```
<#
.SYNOPSIS
    WpfGui.ps1 — WPF/XAML GUI for IPTCompile
    Replaces the WinForms GUI region in Main.ps1 with a modern WPF interface.

.DESCRIPTION
    Drop-in replacement: exposes identical variable names ($form, $txtLSP, $btnSc
    $clbCsv, $clbNeg, $clbPos, $clbLsp, $outputBox, $btnBuild, $grpSign, $txtSign
    $chkWriteSign, $chkOverwriteSign, $slCount, $slWork, $pbWork, $slBatchLink,
    $btnMove3, $btnMove4, $menuStrip, etc.) via thin adapter objects so that ALL
    existing event-handler code in Main.ps1 works without modification.

    Usage: Replace the #region GUI block in Main.ps1 with:
        . (Join-Path $modulesRoot 'WpfGui.ps1')

.NOTES
    Requires: PresentationFramework, PresentationCore, WindowsBase (built into .N
    Also loads System.Windows.Forms for OpenFileDialog and MessageBox compatibili
#>

# ================================================================================
#region ASSEMBLIES
# ================================================================================
Add-Type -AssemblyName PresentationFramework
Add-Type -AssemblyName PresentationCore
Add-Type -AssemblyName WindowsBase
# Keep WinForms loaded — needed for OpenFileDialog, MessageBox, and clipboard
try { Add-Type -AssemblyName System.Windows.Forms -ErrorAction SilentlyContinue }
try { Add-Type -AssemblyName System.Drawing -ErrorAction SilentlyContinue } catch
#endregion ASSEMBLIES

# ================================================================================
#region ADAPTER CLASSES
# ================================================================================

<#
  CheckedListBoxAdapter — wraps a WPF ListView to expose the same API as
  System.Windows.Forms.CheckedListBox so that all callers in Main.ps1
  (Add-CLBItems, Get-CheckedFilePath, Get-AllCheckedFilePaths, Update-StatusBar,
  Update-BuildEnabled, onExclusive, onCsvMax2) work unchanged.
#>
Add-Type -Language CSharp -ReferencedAssemblies @(
    'PresentationFramework','PresentationCore','WindowsBase','System','System.Xam
) -TypeDefinition @'
```

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.ComponentModel;

namespace IPT.Wpf
{
    public class CheckedItem : INotifyPropertyChanged
    {
        private bool _isChecked;
        private object _item;
        private string _displayText;

        public bool IsChecked
        {
            get { return _isChecked; }
            set { _isChecked = value; OnPropertyChanged("IsChecked"); }
        }
        public object Item { get { return _item; } set { _item = value; } }
        public string DisplayText { get { return _displayText; } set { _displayTe

        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged(string name)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(name));
        }

        public override string ToString() { return _displayText ?? (_item != null
    }

    public class CheckedItemCollection
    {
        private ListView _lv;
        public CheckedItemCollection(ListView lv) { _lv = lv; }

        public int Count
        {
            get { return _lv.Items.Count; }
        }

        public void Clear()
        {
            _lv.Items.Clear();
        }
```

```
    public int Add(object item, bool isChecked)
    {
        string display = "";
        if (item != null)
        {
            var prop = item.GetType().GetProperty("Name");
            if (prop != null) display = (string)prop.GetValue(item, null);
            else display = item.ToString();
        }
        var ci = new CheckedItem { Item = item, IsChecked = isChecked, Displa
        return _lv.Items.Add(ci);
    }

    public object this[int index]
    {
        get
        {
            var ci = _lv.Items[index] as CheckedItem;
            return ci != null ? ci.Item : _lv.Items[index];
        }
    }

    public void AddRange(object[] items)
    {
        foreach (var item in items)
        {
            if (item is CheckedItem)
                _lv.Items.Add(item);
            else
                Add(item, false);
        }
    }
}

public class CheckedItemsList
{
    private ListView _lv;
    public CheckedItemsList(ListView lv) { _lv = lv; }

    public int Count
    {
        get
        {
            int n = 0;
            foreach (var item in _lv.Items)
            {
```

```
                    var ci = item as CheckedItem;
                    if (ci != null && ci.IsChecked) n++;
                }
                return n;
            }
        }

        public IEnumerator GetEnumerator()
        {
            var list = new List<object>();
            foreach (var item in _lv.Items)
            {
                var ci = item as CheckedItem;
                if (ci != null && ci.IsChecked) list.Add(ci.Item);
            }
            return list.GetEnumerator();
        }
    }
}
'@ -ErrorAction SilentlyContinue

# PowerShell adapter class
function New-CheckedListBoxAdapter {
    param([System.Windows.Controls.ListView]$ListView)

    $adapter = [pscustomobject]@{
        _lv            = $ListView
        Items          = New-Object IPT.Wpf.CheckedItemCollection($ListView)
        CheckedItems   = New-Object IPT.Wpf.CheckedItemsList($ListView)
        CheckOnClick   = $true
        DisplayMember  = 'Name'
        IntegralHeight = $false
        Dock           = 'Fill'
        Height         = 70
        Margin         = $null
        _itemCheckHandlers = [System.Collections.Generic.List[scriptblock]]::new(
    }

    # Adapter methods
    $adapter | Add-Member -MemberType ScriptMethod -Name 'GetItemChecked' -Value
        param([int]$index)
        if ($index -lt 0 -or $index -ge $this._lv.Items.Count) { return $false }
        $ci = $this._lv.Items[$index]
        if ($ci -is [IPT.Wpf.CheckedItem]) { return $ci.IsChecked }
        return $false
    }
```

```
    $adapter | Add-Member -MemberType ScriptMethod -Name 'SetItemChecked' -Value {
        param([int]$index, [bool]$checked)
        if ($index -lt 0 -or $index -ge $this._lv.Items.Count) { return }
        $ci = $this._lv.Items[$index]
        if ($ci -is [IPT.Wpf.CheckedItem]) { $ci.IsChecked = $checked }
    }

    $adapter | Add-Member -MemberType ScriptMethod -Name 'BeginUpdate' -Value { }
    $adapter | Add-Member -MemberType ScriptMethod -Name 'EndUpdate' -Value { }

