March 17, 2024 | Windows 10 Windows Server 2016 Windows Server 2019

How to Shadow (Remote Control) a User's RDP session on Windows Server RDS

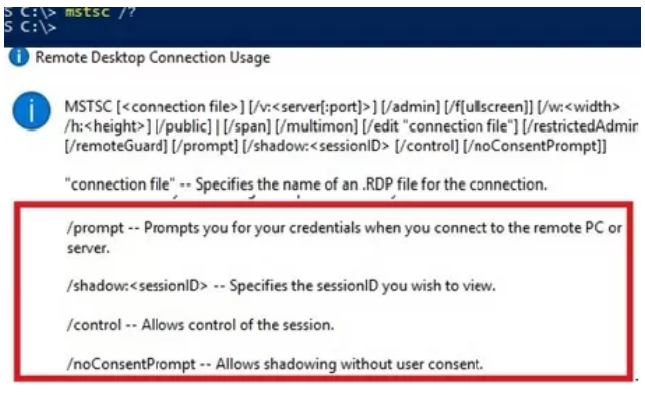
Shadow session mode allows RDS administrators to view and interact with the user's desktop. Remote Desktop Shadowing mode works on all modern versions of Windows starting from Windows Server 2012 R2 and Windows 8.1 (except for Windows Server 2012, due to the transfer of the RDP stack from kernel to user mode). In this article, we'll look at how to configure and use RDS Shadowing to connect and manage active RDP user sessions on Windows Server 2016 and Windows 10.

Contents:

Shadow Connection Options in the Windows RDP Client (mstsc.exe)

On Windows Server 2016/Windows 10, the built-in RDP client (`mstsc.exe`) has several special options that can be used to remotely shadow connect to an active RDP session of any user:

```
Mstsc.exe [/shadow:sessionID [/v:Servername] [/control] [/noConsentPrompt] [/prompt]]
```



- **/shadow:ID** – connect to the user's RDP session with the specified ID;
- **/v:servername** – you can specify the hostname or IP address of the remote RDP/RDS host. If not set, connections are made to local user sessions on the current host;
- **/control** – allows to interact with the user session (desktop). The administrator can control the user's mouse, input data from the keyboard. If this parameter is not set, the user's session view mode is used;
- **/noConsentPrompt** – the option allows the administrator to force the connection to any session without asking the user to confirm the connection;
- **/prompt –** allows to connect with other credentials. The user name and password are requested to connect to the remote computer.

Shadow sessions can be used to connect to user sessions on computers and servers in both an Active Directory domain and a workgroup. In addition, it is not necessary to have administrator privileges on the RDS host on this the user's RDP session is running . Administrators can delegate RDS Shadowing permissions to any user account, even for non-admins (more on this below).

Using Remote Desktop Shadow from the Windows GUI

You can connect to a user session using `mstsc.exe` or directly from Server Manager graphical console. To do it, open the Server Manager console on the RDS server, go to the Remote Desktop Services section -> select your collection, for example `QuickSessionCollection`.

The list on the right will contain a list of users who have sessions on this RDS server. Right-click on the user session you want, select **Shadow** from the drop-down menu.

CATEGORIES

- Active Directory
- Group Policies
- Exchange Server
- Microsoft 365
- Azure
- Windows 11
- Windows 10
- Windows Server 2022
- Windows Server 2019
- Windows Server 2016
- PowerShell
- VMWare
- Hyper-V
- Linux
- MS Office

RECENT POSTS

Remove a Specific Device from the Safely Remove Hardware List on Windows
October 16, 2024

Disable BitLocker Automatic Drive Encryption in Windows 11
October 16, 2024

> You can only connect to an active user session. If the session is in a disconnected state (due to the RDS session limit/timeout settings), you cannot connect to such a session:
>
> ```
> Shadow Error - The specified session is not connected.
> ```

A window with Shadow connection parameters will appear. You can either **View** or **Control** a user's RDP session. You can also check **Prompt for user consent** option.

If this option is checked, the following request appears in the user's RDP session:

```
Remote Monitoring Request
woshub\administrator is requesting to view your session remotely. Do you
accept the request?
```

If the user confirms the connection, the administrator will see his desktop in **View** mode, but won't be able to interact with it.

> **Tip.** To disconnect from a user session and exit the Shadow mode, press `ALT+*` on the workstation or `Ctrl+*` on the RDS server (if no alternative combinations are not set).

