PSMail

Skript zum Versenden von Mails

Vorarlberger Standardschulinstallation
Autor: Lukas Franz
Besuchen Sie uns im Internet
http://www.vobs.at/rb

Inhalt

1. Vorbemerkung

Das Skript PSMail30.ps1 dient zum einfachen Versenden von Mails mit vorher definiertem Absender und Empfänger. Es wird in erster Linie aus anderen Skripts heraus aufgerufen werden. Dabei können Betreff, Inhalt der Mail und bei Bedarf auch Anhänge übergeben werden.
Die Skripts wurden ausgiebig getestet und sollen zur Erleichterung der täglichen Administrator-Arbeit dienen.
Vom Autor wird keinerlei Verantwortung für eventuelle Schäden, Daten- oder Zeitverluste übernommen! Da sämtliche Skript-Zeilen aus diversen Internet-Foren und Hilfeseiten stammen, steht es jedem Anwender frei, die Dateien weiter zu verwenden, zu verändern und nach eigenen Bedürfnissen anzupassen.
Das Skript funktioniert grundsätzlich in allen aktuellen PowerShell-Versionen. Da verschiedene PowerShell-Versionen aber verschiedene Standard-Zeichensätze verwenden, kann es bei der Ausgabe von Text dazu kommen, dass Umlaute falsch dargestellt werden. Auf die Funktion der Programmteile an sich hat dies keinen Einfluss! Grundsätzlich sollte auf Umlaute in Benutzernamen, Kennwörtern usw. verzichtet werden.

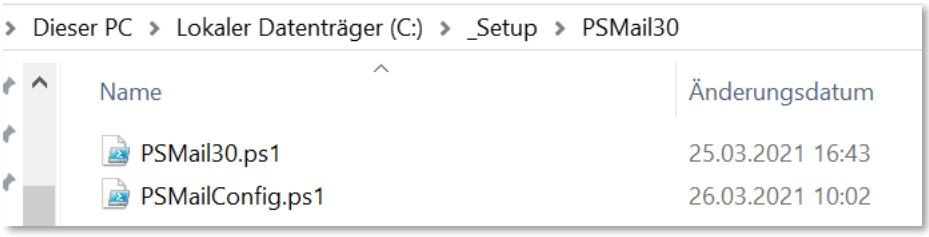
2. Kurzbeschreibung und Installation

Das Paket besteht aus 2 PowerShell-Skripts:

PSMail30.ps1: Dies ist das eigentliche Skript zum Versenden von Mails. In diesem muss nichts konfiguriert werden. Die Anwendung des Skripts wird im Punkt 3 erläutert.

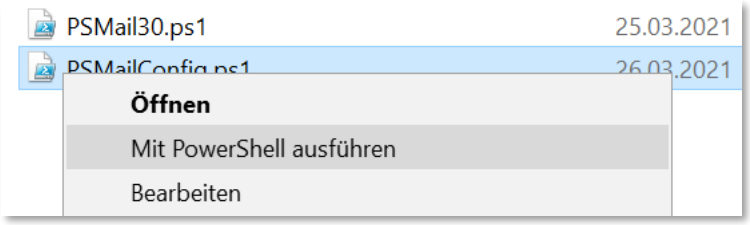
PSMailConfig.ps1: Mit diesem Skript werden zwei Konfigurationsdateien erstellt, welche mit individuellen Informationen (Mailserver-Daten, Passwörter…) befüllt werden.

Die Installation an sich erfolgt durch das einfache Kopieren der beiden ps1-Dateien in ein beliebiges Verzeichnis. Empfehlung: C:_Setup\PSMail30.