    $adapter | Add-Member -MemberType ScriptMethod -Name 'add_ItemCheck' -Value {
        param([scriptblock]$handler)
        $this._itemCheckHandlers.Add($handler)
    }

    $adapter | Add-Member -MemberType ScriptMethod -Name 'BeginInvoke' -Value {
        param([Action]$action)
        $this._lv.Dispatcher.BeginInvoke($action)
    }

    return $adapter
}

#endregion ADAPTER CLASSES

# ============================================================================
#region COLOR PALETTE
# ============================================================================
$script:WpfColors = @{
    # Dark theme
    DarkBg         = '#0F1923'
    DarkCard       = '#1A2736'
    DarkCardHover  = '#223344'
    DarkBorder     = '#2A3F54'
    DarkText       = '#E0E6ED'
    DarkTextDim    = '#7B8FA3'
    DarkAccent     = '#4A9BD9'
    DarkAccentHov  = '#5AABE9'
    DarkHeader     = '#1B3A5C'
    DarkInput      = '#152231'
    DarkInputBrd   = '#2A4562'
    DarkLogBg      = '#101D2A'
    DarkOk         = '#66BB6A'
    DarkWarn       = '#FFB74D'
    DarkError      = '#EF5350'

    # Light theme
```

```
    LightBg        = '#F5F7FA'
    LightCard      = '#FFFFFF'
    LightBorder    = '#D5DDE5'
    LightText      = '#1A2530'
    LightTextDim   = '#5A6B7A'
    LightAccent    = '#2678B2'
    LightAccentHov = '#3088C2'
    LightHeader    = '#4682B4'
    LightInput     = '#FFFFFF'
    LightInputBrd  = '#C8D4DE'
    LightLogBg     = '#FFFFFF'
}
#endregion COLOR PALETTE

# ================================================================================
#region XAML DEFINITION
# ================================================================================
$wpfXaml = @'
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="WINDOW_TITLE_PLACEHOLDER"
        Width="1020" Height="960"
        MinWidth="900" MinHeight="750"
        WindowStartupLocation="CenterScreen"
        Background="#0F1923"
        FontFamily="Segoe UI"
        FontSize="13">
    <Window.Resources>
        <!-- BRUSHES -->
        <SolidColorBrush x:Key="BgBrush"       Color="#0F1923"/>
        <SolidColorBrush x:Key="CardBrush"      Color="#1A2736"/>
        <SolidColorBrush x:Key="CardHoverBrush" Color="#223344"/>
        <SolidColorBrush x:Key="BorderBrush"    Color="#2A3F54"/>
        <SolidColorBrush x:Key="TextBrush"      Color="#E0E6ED"/>
        <SolidColorBrush x:Key="TextDimBrush"   Color="#7B8FA3"/>
        <SolidColorBrush x:Key="AccentBrush"    Color="#4A9BD9"/>
        <SolidColorBrush x:Key="AccentHovBrush" Color="#5AABE9"/>
        <SolidColorBrush x:Key="HeaderBrush"    Color="#1B3A5C"/>
        <SolidColorBrush x:Key="InputBrush"     Color="#152231"/>
        <SolidColorBrush x:Key="InputBrdBrush"  Color="#2A4562"/>
        <SolidColorBrush x:Key="LogBgBrush"     Color="#101D2A"/>
        <SolidColorBrush x:Key="OkBrush"        Color="#66BB6A"/>
        <SolidColorBrush x:Key="WarnBrush"      Color="#FFB74D"/>
        <SolidColorBrush x:Key="ErrorBrush"     Color="#EF5350"/>

        <!-- TEXTBOX STYLE -->
        <Style x:Key="ModernTextBox" TargetType="TextBox">
```

```
    <Setter Property="Background" Value="{StaticResource InputBrush}"/>
    <Setter Property="Foreground" Value="{StaticResource TextBrush}"/>
    <Setter Property="BorderBrush" Value="{StaticResource InputBrdBrush}"
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="Padding" Value="8,6"/>
    <Setter Property="FontSize" Value="13"/>
    <Setter Property="CaretBrush" Value="{StaticResource AccentBrush}"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="TextBox">
                <Border x:Name="bd" Background="{TemplateBinding Backgrou
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness
                        CornerRadius="6" Padding="{TemplateBinding Paddin
                    <ScrollViewer x:Name="PART_ContentHost" Focusable="fa
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsFocused" Value="True">
                        <Setter TargetName="bd" Property="BorderBrush" Va
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="bd" Property="BorderBrush" Va
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- PRIMARY BUTTON -->
<Style x:Key="PrimaryButton" TargetType="Button">
    <Setter Property="Background" Value="{StaticResource AccentBrush}"/>
    <Setter Property="Foreground" Value="White"/>
    <Setter Property="FontWeight" Value="SemiBold"/>
    <Setter Property="FontSize" Value="13"/>
    <Setter Property="Padding" Value="16,8"/>
    <Setter Property="Cursor" Value="Hand"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border x:Name="bd" Background="{TemplateBinding Backgrou
                        CornerRadius="6" Padding="{TemplateBinding Paddin
                    <ContentPresenter HorizontalAlignment="Center" Vertic
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsMouseOver" Value="True">
```

```
                        <Setter TargetName="bd" Property="Background" Val
                    </Trigger>
                    <Trigger Property="IsEnabled" Value="False">
                        <Setter TargetName="bd" Property="Background" Val
                        <Setter Property="Foreground" Value="#556677"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- SECONDARY BUTTON -->
<Style x:Key="SecondaryButton" TargetType="Button">
    <Setter Property="Background" Value="#1E3044"/>
    <Setter Property="Foreground" Value="{StaticResource TextDimBrush}"/>
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Padding" Value="12,6"/>
    <Setter Property="Cursor" Value="Hand"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="Button">
                <Border x:Name="bd" Background="{TemplateBinding Backgrou
                        CornerRadius="5" Padding="{TemplateBinding Paddin
                    <ContentPresenter HorizontalAlignment="Center" Vertic
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="bd" Property="Background" Val
                    </Trigger>
                    <Trigger Property="IsEnabled" Value="False">
                        <Setter TargetName="bd" Property="Background" Val
                        <Setter Property="Foreground" Value="#3A4A5A"/>
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<!-- CARD BORDER -->
<Style x:Key="CardBorder" TargetType="Border">
    <Setter Property="Background" Value="{StaticResource CardBrush}"/>
    <Setter Property="CornerRadius" Value="10"/>
    <Setter Property="Padding" Value="16,12"/>
    <Setter Property="Margin" Value="0,4"/>
```