If a user rejects the administrative Shadow RDS connection, the following message appears:

```
Shadow Error: The operator or administrator has refused the request.
```

> You cannot use the tsadmin.msc graphical snap-in from Windows Server 2008 R2 for shadow connections to RDP sessions on newer versions of Windows Server.

If you attempt to connect to a user's session without prompting for confirmation, you receive an error message:

```
Shadow Error: The Group Policy setting is configured to require the use
```

Check the Software Installation/Removal History in Windows
October 8, 2024

Run PowerShell Scripts on a Schedule with Task Scheduler
October 3, 2024

Graylog: Centralized Log Collection and Analysis
October 1, 2024

Disable and Completely Remove Widgets from Taskbar in Windows 11
September 26, 2024

Configure Kiosk Mode on Windows 11 (Single or Multi-App)
September 24, 2024

How to Cast/Mirror Android Screen to Windows PC
September 11, 2024

Get Started with Docker on Windows (WSL2) without Docker Desktop
September 4, 2024

Adding Multiple Alternate DNS Names for a Windows Computer
September 3, 2024

```
r's consent. Verify the configuration of the policy settings.
```

If you need to audit RDS shadow connection events for user sessions, use the following filtered events from the **Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational** log:

- Event ID **20508**: Shadow View Permission Granted;
- Event ID **20503**: Shadow View Session Started;
- Event ID **20504**: Shadow View Session Stopped.

Configuring RDS Shadow Rules on Windows Using GPO

The settings for remote connections to RDS user sessions are configured using the Group Policy parameter **Set rules for remote control of Remote Desktop Services user sessions**, which is located under the User and Computer sections of the GPO: Policies -> Administrative Templates -> Windows components -> Remote Desktop Services -> Remote Session Host -> Connections. This policy corresponds to the DWORD **Shadow** parameter under the registry key `HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services` (the values of this parameter corresponding to the policy settings are specified in brackets).

This policy can be used to configure the following RD Shadow connection options:

- **No remote control allowed** (corresponds to the value of the registry parameter `Shadow = 0`);
- **Full Control with user's permission** (`1`);
- **Full Control without user's permission** (`2`);
- **View Session with user's permission** (`3`);
- **View Session without user's permission** (`4`).

> To set the parameter of this policy directly through the registry, you can use the command:
>
> `reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v Shadow /t REG_DWORD /d 4`
>
> After changing the session shadow behavior, update the Group Policy settings on the RDP/RDS host.

You can configure rules for remote shadow connections in the AD domain from the `gpmc.msc` console using the policy parameter described above, or by using Group Policy Preferences, which directly modify the registry parameter (the latter option allows you to more accurately target the policy to specific computers using the Group Policy Item Level Targeting).

Shadowing RDP Session with PowerShell

You can also use the Remote Desktop Services Shadow features to connect to a user's session from PowerShell. First of all, you need to get a list of sessions on the RDS host (user sessions will be grouped according to their state):

```
Get-RDUserSession | ft Username, UnifiedSessionId, SessionState, HostServer, ApplicationType -GroupBy Sessionstate
```

There are three active RDP user sessions on this server. Let's connect to the user session with the session ID 3:

```
Mstsc /shadow:3 /control
```

Also, to get a list of all RDP sessions on the server (or on the Windows 10 desktop to which multiple RDP connections are allowed), you can use the command:

```
quser
```

or

```
qwinsta
```

The screen will display a list of RDP user sessions, their IDs and states: `Active` or `Disconnected`.

To show the list of sessions on a remote server, run the following command:

```
query session /server:servername
```

To connect to a user session on a remote server, use the command:

```
mstsc /v:rdsh2:3389 /shadow:3 /control
```

> The dynamic port range (RPC) 49152 to 65535 is used to establish a remote shadow connection session instead of the default TCP/3389 RDP port.

For more convenient shadow connection to RDP user sessions, you can use the following batch script. It prompts you to enter the name of the remote RDS server, displays a list of all sessions and prompts you to specify the session (ID) to which you want to connect to:

shadow.bat

```
@echo off
set /P rcomp="Enter name or IP of a Remote PC: "
query session /server:%rcomp%
set /P rid="Enter RDP user ID: "
start mstsc /shadow:%rid% /v:%rcomp% /control
```

You can save this bat file in the `%Windir%\System32` directory. As a result, you just need to run the **shadow** command to start the shadow connection.

