Example Powershell Script for GPO Deployment of BCS Suite for Windows

https://campus.barracuda.com/doc/89096919/

This sample Powershell script is ONLY an example, which you can follow as a template for your GPO deployment of the Barracuda Content Shield Suite for Windows. This script is not guaranteed by Barracuda, and you must replace the parameter values with your own.

Note that you can change the filenames of the powershell script (ExampleConfigBCS.ps1) and/or batch file (ConfigureBCSPlus.bat), but make sure to update the batch file with the revised powershell script name, and that the GPO procedure references the new file names.

```
param(
    [string]$action='',
    [string]$workDir='',
    [string]$setupName='',
    [string]$keyName='',
    [string]$version='',
    [string]$feature='',
    [string]$userPass=''
) #Must be the first statement in the script
#### CUSTOM SETUP
#####################################################################
<#
=NOTES
================================================================================

 -- TESTED WITH WIN10. Needs adjustments for running on WIN7 (Powershell <
3.0), i.e. "Tee-Object" --
 Make sure to adjust the parameter contents for install/upgrade or uninstall
to your setup.

 Deploy via GPO:
  Create a per-machine GPO that executes a Startup Script (make sure it runs
with elevated permissions) that invokes the powershell script - or any other
script that you created to install / remove the agent suite installer
(executable).
 Return codes:
  3010 - Uninstall successful. Reboot pending: On uninstall, a reboot is
required before reinstalling the agent.
  0    - Installation successful. (Also on a "remove" action, if the agent is
not installed).
  1601 - Install aborted. SETUP.EXE not found.
  1602 - Install aborted. Check KEYPATH value.
```

```
  1603 - Uninstall canceled. Most likely because either the Tamper Proof
feature is disabled, or the wrong password was provided.
  For any of the 16xx return codes, please check the component MSI logs which
can be found in the %temp% folder of the process owner.
================================================================================
=======
#>
# ACTION: Set the action for this script:( install | remove )
$ACTION=$action
# BASE_DIR: Set your network share folder as base dir and some other values.
$BASE_DIR       =$workDir
# SETUP_FILENAME: By default named BarracudaContentShieldSetup-[VERSION].exe.
If you rename this field, make sure to also fill in the VERSION field below.
$SETUP_FILENAME =$setupName
# KEY_FILENAME: By default named bcs.key. Change this property if you plan to
rename this key (but make sure to keep ".key" as the extension).
$KEY_FILENAME   =$keyName
#VERSION: There is no need to set this if you plan to keep the filename
original (which contains the version). In that case the following lines
extract the version from the file name.
$VERSION        =$version
#FEATURE: Only set this to override the default complete install - to install
ONLY EITHER WebFiltering OR Malware Protection - possible values:
"WebFiltering" | "MalwarePrevention"
$FEATURE        =$feature
#### SETUP DONE
#########################################################################
if($VERSION -eq ""){
    #Try to get it from the setup name. This can only work if the name is not
modified
    $VERSION        =($SETUP_FILENAME).split("-""")[1]
    $VERSION = $VERSION.Substring(0,$VERSION.Length-4)
    # TODO: Some regex to validate the result is, in fact, a version number.
}
$ErrorActionPreference="Stop"
$level=@{INFO="[INFO]";WARNING="[WARN]";ERROR="[ERROR]"}
$date=(Get-Date).ToString('MMddyy_HHmmss')
$MMddyyyy=(Get-Date).ToString('MM-dd-yyyy');
# Creates an operational log file in that same directory.
$LOG_FILE=$BASE_DIR + "\install.log"
# Starting script execution
Write-Output "*** START *** $(Get-Date)
**********************************************" | Tee-Object -FilePath
$LOG_FILE -Append #| Out-File $LOG_FILE -Append -Force;
Write-Output "$(Get-Date) : $($level["INFO"]) Starting the script " | Tee-
Object -FilePath $LOG_FILE -Append
```

```
Write-Output "$(Get-Date) : $($level["INFO"]) Params: ACTION=$action,
WORKDIR=$workDir, SETUP_NAME=$setupName, KEY_NAME=$keyName, VERSION=$version,
FEATURE=$feature" | Tee-Object -FilePath $LOG_FILE -Append
$bcs_suite_installed=''
$bcs_cpa_installed=''
$bcs_wca_installed=''
function ExitWithCode
{
    param
    (
        [Int32]$exitcode
    )
    if($exitcode-eq 3010){
        # Expect exitcode 3010, as this just states pending reboot.
        # You can check the component logs in %temp% folder (MSI*.log) for
further details about the installation of each component.
        Write-Host "Requested action '$ACTION' finished with return code:
$exitcode. The actions was successful. A Reboot is pending."