```
        <Setter Property="BorderBrush" Value="{StaticResource BorderBrush}"/>
        <Setter Property="BorderThickness" Value="1"/>
    </Style>

    <!-- CHECKBOX STYLE -->
    <Style x:Key="ModernCheckBox" TargetType="CheckBox">
        <Setter Property="Foreground" Value="{StaticResource TextBrush}"/>
        <Setter Property="FontSize" Value="12"/>
        <Setter Property="VerticalContentAlignment" Value="Center"/>
        <Setter Property="Margin" Value="0,2"/>
    </Style>

    <!-- LISTVIEW ITEM (for file lists with checkboxes) -->
    <Style x:Key="FileListView" TargetType="ListView">
        <Setter Property="Background" Value="{StaticResource InputBrush}"/>
        <Setter Property="Foreground" Value="{StaticResource TextBrush}"/>
        <Setter Property="BorderBrush" Value="{StaticResource InputBrdBrush}"
        <Setter Property="BorderThickness" Value="1"/>
        <Setter Property="FontSize" Value="11.5"/>
        <Setter Property="Padding" Value="2"/>
    </Style>

    <!-- MENU STYLE -->
    <Style x:Key="TopMenu" TargetType="Menu">
        <Setter Property="Background" Value="#14222F"/>
        <Setter Property="Foreground" Value="{StaticResource TextBrush}"/>
        <Setter Property="FontSize" Value="12.5"/>
        <Setter Property="Padding" Value="6,4"/>
    </Style>
</Window.Resources>

<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>   <!-- Menu -->
        <RowDefinition Height="Auto"/>   <!-- Header -->
        <RowDefinition Height="*"/>      <!-- Content -->
        <RowDefinition Height="Auto"/>   <!-- Status bar -->
    </Grid.RowDefinitions>

    <!-- ===== MENU BAR ===== -->
    <Menu x:Name="menuStrip" Grid.Row="0" Style="{StaticResource TopMenu}">
        <MenuItem x:Name="miArkiv" Header="🗂️ Arkiv">
            <MenuItem x:Name="miNew" Header="🆕 Nytt"/>
            <MenuItem x:Name="miOpenRecent" Header="📂 Öppna senaste rapport"
            <Separator/>
            <MenuItem x:Name="miExit" Header="❌ Avsluta"/>
        </MenuItem>
```

```
        <MenuItem x:Name="miVerktyg" Header="🛠️ Verktyg">
            <MenuItem x:Name="miScript1" Header="📜 Kontrollprovsfil-skript"/
            <MenuItem x:Name="miScript2" Header="📜 Aktivera Kontrollprovsfil
            <MenuItem x:Name="miScript3" Header="📅 Ändra datum på filer"/>
            <MenuItem x:Name="miScript4" Header="📦 AutoMappscript Control"/>
            <MenuItem x:Name="miScript5" Header="📄 AutoMappscript Dashboard"
            <MenuItem x:Name="miToggleSign" Header="✅ Aktivera Seal Test-sig
        </MenuItem>
        <MenuItem x:Name="miGenvagar" Header="🔗 Genvägar"/>
        <MenuItem x:Name="miSettings" Header="⚙️ Inställningar">
            <MenuItem x:Name="miTheme" Header="🎨 Tema">
                <MenuItem x:Name="miLightTheme" Header="☀️ Ljust"/>
                <MenuItem x:Name="miDarkTheme" Header="🌙 Mörkt (standard)"/>
            </MenuItem>
        </MenuItem>
        <MenuItem x:Name="miHelp" Header="📖 Instruktioner">
            <MenuItem x:Name="miShowInstr" Header="📖 Visa instruktioner"/>
            <MenuItem x:Name="miFAQ" Header="❓ FAQ"/>
            <MenuItem x:Name="miHelpDlg" Header="🆘 Hjälp"/>
        </MenuItem>
        <MenuItem x:Name="miAbout" Header="ℹ️ Om">
            <MenuItem x:Name="miOm" Header="ℹ️ Om det här verktyget"/>
        </MenuItem>
    </Menu>

    <!-- ═══ HEADER ═══ -->
    <Border Grid.Row="1" Background="{StaticResource HeaderBrush}" Padding="1
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <!-- Logo placeholder -->
            <Border x:Name="picLogoContainer" Grid.Column="0" Width="52" Heig
                    CornerRadius="8" Background="#2A5580" Margin="0,0,14,0"
                    VerticalAlignment="Center">
                <TextBlock Text="IPT" FontSize="16" FontWeight="Bold" Foregro
                           HorizontalAlignment="Center" VerticalAlignment="Ce
            </Border>
            <StackPanel Grid.Column="1" VerticalAlignment="Center">
                <TextBlock x:Name="lblTitle" FontSize="17" FontWeight="SemiBo
                    <Run x:Name="titleRun" Text="IPTCompile"/>
                    <Run x:Name="titleVersionRun" Text="" Foreground="#8BB8D9
                </TextBlock>
                <TextBlock x:Name="lblUpdate" Text="IPT Rapport-generator — W
                           FontSize="11" Foreground="#8BB8D9" Margin="0,2,0,0
                <TextBlock x:Name="lblExtra" Text="CSV för Original + Resampl
```

```
                    FontSize="10" Foreground="#7AAAC8" Margin="0,1,0,0
        </StackPanel>
    </Grid>
</Border>

<!-- ===== MAIN CONTENT ===== -->
<ScrollViewer Grid.Row="2" VerticalScrollBarVisibility="Auto" Padding="14
    <StackPanel>

