

# Kodgranskning av LogTiter→MISQUANTITATION i IPTCompile

## Sammanfattande bedömning

Den nya utbyggnaden (QuantSpecRules + LogTiter-extraktion + MISQ-markering) är i grunden väl upplagd och går i rätt riktning (separation av "call-regler" och "kvant-specs" är en bra design). Däremot finns två konkreta problem som gör att implementationen i nuläget inte är fullt robust:

1) **Kritisk runtime-risk i RuleEngine:** Apply-LogTiterQuantSpec tar RuleBank som [hashtable], men Invoke-RuleEngine arbetar med RuleBank som [pscustomobject]. Det här leder normalt till ParameterBindingException eftersom PowerShell försöker typkonvertera vid parameterbindning. <sup>1</sup>

2) **Sammanfattningsbladet räknar inte MISQ:** MISQ visas korrekt i detaljtabellen (Error Code + Avvikelse) och färgas korrekt, men i "Resultat Avvikelser"-sektionen listas bara FP/FN som Major. Då kan MISQ hamna "osynligt" i sammanfattningen och logiken för "Inga avvikelser" kan bli missvisande när MISQ finns.

## PASS/FAIL enligt din checklista

Område	Status	Kort motivering
A) RuleBank.compiled.ps1	PASS	QuantSpecRules finns, 10 rader, konsekvent schema (AssayPattern/MatchType/ControlCode/Min/Max/FailMode/...) och värden matchar din spec.
B) Modules/ RuleEngine.ps1	FAIL	Kritisk typmismatch för -RuleBank i Apply-LogTiterQuantSpec + MISQ saknas i sammanfattningssektionen. <sup>1</sup>
C) Edge cases	PASS (med begränsningar)	LogTiter-extraktionen är korrekt för "(log 2.08)" och "(log 2,08)" och fail-silent om saknas, men triggas inte om log skrivs i annat format (vilket är enligt ditt nuvarande krav). <sup>2</sup>

## Granskning av RuleBank.compiled.ps1

### QuantSpecRules finns och är konsekvent

I RuleBank/RuleBank.compiled.ps1 finns QuantSpecRules och den innehåller 10 rader (Meta.Counts visar också 10) med konsekvent nyckeluppsättning:

- AssayPattern, MatchType, Call, ControlCode, Metric
- Min, Max, FailMode
- OnFailErrorCode, OnFailDeviation, OnFailErrorName, OnFailGeneratesRetest

- Enabled, Priority, Note

Detta är en stabil struktur för att bygga ut fler assays/ControlCode senare.

## Observation om matchningsstrategi (EQUALS)

Alla dina QuantSpec-rader använder MatchType = EQUALS. Det är "strikt korrekt" om Assay-fältet i CSV alltid är exakt samma sträng. Om det i praktiken förekommer varianter (t.ex. extra mellanslag, versionstext, eller små stavningsskillnader) så kommer MISQ inte trigga. Detta är inte en "bugg" i sig, men en **driftsrisk**.

Min rekommendation är att behålla EQUALS initialt (minimerar oavsiktliga matcher), men att ni snabbt överväger WILDCARD / REGEX för assay-familjer när ni sett verkliga data.

## Granskning av Modules/RuleEngine.ps1

### LogTiter-extraktionen är korrekt för angivet format

Funktionen Try-ExtractLogTiter använder en regex som matchar parentesformatet och accepterar både punkt och komma, och parsar med invariant kultur. Detta är korrekt för ditt krav och är robust mot whitespace-varianter. <sup>3</sup>

Att använda CultureInfo.InvariantCulture är ett bra val för stabil parsing oberoende av systemets lokala decimalformat. <sup>4</sup>

### QuantSpec-applikationen är logiskt rätt – men har en kritisk typbugg

Du applicerar QuantSpec-reglerna i rätt ordning i flödet: ControlCode (\$cc) beräknas innan Apply-LogTiterQuantSpec anropas, och Test Result cachas och återanvänds. Det är bra.

**Problemet ligger i funktionssignaturen:**

- Invoke-RuleEngine tar RuleBank som [pscustomobject]
- Apply-LogTiterQuantSpec tar RuleBank som [hashtable]

PowerShells parameterbindning försöker typkonvertera argument till parametertyp. När det inte finns en naturlig konvertering mellan PSCustomObject och Hashtable kommer det normalt kasta ett binding-fel. <sup>1</sup>

Detta är en blockerande issue: det kan stoppa hela körningen även om QuantSpecRules inte skulle matcha en viss rad.