3. Erstkonfiguration

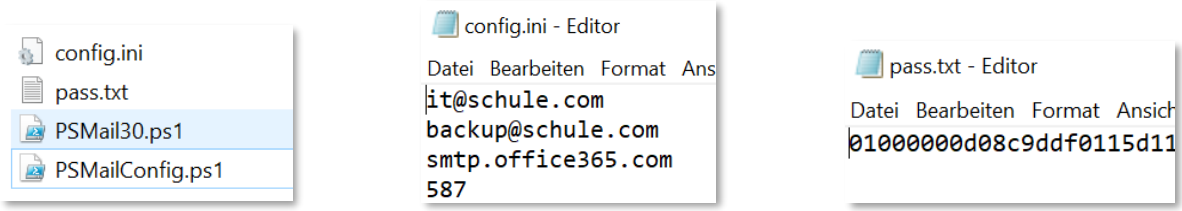
Für die erste Konfiguration wird die Datei PSMailConfig.ps1 mit PowerShell aufgerufen. Achtung: Allenfalls bereits erstellte Konfigurationsdateien werden unmittelbar beim Aufruf des Skripts überschrieben.
HINWEIS: unter Umständen müssen die Ausführungsrichtlinien für PowerShell (siehe Anhang) noch angepasst werden!



Man wird durch das Skript geführt und zur Eingabe der individuellen Informationen für den Versand der Mails aufgefordert.
Am Ende können die eingegebenen Daten kontrolliert werden. Das eingegebene Passwort wird hier nicht angezeigt.

Wenn alles passt und mit „j“ bestätigt wird, werden zwei Dateien ins Programmverzeichnis geschrieben: config.ini und pass.txt.

config.ini: Hier sind die zuvor eingegeben Daten im Klartext gespeichert.
pass.txt: Hier ist das eingegebene Passwort in verschlüsselter Form gespeichert.



… …

Außerdem wird ein Testmail versendet. Dieses Senden kann - verschiedenen Umständen entsprechend - einige Sekunden oder Minuten dauern.

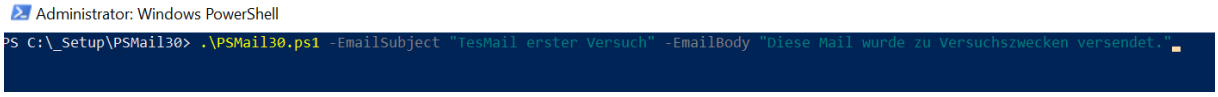
```
Stimmen die Daten? j/n: j

Testmail wird gesendet und Skript beendet...
```

Wenn die Eingaben nicht stimmen und mit „n“ (oder irgendeiner anderen Taste) bestätigt wird, beendet sich das Skript, die Konfigurationsdateien werden nicht erstellt bzw. wieder gelöscht.

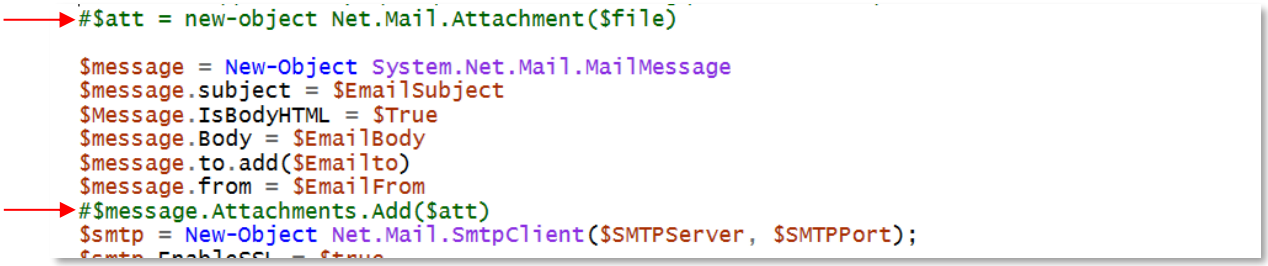
Wird das Testmail korrekt zugestellt, ist PSMail30 richtig konfiguriert und kann entsprechend angewendet werden.

