IPTCompile – Patchanalys & Strukturrekommendationer

Datum: 2026-02-19
Gäller: `IPTCompile_MISQ_SELFCHECKED_20260219.zip`

Del 1: MISQ-patchen – Vad var fel och vad vi fixat

Sammanfattning av den ursprungliga DIFF-patchen

Patchens *syfte* var korrekt: flytta MISQUANTITATION-scanningen från Seal NEG/POS-filerna till Worksheet-filen (som faktiskt innehåller "Data Summary"-bladet). Implementationen hade dock fem kritiska buggar som alla hade orsakat antingen runtime-fel eller tyst felaktigt beteende.

Bugg 1: Parameternamn `-Packages` matchar inte funktionssignaturen

Patch använde	Funktionen definierar (rad 1272)
`-Packages @($tmpPkgMisq)`	`-ExcelPackages`

PowerShell hade gett ett `ParameterBindingException` vid runtime.

Fix: Använder korrekt `-ExcelPackages`.

Bugg 2: Funktionens inre diff matchar en annan version

Patchen refererar till variabelnamn (`$r`, `$ws`, `$hit`, `$sid`, `$s`, `$k`) som inte existerar i den nuvarande koden. Funktionen använder `$row`, `$dataSummaryWs`, `$misqSampleIds`, `$sampleId`, `$status`. Patchen gick inte att applicera mot nuvarande kodbas.

Fix: Vi ändrar inte funktionens inre logik — den fungerar redan korrekt.

Bugg 3: `$script:MisqSampleIds` vs lokal `$misqSampleIds` — scope-mismatch

Patchen tilldelade `$script:MisqSampleIds` men integrationskoden (rad ~1739) läser den lokala `$misqSampleIds`. MISQ-data hade aldrig nått RuleEngine.

Fix: Behåller konsekvent lokal `$misqSampleIds` genom hela flödet.

Bugg 4: Skip-guard orsakade att Worksheet aldrig öppnades för övrig extraktion

Flödet i patchen:

1. Öppna `$tmpPkgMisq` → scanna → **Dispose** i finally-block
2. Sätt `$script:WsPkgWasOpenedForMisq = $true`
3. Senare: skip-guard ser true → hoppar över `$tmpPkg = New-Object ExcelPackage(...)`
4. `$tmpPkg` förblir **null** → All WS-extraktion (Equipment, Headers, QC Reminder) misslyckas tyst

Fix: Ingen skip-guard. Vi öppnar filen en gång för MISQ (dispose direkt), sedan öppnar den normala WS-blocket den igen som vanligt. Dubbel öppning av en read-only fil är helt säker.

Bugg 5: Syntetiska MISQ-rader hade fel schema

Den ursprungliga koden skapade syntetiska RuleEngine-rader med:

- `Deviation = 'Major Functional'` — men RuleEngine-displayen förväntar `'FP'` / `'FN'`
- `AssayName` istället för `Assay`
- `'Cartridge S/N'` istället för `CartridgeSN` (PSObject-fält utan mellanslag)
- 20+ saknade fält som `Write-RuleEngineDebugSheet` försöker läsa

Resultat: MISQ-rader hamnade i `default` -caset och visades som "OK" — komplett omvänt.

Fix: Syntetiska rader skapas nu med exakt samma fält-schema som `Invoke-RuleEngine` producerar (37 fält). `Deviation = 'FP'` matchar display-lagret korrekt.

Del 2: RuleEngine.ps1 — Kompatibilitetsfix

Problem: FP/FN-caset kastar bort ErrorCode

I `Write-RuleEngineDebugSheet` (rad ~2640) fanns:

```
'FP' { $data[$i, 1] = 'Falskt positiv' }   # Överskriven hårdkodat
```

Alla MISQ-rader med `ErrorCode = 'MISQUANTITATION'` fick ErrorCode-kolumnen skriven till

"Falskt positiv" — informationen gick förlorad.

Fix: Kontrollerar nu om `ErrorCode` har ett specifikt värde innan fallback till det generiska textlabeln. Se `PATCH_RuleEngine_MISQ_compat.diff`.

Sammanfattning: DeviationCounts['FP'] fungerar nu korrekt

Summans `DeviationCounts` uppdateras nu med `'FP'` istället för `'Major Functional'`, vilket innebär att MISQ-träffar syns korrekt i:

- Sammanfattningssektionen "❌ Falskt positiv (Major Functional)"
- `$top` / TopDeviations-filtret
- Flikfärgskodningen (röd vid >5 avvikelser)

Del 3: Strukturrekommendationer — Säker refaktorering

Kodstatistik (nuvarande)

Fil	Rader	Funktioner
Main.ps1	4 154	33 (+ 8 inline djupt i buildflödet)
RuleEngine.ps1	2 889	~30
DataHelpers.ps1	1 575	~25
Config.ps1	360	8
Logging.ps1	257	8
SharePointClient.ps1	311	—
Övriga	266	—
Totalt	**~9 812**	**~100+**

🔴 Hög prioritet — Låg risk

1. Extrahera inline-funktioner ur buildflödet

Main.ps1 definierar **8 funktioner** inuti byggblocket (8+ indenteringsnivåer):

```
_EquipTokens, _EquipPretty, _FixMonthText, _EquipEvalList,
_EquipEvalMonth, _Txt, Set-MergedWrapAutoHeight, Add-Hyperlink,
Find-RegexCell, Get-SealHeaderDocInfo, Find-InfoRow, Find-LabelValueRightward
```

Risk: Dessa funktioner omskapas vid *varje* rapport-generering. Flyttas de till modulnivå (t.ex. DataHelpers.ps1) sparar man CPU och gör dem testbara.