### MISQ hamnar rätt i detaljrapporten och färgas korrekt

I detaljtabellen (kolumnerna enligt dina headers) görs rätt sak:

- **Error Code** får MISQUANTITATION
- **Avvikelse** får Major Functional
- Styling för MISQ använder samma MajorBg/MajorFg och markerar både Avvikelse-kolumnen (C) och Error Code-kolumnen (B)

Att använda EPPlus fill/style på det sättet är korrekt och i linje med EPPlus praxis. 5

## Sammanfattningen missar MISQ

I "Resultat Avvikelser"-sektionen plockas OK, FP och FN (och Minor) upp, men **inte MISQ**. Effekten blir:

- MISQ syns i detaljraderna, men inte i sammanfattningsraden för Major
- Texten "✓ Inga avvikelser hittades" kan bli missvisande när MISQ finns, eftersom villkoret inte tar hänsyn till MISQ (och inte heller jämför OK==Total, trots kommentaren).

Detta är inte lika "krasch-kritiskt" som typfelet, men det är ett tydligt funktionellt glapp i rapporteringen.

## Robusthet och edge cases

### POS med ControlCode ≠ 1

Reglerna kräver `ControlCode = 1`, så MISQ triggar inte för andra control levels. Det är i linje med din nuvarande spec.

### NEG-rader

`Apply-LogTiterQuantSpec` returnerar tidigt om `ObservedCall` inte är POS. Det innebär att NEG inte triggar MISQ – också korrekt.

### Test Result utan log-format

`Try-ExtractLogTiter` returnerar `$null` om `(log ...)` saknas och `Apply-LogTiterQuantSpec` gör då "no change". Detta är exakt "fail-silent"-beteendet du efterfrågade.

### Decimal med komma

Både extraktion och Min/Max-parsing normaliserar komma till punkt och använder invariant kultur. Det är en korrekt strategi för att undvika lokala kulturproblem. 3

### Assay med annat log-format

Om instrumentet skriver t.ex. `log=2.08` (utan parentes) eller "Log:" i annan struktur, triggar MISQ inte. Det är inte fel givet din nuvarande avgränsning ("endast LogTiter nu"), men bör dokumenteras som en begränsning.

## Rekommenderade minimala patchar

Nedan är två **minimala** patchar som adresserar de faktiska bristerna, utan att förändra övrig logik.

### Patch för kritisk typmismatch i Apply-LogTiterQuantSpec

Målet är att funktionen accepterar både `PSCustomObject` och `Hashtable`, och att den kontrollerar `QuantSpecRules`-property robust utan `.ContainsKey()`.

```

--- Modules/RuleEngine.ps1
+++ Modules/RuleEngine.ps1
@@
function Apply-LogTiterQuantSpec {
    param(
-        [Parameter(Mandatory)][hashtable]$RuleBank,
+        [Parameter(Mandatory)][object]$RuleBank,
        [Parameter(Mandatory)][string]$Assay,
        [Parameter(Mandatory)][string]$ControlCode,
        [Parameter(Mandatory)][string]$ObservedCall,
@@
-    if (-not $RuleBank.ContainsKey('QuantSpecRules')) { return $out }
-    $rules = @($RuleBank.QuantSpecRules)
+    $rules = @()
+    try { $rules = @($RuleBank.QuantSpecRules) } catch { $rules = @() }
+    if (-not $rules -or $rules.Count -eq 0) { return $out }

-    $ass = ($Assay + '')
+    $ass = (($Assay + '').Trim())
+    $cc = ($ControlCode + '').Trim()

```

Varför detta är rätt: - PowerShells parameterbindning och typkonvertering gör att strikt `[hashtable]` här är riskabelt. <sup>6</sup> - Att hämta `.QuantSpecRules` via `try/catch` gör att funktionen blir bakåtkompatibel även om property saknas.