4. Praktische Anwendung von PSMail30

Ein Mail wird nun versendet, indem aus PowerShell heraus das Skript ausgeführt wird und die beiden Parameter „EmailSubject“ und „EmailBody“ übergeben werden:



Mit dem Parameter „File“ und der Pfadangabe zu einer Datei, könnte auch ein Anhang mitgesendet werden. Die entsprechenden Zeilen müssen dazu jedoch direkt im Skript PSMail30.ps1 durch das Entfernen der Rauten am Beginn der jeweiligen Textzeilen auskommentiert werden.



```
#$att = new-object Net.Mail.Attachment($file)

$message = New-Object System.Net.Mail.MailMessage
$message.subject = $EmailSubject
$Message.IsBodyHTML = $True
$message.Body = $EmailBody
$message.to.add($Emailto)
$message.from = $EmailFrom
#$message.Attachments.Add($att)
$smtp = New-Object Net.Mail.SmtpClient($SMTPServer, $SMTPPort);
```

In der Praxis wird dieses Skript wohl meist in andere Skripts eingebaut werden, und nicht zum manuellen Versenden von Mails verwendet werden.

5. Beispiel zur Anwendung und Aufruf über den Taskplaner

Ein einfaches PowerShell-Skript zum Abrufen des Replikationsstatus auf dem host1 soll die gesammelten Informationen wöchentlich per Mail versenden.

ReHealth.ps1

```
#Variable
$Datum = Get-Date
$Sender = $env:COMPUTERNAME
$Schule = "MS Egg"
$DC = "DCSCHULE"
$SRV = "SERVER"


$ReplicationHealth = (Get-VMReplication -VMNAME $SRV)
    if ($ReplicationHealth.Health -eq "Normal") {
    $HealthSRV = "Normal"
    }
    else {
    $HealthSRV = "Kitisch oder Warnung!"
    }

$ReplicationHealth = (Get-VMReplication -VMNAME $DC)
    if ($ReplicationHealth.Health -eq "Normal") {
    $HealthDC = "Normal"
    }
    else {
    $HealthDC = "Kitisch oder Warnung!"
    }

$Body= "Dieses Mail wurde vom $Sender der $Schule gesendet <br>... $Datum<br>Replikationsstatus $SRV <b>$HealthSRV</b><br>Replikationsstatus $DC <b>$HealthDC</b>"

Set-Location C:\_Setup\PSMail30
.\PSMail30.ps1 -EmailSubject "$Schule Replikationsstatus - $Datum" -EmailBody $Body

```

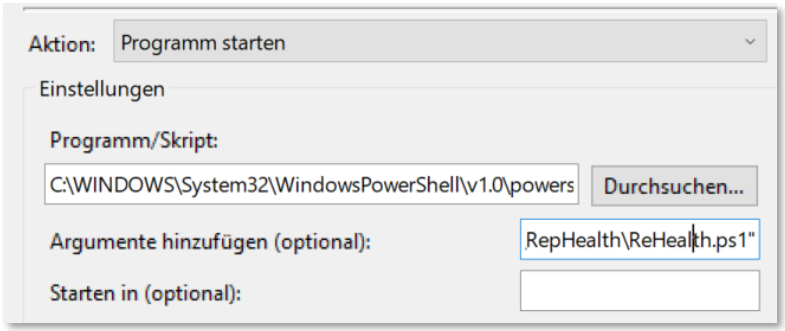
In Zeile 27 muss über das Skript das Arbeitsverzeichnis geändert werden, da sonst im aktuellen Verzeichnis nach den Konfigurationsdateien gesucht wird.
In Zeile 28 wird das Mail-Skript mit den zu übergebenden Parametern aufgerufen.
In Zeile 25 ist zu sehen, dass Variable auch „verschachtelt" verwendet werden können. Da das Mail im html-Format versendet wird, können Elemente wie
 für Zeilenumbrüche oder für fette Schrift verwendet werden.

