Finding RDP sessions on servers using PowerShell

Have you ever needed to use RDP to get to a server console for some local admin work and then been bounced out because there are already active sessions? Or have you had your Active Directory account locked out because of an open RDP session with the old password sending locks to the domain?

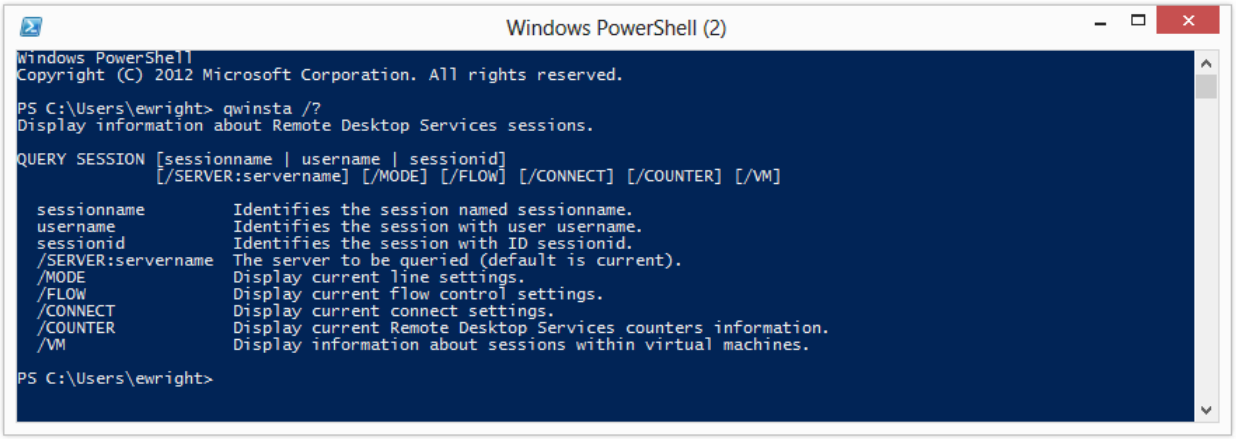
Rather than having to run the Remote Desktop Manager and search between servers, I decided to build a little script which will scan Active Directory for computers running server OS and list out the active and disconnected sessions.

The magic of QWINSTA

One of the challenges with this activity is that PowerShell does not have a native CmdLet to extract RDP information from servers. Luckily, we can use a hybrid approach here to solve that problem.

WIndows ships with two tools named QWINSTA.exe and RWINSTA.exe for querying and resetting Remote Desktop Services sessions. For our purposes we will use QWINSTA (apparently stands for Query WINdows STAtion).

You can find the options in the command line by typing QWINSTA /? as shown here:



The Process

There is a simple flow to the script which is:

1. Query Active Directory for Servers
2. Run QWINSTA to extract the session information
3. If a session exists, read the username and session type
4. Log the username and session type to a variable
5. Email the results

There are some setup options for email parameters that are done and most importantly the Import-Module CmdLet is used to load the Active Directory module so that we can use Get-ADComputer. This will require that you have the AD PowerShell environment on the workstation or server that you are running this process from.

I've added some sections in the script which are commented out but you can uncomment them to have them output to the console so that you can see what is happening while the script is running. This is handy for troubleshooting in the case that there are issues.

Adding the QWINSTA Command

Because we have to use an external command in our script we will simply use the command inline with other script actions. The interesting thing about this is that we can run the command easily and pass the computer name parameter which we have assigned to $ServerName:

```
qwinsta /server:$ServerName
```

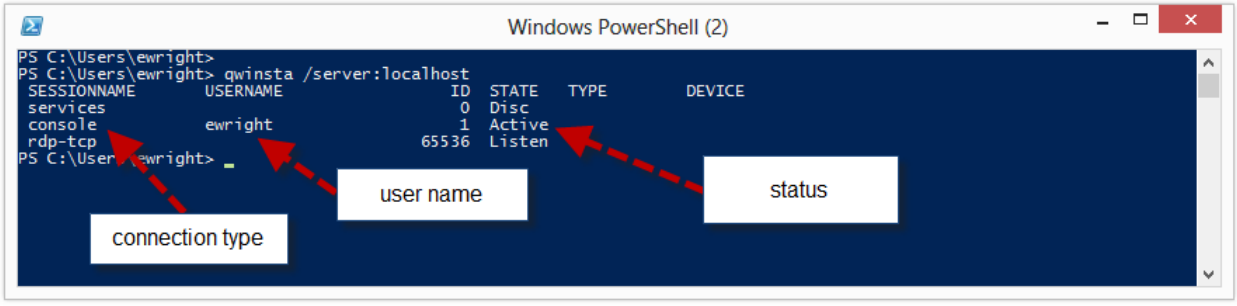
To be able to leverage the PowerShell goodness we have to assign the output to a variable so that we can reuse it as needed:

```
$queryResults = qwinsta /server:$ServerName
```

