Smart Resizing and High DPI issues in Remote Desktop Manager

Devolutions

SHARE WITH YOU NOT ONLY THE PROBLEMS WE'VE ENCOUNTERED, BUT ALSO THE SOLUTIONS WE'VE IMPLEMENTED.

It's been a while since I've posted here, but I thought it was a good time to publish a new blog. I've recently been working on two issues — smart resizing and high DPI in Remote Desktop Manager (RDM is our product that centralizes all your connections and passwords).

I thought it would be a good idea to share with you not only the problems we've encountered, but also the solutions we've implemented.

1. Smart Resize bug in RDP 8.1

Starting with Windows 10, we began experiencing issues with the smart sizing. Sometimes the RDP control wouldn't properly refresh and then it wouldn't be possible to click anywhere in it. It happened every time the parent form was maximized. This issue was often reported on our forum by our users.

https://forum.devolutions.net/topic24395-embedded-rdp-scaling-issues-on-windows-.aspx

This became **one of the most annoying bugs** in Remote Desktop Manager. I was able to reproduce the same issue in RDCMan (Microsoft Remote Desktop Connection Manager) and knew it had something to do with the ActiveX. I've tried many workarounds, such as: forcing a refresh of the window; resizing the window; and switching the Smart Sizing on and off. None of these attempts have succeeded.

Yesterday, however, while testing some new features in the ActiveX, I finally found a way to eliminate the problem. The solution was actually pretty simple and all I needed to do was enable the zoom functionality in the ActiveX. You **don't need** to change the zoom level though.

Just call this method before the Connect and it works like magic!

```
private void SetExtendedProperty(string propertyName, bool value)
{
    IMsRdpExtendedSettings extendedSettings =
        (IMsRdpExtendedSettings)this.Client.GetOcx();
    object boolValue = value;
    try
    {
        extendedSettings.set_Property(propertyName, ref boolValue);
    }
    catch (Exception ex)
    {
        LogManager.LogDebugError(ex);
    }
}
```

Be aware that you will need the RDP 8.1 ActiveX client (AxMsRdpClient9NotSafeForScripting) to fix this issue. This won't work if you don't use the latest version of the interop library.

2. High DPI with RDP

A few months ago we did a major upgrade of our GUI to support High DPI. It was really challenging to add support to a WinForm application. The only missing piece of the puzzle was for the RDP connection itself. At that time, we were unable to scale the connection.

This meant that the display was too small: https://forum.devolutions.net/topic26129-k-screens-and-high-dpi.aspx?lastpage=1#post95709 So last week I started looking for a solution. I've found an API (https://msdn.microsoft.com/en-us/library/mt703457(v=vs.85).aspx) in the ActiveX which looked very promising:

IMsRdpClient9::UpdateSessionDisplaySettings

```
HRESULT UpdateSessionDisplaySettings(
    [in] ULONG ulDesktopWidth,
    [in] ULONG ulDesktopHeight,
    [in] ULONG ulPhysicalWidth,
    [in] ULONG ulPhysicalHeight,
    [in] ULONG ulOrientation,
    [in] ULONG ulDesktopScaleFactor,
    [in] ULONG ulDeviceScaleFactor
);
```

Next, I checked the RDP protocol specifications to get more details and it was exactly what we wanted. We needed to specify the desktop scale factor and the device scale factor:

```
protected uint GetDesktopScaleFactor()
{
    switch (DpiManager.ScalePercent)
    {
        case 125:
            return 125;
        case 150:
        case 175:
            return 150;
        case 200:
            return 200;
    }
    if (DpiManager.ScalePercent > 200)
    {
        return 200;
    }
    return 100;
}
```

```
protected uint GetDeviceScaleFactor()
{
    switch (DpiManager.ScalePercent)
    {
        case 125:
        case 150:
        case 175:
            return 140;
        case 200:
            return 180;
    }
    if (DpiManager.ScalePercent > 200)
    {
        return 180;
    }
    return 100;
}
```

Unfortunately, it failed. It wasn't possible to call it **before** the Connect; as a stack overflow thread mentioned, we needed to call it a few seconds **after** the Connect. How many seconds? Well, that was nearly impossible to find out. Sometimes 1 second, sometimes 2, and other times the login would be longer too!
https://stackoverflow.com/questions/34496466/auto-apply-local-scale-to-non-fullscreen-desktop-connections-rdp-version-8-1-an

So despite trying to find an event to execute the call at the right time, I couldn't find anything. By inspecting the ActiveX, I've found that it is possible to specify the DesktopScaleFactor and the DeviceScaleFactor before the connection with this code by using the *IMsRdpExtendedSettings:*

```
this.SetExtendedProperty("DesktopScaleFactor", this.GetDesktopScaleFac
this.SetExtendedProperty("DeviceScaleFactor", this.GetDeviceScaleFacto
```

This fix works perfectly for Windows 10 and Windows 2012R2.

In Closing

I truly hope these solutions will be useful for those of you integrating the RDP ActiveX. If you are already using Remote Desktop Manager, you can download the update to fix the issue. Note that this is even included in our free edition.

If you're not using our application, the good news is that I'm pretty sure that soon this fix will be among your favorite tools, like mRemote, RDCMan or maybe your own custom application. You may even refer others to this blog to help speed things up.

Please let me know if you find anything else to improve my code or any other workarounds that relate to the RDP ActiveX.