```
<#
.SYNOPSIS
    IPTCompile Metrics Dashboard — WPF/XAML Edition
    Visuell dashboard för tidsbesparingar och kvalitetsdata.

.DESCRIPTION
    Läser Audit_*.csv och *_audit_*.csv från audit-katalogen och presenterar
    data i en modern WPF-dashboard med diagram och KPI-kort.

.PARAMETER AuditDir
    Sökväg till audit-katalogen. Standard: .\audit

.PARAMETER ManualMinutes
    Uppskattad tid (minuter) för manuell rapportgenerering. Standard: 20

.EXAMPLE
    .\IPT_Dashboard_WPF.ps1
    .\IPT_Dashboard_WPF.ps1 -AuditDir "\\server\share\audit" -ManualMinutes 25
#>
param(
    [string]$AuditDir   = 'N:\QC\QC-1\IPT\Skiftspecifika dokument\PQC analyst\JES
    [int]$ManualMinutes = 20,
    [string]$ExportCsv  = 'N:\QC\QC-1\IPT\Skiftspecifika dokument\PQC analyst\JES
)

# ═══════════════════════════════════════════════════════════════════════════
# ASSEMBLIES
# ═══════════════════════════════════════════════════════════════════════════
Add-Type -AssemblyName PresentationFramework
Add-Type -AssemblyName PresentationCore
Add-Type -AssemblyName WindowsBase
Add-Type -AssemblyName System.Windows.Forms

# ═══════════════════════════════════════════════════════════════════════════
# DATA LOADING
# ═══════════════════════════════════════════════════════════════════════════
function Load-AuditData([string]$Dir) {
    $rows = [System.Collections.Generic.List[pscustomobject]]::new()

    if (-not (Test-Path -LiteralPath $Dir)) { return $rows }

    $files = @(Get-ChildItem -LiteralPath $Dir -Filter "Audit_*.csv" -File -Error
    $files += @(Get-ChildItem -LiteralPath $Dir -Filter "*_audit_*.csv" -File -Er
    $files = $files | Select-Object -Unique
```

```
foreach ($f in $files) {
    try {
        $csv = Import-Csv -LiteralPath $f.FullName -Encoding UTF8
        foreach ($r in $csv) {
            $obj = [pscustomobject]@{
                Timestamp  = ''
                Username   = ''
                LSP        = ''
                Assay      = ''
                TestCount  = 0
                Status     = 'OK'
                TotalMs    = 0
                TotalSec   = 0.0
                PhaseJson  = ''
                Violations = 0
                SignSealTest = ''
                SignWsSamm   = ''
                SignWsGransk = ''
            }
            if ($r.Timestamp)    { $obj.Timestamp = $r.Timestamp }
            elseif ($r.DatumTid) { $obj.Timestamp = $r.DatumTid }
            if ($r.Username)       { $obj.Username = $r.Username }
            elseif ($r.Användare)  { $obj.Username = $r.Användare }
            if ($r.LSP)    { $obj.LSP = $r.LSP }
            if ($r.Assay)  { $obj.Assay = $r.Assay }
            if ($r.Status) { $obj.Status = $r.Status }
            if ($r.PhaseJson)      { $obj.PhaseJson = $r.PhaseJson }
            elseif ($r.FasTider_json) { $obj.PhaseJson = $r.FasTider_json }
            try { $obj.TestCount = [int]($r.TestCount) } catch {}
            try { if (-not $obj.TestCount -and $r.AntalTester) { $obj.TestCou
            try { $obj.TotalMs = [int]($r.TotalMs) } catch {}
            try { if (-not $obj.TotalMs -and $r.TotalTid_ms) { $obj.TotalMs =
            try { $obj.TotalSec = [double]($r.TotalSec) } catch {}
            try { if (-not $obj.TotalSec -and $r.TotalTid_s) { $obj.TotalSec
            try { $obj.Violations = [int]($r.Violations) } catch {}
            if ($r.SignaturSkriven)  { $obj.SignSealTest = $r.SignaturSkriven
            if ($r.WsSammSkriven)    { $obj.SignWsSamm   = $r.WsSammSkriven }
            if ($r.WsGranskSkriven)  { $obj.SignWsGransk = $r.WsGranskSkriven
            if ($obj.TotalMs -gt 0 -and $obj.TotalSec -eq 0) {
                $obj.TotalSec = [math]::Round($obj.TotalMs / 1000, 1)
            }
            $rows.Add($obj)
        }
    } catch {}
}
return $rows
```

```
}

# ================================================================================
# XAML UI
# ================================================================================
[xml]$xaml = @'
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="IPTCompile — Metrics Dashboard"
        Width="1280" Height="820"
        MinWidth="900" MinHeight="600"
        WindowStartupLocation="CenterScreen"
        Background="#0F1923">
    <Window.Resources>
        <!-- Card style -->
        <Style x:Key="Card" TargetType="Border">
            <Setter Property="Background" Value="#1A2736"/>
            <Setter Property="CornerRadius" Value="12"/>
            <Setter Property="Padding" Value="20,16"/>
            <Setter Property="Margin" Value="6"/>
            <Setter Property="Effect">
                <Setter.Value>
                    <DropShadowEffect BlurRadius="16" ShadowDepth="2" Opacity="0.
