# **PowerShell Script for Hourly Logged-In User Monitoring.**

This guide provides a PowerShell script and setup instructions to monitor all logged-in users on a Windows system every hour. The script records the results (with Central Time timestamps) in a human-readable text format, and either emails the results or saves them to a log file, depending on email availability. Follow the steps below to implement the script and schedule it in Task Scheduler.

## **PowerShell Script Overview**

The PowerShell script performs the following actions (Source links are colored in light blue):

* **Identify Logged-In Users:** It uses the built-in query user command (also known as query.exe) to list all currently logged on users/sessions ([windows - Powershell Get Active logged in user in local machine - Super User](https://superuser.com/questions/1186568/powershell-get-active-logged-in-user-in-local-machine#:~:text=The%20issue%20with%20%60Get,exe)). This approach is reliable for finding interactive user sessions (local or RDP) on the machine.
* **Timestamp in Central Time:** The script gets the current time and converts it to US Central Time using .NET’s TimeZoneInfo API ([TimeZoneInfo.ConvertTimeFromUtc(DateTime, TimeZoneInfo) Method (System) | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/api/system.timezoneinfo.converttimefromutc?view=net-9.0" \l ":~:text=TimeZoneInfo%20cstZone%20%3D%20TimeZoneInfo.FindSystemTimeZoneById%28,1)). This ensures the log timestamp is in Central Time regardless of the system’s local timezone.  
  \*Note\*: Adjust your timezone accordingly.
* **Format Output:** The script formats the data into a human-readable text line, including the Central Time timestamp and the list of logged-in user accounts. For example:  
  2025-03-05 15:00:00 (Central Time) - Logged-in users: UserA, UserB
* **Email or Log to File:** The script attempts to send the results via an Outlook email if an email client (like Microsoft Outlook) is available. It uses the Outlook COM object to create and send an email ([Send mail to Myself using Powershell - Stack Overflow](https://stackoverflow.com/questions/68666255/send-mail-to-myself-using-powershell#:~:text=%24Outlook%20%3D%20New,%24Mail.Send)). If email sending fails (e.g., no Outlook profile or SMTP configured), the script will fall back to appending the result line to a local log text file.

Below is the complete PowerShell script. You can modify the email recipient, log file path, or other parameters as needed:

## ===========================================================================

## NAME:        LogLoggedInUsers.ps1

## CREATED:     05-MAR-2025

## BY:          DAVID RADOICIC

## VERSION:     1.0

## DESCRIPTION: Checks the Windows system for logged in users and sends an email.

##

##

## NOTE:

## The script first determines the current Central Time timestamp and configures the timezone.

## Script calls query user to list all logged-in users; this command returns all active user sessions on the system​.

## It then parses the output to extract usernames and joins them into a single comma-separated string.

## It composes an email using the Outlook COM interface and then sends the email.

# Timezone configuration: Convert current time to Central Time (CT)

try {

    $CTzone = [System.TimeZoneInfo]::FindSystemTimeZoneById("Central Standard Time")

} catch {

    Write-Error "Central Timezone not found on system."

    $CTzone = $null

}

if ($CTzone) {

    $timestamp = [System.TimeZoneInfo]::ConvertTimeFromUtc((Get-Date).ToUniversalTime(), $CTzone)

    $timestampStr = $timestamp.ToString("yyyy-MM-dd HH:mm:ss")

    $tzLabel = "(Central Time)"

} else {

    # Fallback to local time if CT zone not available

    $timestampStr = (Get-Date).ToString("yyyy-MM-dd HH:mm:ss")

    $tzLabel = "(Local Time)"

}

# Get all logged-in users using 'query user' (fallback to query session if needed)

$rawOutput = try { (query user) } catch { (query session) }  # query session is similar alias

$usersList = "None"

if ($rawOutput) {

    $lines = $rawOutput | Select-Object -Skip 1    # skip header line

    $loggedOnUsers = @()

    foreach ($line in $lines) {

        if (-not [string]::IsNullOrWhiteSpace($line)) {

            # Remove any leading '>' indicator and trim whitespace

            $clean = $line.TrimStart('>').Trim()

            # 'query user' output columns are separated by whitespace; first token is username

            $username = $clean -split '\s+' | Select-Object -First 1

            $loggedOnUsers += $username

        }

    }

    if ($loggedOnUsers.Count -gt 0) {

        $usersList = ($loggedOnUsers | Sort-Object -Unique) -join ', '

    }

}

# Prepare the log message line

$logLine = "$($timestampStr) $tzLabel - Logged-in users: $usersList"

# Try to send email via Outlook (if available)

$sent = $false

try {

    $Outlook = New-Object -ComObject Outlook.Application

    $Mail = $Outlook.CreateItem(0)

    $Mail.To = "YOUR EMAIL HERE"               # recipient email

    $Mail.Subject = "Logged-in Users Report ($($timestampStr) CT)"

    $Mail.Body = $logLine

    $Mail.Send()                                    # send email

    $sent = $true

} catch {

    # Could not send email (Outlook not installed or other issue)

    $sent = $false

}

# If email not sent, write to a log file

if (-not $sent) {

    $logFile = "C:\Logs\LoggedInUsers.log"          # path to log file

    # Ensure directory exists

    $dir = Split-Path $logFile -Parent

    if (-not (Test-Path $dir)) { New-Item -Path $dir -ItemType Directory | Out-Null }

