controleren van vereisten en het voorbereiden van de release folder tot het genereren van detectieprofielen, het verwerken van detectieresultaten en het uitvoeren van WSIT-importen. De volgorde van de menuopties weerspiegelt daarbij rechtstreeks de logische opbouw van de workflow te zien in figuur 1.

Daarnaast bevat de launcher ondersteunende functionaliteit voor beheer en controle. Gebruikers kunnen de meest recente outputbestanden per stap bekijken, outputmappen rechtstreeks openen en met toets 'V' schakelen tussen een beperkte en uitgebreide uitvoermodus.

Door alle functionaliteit achter één consistente interface te plaatsen, verlaagt de launcher de operationele drempel aanzienlijk en ondersteunt hij rechtstreeks het gebruiksgemak van de oplossing.

Figuur 18 EPC-CMDB Tool Launcher menu



```
Windows PowerShell
EPC-CMDB Tool Launcher

Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool root: \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Verbose    OFF

[P] Install / Check Prerequisites
[B] Build / Release Preparation (_dev -> release)
[V] Toggle verbose output

[1] Software Source Maintainer
[2] Detection Profile Generator
[3] Detection Engine Upload Bundle Builder
[4] EPC Result Merger / Normalizer
[5] WSIT Assets Export Script
[6] Comparison Engine
[I] WSIT REST API Import

[7] Show latest outputs
[8] Open output folder

[Q] Quit

Choose: |
```

Prerequisite check

Om de toolketen uitvoerbaar te maken op verschillende werkstations werd een prerequisitecontrole geïntegreerd in de launcher. Deze controle verifieert automatisch de aanwezigheid van de vereiste componenten, namelijk Python 3, PowerShell 7 en .NET 10.

Ontbrekende componenten kunnen rechtstreeks vanuit dezelfde interface worden geïnstalleerd.

Voor de Software Source Maintainer volstaat de .NET Runtime. Wanneer de applicatie ook opnieuw moet worden gecompileerd tijdens de buildfase, is de .NET SDK vereist. Door deze afhankelijkheden centraal te beheren wordt de kans op configuratiefouten beperkt en kunnen nieuwe gebruikers de werkomgeving eenvoudiger voorbereiden.

Figuur 19 *Prerequisites menu*

```
Step P - Prerequisites
Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool root: \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Prerequisites
Required: PowerShell 7, Python 3
Build/publish: .NET 10 SDK
Runtime-only option: .NET 10 Runtime

Current prerequisite status

PowerShell 7      OK  7.6.1
pwsh.exe         C:\Program Files\WindowsApps\Microsoft.PowerShell_7.6.1.0_x64__8wekyb3d8bbwe\pwsh.exe

Python 3          OK  Python 3.14.5
python.exe       C:\Users\DEMEYQ\AppData\Local\Programs\Python\Python314\python.exe

.NET 10 SDK       OK  10.0.300
.NET 10 Runtime  OK  10.0.8
dotnet.exe       C:\Users\DEMEYQ\AppData\Local\Microsoft\dotnet\dotnet.exe

Notes:
- .NET SDK normally includes the .NET runtime.
- User PATH changes may require a new terminal.

[C] Check all prerequisites / refresh status
[A] Install all missing prerequisites

[1] Install/update PowerShell 7
[2] Install/update Python 3
[3] Install/update .NET 10 SDK
[4] Install/update .NET 10 Runtime

[Q] Back to main menu

Choose:
```

Build- en releasevoorbereiding

Om de scheiding tussen ontwikkelomgeving en gebruikersomgeving te behouden, werd een geautomatiseerde build- en releaseprocedure voorzien. Deze stap wordt vanuit de launcher gestart en voert een PowerShell-script uit dat de meest recente scripts vanuit de ontwikkelmap naar de releaseomgeving kopieert.

Daarnaast worden de Python-projecten gecompileerd naar zelfstandige uitvoerbare bestanden met behulp van PyInstaller. Indien gewenst wordt ook de Software Source Maintainer opnieuw gepubliceerd via het .NET publish-proces. Hiervoor is de .NET SDK vereist; voor het louter uitvoeren van de applicatie volstaat de .NET Runtime.

Na afloop bevat de release-map alle bestanden die nodig zijn voor operationeel gebruik.

Hierdoor werken eindgebruikers uitsluitend met de releaseomgeving, terwijl de broncode en projectbestanden gescheiden blijven in de ontwikkelomgeving.