                </Setter.Value>
            </Setter>
        </Style>
        <Style x:Key="KpiValue" TargetType="TextBlock">
            <Setter Property="FontSize" Value="32"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Foreground" Value="#FFFFFF"/>
            <Setter Property="Margin" Value="0,4,0,0"/>
        </Style>
        <Style x:Key="KpiLabel" TargetType="TextBlock">
            <Setter Property="FontSize" Value="12"/>
            <Setter Property="Foreground" Value="#7B8FA3"/>
            <Setter Property="TextWrapping" Value="Wrap"/>
        </Style>
        <Style x:Key="KpiUnit" TargetType="TextBlock">
            <Setter Property="FontSize" Value="13"/>
            <Setter Property="Foreground" Value="#4FC3F7"/>
            <Setter Property="Margin" Value="0,2,0,0"/>
        </Style>
        <Style x:Key="SectionTitle" TargetType="TextBlock">
            <Setter Property="FontSize" Value="15"/>
            <Setter Property="FontWeight" Value="SemiBold"/>
            <Setter Property="Foreground" Value="#B0BEC5"/>
            <Setter Property="Margin" Value="0,0,0,10"/>
```

```
    </Style>
</Window.Resources>

<Grid Margin="16">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <!-- HEADER -->
    <Border Grid.Row="0" Margin="0,0,0,12">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <StackPanel Grid.Column="0" Orientation="Horizontal" VerticalAlig
                <TextBlock Text="" FontSize="28" VerticalAlignment="Center"
                <TextBlock Text="IPTCompile" FontSize="26" FontWeight="Bold"
                <TextBlock Text=" Metrics Dashboard" FontSize="26" FontWeight
            </StackPanel>
            <StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlig
                <TextBlock x:Name="txtPeriod" FontSize="12" Foreground="#607D
                <Border Background="#263544" CornerRadius="6" Padding="10,5">
                    <TextBlock x:Name="txtCount" FontSize="12" Foreground="#8
                </Border>
            </StackPanel>
        </Grid>
    </Border>

    <!-- KPI ROW -->
    <UniformGrid Grid.Row="1" Columns="6" Margin="0,0,0,6">
        <!-- KPI 1: Total Reports -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Totalt rap
                <TextBlock x:Name="kpiReports" Style="{StaticResource KpiValu
                <TextBlock x:Name="kpiReportsSub" Style="{StaticResource KpiU
            </StackPanel>
        </Border>
        <!-- KPI 2: Avg Time -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Medeltid p
                <TextBlock x:Name="kpiAvgTime" Style="{StaticResource KpiValu
```

```
                <TextBlock x:Name="kpiAvgTimeSub" Style="{StaticResource KpiU
            </StackPanel>
        </Border>
        <!-- KPI 3: Speedup -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Hastighets
                <TextBlock x:Name="kpiSpeedup" Style="{StaticResource KpiValu
                <TextBlock x:Name="kpiSpeedupSub" Style="{StaticResource KpiU
            </StackPanel>
        </Border>
        <!-- KPI 4: Saved Hours -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Sparade ar
                <TextBlock x:Name="kpiSaved" Style="{StaticResource KpiValue}
                <TextBlock x:Name="kpiSavedSub" Style="{StaticResource KpiUni