    Add-Content -Path $logFile -Value $logLine

}

## **How it works:**

The script first determines the current Central Time timestamp using FindSystemTimeZoneById("Central Standard Time") and converting from UTC ([TimeZoneInfo.ConvertTimeFromUtc(DateTime, TimeZoneInfo) Method (System) | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/api/system.timezoneinfo.converttimefromutc?view=net-9.0" \l ":~:text=TimeZoneInfo%20cstZone%20%3D%20TimeZoneInfo.FindSystemTimeZoneById%28,1)). Then it calls query user to list all logged-in users; this command returns all active user sessions on the system ([windows - Powershell Get Active logged in user in local machine - Super User](https://superuser.com/questions/1186568/powershell-get-active-logged-in-user-in-local-machine#:~:text=The%20issue%20with%20%60Get,exe)). It parses the output to extract usernames and joins them into a single comma-separated string. Next, it composes an email using the Outlook COM interface – this leverages an installed Outlook client to send the email ([Send mail to Myself using Powershell - Stack Overflow](https://stackoverflow.com/questions/68666255/send-mail-to-myself-using-powershell#:~:text=%24Outlook%20%3D%20New,%24Mail.Send)). The email’s body contains the timestamp and user list. If the email step fails (for example, if Outlook is not available or no email profile is configured), the script logs the output line to a text file (C:\Logs\LoggedInUsers.log).

**Note:** The script uses Outlook automation for emailing. This requires Microsoft Outlook to be installed and configured on the system (with a default mail profile). If you prefer sending via SMTP without Outlook, you could replace that section with the Send-MailMessage cmdlet and appropriate SMTP server settings. For instance, using Send-MailMessage -SmtpServer "your.smtp.server" -To "sampleemail@sample.com" -From "[email protected]" -Subject "..." -Body "..." if an SMTP server is available.

## **Scheduling the Script with Task Scheduler**

Once the PowerShell script is ready, use Windows Task Scheduler to run it automatically every hour. Below are the steps to set up the scheduled task:

1. **Save the Script:** Save the above PowerShell code to a file, for example: C:\Scripts\LogLoggedInUsers.ps1. Make sure the path has no spaces or adjust quoting in the task action accordingly.
2. **Open Task Scheduler:** Click Start, search for "Task Scheduler", and open it. In Task Scheduler, select **Create Task** (do not use the basic task wizard for advanced settings).
3. **General Tab:** Give the task a name like "Log Logged-In Users Hourly". Optionally add a description. Choose **"Run whether user is logged on or not"** if you want the script to run in the background. Use an account with appropriate privileges (e.g., an admin or SYSTEM) so it can query all user sessions. If using a specific user account, check "Do not store password" only if the task doesn't need network access (storing password allows non-interactive run).
4. **Triggers Tab:** Click **New** to create a trigger. Set it to begin **On a schedule**. Choose **Daily**, and set the start date/time (e.g., start at 12:00 AM or any convenient time). Then enable **Advanced settings** -> **Repeat task every** and select **1 hour**, with a duration of **Indefinitely** (so it repeats every hour continuously) ([Windows Task Scheduler top of the hour : r/sysadmin](https://www.reddit.com/r/sysadmin/comments/am2ndc/windows_task_scheduler_top_of_the_hour/#:~:text=Be%20sure%20you%20create%20task,for%20a%20duration%20of%20indefinitely)). Ensure the trigger is enabled.
5. **Actions Tab:** Click **New** to create the action. For *Action*, select **Start a program**. In the **Program/script** field, enter:  
   powershell.exe  
   In the **Add arguments (optional)** field, enter the execution policy bypass and the path to your script. For example:  
   -ExecutionPolicy Bypass -File "C:\Scripts\LogLoggedInUsers.ps1"  
   This tells Task Scheduler to run the PowerShell script. (Using -ExecutionPolicy Bypass ensures the script runs even if the system’s execution policy is restrictive.)
6. **Conditions Tab (optional):** Adjust any conditions as needed. For instance, you can uncheck “Start the task only if the computer is on AC power” if you want it to run on battery. In most cases, defaults are fine.
7. **Settings Tab:** Ensure **Allow task to be run on demand** is checked (so you can manually test it). You can also check **“Run task as soon as possible after a scheduled start is missed”** to catch up if the computer was off at the scheduled time. Make sure **If the task is already running, do not start a new instance** (or choose what fits your needs to prevent overlap if an hourly run takes longer than an hour, though this script should finish quickly).
8. **Save the Task:** Click **OK** to save. If you chose "Run whether user is logged on or not" with a specific account, you’ll be prompted to enter that account’s password for the task to store. After saving, you should see your task listed in Task Scheduler Library.
9. **Test the Task:** Right-click your new task and select **Run** to test it. The script should execute immediately. If Outlook is configured and running, you should receive an email at sampleemail@sample.com with the current logged-in users. If not, check the log file (e.g., C:\Logs\LoggedInUsers.log) for a new entry. Also verify in Task Scheduler’s **History** tab that the task ran successfully. (If History is disabled, you can enable all task history from the Actions pane for debugging.)

By following these steps, the PowerShell script will run every hour via Task Scheduler. It will capture all logged-on user accounts and record the info with a Central Time timestamp. If an email system is available, the results will be emailed to the specified address; otherwise, they will be saved in a log file on the computer for later review. This setup helps in auditing or monitoring user login activity on an hourly basis and can be customized further as needed.