        <!-- Search row -->
        <Border Style="{StaticResource CardBorder}">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <TextBlock Grid.Column="0" Text="LSP:" Foreground="{Stati
                           VerticalAlignment="Center" Margin="0,0,10,0" F
                <TextBox x:Name="txtLSP" Grid.Column="1" Style="{StaticRe
                         Margin="0,0,10,0"/>
                <Button x:Name="btnScan" Grid.Column="2" Content="🔍 Sök
                        Style="{StaticResource PrimaryButton}" MinWidth="
            </Grid>
        </Border>

        <!-- Move downloaded files -->
        <Border x:Name="grpDl" Style="{StaticResource CardBorder}">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="Auto"/>
                </Grid.ColumnDefinitions>
                <StackPanel Grid.Column="0" VerticalAlignment="Center">
                    <TextBlock Text="Flytta nedladdade filer" Foreground=
                               FontWeight="SemiBold" FontSize="12"/>
                    <TextBlock Text="Matchar LSP i filnamn och flyttar fr
                               Foreground="{StaticResource TextDimBrush}"
                </StackPanel>
                <Button x:Name="btnMove4" Grid.Column="1" Content="📁 KLA
                        Style="{StaticResource SecondaryButton}" Margin="
                <Button x:Name="btnMove3" Grid.Column="2" Content="🧪 TES
                        Style="{StaticResource SecondaryButton}" Visibili
            </Grid>
        </Border>
```

```
<!-- Log output -->
<Border Style="{StaticResource CardBorder}" Padding="0" Margin="0
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="180"/>
        </Grid.RowDefinitions>
        <TextBlock Grid.Row="0" Text="📋 Logg" Foreground="{Stati
                   FontSize="11" FontWeight="SemiBold" Margin="14
        <Border Grid.Row="1" Background="{StaticResource LogBgBru
            <TextBox x:Name="outputBox" IsReadOnly="True" TextWra
                     VerticalScrollBarVisibility="Auto" AcceptsRe
                     Background="Transparent" Foreground="#A0B8C8
                     BorderThickness="0" FontFamily="Cascadia Mon
                     FontSize="11" Padding="12,8" CaretBrush="Tra
        </Border>
    </Grid>
</Border>

<!-- File picker -->
<Border Style="{StaticResource CardBorder}">
    <StackPanel>
        <TextBlock Text="📁 Välj filer för rapport" Foreground="{
                   FontWeight="SemiBold" FontSize="13" Margin="0,
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>

            <!-- CSV -->
            <TextBlock Grid.Row="0" Grid.Column="0" Text="CSV-fil
                       Foreground="{StaticResource TextDimBrush}"
                       Margin="0,8,10,0" FontSize="12"/>
            <ListView x:Name="lvCsv" Grid.Row="0" Grid.Column="1"
                      Style="{StaticResource FileListView}" Heigh
            <Button x:Name="btnCsvBrowse" Grid.Row="0" Grid.Colum
                    Content="Bläddra…" Style="{StaticResource Sec
                    VerticalAlignment="Top" Margin="0,4,0,4" MinW
```

```
                <!-- NEG -->
                <TextBlock Grid.Row="1" Grid.Column="0" Text="Seal Te
                           Foreground="{StaticResource TextDimBrush}"
                           Margin="0,8,10,0" FontSize="12"/>
                <ListView x:Name="lvNeg" Grid.Row="1" Grid.Column="1"
                          Style="{StaticResource FileListView}" Heigh
                <Button x:Name="btnNegBrowse" Grid.Row="1" Grid.Colum
                        Content="Bläddra…" Style="{StaticResource Sec
                        VerticalAlignment="Top" Margin="0,4,0,4" MinW

                <!-- POS -->
                <TextBlock Grid.Row="2" Grid.Column="0" Text="Seal Te
                           Foreground="{StaticResource TextDimBrush}"
                           Margin="0,8,10,0" FontSize="12"/>
                <ListView x:Name="lvPos" Grid.Row="2" Grid.Column="1"
                          Style="{StaticResource FileListView}" Heigh
                <Button x:Name="btnPosBrowse" Grid.Row="2" Grid.Colum
                        Content="Bläddra…" Style="{StaticResource Sec
                        VerticalAlignment="Top" Margin="0,4,0,4" MinW

                <!-- Worksheet -->
                <TextBlock Grid.Row="3" Grid.Column="0" Text="Workshe
                           Foreground="{StaticResource TextDimBrush}"
                           Margin="0,8,10,0" FontSize="12"/>
                <ListView x:Name="lvLsp" Grid.Row="3" Grid.Column="1"
                          Style="{StaticResource FileListView}" Heigh
                <Button x:Name="btnLspBrowse" Grid.Row="3" Grid.Colum
                        Content="Bläddra…" Style="{StaticResource Sec
                        VerticalAlignment="Top" Margin="0,4,0,4" MinW
            </Grid>
        </StackPanel>
    </Border>

    <!-- Signature panel (initially collapsed) -->
    <Border x:Name="grpSign" Style="{StaticResource CardBorder}" Visi
        <StackPanel>
            <TextBlock Text="✍ Lägg till signatur i Seal Test-filern
                       Foreground="{StaticResource TextBrush}" FontWe
                       FontSize="13" Margin="0,0,0,10"/>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="Auto"/>
```

```
                    </Grid.RowDefinitions>
                    <TextBlock Grid.Row="0" Grid.Column="0"
                               Text="Fullständigt namn, signatur och datu
                               Foreground="{StaticResource TextDimBrush}"
                               VerticalAlignment="Center" Margin="0,0,10,
                    <TextBox x:Name="txtSigner" Grid.Row="0" Grid.Column=
                             Style="{StaticResource ModernTextBox}" Margi
                    <CheckBox x:Name="chkWriteSign" Grid.Row="1" Grid.Col
                              Content="Signera Seal Test-Filerna"
                              Style="{StaticResource ModernCheckBox}"/>
                    <CheckBox x:Name="chkOverwriteSign" Grid.Row="1" Grid
                              Content="Aktivera" IsEnabled="False"
                              Style="{StaticResource ModernCheckBox}"/>
                </Grid>
            </StackPanel>
        </Border>