Einbau im Taskplaner:
In der Aufgabenplanung wird ein Task erstellt und nach den eigenen Wünschen konfiguriert. Im Reiter Aktionen wird durch Klick auf „neu" eine Aktion Programm starten hinzugefügt. Im Feld Programm/Skript wird auf PowerShell verwiesen:

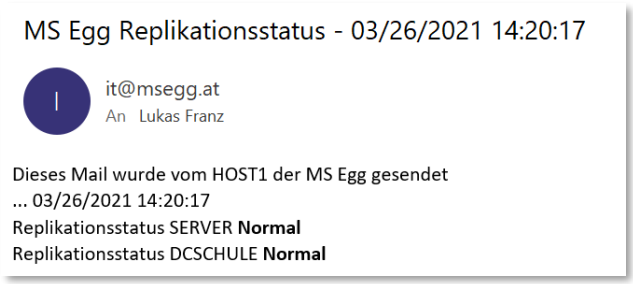
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe

Ins Feld Argumente wird das auszuführende Skript eingetragen:

-ExecutionPolicy Bypass "C:_Setup\RepHealth\ReHealth.ps1"



Hier werden mit „-ExecutionPolicy Bypass" die Ausführungsrichtlinien unabhängig von den Standardeinstellungen umgangen.
Nun wird z.B. wöchentlich ein Statusmail für die Replikation an die gewünschte Emailadresse gesendet.



6. Anhang – Inhalt der Skript-Dateien und weitere Informationen

6.1. PSMailConfig.ps1

```
#GrundSetup Skript
$HOST.UI.RawUI.BackgroundColor   = "White"
$HOST.UI.RawUI.ForegroundColor   = "Black"
Clear-Host

# Begrüßngstext Erstkonfiguration
write-host '
Mit diesem Powershell-Skript werden verschiedene Variable für
das Skript PSMail festgelegt und in ein Config-File geschrieben.
Außerdem wird das Passwort fuer den Mailversand verschlüsselt in
der Datei pass.txt abgepeichert.

'

# Eingabe der Variablen - in Config-File schreiben
"
Eingabe der Daten für den Versand der Mails:

"
new-item -Path config.ini -Force > $Null

Read-Host -Prompt "Emailadresse Absender" >> config.ini
$Mailto = Read-Host -Prompt "Emailadresse Empfänger" >> config.ini
Read-Host -Prompt "SMTP-Server" >> config.ini
Read-Host -Prompt "SMTP-Port (meistens 587)" >> config.ini

# Eingabe Passwort und Erstellen der verschlüsselten pass.txt
$password = Read-Host -Prompt "Zugehoeriges Passwort für den smtp-Versand eingeben" -AsSecureString
$password | ConvertFrom-SecureString | Out-File .\pass.txt

# Kontrolle mit bestätigen
"
Kontrolle der Daten: Wenn die Daten richtig sind, bestätigen sie mit 'j' - es wird ein Testmail versendet.
Ansonsten beendet sich das Skript und muss erneut ausgeführt werden.
"
$config = get-content .\config.ini
$SMTPServer = $config[2]
$SMTPPort = $config[3]
$EmailUser = $config[0]
$EmailFrom = $config[0]
$EmailTo = $config[1]
Write-Host "Emailadresse Absender - $EmailFrom"  -ForegroundColor Red
Write-Host "Emailadresse Empfänger - $EmailTo" -ForegroundColor Red
Write-Host "SMTP-Server - $SMTPServer" -ForegroundColor Red
Write-Host "SMTP-Port - $SMTPPort`n" -ForegroundColor Red

$proof = Read-Host -Prompt "Stimmen die Daten? j/n"

if ($proof -eq 'j') {
    #"Sende Testmail..."
    "`nTestmail wird gesendet und Skript beendet..."
    sleep 5
    & .\PSMail30.ps1 -EmailSubject "TestMail PSMail30" -EmailBody "Dieses Mail wurde zum Test versendet an $Emailto."
    }
    else {
        remove-item .\config.ini -Force
        remove-item .\pass.txt -Force
        #GrundSetup Farbe wieder zurückstellen
        $HOST.UI.RawUI.BackgroundColor   = "Black"
        $HOST.UI.RawUI.ForegroundColor   = "Gray"
        Clear-Host
        Write-Host "`nSkript wird beendet und muss erneut ausgeführt werden!" -ForegroundColor Red
        sleep 5
        exit
    }

#GrundSetup Farbe wieder zurückstellen
$HOST.UI.RawUI.BackgroundColor   = "Black"
$HOST.UI.RawUI.ForegroundColor   = "Gray"
Clear-Host
```

