

Upload Log Files to Azure Blob Storage via PowerShell

When troubleshooting Windows installations, upgrades, or updates, collecting log files like setupact.log or WindowsUpdate.log is essential. This blog provides a simple PowerShell script to automatically upload these files to Azure Blob Storage, making it easier to gather diagnostics from end-user devices. The script is designed for flexible use without any requirement for PowerShell modules. It can either be run manually or deployed through tools like Intune, SCCM, or NinjaOne. The goal is to collect logs without interfering the work of end users.

Security considerations

To keep data secure, use a SAS (Shared Access Signature) token that is limited to write-only access and has a defined expiration. Make sure to use a separate container for this automation. Don't let the SAS token be able to access other locations than the specified container. Avoid hardcoding credentials into shared scripts, and store tokens securely if possible.

Prerequisites

To use this script, you'll need an Azure Storage account with a container and a valid SAS (Shared Access Signature) token that allows write access (More information: [Create SAS tokens for storage containers and blobs | Microsoft Learn](#)). Make sure that the script gets executed with administrator rights, since some paths for the log sources are in system protected folders. Finally, ensure the device has internet access and can reach Azure Blob endpoints over HTTPS.

PowerShell Script

This script gathers key log files from standard Windows locations and uploads them securely to Azure Blob Storage using a SAS token. Uploaded logs are automatically named and organized into folders based on the device's hostname and timestamp. The script also includes basic error handling and checks if files exist before attempting upload.

Storage and File Locations

The script uploads files to a specific Azure Blob container under a folder named after the device's hostname. Each file is named with a timestamp, hostname, and sanitized file path, such as:

```
PC123_logs/250614_135200_PC123_C_Windows_Panther_setupact.log
```

```
function Upload-LogFilesToBlob {
    param (
        [Parameter(Mandatory = $true)]
        [string[]]$Path
    )

    # Set your static SAS URL prefix
    $baseUploadUrl = "https://<storageaccountname>.blob.core.windows.net/<rootfolder>"
    $sasToken = "<storageaccountsastoken>"

    foreach ($filePath in $Path) {
        if (-Not (Test-Path $filePath)) {
            Write-Warning "File not found: $filePath"
            continue
        }

        # Escape full path to make it filename-safe
        $escapedPath = $filePath -replace '[\V:*?"<>|]', '_'

        # Construct blob name inside a folder named by hostname
        $blobFolder = "$(hostname)_logs"
        $blobName = "$blobFolder/$(Get-Date -Format 'yyMMdd_HHmmss')_$(hostname)_$escapedPath"

        $uploadUrl = "$baseUploadUrl/$blobName$sasToken"

        $headers = @{
            "x-ms-blob-type" = "BlockBlob"
        }

        $fileBytes = [System.IO.File]::ReadAllBytes($filePath)

        try {
            Invoke-RestMethod -Uri $uploadUrl -Method Put -Headers $headers -Body $fileBytes
            Write-Output "Uploaded: $filePath"
        }
    }
}
```

```
}  
catch {  
    Write-Warning "Failed to upload: $_"  
}  
}  
}  
  
Get-WindowsUpdateLog -LogPath "C:\Windows\Logs\WindowsUpdateLog.log"  
  
Upload-LogFilesToBlob -Path @(  
    "C:\Windows\Panther\setupact.log",  
    "C:\Windows\Panther\setuperr.log",  
    "C:\Windows.old\Windows\Panther\setupact.log",  
    "C:\Windows.old\Windows\Panther\setuperr.log",  
    "C:\Windows\Logs\WindowsUpdateLog.log"  
)
```

Revision #2

Created 12 June 2025 11:30:04 by Luca Noah Caprez

Updated 16 June 2025 11:14:14 by Luca Noah Caprez