> To connect to the console session, you can use this script:
>
> **shadow_console.bat**
>
> ```
> @echo off
> set /P rcomp="Enter name or IP of a Remote PC: "
> for /f "tokens=3 delims= " %%G in ('query session console /server:%rcomp%') do
> set rid=%%G
> start mstsc /shadow:%rid% /v:%rcomp% /control
> ```

You can also use the following PowerShell script with a simple GUI (**shadow_user_rdp_session.ps1**) for shadow connection:

```
Add-Type -assembly System.Windows.Forms
$Header = "SESSIONNAME", "USERNAME", "ID", "STATUS"
$gForm = New-Object System.Windows.Forms.Form
$gForm.Text ='Shadow Session Connect'
```

```
$gForm.Width = 400
$gForm.AutoSize = $true
$dBttn = New-Object System.Windows.Forms.Button
$dBttn.Text = 'Control'
$dBttn.Location = New-Object System.Drawing.Point(15,10)
$gForm.Controls.Add($dBttn)
$dList = New-Object System.Windows.Forms.ListView
$dList.Location = New-Object System.Drawing.Point(0,50)
$dList.Width = $gForm.ClientRectangle.Width
$dList.Height = $gForm.ClientRectangle.Height
$dList.Anchor = "Top, Left, Right, Bottom"
$dList.MultiSelect = $False
$dList.View = 'Details'
$dList.FullRowSelect = 1;
$dList.GridLines = 1
$dList.Scrollable = 1
$gForm.Controls.add($dList)
foreach ($column in $Header){
$dList.Columns.Add($column) | Out-Null
}
$(qwinsta.exe | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," |
ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
$dBttn.Add_Click(
{
$SelectedItem = $dList.SelectedItems[0]
if ($SelectedItem -eq $null){
[System.Windows.Forms.MessageBox]::Show("Select a user session to connect ")
}else{
$session_id = $SelectedItem.subitems[2].text
$(mstsc /shadow:$session_id /control)
#[System.Windows.Forms.MessageBox]::Show($session_id)
}
}
)
$gForm.ShowDialog()
```

This script displays a simple graphical form with a list of active RDP sessions on the local host. You just need to select a user account and click the **Connect** button.

> To run PowerShell scripts (*.ps1) on the computer, you need to configure the PowerShell Execution policy.

You can use the shadow user connection not only on Windows Server with the Remote Desktop Services role, but also to connect to users' desktops running Windows 10 (Using Remote Desktop Session Shadowing Mode in Windows 10).

How-to Allow Non-admin Users to Shadow RDS Sessions?

In the examples above, using the shadow connection to RDS user sessions requires local administrator privileges on the RDS server. However, you can allow a non-admin user to shadow RDP sessions without granting local admin permissions on the computer/server.

For instance, you want to allow members of the AllowRDSShadow group to use a shadow connection to RDP user sessions. Open the elevated command prompt (cmd.exe) and run the command:

```
wmic /namespace:\\root\CIMV2\TerminalServices PATH Win32_TSPermissionsSetting WHERE (TerminalName="RDP-Tcp") CALL AddAccount "woshub\AllowRDSShadow",2
```


previous post
Configuring Proxy Settings on Windows Using Group Policy Preferences

next post
How to Sign a PowerShell Script File (PS1)

It is too bad that MS removed it from 2012 server. Thankfully they put it back in R2. We desperately needed shadowing for our terminal server users since they need support for their software packages all the time. We finally found a 3rd party product that allows us to do it in a much easier way: http://www.intelliadmin.com/index.php/2012/04/easily-shadow-remote-desktop-sessions/
We tried to see if dameware, or radmin did these this but no luck. So far it is the only one we have found. Wondering if anyone knows of a free solution.

PHILIPPE FERRUCCI Reply

November 23, 2016 - 1:18 pm

Starting a remote control by mstsc does not work. It gives an error "Access denied". You can use the Server Manager only.
BTW, to find the logged users, just issue "quser". No need to fire up PowerShell for that.