            </StackPanel>
        </Border>
        <!-- KPI 5: Quality -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Felfria ra
                <TextBlock x:Name="kpiQuality" Style="{StaticResource KpiValu
                <TextBlock x:Name="kpiQualitySub" Style="{StaticResource KpiU
            </StackPanel>
        </Border>
        <!-- KPI 6: Signing -->
        <Border Style="{StaticResource Card}">
            <StackPanel>
                <TextBlock Style="{StaticResource KpiLabel}" Text="Signerings
                <TextBlock x:Name="kpiSigning" Style="{StaticResource KpiValu
                <TextBlock x:Name="kpiSigningSub" Style="{StaticResource KpiU
            </StackPanel>
        </Border>
    </UniformGrid>

    <!-- CHARTS ROW -->
    <Grid Grid.Row="2" Margin="0,6,0,6">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="2*"/>
            <ColumnDefinition Width="1.2*"/>
            <ColumnDefinition Width="1.2*"/>
        </Grid.ColumnDefinitions>

        <!-- Chart 1: Reports per day -->
        <Border Grid.Column="0" Style="{StaticResource Card}">
```

```
            <DockPanel>
                <TextBlock DockPanel.Dock="Top" Style="{StaticResource Sectio
                <Canvas x:Name="chartDaily" ClipToBounds="True"/>
            </DockPanel>
        </Border>

        <!-- Chart 2: Phase timing -->
        <Border Grid.Column="1" Style="{StaticResource Card}">
            <DockPanel>
                <TextBlock DockPanel.Dock="Top" Style="{StaticResource Sectio
                <Canvas x:Name="chartPhases" ClipToBounds="True"/>
            </DockPanel>
        </Border>

        <!-- Chart 3: Per user + per assay -->
        <Border Grid.Column="2" Style="{StaticResource Card}">
            <DockPanel>
                <TextBlock DockPanel.Dock="Top" Style="{StaticResource Sectio
                <Canvas x:Name="chartUsers" ClipToBounds="True" DockPanel.Doc
                <TextBlock Style="{StaticResource SectionTitle}" Text="Rappor
                <Canvas x:Name="chartAssays" ClipToBounds="True"/>
            </DockPanel>
        </Border>
    </Grid>

    <!-- FOOTER -->
    <Border Grid.Row="3" Margin="0,4,0,0">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <TextBlock x:Name="txtFooter" Grid.Column="0" FontSize="11" Foreg
            <Button x:Name="btnExport" Grid.Column="1" Content="📥 Exportera
                    Background="#263544" Foreground="#90A4AE" BorderThickness
                    Padding="16,6" FontSize="12" Cursor="Hand">
                <Button.Template>
                    <ControlTemplate TargetType="Button">
                        <Border x:Name="bd" Background="{TemplateBinding Back
                            <ContentPresenter HorizontalAlignment="Center" Ve
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter TargetName="bd" Property="Background"
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
```

```
                    </Button.Template>
                </Button>
            </Grid>
        </Border>
    </Grid>
</Window>
'@

# ================================================================================
# CREATE WINDOW
# ================================================================================
$reader = [System.Xml.XmlNodeReader]::new($xaml)