        <!-- Build button -->
        <Button x:Name="btnBuild" Content="✅ Skapa rapport"
                Style="{StaticResource PrimaryButton}" IsEnabled="False"
                FontSize="15" Padding="20,12" Margin="0,8,0,4" Horizontal

    </StackPanel>
</ScrollViewer>

<!-- ===== STATUS BAR ===== -->
<Border Grid.Row="3" Background="#0D1720" Padding="12,7" BorderBrush="#1A
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
            <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
        <TextBlock x:Name="slCount" Grid.Column="0" Text="0 filer valda"
                   Foreground="{StaticResource TextDimBrush}" FontSize="1
        <TextBlock x:Name="slWork" Grid.Column="1" Text=""
                   Foreground="{StaticResource TextDimBrush}" FontSize="1
                   VerticalAlignment="Center" Margin="16,0,0,0"/>
        <ProgressBar x:Name="pbWork" Grid.Column="2" Width="140" Height="
                     Visibility="Collapsed" Margin="10,0" VerticalAlignme
                     Foreground="{StaticResource AccentBrush}" Background
                     BorderThickness="0"/>
        <Button x:Name="btnSpConnect" Grid.Column="3" Content="🔌 Anslut
                Style="{StaticResource SecondaryButton}" FontSize="11" Ma
                Padding="8,3" VerticalAlignment="Center"/>
```

```
                <TextBlock x:Name="slBatchLink" Grid.Column="4" Text="SharePoint:
                           Foreground="{StaticResource TextDimBrush}" FontSize="1
                           VerticalAlignment="Center" Margin="12,0,0,0" Cursor="H
            </Grid>
        </Border>
    </Grid>
</Window>
'@
#endregion XAML DEFINITION

# ================================================================================
#region BUILD WINDOW
# ================================================================================

# Inject version into title
$wpfXaml = $wpfXaml -replace 'WINDOW_TITLE_PLACEHOLDER', "IPTCompile – $ScriptVer

$reader = [System.Xml.XmlNodeReader]::new(([xml]$wpfXaml))
$wpfWindow = [Windows.Markup.XamlReader]::Load($reader)

# Set title version run
try {
    $tvr = $wpfWindow.FindName('titleVersionRun')
    if ($tvr) { $tvr.Text = " – $ScriptVersion" }
} catch {}

# ================================================================================
#region ELEMENT REFERENCES + ADAPTER BINDING
# ================================================================================

# --- Direct WPF controls ---
$txtLSP          = $wpfWindow.FindName('txtLSP')
$btnScan         = $wpfWindow.FindName('btnScan')
$btnBuild        = $wpfWindow.FindName('btnBuild')
$outputBox       = $wpfWindow.FindName('outputBox')
$txtSigner       = $wpfWindow.FindName('txtSigner')
$chkWriteSign    = $wpfWindow.FindName('chkWriteSign')
$chkOverwriteSign = $wpfWindow.FindName('chkOverwriteSign')
$btnCsvBrowse    = $wpfWindow.FindName('btnCsvBrowse')
$btnNegBrowse    = $wpfWindow.FindName('btnNegBrowse')
$btnPosBrowse    = $wpfWindow.FindName('btnPosBrowse')
$btnLspBrowse    = $wpfWindow.FindName('btnLspBrowse')
$btnMove3        = $wpfWindow.FindName('btnMove3')
$btnMove4        = $wpfWindow.FindName('btnMove4')
$grpSign         = $wpfWindow.FindName('grpSign')
$grpDl           = $wpfWindow.FindName('grpDl')
$menuStrip       = $wpfWindow.FindName('menuStrip')
```

```
$slCount         = $wpfWindow.FindName('slCount')
$slWork          = $wpfWindow.FindName('slWork')
$pbWork          = $wpfWindow.FindName('pbWork')
$slBatchLink     = $wpfWindow.FindName('slBatchLink')

# Menu items
$miArkiv     = $wpfWindow.FindName('miArkiv')
$miVerktyg   = $wpfWindow.FindName('miVerktyg')
$miSettings  = $wpfWindow.FindName('miSettings')
$miHelp      = $wpfWindow.FindName('miHelp')
$miAbout     = $wpfWindow.FindName('miAbout')
$miNew       = $wpfWindow.FindName('miNew')
$miOpenRecent = $wpfWindow.FindName('miOpenRecent')
$miScan      = $wpfWindow.FindName('miScan')
$miBuild     = $wpfWindow.FindName('miBuild')
$miExit      = $wpfWindow.FindName('miExit')
$miScript1   = $wpfWindow.FindName('miScript1')
$miScript2   = $wpfWindow.FindName('miScript2')
$miScript3   = $wpfWindow.FindName('miScript3')
$miScript4   = $wpfWindow.FindName('miScript4')
$miScript5   = $wpfWindow.FindName('miScript5')
$miToggleSign = $wpfWindow.FindName('miToggleSign')
$miTheme     = $wpfWindow.FindName('miTheme')
$miLightTheme = $wpfWindow.FindName('miLightTheme')
$miDarkTheme  = $wpfWindow.FindName('miDarkTheme')
$miShowInstr = $wpfWindow.FindName('miShowInstr')
$miFAQ       = $wpfWindow.FindName('miFAQ')
$miHelpDlg   = $wpfWindow.FindName('miHelpDlg')
$miGenvagar  = $wpfWindow.FindName('miGenvagar')
$miOm        = $wpfWindow.FindName('miOm')
$btnSpConnect = $wpfWindow.FindName('btnSpConnect')
$script:btnSpConnect = $btnSpConnect

# SP connect button
$script:btnMove3 = $btnMove3

# --- CheckedListBox adapters ---
$clbCsv = New-CheckedListBoxAdapter -ListView $wpfWindow.FindName('lvCsv')
$clbNeg = New-CheckedListBoxAdapter -ListView $wpfWindow.FindName('lvNeg')
$clbPos = New-CheckedListBoxAdapter -ListView $wpfWindow.FindName('lvPos')
$clbLsp = New-CheckedListBoxAdapter -ListView $wpfWindow.FindName('lvLsp')

