Dashboard for RPA

HTML/CSS + PowerShell

Two components are needed for this solution. The principle is to take screenshots on each virtual machine at a specified interval, then save it to disk and overwrite it through the next. In this way, we create monitoring a'la real time. These snapshots are then read by HTML, which also refreshes the display area at a similar time to the time between taking the snapshot.

HTML/CSS

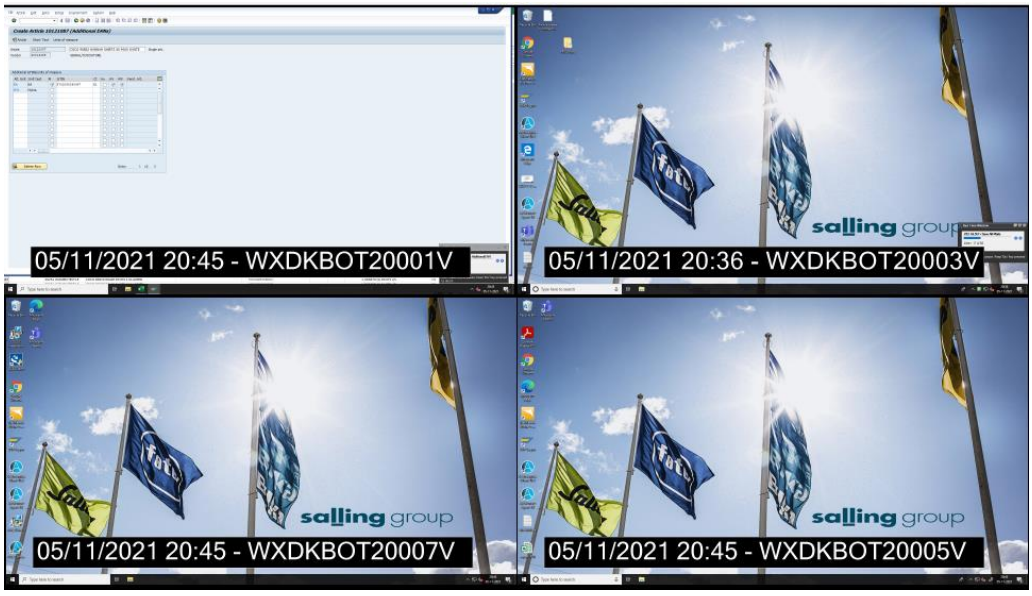
Simple to use and write HTML/CSS code, refreshing every x (for the pasted code it is 20 seconds), area is 2x2, 4 images displayed in the initial phase and another 4 screenshots after 10 seconds.

The code has been adjusted to fit the TV resolution, the sliders on the sides of the browser have been removed.

The flashing after refresh has been reduced by using a dark background.

The code can be found under the Dashboard track.

The result of this solution itself looks like in the screenshot below:



Powered by: Filip Rysz

PowerShell

Parameters

```
[CmdletBinding()] Param(

[Parameter(Mandatory=$True)]          // Time interval
[int32] $Interval,

[Parameter(Mandatory=$True)]          // File name that
[string] $FileName,

[Parameter(Mandatory=$false)]
[bool]$AddTimestamp=$false
                                   // If value of this parametr is set
to $true the table with timestamp and string will be included on
screenshot
```

This script, written in powershell, is designed to take screenshots on the machine it is running on. Appropriate parameters used in the implementation of the code draw a table with the name of the machine and the time at which the screenshot was taken.

The $path variable in the code is responsible for the path where the screenshot should be saved. Change with caution.

Loops responsible for running the screenshot function looks like this:

```
While($true)
{
    Try
    {

        Do
        {
            $FilePath = (Join-Path $Path $FileName)

            PrintScreen
            Start-Sleep -Seconds $Interval
        }
        While($true)
    }
 Catch {Write-Warning "$Error[0].ToString() + $Error[0].InvocationInfo.PositionMessage"}
}
```

No need to change anything in those two loops.

Powered by: Filip Rysz

The function responsible for taking screenshots is as follows

```
Function PrintScreen
{
   $ScreenBounds = [Windows.Forms.SystemInformation]::VirtualScreen
   $ScreenshotObject = New-Object Drawing.Bitmap $ScreenBounds.Width, $ScreenBounds.Height
   $DrawingGraphics = [Drawing.Graphics]::FromImage($ScreenshotObject)
   $DrawingGraphics.CopyFromScreen($ScreenBounds.Location, [Drawing.Point]::Empty, $ScreenBounds.Size)
   if($AddTimestamp)
   {
        $Timestamp_Font = New-Object System.Drawing.Font("Segoe UI Bold", 69);
        $Timestamp_Brush_Fg = [System.Drawing.Brushes]::White;
        $Timestamp_Brush_Bg = [System.Drawing.Brushes]::Black;

        [String]$Timestamp_DateTimeString = Get-Date -UFormat "%d/%m/%Y %R";
        [String]$nazwamaszyny = hostname
        $Bobo = "$Timestamp_DateTimeString - $nazwamaszyny"

        $Timestamp_Font_BackgroundSize = $DrawingGraphics.MeasureString($Bobo, $Timestamp_Font);
        $Timestamp_Font_BackgroundRect = New-Object System.Drawing.RectangleF(100,900, $Timestamp_Font_BackgroundSize.Width,
$Timestamp_Font_BackgroundSize.Height);

        $DrawingGraphics.FillRectangle($Timestamp_Brush_Bg, $Timestamp_Font_BackgroundRect);
        $DrawingGraphics.DrawRectangle($Timestamp_Brush_Fg, [System.Drawing.Rectangle]::Round($Timestamp_Font_BackgroundRect));
        $DrawingGraphics.DrawString($Bobo, $Timestamp_Font, $Timestamp_Brush_Fg, 100, 900);
   }
   $DrawingGraphics.Dispose()
   $ScreenshotObject.Save($FilePath)
   $ScreenshotObject.Dispose()
}
```

if you need to change the appearance or size related to the machine name table and the time, change the values assigned to the variables:

```
$Timestamp_Font
$Timestamp_Brush_Fg
$Timestamp_Brush_Bg
```

If there is a need to change position of that table, change the values 100 and 900 where 100 is an x and 900 is y in:

```
$Timestamp_Font_BackgroundRect
$DrawingGraphics.DrawString
```

Script calling example:

```
PrintScreenFun -Interval 20 -FileName Screen1.jpg -AddTimestamp $true
```

Where -interval = time between making screenshot, -filename is just a file name and -AddtimeStamp defines if there is a need for a box with machine name and timestamp ($true = yes)

Regarding the method of calling the script on user login:
Enter path:
\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Then, create CMD file (txt file but save it as .cmd) with the following string:
PowerShell -windowstyle hidden -executionpolicy bypass PATH_TO_THE_SCRIPT
(example: PowerShell -windowstyle hidden -executionpolicy bypass C:\Users\sacri\Desktop\PrintScreenFunv6.ps1)

Both, HTML/CSS and Script in attachment.

Powered by: Filip Rysz