Figuur 20 Build / Release Preparation menu

```
Step B - Build / Release Preparation

Root:      \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten
Tool root:  \\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkFiles Quinten\release

Copies/builds latest files from _dev to release.

Skip Software Source Maintainer dotnet publish? [Y/n/q]:

Tool      C:\Program Files\WindowsApps\Microsoft.PowerShell_7.6.1.0_x64__8wekyb3d8bbwe\pwsh.exe
Arguments -NoProfile -ExecutionPolicy Bypass -File "\\puupamlanfsp\ICSdata\Scripts\Stagiair\WorkF
Output    quiet, tail after completion

Execute? [Y/n/q]:

Running Build / Release Preparation
Press Q to stop this process.

SUCCESS: Build / Release Preparation
Duration: 00:01:23

Last output

20318 INFO: Performing binary vs. data reclassification (13 entries)
20335 INFO: Looking for ctypes DLLs
20352 INFO: Analyzing run-time hooks ...
20355 INFO: Including run-time hook 'pyi_rth_inspect.py' from 'C:\\Users\\DEMEYQ\\AppData\\Loc...
20376 INFO: Creating base_library.zip...
20459 INFO: Looking for dynamic libraries
20638 INFO: Extra DLL search directories (AddDllDirectory): []
20638 INFO: Extra DLL search directories (PATH): []
20946 INFO: Warnings written to C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1\\EPC-CMDB-Tool-build\\compa...
21007 INFO: Graph cross-reference written to C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1\\EPC-CMDB-Too...
21126 INFO: checking PYZ
21126 INFO: Building PYZ because PYZ-00.toc is non existent
21126 INFO: Building PYZ (ZlibArchive) C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1\\EPC-CMDB-Tool-buil...
21566 INFO: Building PYZ (ZlibArchive) C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1\\EPC-CMDB-Tool-buil...
21593 INFO: checking PKG
21593 INFO: Building PKG because PKG-00.toc is non existent
21593 INFO: Building PKG (CArchive) comparison-engine.pkg
24427 INFO: Building PKG (CArchive) comparison-engine.pkg completed successfully.
24438 INFO: Bootloader C:\\Users\\DEMEYQ\\AppData\\Local\\Programs\\Python\\Python314\\Lib\\site-packag...
24438 INFO: checking EXE
24439 INFO: Building EXE because EXE-00.toc is non existent
24439 INFO: Building EXE from EXE-00.toc
24439 INFO: Copying bootloader EXE to C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1\\EPC-CMDB-Tool-build...
24517 INFO: Copying icon to EXE
24560 INFO: Copying 0 resources to EXE
24560 INFO: Embedding manifest in EXE
24603 INFO: Appending PKG archive to EXE
24686 INFO: Fixing EXE headers
26596 INFO: Building EXE from EXE-00.toc completed successfully.
26607 INFO: Build complete! The results are available in: C:\\Users\\DEMEYQ\\AppData\\Local\\Temp\\1...
```

Configuratie, outputrouting en documentatie

Om de oplossing duurzaam beschikbaar te maken na afloop van de stage, werden twee maatregelen genomen op het vlak van infrastructuur en documentatie. Ten eerste is een permanente

netwerkshare aangevraagd die specifiek is toegewezen aan de tool. Tijdens de ontwikkeling werd gebruikgemaakt van een persoonlijke netwerkshare, maar die wordt binnen Pfizer elke dertig dagen gewist. De permanente share garandeert dat bronbestanden, scripts en resultaten beschikbaar blijven voor toekomstige verantwoordelijken. Ten tweede werd een Confluence-paginastructuur opgezet als centrale handleiding voor collega's binnen Pfizer. De hoofdpagina biedt een overzicht van de volledige workflow met het architectuurdiagram, de vereisten voor de lokale werkomgeving en een genummerde lijst van stappen. Elke stap verwijst naar een afzonderlijke Confluence-detailpagina. Op die detailpagina's wordt per tool beschreven wat de functionaliteit is, welke voorwaarden gelden, hoe de tool wordt gebruikt en welke output wordt geproduceerd, aangevuld met stapsgewijze afbeeldingen en concrete instructies. In het projectplan werd overdrachtsdocumentatie expliciet als bedrijfsdeliverable benoemd (De Meyer, 2026a, sectie 9.1).