$window = [Windows.Markup.XamlReader]::Load($reader)

# Named element references
$names = @(
    'txtPeriod','txtCount','txtFooter',
    'kpiReports','kpiReportsSub',
    'kpiAvgTime','kpiAvgTimeSub',
    'kpiSpeedup','kpiSpeedupSub',
    'kpiSaved','kpiSavedSub',
    'kpiQuality','kpiQualitySub',
    'kpiSigning','kpiSigningSub',
    'chartDaily','chartPhases','chartUsers','chartAssays',
    'btnExport'
)
$ui = @{}
foreach ($n in $names) { $ui[$n] = $window.FindName($n) }

# ================================================================================
# CHART HELPERS
# ================================================================================
function New-Rect([double]$X, [double]$Y, [double]$W, [double]$H, [string]$Fill,
    $r = New-Object Windows.Shapes.Rectangle
    $r.Width = $W; $r.Height = $H
    $r.RadiusX = $Radius; $r.RadiusY = $Radius
    $r.Fill = [Windows.Media.BrushConverter]::new().ConvertFrom($Fill)
    [Windows.Controls.Canvas]::SetLeft($r, $X)
    [Windows.Controls.Canvas]::SetTop($r, $Y)
    return $r
}

function New-Label([double]$X, [double]$Y, [string]$Text, [double]$Size = 11, [st
    $tb = New-Object Windows.Controls.TextBlock
    $tb.Text = $Text
    $tb.FontSize = $Size
    $tb.Foreground = [Windows.Media.BrushConverter]::new().ConvertFrom($Color)
```

```
    if ($Align -eq 'Right') { $tb.TextAlignment = 'Right' }
    [Windows.Controls.Canvas]::SetLeft($tb, $X)
    [Windows.Controls.Canvas]::SetTop($tb, $Y)
    return $tb
}

function Draw-HorizontalBars([Windows.Controls.Canvas]$Canvas, [hashtable[]]$Data
    # Data = @( @{Label='x'; Value=5}, ... )  sorted desc
    $Canvas.Children.Clear()
    if (-not $Data -or $Data.Count -eq 0) { return }

    $canvas.Dispatcher.Invoke([Action]{
        $cw = $Canvas.ActualWidth; $ch = $Canvas.ActualHeight
        if ($cw -le 0) { $cw = 260 }
        if ($ch -le 0) { $ch = 200 }

        $barH = [math]::Min(26, [math]::Max(14, ($ch - 10) / [math]::Max($Data.Co
        $gap = 4
        $labelW = 110
        $maxVal = ($Data | ForEach-Object { $_.Value } | Measure-Object -Maximum)
        if ($maxVal -le 0) { $maxVal = 1 }
        $barArea = $cw - $labelW - 50

        for ($i = 0; $i -lt $Data.Count; $i++) {
            $d = $Data[$i]
            $y = $i * ($barH + $gap)
            if ($y + $barH -gt $ch) { break }

            # Label
            $lbl = $d.Label
            if ($lbl.Length -gt 16) { $lbl = $lbl.Substring(0, 14) + '…' }
            $Canvas.Children.Add((New-Label 0 ($y + 1) $lbl 10.5 '#90A4AE')) | Ou

            # Bar
            $bw = [math]::Max(3, ($d.Value / $maxVal) * $barArea)
            $Canvas.Children.Add((New-Rect $labelW $y $bw $barH $BarColor 3)) | O

            # Value label
            $Canvas.Children.Add((New-Label ($labelW + $bw + 6) ($y + 1) "$($d.Va
        }
    }, [Windows.Threading.DispatcherPriority]::Loaded)
}

function Draw-VerticalBars([Windows.Controls.Canvas]$Canvas, [hashtable[]]$Data,
    $Canvas.Children.Clear()
    if (-not $Data -or $Data.Count -eq 0) { return }
```

```
$Canvas.Dispatcher.Invoke([Action]{
    $cw = $Canvas.ActualWidth; $ch = $Canvas.ActualHeight
    if ($cw -le 0) { $cw = 500 }