JELLE AMEYE Reply

February 13, 2017 - 8:36 am

Hello,
Copy / paste this into notepad and save as batch file (.bat). Edit 'servername' into your servers name…
@echo off
qwinsta
set /P id=Enter id: %=%
mstsc /v:servername /shadow:%id% /control /noConsentPrompt
Jelle

ITPRO Reply

August 1, 2017 - 9:26 am

To allow non-administrator users connect to other user session via shadowing on Win 2012 R2 RDS, you ned to execute the following command:
wmic /namespace:\\root\CIMV2\TerminalServices PATH Win32_TSPermissionsSetting WHERE (TerminalName="RDP-Tcp") CALL AddAccount "corp\RDSSupport",2

ALEXEY Reply

November 15, 2019 - 11:24 am

It doesn't work for me too. When I try to shadow to user's session I get the message "Access denied". The WMIC command above doesn't help either. It looks like local admin rights are mandatory.

WINDOWS: PER REMOTEDESKTOPVERBINDUNG AUF DIE KONSOLE VERBINDEN | ANDYS BLOG – LINUX & WINDOWS Reply

October 16, 2018 - 8:57 pm

[…] Quelle: Windows OS Hub – How to Shadow (Remote Control) a User RDP session on RDS Windows Server 2016 … […]

NYEBODNYE Reply

February 11, 2020 - 2:50 pm

shadow.bat
@echo off
for /f "tokens=2-4" %%a in ('query session /server:%1') do @if "%%c"=="Active" set session=%%b
mstsc.exe /multimon /span /v:%1 /shadow:%session% /control
multimon and span don't seem to work though

NYEBODNYE Reply

February 11, 2020 - 3:51 pm

RE: previous post. You need to fix the single and double quotes to NOT be backwards and forwards or this will not work.
It looks like the comments system decided to change them for no apparent reason.

MAX Reply

February 22, 2021 - 12:21 pm

In January 2018, after installing the update KB4056898 (the Windows patch against Meltdown and Specter), users encountered that the RDS shadow stopped working. When attempting to perform a shadow connection to a user session, the message "An unidentified error" appeared (there is a STATUS_BAD_IMPERSONATION_LEVEL error in the Windows logs). A similar problem arose on the RDS farm Windows Server 2016.
To solve the problem, you need to install the following patches:
for Windows Server 2016 — KB4057142 (January 17, 2018)
for Windows Server 2012 R2 — KB4057401 (January 17, 2018)

WINNI Reply

March 13, 2021 - 10:50 am

can you also etablsih a rdp shadow connection over the internet? i tried this and nat forward the port 445 but it didn't work. does anybodyknow if you can do this also from an external acces with directing to the public ip of the remote host? any ideas?

NYEBODNYE Reply

June 29, 2021 - 4:26 pm

Port 445 is SMB, not RDS/RDP. RDS/RDP uses port 3389 TCP and UDP.
This is dangerous to allow on the internet because bad actors frequently scan these ports looking for desktops to exploit.

GUICH Reply

July 19, 2021 - 1:27 pm

Hi and thank you for this.
Any idea on how to offer the user who is sharing is session to see in live who is connected to it's session through shadowing ?
For now, only thing I've got is looking inside eventlogs, but it's not really user friendly.
Thanks

HARDYHARD Reply

August 20, 2021 - 7:42 am

I had to execute the wmic command for a domain admin account on a domain-joined server too, otherwise I got "Access denied", just in case somebody else experiences the same problem…

HARDYHARD Reply

August 24, 2021 - 12:57 pm

And it seems I also have to start cmd in Admin-Mode, otherwise I get access denied as well.

NYEBODNYE Reply

October 26, 2021 - 8:40 am

If you want to see if anyone is connected try netstat and look for connections on port 3389

```
@echo off
rem Requires registry/group policy setting to allow shadowing
rem https://woshub.com/rds-shadow-how-to-connect-to-a-user-session-in-windows-server-2012-r2/
for /f "tokens=2-4" %%a in ('query session /server:%1') do @if "%%c"=="active" set session=%%b
mstsc.exe /multimon /span /v:%1 /shadow:%session% /control
```