Naast documentatie werd ook ingezet op technische maatregelen om de drempel voor gebruik te verlagen. Om de workflow als geheel uitvoerbaar te maken zonder kennis van individuele commando's of bestandspaden, werd een centrale *launcher* ontwikkeld. Dit interactieve PowerShell-menu biedt alle stappen van de toolketen aan en bevat een resultaatweergave waarmee de meest recente output per stap kan worden bekeken en geopend. De *launcher* werkt uitsluitend met de bestanden in de release-map.

De oplossing is uitgebreid met een configuratiebestand (*config.json*) waarin de *work_root*, *dev_folder*, *release_folder*, *source_folder* en *output_folder* zijn gedefinieerd. Dit minimaliseert de onderhoudslast: bij een verhuizing naar een nieuwe netwerkshare hoeft enkel de *work_root* aangepast te worden.

Daarnaast werd een *buildscript* voorzien dat de Python-tools compileert naar zelfstandige uitvoerbare bestanden via *PyInstaller* en deze naar de release-map kopieert. Hierdoor blijft de scheiding tussen ontwikkelomgeving en gebruikersomgeving gewaarborgd.

De buildflow voor de Software Source Maintainer is geoptimaliseerd door over te stappen op een *non-self-contained* architectuur. Dit verkort de opstarttijd van de applicatie aanzienlijk. En geeft

hetzelfde eenvoudige opstartprocedure aangezien dit via het launcher-menu gebeurt. Voor het gebruik van deze tool is de installatie van .NET 10 vereist op de werkomgeving, wat door beheerders eenvoudig via winget kan worden uitgevoerd.

4. Beperkingen

Een eerste bewuste beperking is dat de tool enkel actieve, vooraf gemodelleerde in-house software detecteert. Gedecommissioneerde software wordt niet actief opgespoord. Die keuze is niet technisch ingegeven: het YAML-bronbestand zou uitgebreid kunnen worden met een statusveld voor gedecommissioneerde software. De beperking ligt vooral in de onderhoudslast. Het structureel bewaren en onderhouden van historische softwaredefinities zou het bronbestand aanzienlijk vergroten, terwijl de primaire controle op correcte uitfasering binnen Pfizer via de bestaande procedure- en ticketflow verloopt. De tool moet daarom gelezen worden als een detector voor actieve, gemodelleerde software binnen scope, niet als een universele detector voor alle ooit aanwezige in-house software.

Een tweede beperking heeft betrekking op uitvoerbaarheid op oudere of afwijkende endpointtypes. De detectie-engine werd succesvol gevalideerd op 64-bit endpoints met onder meer Windows 7, Windows Server 2008, Windows Server 2012 en Windows 10. Voor 32-bit endpoints bleek de beperking echter niet in de detectie-engine zelf te zitten, maar in de manier waarop Endpoint Central scripts uitvoert met de vereiste administratieve context. Deze endpoints vallen daardoor buiten de actuele uitvoerbare scope.

Daarnaast bleven bepaalde Windows 7 Enterprise/Ultimate x64-endpoints problematisch door een access violation, vermoedelijk gelinkt aan de patch-baseline of execution context. Ook op Endpoint Central server 02 werd vastgesteld dat custom script execution niet bruikbaar was onder de geteste accounts. Deze beperkingen werden gedocumenteerd, maar hadden beperkte impact op de totale bruikbaarheid van de tool omdat de finale productiescan succesvol kon worden uitgevoerd op duizenden Windows 10-machines.

Een laatste geregistreerde beperking situeert zich in de importfase via de WSIT REST API. Tijdens de validatie op de WSIT-testomgeving is gebleken dat de automatische import faalt op endpoints die een corrupt of ongeldig, ongerelateerd attribuut bevatten. De onderliggende Jira-architectuur weigert de update van het object in dat geval volledig, ongeacht of de nieuwe

softwarerelatie zelf wel correct is. Deze blokkade is momenteel bevestigd op de testomgeving (WSIT TEST); de impact hiervan op de productieomgeving (WSIT PROD) is nog niet getest.

5. Besluit

De gerealiseerde toolketen beantwoordt de centrale onderzoeksvraag uit het projectplan: er bestaat een onderhoudbare en overdraagbare workflow waarmee in-house software binnen de afgesproken scope op endpoints kan worden gedetecteerd, vergeleken met de CMDB en semiautomatisch kan worden opgevolgd.