# --- ListView DataTemplate injection (checkbox + text) ---
$dtXaml = @'
<DataTemplate xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
    <StackPanel Orientation="Horizontal" Margin="2,1">
        <CheckBox IsChecked="{Binding IsChecked, Mode=TwoWay}" VerticalAlignment=
```

```
        <TextBlock Text="{Binding DisplayText}" VerticalAlignment="Center" Foregr
    </StackPanel>
</DataTemplate>
'@
$dt = [Windows.Markup.XamlReader]::Parse($dtXaml)
foreach ($lv in @($wpfWindow.FindName('lvCsv'), $wpfWindow.FindName('lvNeg'),
                  $wpfWindow.FindName('lvPos'), $wpfWindow.FindName('lvLsp'))) {
    $lv.ItemTemplate = $dt
}

# ================================================================================
#region FORM ADAPTER (compatibility with WinForms $form references)
# ================================================================================
# Create a thin proxy so that $form.Close(), $form.Text, $form.Height etc. work
$form = [pscustomobject]@{
    _window    = $wpfWindow
    Text       = $wpfWindow.Title
    BackColor  = $null
    Height     = $wpfWindow.Height
    Width      = $wpfWindow.Width
    Font       = $null
    AcceptButton = $null
    Cursor     = $null
    MaximizeBox = $true
    MainMenuStrip = $menuStrip
}

$form | Add-Member -MemberType ScriptMethod -Name 'Close'         -Value { $this.
$form | Add-Member -MemberType ScriptMethod -Name 'Refresh'       -Value { try {
$form | Add-Member -MemberType ScriptMethod -Name 'SuspendLayout' -Value { }
$form | Add-Member -MemberType ScriptMethod -Name 'ResumeLayout'  -Value { }
$form | Add-Member -MemberType ScriptMethod -Name 'PerformLayout' -Value { }
$form | Add-Member -MemberType ScriptMethod -Name 'ShowDialog'    -Value { $this.

$form | Add-Member -MemberType ScriptMethod -Name 'add_KeyDown' -Value {
    param([scriptblock]$handler)
    $this._window.add_KeyDown({
        param($sender, $e)
        # Map WPF KeyEventArgs to something the handler can use
        & $handler $sender $e
    })
}

$form | Add-Member -MemberType ScriptMethod -Name 'add_FormClosed' -Value {
    param([scriptblock]$handler)
    $this._window.add_Closed($handler)
}
```

```
$form | Add-Member -MemberType ScriptMethod -Name 'add_Shown' -Value {
    param([scriptblock]$handler)
    $this._window.add_ContentRendered($handler)
}

# ================================================================================
#region STATUS BAR ADAPTERS
# ================================================================================

# $slCount and $slWork are TextBlocks — add .Text property alias
# (TextBlock already has .Text, so these work natively)

# $slBatchLink adapter — needs IsLink, Enabled, Tag, ToolTipText, add_Click
$slBatchLink | Add-Member -MemberType NoteProperty -Name 'IsLink'      -Value $tr
$slBatchLink | Add-Member -MemberType NoteProperty -Name 'Enabled'     -Value $fa
$slBatchLink | Add-Member -MemberType NoteProperty -Name 'Tag'         -Value $nu
$slBatchLink | Add-Member -MemberType NoteProperty -Name 'ToolTipText' -Value ''

$slBatchLink.add_MouseLeftButtonUp({
    if ($this.Enabled -and $this.Tag) {
        try { Start-Process -FilePath ([string]$this.Tag) } catch {}
    }
})

# $pbWork adapter
$pbWork | Add-Member -MemberType NoteProperty -Name 'Style'    -Value 'Blocks' -F
$pbWork | Add-Member -MemberType NoteProperty -Name 'Minimum'  -Value 0 -Force
$pbWork | Add-Member -MemberType NoteProperty -Name 'Maximum'  -Value 100 -Force
$pbWork | Add-Member -MemberType NoteProperty -Name 'MarqueeAnimationSpeed' -Valu

# $btnSpConnect adapter — needs .Text, .ToolTipText, .Enabled
$btnSpConnect | Add-Member -MemberType NoteProperty -Name 'Text'        -Value $b
$btnSpConnect | Add-Member -MemberType NoteProperty -Name 'ToolTipText' -Value ''
$btnSpConnect | Add-Member -MemberType NoteProperty -Name 'DisplayStyle' -Value '

# Sync .Text ↔ .Content
$btnSpConnect | Add-Member -MemberType ScriptMethod -Name '_syncText' -Value {
    $this.Content = $this.Text
}

# ================================================================================
#region SIGNATURE CHECKBOX LINKAGE
# ================================================================================
$chkWriteSign.add_Checked({  $chkOverwriteSign.IsEnabled = $true })
$chkWriteSign.add_Unchecked({ $chkOverwriteSign.IsEnabled = $false; $chkOverwrite
```

```
# Checkbox adapter: .Checked ↔ .IsChecked
foreach ($cb in @($chkWriteSign, $chkOverwriteSign)) {
    $cb | Add-Member -MemberType ScriptProperty -Name 'Checked' -Value {
        $this.IsChecked
    } -SecondValue {
        param($val)
        $this.IsChecked = [bool]$val
    } -Force
    $cb | Add-Member -MemberType ScriptMethod -Name 'add_CheckedChanged' -Value {
        param([scriptblock]$handler)
        $this.add_Checked($handler)
        $this.add_Unchecked($handler)
    } -Force
}