    if ($ch -le 0) { $ch = 280 }

    $maxVal = ($Data | ForEach-Object { $_.Value } | Measure-Object -Maximum)
    if ($maxVal -le 0) { $maxVal = 1 }

    $labelH = 40
    $chartH = $ch - $labelH - 10
    $barW = [math]::Min(32, [math]::Max(6, ($cw - 20) / [math]::Max($Data.Cou
    $gap = [math]::Max(2, ($cw - ($barW * $Data.Count)) / [math]::Max($Data.C

    # Grid lines
    for ($g = 0; $g -le 4; $g++) {
        $gy = 5 + ($chartH * (1 - $g / 4))
        $line = New-Object Windows.Shapes.Line
        $line.X1 = 0; $line.X2 = $cw
        $line.Y1 = $gy; $line.Y2 = $gy
        $line.Stroke = [Windows.Media.BrushConverter]::new().ConvertFrom('#1E
        $line.StrokeThickness = 1
        $Canvas.Children.Add($line) | Out-Null

        $gVal = [math]::Round($maxVal * $g / 4)
        if ($g -gt 0) {
            $Canvas.Children.Add((New-Label ($cw - 30) ($gy - 14) "$gVal" 9 '
        }
    }

    # Bars
    $colors = @('#4FC3F7','#66BB6A','#FFB74D','#EF5350','#AB47BC','#26C6DA','
                '#9CCC65','#FF7043','#5C6BC0','#29B6F6','#FFCA28')

    for ($i = 0; $i -lt $Data.Count; $i++) {
        $d = $Data[$i]
        $x = $gap + $i * ($barW + $gap)
        $bh = [math]::Max(2, ($d.Value / $maxVal) * $chartH)
        $y = 5 + $chartH - $bh

        $clr = $colors[$i % $colors.Count]
        $Canvas.Children.Add((New-Rect $x $y $barW $bh $clr 2)) | Out-Null

        # X label (rotated date)
        $lbl = $d.Label
        if ($lbl.Length -gt 10) { $lbl = $lbl.Substring(5) }  # Strip year
        $tb = New-Object Windows.Controls.TextBlock
        $tb.Text = $lbl
```

```
            $tb.FontSize = 9
            $tb.Foreground = [Windows.Media.BrushConverter]::new().ConvertFrom('#
            $tb.RenderTransform = New-Object Windows.Media.RotateTransform(45)
            [Windows.Controls.Canvas]::SetLeft($tb, $x - 2)
            [Windows.Controls.Canvas]::SetTop($tb, ($ch - $labelH + 4))
            $Canvas.Children.Add($tb) | Out-Null
        }
    }, [Windows.Threading.DispatcherPriority]::Loaded)
}

# ================================================================================
# POPULATE DATA
# ================================================================================
$allData = Load-AuditData $AuditDir
$dataForExport = $allData

if ($allData.Count -eq 0) {
    # Show empty state
    $ui['kpiReports'].Text = '0'
    $ui['kpiReportsSub'].Text = 'Ingen audit-data hittades'
    $ui['txtFooter'].Text = "Sökväg: $AuditDir — Kör IPTCompile för att börja sam
    $ui['kpiAvgTime'].Text = '—'
    $ui['kpiSpeedup'].Text = '—'
    $ui['kpiSaved'].Text = '—'
    $ui['kpiQuality'].Text = '—'
    $ui['kpiSigning'].Text = '—'
} else {
    # --- Period ---
    $dates = @()
    foreach ($r in $allData) { try { $dates += [datetime]::Parse($r.Timestamp) }
    if ($dates.Count -gt 0) {
        $first = ($dates | Sort-Object | Select-Object -First 1).ToString('yyyy-M
        $last  = ($dates | Sort-Object | Select-Object -Last 1).ToString('yyyy-MM
        $ui['txtPeriod'].Text = "$first  →  $last"
    }
    $ui['txtCount'].Text = "$($allData.Count) rapporter laddade"

    # --- KPI: Reports ---
    $totalTests = ($allData | ForEach-Object { $_.TestCount } | Measure-Object -S