Terugkoppelend naar het evaluatiekader (De Meyer, 2026a, sectie 7.3) kan de realisatie op drie criteria worden beoordeeld: betrouwbaarheid, onderhoudbaarheid en overdraagbaarheid. Om de betrouwbaarheid van de detectie te valideren, is software handmatig geïnstalleerd op testsystemen volgens de bijhorende IID's en zijn de detectieresultaten vergeleken met de verwachte toestand. De detectie-engine leverde daarbij reproduceerbare resultaten op zonder evidente foutpositieven. De onderhoudbaarheid werd in de praktijk bevestigd doordat tijdens de stage meerdere iteraties van softwaredefinities werden doorgevoerd die correct doorstroomden naar profielen en detectie, zonder wijzigingen aan de scripts zelf. De overdraagbaarheid wordt ondersteund door het grafische hulpmiddel voor bronbestandbeheer, de vaste mappenstructuur met automatische input- en outputrouting, de centrale launcher en de release-map met gebruikersklare bestanden. Daarnaast dragen ook de dynamische endpointgroepen in Endpoint Central en de Confluence-documentatie met gedetailleerde instructies per stap bij aan de overdraagbaarheid.

Als vooruitblik kan worden opgemerkt dat de huidige architectuur ruimte laat voor toekomstige uitbreiding. Het bronbestand kan worden aangevuld met nieuwe software zonder wijzigingen aan de detectie-engine. De mappenstructuur en de automatische routing maken het mogelijk om nieuwe stappen aan de keten toe te voegen. Een mogelijke uitbreiding is de implementatie van een volledig geautomatiseerde workflow. Dit vereist dat het YAML-bronbestand ongewijzigd blijft en de *Custom Script Configurations* periodiek worden uitgevoerd. Onder deze voorwaarden kan de tool in een automatische modus opereren. Het systeem doorloopt dan

autonoom het volledige proces, van initiatie tot en met de rapportage. Na visuele verificatie verleent de gebruiker goedkeuring. Tot slot importeert de tool de gegevens in de CMDB.

De Confluence-documentatie biedt collega's een startpunt om de workflow zelfstandig te gebruiken en uit te breiden. De realisatie toont aan dat het binnen een complexe beheeromgeving mogelijk is om een gestructureerde workflow uit te werken voor in-house softwaredetectie en -vergelijking met de CMDB, die bruikbaar en uitbreidbaar blijft voor toekomstig beheer.

Literatuurlijst

Atlassian. (z.d.). *Managing your assets with Assets*.

<https://confluence.atlassian.com/servicemanagementserver/managing-your-assets-with-assets-1044784300.html>

De Meyer, Q. (2026a). *Projectplan — Onderhoudbare detectie- en vergelijkingsketen voor in-house software in EPC en WSIT/CMDB* [Ongepubliceerd document]. Thomas More / Pfizer Puurs.

De Meyer, Q. (2026b). *Verslag Stage: Pfizer Puurs* [Ongepubliceerd document]. Thomas More.

ManageEngine. (z.d.-a). *Custom groups — Creating and managing custom groups*.

https://www.manageengine.com/products/desktop-central/help/configuring_desktop_central/dynamic-custom-group.html

ManageEngine. (z.d.-b). *Custom Scripts — Adding & executing custom scripts for Windows desktop administration*. <https://www.manageengine.com/products/desktop-central/custom-scripts.html>

Microsoft. (z.d.). *PowerShell documentation*. <https://learn.microsoft.com/en-us/powershell/>

Pavlov, I. (z.d.). *7-Zip*. <https://www.7-zip.org/>

Pfizer Belgium. (z.d.). *Productie-Puurs*. <https://www.pfizer.be/nl/over-ons/pfizer-in-belgie/productie-puurs>

Python Software Foundation. (z.d.). *Python 3 documentation*. <https://docs.python.org/3/>

Bijlagen

5.1 Bijlage A — Begrippenlijst

AQL: *Assets Query Language*; querytaal binnen Jira Assets waarmee objecten gefilterd en opgezocht kunnen worden.

CD-shares: gedeelde netwerklocaties waarop installatiebestanden van software beschikbaar worden gesteld, doorgaans aangeduid met een cd-nummer in het formaat CDxxxx.

CMDB: *Configuration Management Database*; de asset-database die binnen WSIT wordt beheerd via Jira Assets. Hierin worden software en endpoints als objecten beheerd in een objectschema.

Discrepantie: verschil tussen de software-endpointrelaties in de CMDB en de feitelijke softwaretoestand op een endpoint.