# ================================================================================
#region SHORTCUT MENU BUILDER
# ================================================================================
# Override Add-ShortcutItem to work with WPF MenuItems
function Add-ShortcutItem {
    param(
        [System.Windows.Controls.MenuItem]$Parent,
        [string]$Text,
        [string]$Target
    )
    $it = New-Object System.Windows.Controls.MenuItem
    $it.Header = $Text
    $it.Tag = $Target
    $it.add_Click({
        param($s, $e)
        $t = [string]$s.Tag
        try {
            if ($t -match '^(?i)https?://') {
                Start-Process -FilePath $t
            }
            elseif (Test-Path -LiteralPath $t) {
                $gi = Get-Item -LiteralPath $t
                if ($gi.PSIsContainer) {
                    Start-Process -FilePath explorer.exe -ArgumentList "`"$t`""
                } else {
                    Start-Process -FilePath $t
                }
            } else {
                [System.Windows.Forms.MessageBox]::Show("Hittar inte sökvägen:`n$
            }
        } catch {
            [System.Windows.Forms.MessageBox]::Show("Kunde inte öppna:`n$t`n$($_.
```

```
        }
    })
    [void]$Parent.Items.Add($it)
}

# Build shortcut menus from config
$ShortcutGroups = Get-ConfigValue -Name 'ShortcutGroups' -Default $null -ConfigOv
if (-not $ShortcutGroups) {
    $ShortcutGroups = @{
        '🗂️ IPT-mappar' = @(
            @{ Text='📁 IPT - PÅGÅENDE KÖRNINGAR';        Target='N:\QC\QC-1\IPT\
            @{ Text='📁 IPT - KLART FÖR SAMMANSTÄLLNING'; Target='N:\QC\QC-1\IPT\
            @{ Text='📁 IPT - KLART FÖR GRANSKNING';      Target='N:\QC\QC-1\IPT\
            @{ Text='📁 SPT Macro Assay';                 Target='N:\QC\QC-0\SPT\
        )
        '📄 Dokument' = @(
            @{ Text='🧰 Utrustningslista';    Target=$UtrustningListPath },
            @{ Text='🧪 Kontrollprovsfil';    Target=$RawDataPath }
        )
        '🌐 Länkar' = @(
            @{ Text='⚡ IPT App';              Target='https://apps.powerapps.com
            @{ Text='🌐 MES';                  Target='http://mes.cepheid.pri/cam
            @{ Text='🌐 CSV Uploader';         Target='http://auw2wgxtpap01.cepaw
            @{ Text='🌐 BMRAM';                Target='https://cepheid62468.coolb
            @{ Text='🌐 Agile';                Target='https://agileprod.cepheid.
        )
    }
}
foreach ($grp in $ShortcutGroups.GetEnumerator()) {
    $grpMenu = New-Object System.Windows.Controls.MenuItem
    $grpMenu.Header = $grp.Key
    foreach ($entry in $grp.Value) { Add-ShortcutItem -Parent $grpMenu -Text $ent
    [void]$miGenvagar.Items.Add($grpMenu)
}

# ================================================================================
#region COMPATIBILITY SHIMS
# ================================================================================

# Gui-Log adapter: WPF TextBox uses .Text + AppendText
function Set-LogOutputControl {
    param($Control)
    $script:LogOutputControl = $Control
}
Set-LogOutputControl -Control $outputBox

# Override Invoke-UiPump for WPF
```

```
function Invoke-UiPump {
    try {
        [System.Windows.Threading.Dispatcher]::CurrentDispatcher.Invoke(
            [Action]{ },
            [System.Windows.Threading.DispatcherPriority]::Background
        )
    } catch {}
}

# Set-UiBusy for WPF
function Set-UiBusy {
    param(
        [Parameter(Mandatory)][bool]$Busy,
        [string]$Message = ''
    )
    try {
        if ($Busy) {
            $slWork.Text = $Message
            $pbWork.Visibility = 'Visible'
            $pbWork.IsIndeterminate = $true
            $wpfWindow.Cursor = [System.Windows.Input.Cursors]::Wait
            $btnScan.IsEnabled  = $false
            $btnBuild.IsEnabled = $false
        } else {
            $pbWork.Visibility = 'Collapsed'
            $pbWork.IsIndeterminate = $false
            $pbWork.Value   = 0
            $slWork.Text = ''
            $wpfWindow.Cursor = $null
            $btnScan.IsEnabled = $true
            Update-BuildEnabled
        }
        Invoke-UiPump
    } catch {}
}

function Set-UiStep {
    param(
        [Parameter(Mandatory)][int]$Percent,
        [string]$Message = ''
    )
    try {
        if ($Percent -lt 0) { $Percent = 0 }
        if ($Percent -gt 100) { $Percent = 100 }
        $slWork.Text = $Message
        $pbWork.Visibility = 'Visible'
        $pbWork.IsIndeterminate = $false
```

```
        $pbWork.Minimum = 0
        $pbWork.Maximum = 100
        $pbWork.Value = $Percent
        Invoke-UiPump
    } catch {}
}

# Update-StatusBar adapter
function Update-StatusBar { $slCount.Text = "$(Get-SelectedFileCount) filer valda

# Theme engine for WPF
function Set-Theme {
    param([string]$Theme)
    $global:CurrentTheme = $Theme
    $bc = [System.Windows.Media.BrushConverter]::new()

    if ($Theme -eq 'light') {
        $wpfWindow.Background = $bc.ConvertFrom('#F5F7FA')
        $outputBox.Foreground = $bc.ConvertFrom('#1A2530')
        # Simplified — full theme would update all resource brushes
    } else {
        $wpfWindow.Background = $bc.ConvertFrom('#0F1923')
        $outputBox.Foreground = $bc.ConvertFrom('#A0B8C8')
    }
}

# grpSign visibility adapter
$grpSign | Add-Member -MemberType ScriptProperty -Name 'Visible' -Value {
    $this.Visibility -eq [System.Windows.Visibility]::Visible
} -SecondValue {
    param($val)
    $this.Visibility = if ($val) { [System.Windows.Visibility]::Visible } else {
} -Force

# btnMove3 visibility adapter
$btnMove3 | Add-Member -MemberType ScriptProperty -Name 'Visible' -Value {
    $this.Visibility -eq [System.Windows.Visibility]::Visible
} -SecondValue {
    param($val)
    $this.Visibility = if ($val) { [System.Windows.Visibility]::Visible } else {
} -Force