## Patch för MISQ i sammanfattningen

Detta är en **rapportfix**: MISQ ska synas som Major i "Resultat Avvikelser" och "Inga avvikelser"-raden ska inte visas om MISQ finns.

```

--- Modules/RuleEngine.ps1
+++ Modules/RuleEngine.ps1
@@
# FN - Falskt negativ (Major Functional) - Mörkröd
$fnCount = 0
if ($sum.DeviationCounts.ContainsKey('FN')) { $fnCount =
$sum.DeviationCounts['FN'] }
if ($fnCount -gt 0) {
    Write-KV -Row $row -Key '  Falskt negativ (Major Functional)' -Value
$fnCount -Col 1 -Major
    $row++
}

+# MISQ - Misquantitation (Major Functional) - Mörkröd
+$misqCount = 0
+if ($sum.DeviationCounts.ContainsKey('MISQ')) { $misqCount =
+$sum.DeviationCounts['MISQ'] }
+if ($misqCount -gt 0) {
+    Write-KV -Row $row -Key '  Misquantitation (Major Functional)' -Value

```

```

$misqCount -Col 1 -Major
+   $row++
+}
+
# Minor Functional - Ljusröd
if ($sum -and $sum.MinorFunctionalError -ne $null -and
$sum.MinorFunctionalError -gt 0) {
    Write-KV -Row $row -Key '⚠ Minor Functional' -Value
$sum.MinorFunctionalError -Col 1 -Minor
    $row++
}

-# Visa om inga avvikelser (om OK = Total och inga FP/FN/Minor)
-if ($fpCount -eq 0 -and $fnCount -eq 0 -and ($sum.MinorFunctionalError -eq
$null -or $sum.MinorFunctionalError -eq 0)) {
+# Visa om inga avvikelser (om OK = Total och inga FP/FN/MISQ/Minor)
+if ($fpCount -eq 0 -and $fnCount -eq 0 -and $misqCount -eq 0 -and
($sum.MinorFunctionalError -eq $null -or $sum.MinorFunctionalError -eq 0)) {
    $ws.Cells.Item($row, 1).Value = '✓ Inga avvikelser hittades'
    ...
}

```

Detta ändrar inte sammanfattningen när MISQ inte finns (bakåtkompatibelt i praktiken), men gör MISQ synligt när det väl förekommer.

## Bakåtkompatibilitet

### Nuvarande status i din zip

- **Inte fullt bakåtkompatibelt i runtime** på grund av `[hashtable]$RuleBank` i `Apply-LogTiterQuantSpec` (risk för bindningsfel). <sup>1</sup>
- Detaljrapportens kolumner och färgning är i övrigt kompatibla.

### Efter rekommenderade patchar

- Om inga QuantSpecRules matchar: `Apply-LogTiterQuantSpec` returnerar "no change", vilket gör output identisk rad-för-rad jämfört med innan (förutom att du nu har kod som är redo). <sup>6</sup>
- Om MISQ förekommer: samma rader får `Error Code = MISQUANTITATION` och `Avvikelse = Major Functional` samt Major-färg (mörkröd/vit), vilket uppfyller dina krav. EPlus-stylingstrategin är korrekt. <sup>5</sup>

<sup>1</sup> <sup>6</sup> [about\\_Parameter\\_Binding - PowerShell | Microsoft Learn](https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_parameter_binding?view=powershell-7.5&utm_source=chatgpt.com)

[https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\\_parameter\\_binding?view=powershell-7.5&utm\\_source=chatgpt.com](https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_parameter_binding?view=powershell-7.5&utm_source=chatgpt.com)

<sup>2</sup> <sup>3</sup> <sup>4</sup> [System.Globalization.CultureInfo.InvariantCulture property - .NET | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/fundamentals/runtime-libraries/system-globalization-cultureinfo-invariantculture?utm_source=chatgpt.com)

[https://learn.microsoft.com/en-us/dotnet/fundamentals/runtime-libraries/system-globalization-cultureinfo-invariantculture?utm\\_source=chatgpt.com](https://learn.microsoft.com/en-us/dotnet/fundamentals/runtime-libraries/system-globalization-cultureinfo-invariantculture?utm_source=chatgpt.com)

5 Formatting and styling · EPPlusSoftware/EPPlus Wiki · GitHub

[https://github.com/EPPlusSoftware/EPPlus/wiki/Formatting-and-styling?utm\\_source=chatgpt.com](https://github.com/EPPlusSoftware/EPPlus/wiki/Formatting-and-styling?utm_source=chatgpt.com)