Säker metod:

1. Kopiera funktionen till modulen
2. Lägg till `if (-not (Get-Command FuncName -EA SilentlyContinue))` guard (redan existerande mönster)
3. Testa att rapporten genererar identisk output
4. Ta bort inline-kopian

2. Eliminera 78+ tomma catch-block i Main.ps1

```
# Nuvarande (78 instanser):
try { ... } catch {}

# Rekommenderat:
try { ... } catch { <# Avsiktlig: Excel-cell kan vara tom #> }
```

Vinst: Debugging tar hälften av tiden. Om en bugg döljs i ett tomt catch vet du aldrig att den hände.

Säker metod: Gå igenom batch-vis, lägg till kommentarer. Inga funktionella ändringar.

3. Konsolidera `$script:` -variabler

72 stycken `$script:` -variabler i Main.ps1. Minst 10 av dessa används bara för att föra data mellan build-blockets sektioner. En enkel `$buildState` -hashtable minskar risken för namn-kollisioner och gör det tydligare vad som är persisterat tillstånd vs. tillfälligt.

```
# Istället för:
$script:RuleEngineShadow = $re
$script:RuleBankCache = $rb
$script:RuleEngineCsvObjs = $csvObjs

# Samla i:
$script:BuildCtx = @{
    RuleEngineShadow = $re
```

```
    RuleBankCache     = $rb
    CsvObjs           = $csvObjs
}
```

🟡 Medel prioritet — Medel risk

4. Bryt ut rapportlogiken ur Main.ps1

Main.ps1:s byggblock (rad 1470–4140, ~2 670 rader) mixar:

- GUI-uppdateringar (Set-UiStep, Gui-Log)
- Fil-I/O (EPPlus-öppning, CSV-import)
- Affärslogik (MISQ-scan, RuleEngine, signaturjämförelse)
- Excel-rapportskrivning (30+ sektioner med celler, färger, formler)

Mål: `Main.ps1` = GUI + orkestrator (~1 500 rader), ny `ReportBuilder.ps1` = rapport-generering (~1 500 rader).

Säker metod:

1. Skapa `Modules/ReportBuilder.ps1`
2. Flytta en sektion i taget (börja med Equipment-blocket, det är mest isolerat)
3. Parameterisera: skicka `$Config`, `$Pkg`, `$data` som argument — inte `$script:`
4. Testa varje sektion isolerat före nästa flytt

5. Minska try/catch-djupet

158 try/catch i Main.ps1 — minst 20 fall av try-inuti-try. En vanlig pattern:

```
try {
    try {
        try { $status = ($ws.Cells[$r, 3].Text + '') } catch { $status = $null }
    } catch {}
} catch {}
```

Refaktor: Skapa en helper:

```
function Safe-CellText([OfficeOpenXml.ExcelWorksheet]$Ws, [int]$Row, [int]$Col) {
    try { return ($Ws.Cells[$Row, $Col].Text + '').Trim() } catch { return '' }
}
```

Denna enda funktion kan ersätta 30-40 try/catch-block och göra koden dramatiskt mer läsbar.

🟢 Låg prioritet — Hög vinst på sikt

6. Inför Pester-tester för RuleEngine

RuleEngine.ps1 är redan väl isolerad (ren input/output, inga GUI-beroenden). Det gör den idealisk för enhetstester:

```
Describe 'Invoke-RuleEngine' {
    It 'classifies MISQUANTITATION as FP' {
        $result = Invoke-RuleEngine -CsvObjects $mockCsv -RuleBank $rb
        $misq = $result.Rows | Where-Object { $_.ErrorCode -eq 'MISQUANTITATION'
        $misq.Deviation | Should -Be 'FP'
    }
}
```

Vinst: Varje framtida ändring i RuleEngine kan verifieras automatiskt.

7. Versionerad RuleBank med hash-validering

RuleBank.compiled.ps1 laddas via `& $cp` (dot-source av körbar fil). Lägg till en SHA256-hash-validering för att säkerställa att ingen har ändrat filen oavsiktligt:

```
$hash = (Get-FileHash $cp -Algorithm SHA256).Hash
if ($hash -ne $expectedHash) { throw "RuleBank integrity check failed" }
```

Del 4: Filer i detta paket

Fil	Beskrivning
`Main.ps1`	**Patchad** – MISQ scannar Worksheet, korrigerat rad-schema, scope-fix
`Modules/RuleEngine.ps1`	**Patchad** – FP/FN bevarar ErrorCode
`PATCH_Main_MISQ_fix.diff`	Unified diff för Main.ps1 (granska innan deploy)
`PATCH_RuleEngine_MISQ_compat.diff`	Unified diff för RuleEngine.ps1

`ANALYS_OCH_REKOMMENDATIONER.md`	Detta dokument

Hur man applicerar (om du vill använda diff istället för patchade filer)

```
# Från projektets rotmapp:
patch -p0 < PATCH_Main_MISQ_fix.diff
patch -p0 < PATCH_RuleEngine_MISQ_compat.diff
```

Verifieringssteg

1. Kör en VL-assay (t.ex. Xpert HIV-1 Viral Load) med känd MISQUANTITATION i Worksheet
2. Kontrollera att 🧪 `MISQ (Data Summary): träffar: N` visas i loggen
3. Kontrollera att CSV Sammanfattning-bladet visar MISQ-rader som "Major Functional" med mörkröd bakgrund
4. Kontrollera att sammanfattningssektionen visar "❌ Falskt positiv (Major Functional)" med korrekt antal
5. Kör en icke-VL-assay och verifiera att MISQ-logiken hoppas över utan fel
6. Verifiera att WS-extraktion (Equipment, Headers, QC Reminder) fortfarande fungerar