# Button adapters: .Enabled ↔ .IsEnabled, .Text ↔ .Content
foreach ($btn in @($btnScan, $btnBuild, $btnMove3, $btnMove4, $btnCsvBrowse, $btn
    $btn | Add-Member -MemberType ScriptProperty -Name 'Enabled' -Value {
        $this.IsEnabled
    } -SecondValue {
```

```
        param($val)
        $this.IsEnabled = [bool]$val
    } -Force

    $btn | Add-Member -MemberType ScriptProperty -Name 'Text' -Value {
        $this.Content
    } -SecondValue {
        param($val)
        $this.Content = $val
    } -Force

    # PerformClick shim
    $btn | Add-Member -MemberType ScriptMethod -Name 'PerformClick' -Value {
        $args = New-Object System.Windows.RoutedEventArgs([System.Windows.Control
        $this.RaiseEvent($args)
    } -Force
}

# TextBox adapter: .Clear()
$outputBox | Add-Member -MemberType ScriptMethod -Name 'Clear' -Value { $this.Tex
$txtLSP    | Add-Member -MemberType ScriptMethod -Name 'Clear' -Value { $this.Tex
$txtSigner | Add-Member -MemberType ScriptMethod -Name 'Clear' -Value { $this.Tex

# MenuItem adapters: add_Click
foreach ($mi in @($miArkiv,$miVerktyg,$miSettings,$miHelp,$miAbout,$miNew,$miOpen
                  $miExit,$miScript1,$miScript2,$miScript3,$miScript4,$miScript5,
                  $miToggleSign,$miLightTheme,$miDarkTheme,$miShowInstr,$miFAQ,$m
    if (-not $mi) { continue }
    $mi | Add-Member -MemberType ScriptMethod -Name 'add_Click' -Value {
        param([scriptblock]$handler)
        $this.add_Click($handler)
    } -Force -ErrorAction SilentlyContinue

    # .Text ↔ .Header
    $mi | Add-Member -MemberType ScriptProperty -Name 'Text' -Value {
        $this.Header
    } -SecondValue {
        param($val)
        $this.Header = $val
    } -Force -ErrorAction SilentlyContinue

    # .ToolTipText
    $mi | Add-Member -MemberType ScriptProperty -Name 'ToolTipText' -Value {
        $this.ToolTip
    } -SecondValue {
        param($val)
        $this.ToolTip = $val
```

```
    } -Force -ErrorAction SilentlyContinue
}

# ========================================================================
#region KEYBOARD SHORTCUTS
# ========================================================================
$wpfWindow.add_KeyDown({
    param($sender, $e)
    if ($e.Key -eq [System.Windows.Input.Key]::Escape) { $wpfWindow.Close(); retu
    # Dev toggle: Ctrl + Alt + T
    if ($e.KeyboardDevice.Modifiers -band [System.Windows.Input.ModifierKeys]::Co
        $e.KeyboardDevice.Modifiers -band [System.Windows.Input.ModifierKeys]::Al
        $e.Key -eq [System.Windows.Input.Key]::T) {
        if ($script:btnMove3) {
            $script:btnMove3.Visible = -not $script:btnMove3.Visible
        }
        $e.Handled = $true
    }
    # Enter → Scan
    if ($e.Key -eq [System.Windows.Input.Key]::Return) {
        if ($txtLSP.IsFocused -and $btnScan.IsEnabled) {
            $btnScan.RaiseEvent((New-Object System.Windows.RoutedEventArgs([Syste
            $e.Handled = $true
        }
    }
})

# ========================================================================
#region LOGO LOADING
# ========================================================================
try {
    if ($ikonSokvag -and (Test-Path -LiteralPath $ikonSokvag)) {
        $bmp = New-Object System.Windows.Media.Imaging.BitmapImage
        $bmp.BeginInit()
        $bmp.UriSource = New-Object Uri($ikonSokvag, [UriKind]::Absolute)
        $bmp.CacheOption = [System.Windows.Media.Imaging.BitmapCacheOption]::OnLo
        $bmp.EndInit()
        $img = New-Object System.Windows.Controls.Image
        $img.Source = $bmp
        $img.Stretch = [System.Windows.Media.Stretch]::Uniform
        $logoContainer = $wpfWindow.FindName('picLogoContainer')
        if ($logoContainer) {
            $logoContainer.Child = $img
            $logoContainer.Background = [System.Windows.Media.Brushes]::Transpare
        }
    }
} catch {}
```

```
# ================================================================================
#region COMPATIBILITY ALIASES FOR $form
# ================================================================================
# The real WPF window (needed for ShowDialog at the end of Main.ps1)
$script:WpfWindow = $wpfWindow

# Override Application.Run equivalent
function Start-WpfApplication {
    [System.Windows.Threading.Dispatcher]::CurrentDispatcher.InvokeShutdown()
    $wpfWindow.ShowDialog() | Out-Null
}

# Panel/content references (used by Set-Theme, Enable-DoubleBuffer)
$content     = $wpfWindow
$panelHeader = $wpfWindow.FindName('picLogoContainer')
$pLog        = $outputBox
$grpPick     = $null
$tlSearch    = $null

# ToolTip adapter (no-op — WPF controls have native ToolTip property)
$toolTip = [pscustomobject]@{}
$toolTip | Add-Member -MemberType ScriptMethod -Name 'SetToolTip' -Value {
    param($control, [string]$text)
    try { $control.ToolTip = $text } catch {}
}

# ToolStripStatusLabel Spring property (no-op)
$slCount | Add-Member -MemberType NoteProperty -Name 'Spring' -Value $false -Forc
$slWork  | Add-Member -MemberType NoteProperty -Name 'Spring' -Value $true -Force

# Enable-DoubleBuffer (no-op in WPF)
function Enable-DoubleBuffer { }

# Status strip refresh (no-op)
$status = [pscustomobject]@{}
$status | Add-Member -MemberType ScriptMethod -Name 'Refresh' -Value { }

Write-Host "[WPF] GUI loaded successfully."
#endregion
```