VALNT9

December 14, 2023 - 6:43 pm

```
################################################################################################################################
## Author……: Doug Valentine DateCreated: 2023-12-14 DateModified: ##
## Description.: PowerShell script to query RDSH servers and list session in USERNAME order, upon Click open a RDP SHADOW session. ##
################################################################################################################################
Add-Type -assembly System.Windows.Forms
## Set up the environment
Add-Type -AssemblyName System.Windows.Forms
$LastColumnClicked = 0 # tracks the last column number that was clicked
$LastColumnAscending = $false # tracks the direction of the last sort of this column
# Calculating the factors to multiply the Width and Height
$monitor = [System.Windows.Forms.Screen]::PrimaryScreen
$widthFactor = 850 / $monitor.WorkingArea.Width
$heightFactor = 800 / $monitor.WorkingArea.Height
# Create a FORM & Configure
$gForm = New-Object System.Windows.Forms.Form
$gForm.Text ='Shadow Session Connect'
$gForm.Width = $gForm.ClientRectangle.Width
$gForm.AutoSize = $true
$gForm.Size = New-Object System.Drawing.Size (1920 * $widthFactor), (1080 * $heightFactor)
$dBttn = New-Object System.Windows.Forms.Button
$dBttn.Text = 'Control'
$dBttn.Location = New-Object System.Drawing.Point(15,10)
$gForm.Controls.Add($dBttn)
$gForm.StartPosition = "CenterScreen"
#$gForm.StartPosition.BringToFront()
# Create a LIST & Configure
$dList = New-Object System.Windows.Forms.ListView
$dList.Location = New-Object System.Drawing.Point(0,50)
$dList.Width = $gForm.ClientRectangle.Width
$dList.Height = $gForm.ClientRectangle.Height
$dList.Anchor = "Top, Left, Right, Bottom"
$dList.MultiSelect = $False
$dList.View = 'Details'
```

```
$dList.FullRowSelect = 1;
$dList.GridLines = 1
$dList.Scrollable = 1
$gForm.Controls.add($dList)
# Add columns to the ListView
$Header = "SESSIONNAME", "USERNAME","ID", "STATUS", "RDS HOST", "COMMAND"
$dList.Columns.Add("SESSIONNAME") | Out-Null
$dList.Columns.Add("USERNAME") | Out-Null
$dList.Columns.Add("ID") | Out-Null
$dList.Columns.Add("STATUS") | Out-Null
$dList.Columns.Add("RDS HOST") | Out-Null
$dList.Columns.Add("COMMAND") | Out-Null
## Query each RDS session. if we add an RDSH server we want to add the below code for the new RDSH server.
## RDSH = SERVER:RDS-10
$(qwinsta.exe /SERVER:RDS-10 | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," | ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dListItem.Subitems.Add("RDS-10") | Out-Null
$dListItem.Subitems.Add("mstsc /v:RDS-10 /shadow:" + $_.ID + " /control /noConsentPrompt") | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
## RDSH = SERVER:RDS-11
$(qwinsta.exe /SERVER:RDS-11 | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," | ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dListItem.Subitems.Add("RDS-11") | Out-Null
$dListItem.Subitems.Add("mstsc /v:RDS-11 /shadow:" + $_.ID + " /control /noConsentPrompt") | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
## RDSH = SERVER:RDS-12
$(qwinsta.exe /SERVER:RDS-12 | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," | ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dListItem.Subitems.Add("RDS-12") | Out-Null
$dListItem.Subitems.Add("mstsc /v:RDS-12 /shadow:" + $_.ID + " /control /noConsentPrompt") | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
## RDSH = SERVER:RDS-13
$(qwinsta.exe /SERVER:RDS-13 | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," | ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dListItem.Subitems.Add("RDS-13") | Out-Null
$dListItem.Subitems.Add("mstsc /v:RDS-13 /shadow:" + $_.ID + " /control /noConsentPrompt") | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
## RDSH = SERVER:RDS-14
$(qwinsta.exe /SERVER:RDS-14 | findstr "Active") -replace "^[\s>]" , "" -replace "\s+" , "," | ConvertFrom-Csv -Header $Header | ForEach-Object {
$dListItem = New-Object System.Windows.Forms.ListViewItem($_.SESSIONNAME)
$dListItem.Subitems.Add($_.USERNAME) | Out-Null
$dListItem.Subitems.Add($_.ID) | Out-Null
$dListItem.Subitems.Add($_.STATUS) | Out-Null
$dListItem.Subitems.Add("RDS-14") | Out-Null
$dListItem.Subitems.Add("mstsc /v:RDS-14 /shadow:" + $_.ID + " /control /noConsentPrompt") | Out-Null
$dList.Items.Add($dListItem) | Out-Null
}
## if the column header clicked, note the column selected.
$dList.add_ColumnClick({SortListView $_.Column})
## Code to sort selected column Ascending or Descending.
## PowerShell: Sort a Windows Forms ListView without a Custom Comparer
## https://etechgoodness.wordpress.com/2014/02/25/sort-a-windows-forms-listview-in-powershell-without-a-custom-comparer/
function SortListView
{
param([parameter(Position=0)][UInt32]$Column)
$Numeric = $true # determine how to sort
# if the user clicked the same column that was clicked last time, reverse its sort order. otherwise, reset for normal ascending sort
if($Script:LastColumnClicked -eq $Column)
{
$Script:LastColumnAscending = -not $Script:LastColumnAscending
}
else
{
$Script:LastColumnAscending = $true
}
$Script:LastColumnClicked = $Column
$ListItems = @(@(@())) # three-dimensional array; column 1 indexes the other columns, column 2 is the value to be sorted on, and column 3 is the System.Windows.Forms.ListViewItem o
foreach($ListItem in $dList.Items)
{
```

```
# if all items are numeric, can use a numeric sort
if($Numeric -ne $false) # nothing can set this back to true, so don't process unnecessarily
{
try
{
$Test = [Double]$ListItem.SubItems[[int]$Column].Text
}
catch
{
$Numeric = $false # a non-numeric item was found, so sort will occur as a string
}
}
$ListItems += ,@($ListItem.SubItems[[int]$Column].Text,$ListItem)
}
# create the expression that will be evaluated for sorting
$EvalExpression = {
if($Numeric)
{ return [Double]$_[0] }
else
{ return [String]$_[0] }
}
# all information is gathered; perform the sort
$ListItems = $ListItems | Sort-Object -Property @{Expression=$EvalExpression; Ascending=$Script:LastColumnAscending}
## the list is sorted; display it in the listview
$dList.BeginUpdate()
$dList.Items.Clear()
foreach($ListItem in $ListItems)
{
$dList.Items.Add($ListItem[1])
}
$dList.EndUpdate()
}
$dBttn.Add_Click( {
$SelectedItem = $dList.SelectedItems[0]
if ($SelectedItem -eq $null) {
[System.Windows.Forms.MessageBox]::Show("Select a user session to connect ")
}else
{
$session_id = $SelectedItem.subitems[2].text
$rds_host = $SelectedItem.subitems[4].text
$(mstsc /v:$rds_host /shadow:$session_id /control /noConsentPrompt)
"mstsc /v:" +$rds_host+ " /shadow:" +$session_id+ " /control /noConsentPrompt" | Clip ## copy to clipboard ##
#[System.Windows.Forms.MessageBox]::Show($session_id & " – " & $rds_host)
}
}
)
## Sort the list view "USERNAME" ascending order
SortListView(1)
$dList.columns[0].width = 75
$dList.columns[1].width = 150
$dList.columns[2].width = 50
$dList.columns[3].width = 75
$dList.columns[4].width = 100
$dList.columns[5].width = 400
## Show form to user.
$gForm.TopMost = $true
$gForm.ShowDialog()
##$gForm.topmost = $true
```