Luckily PowerShell makes it just that easy to capture the output of external commands as well as native CmdLets.

Managing the QWINSTA Output

When you look at the script you can see that the QWINSTA command line is not quite as simple as I've shown above. This is because the output of the QWINSTA command isn't in a neat and tidy one line format like we really would prefer:



This is why we have a much more complex command line in the script:

```
$queryResults = (qwinsta /server:$ServerName | foreach { (($_.trim() -replace
“s+”,”,”))} | ConvertFrom-Csv)
```

We are taking each line of the output from the command, trimming the content and replacing the spaces with commas and then piping it to a ConvertFrom-CSV to put the content into a table array with the appropriate headers of USERNAME and SESSIONNAME which is the information we are looking for.

Logging and Emailing the Results

The logging of the output is quite simple. We just want to collect each line as we loop through the sessions and the add the content into a variable ($SessionList).

Once we have completed querying each of the computers, the loop exits and then we put together our email message using the Send-MailMessage CmdLet:

```
Send-MailMessage -To $emailTo -Subject $subject -Body $SessionList -SmtpServer
$smtpServer -From $emailFrom -Priority $priority
```

The parameters for the Send-MailMessage command come from those defined in the static variables at the beginning of the script and with the $SessionList which was created during the script run.

The Script

Once we’ve put it all together it is a nifty and fairly lightweight script. I’ve put the script up on my TechNet Gallery and you can find it here:
http://gallery.technet.microsoft.com/PowerShell-script-to-Find-d2ba4252

Here is the script in it’s full form to read through. I hope that you are able to make use of it. An ideal situation is to have it run as a batch process every night to scan the environment and email your Systems Admin team so that they know what sessions are left overnight.

```
# Import the Active Directory module for the Get-ADComputer CmdLet

Import-Module ActiveDirectory

# Get today’s date for the report

$today = Get-Date

# Setup email parameters

$subject = “ACTIVE SERVER SESSIONS REPORT – ” + $today

$priority = “Normal”

$smtpServer = “YourMailServer”
```

```
$emailFrom = “email@yourdomain.com”

$emailTo = “email@yourdomain.com”

# Create a fresh variable to collect the results. You can use this to output as desired

$SessionList = “ACTIVE SERVER SESSIONS REPORT – ” + $today + “`n`n”

# Query Active Directory for computers running a Server operating system

$Servers = Get-ADComputer -Filter {OperatingSystem -like “*server*”}

# Loop through the list to query each server for login sessions

ForEach ($Server in $Servers) {

$ServerName = $Server.Name

# When running interactively, uncomment the Write-Host line below to show which server is being queried

# Write-Host “Querying $ServerName”

# Run the qwinsta.exe and parse the output

$queryResults = (qwinsta /server:$ServerName | foreach { (($_.trim() -replace “s+”,”,”))} | ConvertFrom-Csv)

# Pull the session information from each instance

ForEach ($queryResult in $queryResults) {

$RDPUser = $queryResult.USERNAME

$sessionType = $queryResult.SESSIONNAME

# We only want to display where a “person” is logged in. Otherwise unused sessions show up as USERNAME as a number

If (($RDPUser -match “[a-z]”) -and ($RDPUser -ne $NULL)) {

# When running interactively, uncomment the Write-Host line below to show the output to screen

# Write-Host $ServerName logged in by $RDPUser on $sessionType

$SessionList = $SessionList + “`n`n” + $ServerName + ” logged in by ” + $RDPUser + ” on ” + $sessionType    }

}

}

# Send the report email
```

```
Send-MailMessage -To $emailTo -Subject $subject -Body $SessionList -SmtpServer $smtpServer -From $emailFrom -Priority $priority

# When running interactively, uncomment the Write-Host line below to see the full list on screen

# $SessionList
```