    $users = @($allData | ForEach-Object { $_.Username } | Where-Object { $_ } |
    $ui['kpiReports'].Text = "$($allData.Count)"
    $ui['kpiReportsSub'].Text = "$totalTests tester  ·  $($users.Count) användare

    # --- KPI: Avg Time ---
    $timedRows = @($allData | Where-Object { $_.TotalMs -gt 0 })
    $avgSec = 0; $medianSec = 0; $minSec = 0; $maxSec = 0
    if ($timedRows.Count -gt 0) {
```

```
    $secs = @($timedRows | ForEach-Object { $_.TotalSec })
    $stats = $secs | Measure-Object -Average -Minimum -Maximum
    $avgSec = $stats.Average
    $minSec = $stats.Minimum
    $maxSec = $stats.Maximum
    $sorted = @($secs | Sort-Object)
    $medianSec = $sorted[[math]::Floor($sorted.Count / 2)]
    $ui['kpiAvgTime'].Text = "{0:N1}s" -f $avgSec
    $ui['kpiAvgTimeSub'].Text = "median {0:N1}s  ·  min {1:N1}s  ·  max {2:N1
} else {
    $ui['kpiAvgTime'].Text = '—'
    $ui['kpiAvgTimeSub'].Text = 'Inga tidsmätningar ännu'
}

# --- KPI: Speedup ---
$manualSec = $ManualMinutes * 60
if ($avgSec -gt 0) {
    $speedup = [math]::Round($manualSec / $avgSec)
    $ui['kpiSpeedup'].Text = "${speedup}x"
    $savedPerReport = [math]::Round(($manualSec - $avgSec) / 60, 1)
    $ui['kpiSpeedupSub'].Text = "$savedPerReport min sparas per rapport"
} else {
    $ui['kpiSpeedup'].Text = '—'
    $ui['kpiSpeedupSub'].Text = "vs $ManualMinutes min manuellt"
}

# --- KPI: Saved Hours ---
if ($avgSec -gt 0) {
    $totalSavedHrs = [math]::Round(($manualSec - $avgSec) * $allData.Count /
    $ui['kpiSaved'].Text = "$totalSavedHrs h"
    $perMonth = [math]::Round($totalSavedHrs / [math]::Max(1, ($dates | Sort-
    $ui['kpiSavedSub'].Text = "~$perMonth h/månad"
} else {
    $ui['kpiSaved'].Text = '—'
    $ui['kpiSavedSub'].Text = ''
}

# --- KPI: Quality ---
$okCount = @($allData | Where-Object { $_.Status -eq 'OK' }).Count
if ($allData.Count -gt 0) {
    $okPct = [math]::Round(($okCount / $allData.Count) * 100, 1)
    $ui['kpiQuality'].Text = "$okPct%"
    $warnCount = $allData.Count - $okCount
    $ui['kpiQualitySub'].Text = "$okCount OK  ·  $warnCount med avvikelser"
} else {
    $ui['kpiQuality'].Text = '—'
    $ui['kpiQualitySub'].Text = ''
```

```
}

# --- KPI: Signing ---
$signedRows = @($allData | Where-Object {
    ($_.SignSealTest -and $_.SignSealTest -imatch '^Ja') -or
    ($_.SignWsSamm   -and $_.SignWsSamm   -imatch '^Ja') -or
    ($_.SignWsGransk -and $_.SignWsGransk -imatch '^Ja')
})
$unsignedRows = @($allData | Where-Object {
    ($_.SignSealTest -ieq 'Nej') -and ($_.SignWsSamm -ieq 'Nej') -and ($_.Sig
})
$hasSignData = @($allData | Where-Object { $_.SignSealTest -or $_.SignWsSamm
if ($hasSignData.Count -gt 0) {
    $signPct = [math]::Round(($signedRows.Count / $hasSignData.Count) * 100,
    $verifiedCount = @($allData | Where-Object {
        ($_.SignSealTest -imatch 'verifierad') -or
        ($_.SignWsSamm   -imatch 'verifierad') -or
        ($_.SignWsGransk -imatch 'verifierad')
    }).Count
    if ($signPct -ge 95) {
        $ui['kpiSigning'].Foreground = [Windows.Media.BrushConverter]::new().
    } elseif ($signPct -ge 80) {
        $ui['kpiSigning'].Foreground = [Windows.Media.BrushConverter]::new().
    } else {
        $ui['kpiSigning'].Foreground = [Windows.Media.BrushConverter]::new().