EIGHTY

Reply

June 5, 2024 - 6:19 am

We use Shadowing to support users on AVD hosts where no one should have local admin permissions especially support staff who may not understand config changes and app installs need to be made to all hosts in a host pool or user experience will be inconsistent.
How can we perform the following config via GPO?
wmic /namespace:\\root\CIMV2\TerminalServices PATH Win32_TSPermissionsSetting WHERE (TerminalName="RDP-Tcp") CALL AddAccount "woshub\AllowRDSShadow",2

ADMIN

Reply

June 6, 2024 - 8:49 am

You can run this code as a GPO logon script:
https://woshub.com/running-powershell-startup-scripts-using-gpo/
to run this script only once on each computer:
https://woshub.com/run-gpo-logon-script-once/

LEAVE A COMMENT

Your Comment

Name*

Email*

Website

NOTIFY ME OF FOLLOWUP COMMENTS VIA E-MAIL. YOU CAN ALSO SUBSCRIBE WITHOUT COMMENTING.

POST COMMENT

CURRENT YE@R *

4.4

FACEBOOK TWITTER TELEGRAM

Popular Posts

How to Repair EFI/GPT Bootloader on Windows 10 or 11

How to Restore Deleted EFI System Partition in Windows

Network Computers are not Showing Up in Windows 10/11

Install and Manage Windows Updates with PowerShell (PSWindowsUpdate)

How to Download Offline Installer (APPX/MSIX) for Microsoft Store App

Updating List of Trusted Root Certificates in Windows

Fix: Windows Cannot Connect to a Shared Printer

BACK TO TOP