6.2. PSMail30.ps1

```
#Definition der Parameter, welche übergeben werden
Param(
    [string]$EmailSubject,
    [string]$EmailBody,
    [string]$File
)

#Variable aus config.ini importieren
$config = get-content .\config.ini

$SMTPServer = $config[2]
$SMTPPort = $config[3]
$EmailUser = $config[0]
$EmailFrom = $config[0]
$Password = Get-Content .\pass.txt  | ConvertTo-SecureString
$EmailTo = $config[1]
#$EmailSubject = "BeispielSubject - könnte hier auch manuell eingetragen werden"
#$EmailBody = "BeispielBody - könnte hier auch manuell eingetragen werden"

#Wird ein Anhang mitgesendet, müssen die entsprechenden Zeilen auskommentiert
#und allenfalls der richtige Parameter übergeben werden
#$att = new-object Net.Mail.Attachment($File)

$message = New-Object System.Net.Mail.MailMessage
$message.subject = $EmailSubject
$Message.IsBodyHTML = $True
$message.Body = $EmailBody
$message.to.add($Emailto)
$message.from = $EmailFrom
#$message.Attachments.Add($att)
$smtp = New-Object Net.Mail.SmtpClient($SMTPServer, $SMTPPort);
$smtp.EnableSSL = $true
$smtp.Credentials = New-Object System.Net.NetworkCredential($EmailUser, $Password);
$smtp.send($message)
```

6.3. RepHealth.ps1

```
#Variable
$Datum = Get-Date
$Sender = $env:COMPUTERNAME
$Schule = "MS Egg"
$DC = "DCSCHULE"
$SRV = "SERVER"


$ReplicationHealth = (Get-VMReplication -VMNAME $SRV)
    if ($ReplicationHealth.Health -eq "Normal") {
    $HealthSRV = "Normal"
    }
    else {
    $HealthSRV = "Kritisch oder Warnung!"
    }

$ReplicationHealth = (Get-VMReplication -VMNAME $DC)
    if ($ReplicationHealth.Health -eq "Normal") {
    $HealthDC = "Normal"
    }
    else {
    $HealthDC = "Kritisch oder Warnung!"
    }

$Body= "Dieses Mail wurde vom $Sender der $Schule gesendet <br>... $Datum<br>Replikationsstatus $SRV <b>$HealthSRV</b><br>Replikationsstatus
$DC <b>$HealthDC</b>"

Set-Location C:\_Setup\PSMail30
.\PSMail30.ps1 -EmailSubject "$Schule Replikationsstatus - $Datum" -EmailBody $Body
```

6.4. Ausführungsrichtlinien (Execution Policy) für PowerShell-Scripts über GPO setzen:

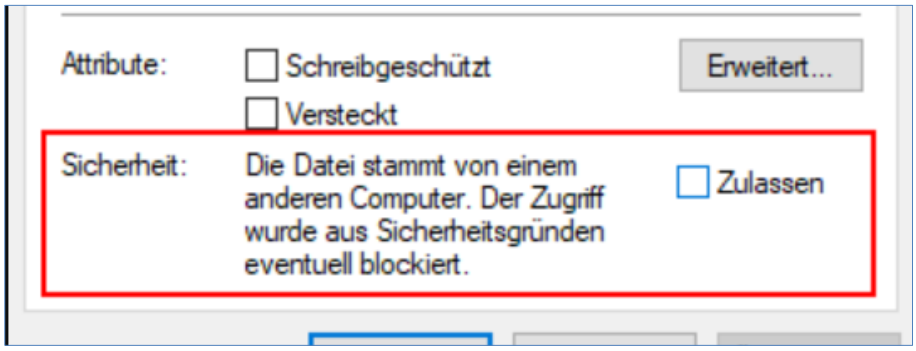
https://www.windowspro.de/script/ausfuehrungsrichtlinien-executionpolicy-fuer-powershell-scripts-ueber-gpo-setzen