    }
    $ui['kpiSigning'].Text = "$signPct%"
    $subParts = @("$($signedRows.Count)/$($hasSignData.Count) signerade")
    if ($verifiedCount -gt 0) { $subParts += "$verifiedCount GDP-verifierade"
    if ($unsignedRows.Count -gt 0) { $subParts += "$($unsignedRows.Count) osi
    $ui['kpiSigningSub'].Text = $subParts -join '  ·  '
} else {
    $ui['kpiSigning'].Text = '—'
    $ui['kpiSigningSub'].Text = 'Signeringsdata saknas (äldre format)'
}

# --- Chart: Daily ---
$byDay = @{}
foreach ($r in $allData) {
    try {
        $d = [datetime]::Parse($r.Timestamp).ToString('yyyy-MM-dd')
        if (-not $byDay.ContainsKey($d)) { $byDay[$d] = 0 }
        $byDay[$d]++
    } catch {}
}
$dailyData = @($byDay.Keys | Sort-Object | Select-Object -Last 21 | ForEach-O
```

```
    # --- Chart: Phases ---
    $phaseTotals = [ordered]@{}
    $phaseCount = 0
    foreach ($r in $timedRows) {
        if (-not $r.PhaseJson) { continue }
        try {
            $phases = $r.PhaseJson | ConvertFrom-Json
            $phaseCount++
            foreach ($prop in $phases.PSObject.Properties) {
                if (-not $phaseTotals.Contains($prop.Name)) { $phaseTotals[$prop.
                $phaseTotals[$prop.Name] += [double]$prop.Value
            }
        } catch {}
    }
    $phaseData = @()
    if ($phaseCount -gt 0) {
        $phaseData = @($phaseTotals.Keys |
            Sort-Object @{ Expression = { $phaseTotals[$_] } } -Descending |
            ForEach-Object { @{ Label = $_; Value = [math]::Round($phaseTotals[$_
    }

    # --- Chart: Users ---
    $byUser = $allData | Group-Object Username | Sort-Object Count -Descending
    $userData = @($byUser | Select-Object -First 8 | ForEach-Object {
        $name = if ($_.Name) { $_.Name } else { '(okänd)' }
        @{ Label = $name; Value = $_.Count }
    })

    # --- Chart: Assays ---
    $byAssay = $allData | Where-Object { $_.Assay } | Group-Object Assay | Sort-O
    $assayData = @($byAssay | Select-Object -First 8 | ForEach-Object { @{ Label

    # --- Footer ---
    $ui['txtFooter'].Text = "Datakälla: $AuditDir  |  Manuell referenstid: $Manua
}

# ================================================================================
# RENDER CHARTS ON LOADED
# ================================================================================
$window.Add_ContentRendered({
    if ($dailyData -and $dailyData.Count -gt 0) {
        Draw-VerticalBars $ui['chartDaily'] $dailyData '#4FC3F7'
    }
    if ($phaseData -and $phaseData.Count -gt 0) {
        Draw-HorizontalBars $ui['chartPhases'] $phaseData '#66BB6A'
    }
    if ($userData -and $userData.Count -gt 0) {
```

```
        Draw-HorizontalBars $ui['chartUsers'] $userData '#FFB74D'
    }
    if ($assayData -and $assayData.Count -gt 0) {
        Draw-HorizontalBars $ui['chartAssays'] $assayData '#AB47BC'
    }
}.GetNewClosure())

# ================================================================================
# EXPORT BUTTON
# ================================================================================
$ui['btnExport'].Add_Click({
    $dlg = New-Object Microsoft.Win32.SaveFileDialog
    $dlg.Filter = 'CSV-filer (*.csv)|*.csv'
    $dlg.FileName = "IPTCompile_Metrics_$((Get-Date).ToString('yyyyMMdd')).csv"
    if ($dlg.ShowDialog()) {
        try {
            $exportRows = foreach ($r in $dataForExport) {
                [pscustomobject]@{
                    Timestamp  = $r.Timestamp
                    Username   = $r.Username
                    LSP        = $r.LSP
                    Assay      = $r.Assay
                    TestCount  = $r.TestCount
                    Status     = $r.Status
                    TotalSec   = $r.TotalSec
                    Violations = $r.Violations
                    SignSealTest = $r.SignSealTest
                    SignWsSamm   = $r.SignWsSamm
                    SignWsGransk = $r.SignWsGransk
                }
            }
            $exportRows | Export-Csv -Path $dlg.FileName -NoTypeInformation -Enco
            [System.Windows.MessageBox]::Show("Exporterat $($exportRows.Count) ra
        } catch {
            [System.Windows.MessageBox]::Show("Kunde inte exportera: $($_.Excepti
        }
    }
}.GetNewClosure())

# ================================================================================
# RUN
# ================================================================================
$window.ShowDialog() | Out-Null
```