Dynamic Custom Group: endpointgroep in Endpoint Central waarvan het lidmaatschap automatisch wordt bepaald op basis van filtercriteria zoals besturingssysteem of service pack.

Endpointrealiteit: de feitelijke toestand van software-installaties op een endpoint, vastgesteld door de detectie-engine op basis van JSON-detectieprofielen. Dit begrip wordt gebruikt als tegenhanger van de administratieve registratie in de CMDB.

EPC: Endpoint Central; platform voor endpointbeheer, scriptuitvoering op endpoints en aanvullende beheerfunctionaliteiten.

IID: *Installation Instruction Document*; documentatie waarin de installatie-informatie en -procedure van in-house software worden beschreven, inclusief een deïnstallatieprocedure.

In-house software: intern ontwikkelde en niet-standaardsoftware die niet betrouwbaar door de standaardinventarisatie van EPC wordt gedetecteerd.

JSON-detectieprofielen: gegenereerde artefacten op basis van het YAML-bronbestand, die door de detectie-engine worden gebruikt voor endpointcontrole.

Object key: unieke identificatiesleutel van een object in Jira Assets, gebruikt om endpoints en software eenduidig te koppelen bij import en export.

PowerShell SecretStore: module voor PowerShell waarmee gevoelige informatie, zoals authenticatietokens, lokaal en versleuteld kan worden opgeslagen.

PyInstaller: open-source hulpmiddel waarmee Python-scripts worden gecompileerd naar zelfstandige uitvoerbare bestanden die zonder Python-installatie kunnen worden uitgevoerd.

REST API: *Representational State Transfer Application Programming Interface*; waarmee registraties uit Jira Assets programmatisch kunnen worden opgehaald en toegevoegd.

WSIT: *Workstation and Software Inventory Tool*; de overkoepelende Jira Service Management-omgeving van Pfizer, waarbinnen onder meer de CMDB via Jira Assets wordt beheerd.

YAML-bronbestand: centraal bronbestand waarin in-house software gestructureerd wordt beschreven, inclusief installatiepaden, bronmedia, versies en detectiekenmerken.

5.2 Bijlage B — Mappenstructuur .OUTPUT

```
.OUTPUT\  
├── 2_generate_detection_profiles_output\  
│   ├── <datum>\  
│   │   └── detection_profiles_<tijdstempel>\  
│   │       └── detection_profiles.json  
├── 3_detection_engine_output\  
│   ├── <datum>\  
│   │   └── detection_engine_<endpoint>_<tijdstempel>\  
│   │       ├── detection_engine.log  
│   │       ├── detection_full_details.json  
│   │       ├── detection_human_readable_summary.txt  
│   │       └── detection_summary.json  
├── 4_epc_result_merger_output\  
│   ├── <datum>\  
│   │   └── epc_inventory_merge_<tijdstempel>\  
│   │       ├── epc_inventory_endpoint_view_<tijdstempel>.json  
│   │       └── epc_inventory_merge_report_<tijdstempel>.csv  
├── 5_wsit_api_export_output\  
│   ├── <datum>\  
│   │   └── wsit_api_export_<tijdstempel>\  
│   │       ├── wsit_api_export.log  
│   │       ├── wsit_api_export.json  
│   │       ├── wsit_api_export_endpoint_view.json  
│   │       └── wsit_api_export_summary.json  
├── 6_wsit_epc_compare_output\  
│   ├── <datum>\  
│   │   └── 6_Comparison_Engine_<tijdstempel>\  
│   │       ├── comparison_discrepancy_report.html  
│   │       └── Endpoint-Software_WSIT-importfile.csv  
└── I_wsit_rest_apply_output\  
    ├── <datum>\  
    │   └── wsit_apply_<tijdstempel>\  
    │       ├── wsit_apply_plan_<tijdstempel>.csv  
    │       ├── wsit_apply_result_<tijdstempel>.csv  
    │       ├── wsit_apply_summary_<tijdstempel>.json  
    │       └── wsit_apply_log_<tijdstempel>.log
```

5.3 Bijlage C — Launcher-configuratie

```
{  
  "work_root": "\\puupamlanfsp\\ICSdata\\Scripts\\Stagiair\\WorkFiles Quinten",  
  "dev_folder": "_dev",  
  "release_folder": "release",  
  "source_folder": ".SOURCE",  
  "output_folder": ".OUTPUT"  
}
```