6.5. SMTP-Daten für MS365 bzw. Selbstverwalterschulen:

Servername: smtp.office365.com
Port: 587
Verschlüsselungsmethode: STARTTLS
Die Verschlüsselungsmethode ist im Skript vordefiniert.

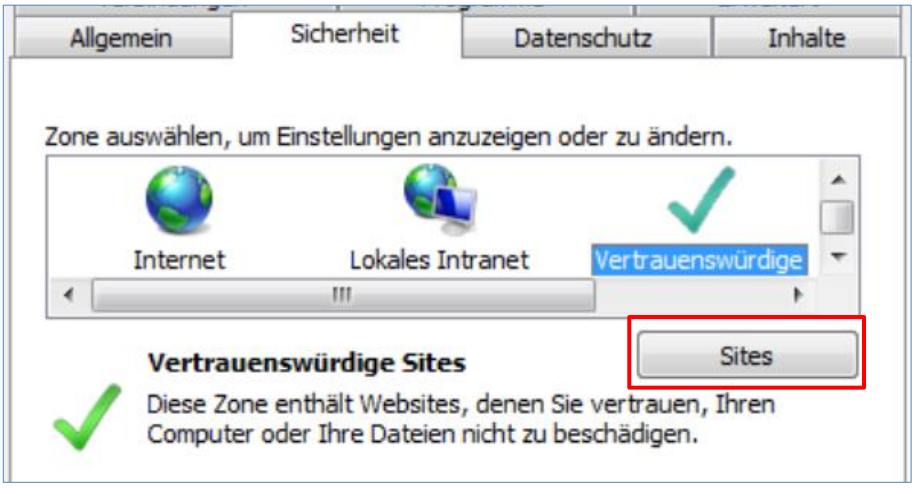
6.6. Mail wird bei „händischer“ Ausführung versandt, nicht aber mittels Taskplaner

Das Blockieren des Ausführens von Dateien aufheben
Werden nach dem Download einer ZIP-Datei die Eigenschaften dieser angezeigt, so erschient möglicherweise ein Block „Sicherheit“ mit folgendem Hinweis: „Die Datei stammt von einem anderen Computer. Der Zugriff wurde aus Sicherheitsgründen eventuell blockiert.“



Ein Klick auf „Zulassen“ lässt diese Sicherheitswarnung verschwinden. Macht man das nicht, wirkt es sich auf alle im ZIP-File enthaltenen ausführbaren Dateien aus.
Hier würde das bedeuten, dass die PSMail-PowerShell-Skripts nicht ausgeführt werden können, wenn sie im Taskplaner eingebaut sind. Und genau das ist bei den meisten ja vorgesehen.

Eine andere Möglichkeit, das Setzen dieses Sicherheits-Attributs vorab zu unterbinden ist, die Downloadseite https://www.vobs.at den Vertrauenswürdigen Sites in den Internetoptionen hinzuzufügen: Über Systemsteuerung – Internetoptionen wählen wir den Reiter Sicherheit.



Über Sites können URLs hinzugefügt werden, bei denen am Ende eines Downloads keine Sicherheitsüberprüfung ausgeführt wird. Damit entfällt für alle vom VOBS heruntergeladenen Dateien der oben beschriebene Klick auf „Zulassen“.
Dies ist unabhängig davon, ob Dateien über Internetexplorer oder Chrome heruntergeladen werden.