        exit 0   #handle as success
    }
    else{
        Write-Host "Action '$ACTION' finished with return code: $exitcode"
    }
    $host.SetShouldExit($exitcode)
    exit $exitcode
}
function InstallOrUpgradeBCSPlus
{
    $bcsExePath = $(Join-Path $BASE_DIR $SETUP_FILENAME)
    $bcsKeyPath = $(Join-Path $BASE_DIR $KEY_FILENAME)
    if(-not (Test-Path $bcsExePath)) {
        Write-Output "$(Get-Date) : $($level["ERROR"]) $($bcsExePath) does
not exist" | Tee-Object -FilePath $LOG_FILE -Append
        ExitWithCode -exitcode 1601
    }
    if(-not (Test-Path $bcsKeyPath)) {
        Write-Output "$(Get-Date) : $($level["ERROR"]) $($bcsKeyPath) does
not exist" | Tee-Object -FilePath $LOG_FILE -Append
        ExitWithCode -exitcode 1602
    }
    [string[]]$arguments = @()
    If ($KEY_FILENAME -ne ""){
        $arguments += "KEYPATH=`"$($bcsKeyPath)`""
    }
    If ($feature -ne ""){
        $arguments += "ISFeatureInstall=$($($feature -split "," |
```

```
foreach({"$($_.Trim())"})) -join ','")"
    }
    $arguments += "/silent"
    write-output $arguments
    $process = (Start-Process -FilePath "$($bcsExePath)" -ArgumentList
$arguments -PassThru -Wait -NoNewWindow)
    ExitWithCode -exitcode $process.ExitCode
}
function UninstallBCSPlus {
    # Uninstall Barracuda Content Shield Suite from system
    $process = (Start-Process -FilePath "$BASE_DIR\$SETUP_FILENAME" -
ArgumentList "USER_PASS=$userPass","/silent","/remove" -PassThru -Wait -
NoNewWindow)
    ExitWithCode -exitcode $process.ExitCode
}
function CheckInstallationBCSPlus
{
    # ---- Check registry to see if BCS is installed ----
    $UninstallKeys =
"HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall",
"HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall"
    $guid = @{ "CPA" = "`{0F536D44-C0A5-4DCC-BAC9-5724FFB91D64`}"; "WCA" =
"`{CEB35DBB-E633-452D-A876-0CA0DC9D4DC0`}"; "Suite" = "`{3877F3A3-358E-49C2-
BFA7-9A63994D2FCA`}"}
    $suite_ver=''
    $cpa_ver=''
    $wca_ver=''
    foreach($key in $UninstallKeys)
    {
        if(Test-Path "$key\$($guid.Suite)" -ErrorAction SilentlyContinue){
            #Find out the version installed
            $display_version = Get-ItemProperty -Path "$key\$($guid.Suite)" -
Name DisplayVersion
            $suite_ver="$($display_version.DisplayVersion)"
        }
        if(Test-Path "$key\$($guid.CPA)" -ErrorAction SilentlyContinue){
            #Find out the version installed
            $display_version = Get-ItemProperty -Path "$key\$($guid.CPA)" -
Name DisplayVersion
            $cpa_ver="$($display_version.DisplayVersion)"
        }
        if(Test-Path "$key\$($guid.WCA)" -ErrorAction SilentlyContinue){
            $display_version = Get-ItemProperty -Path "$key\$($guid.WCA)" -
Name DisplayVersion
            $wca_ver="$($display_version.DisplayVersion)"
        }
```

```
    }
    Return $suite_ver, $cpa_ver, $wca_ver
}
# Get versions installed
$bcs_suite_installed ,$bcs_cpa_installed, $bcs_wca_installed =
CheckInstallationBCSPlus
Write-Output "$(get-date) : $($level["INFO"]) SUITE: $bcs_suite_installed,
CPA: $bcs_cpa_installed, WCA: $bcs_wca_installed" | Tee-Object -FilePath
$LOG_FILE -Append
if ($ACTION -eq "install")
{
    if($bcs_suite_installed -eq '')
    {
        Write-Output "$(get-date) : $($level["INFO"])  INSTALL BCS v$VERSION"
| Tee-Object -FilePath $LOG_FILE -Append
        InstallOrUpgradeBCSPlus
    }
    else
    {
        if($VERSION -gt $bcs_suite_installed)
        {
            Write-Output "$(get-date) : $($level["INFO"])  UPGRADE ($VERSION
> $bcs_suite_installed)" | Tee-Object -FilePath $LOG_FILE -Append
            InstallOrUpgradeBCSPlus
        }
        elseif($VERSION -lt $bcs_suite_installed)
        {
            Write-Output "$(get-date) : $($level["INFO"])  NO ACTION (up to
date: $VERSION < $bcs_suite_installed)" | Tee-Object -FilePath $LOG_FILE -
Append
        }
        elseif($VERSION -eq $bcs_suite_installed)
        {
            # NOTE:
            # If you run 'install' on the same version installed, it will try
to uninstall the installed version.
            # If you intend to uninstall, you will need to enter the password
parameter.
            Write-Output "$(get-date) : $($level["INFO"])  NO ACTION (same:
$VERSION == $bcs_suite_installed)" | Tee-Object -FilePath $LOG_FILE -Append
        }
    }
}
elseif ($ACTION -eq "remove")
{
    if ($bcs_suite_installed -ne '')
```

```
    {
        Write-Output "$(get-date) : $($level["INFO"])  UNINSTALL BCS..." |
Tee-Object -FilePath $LOG_FILE -Append
        UninstallBCSPlus
    }
    else{
        Write-Output "$(get-date) : $($level["INFO"])  NO ACTION (No BCS
installed. No action necessary.)" | Tee-Object -FilePath $LOG_FILE -Append
    }
}
else
{
    Write-Output "$(get-date) : $($level["ERROR"])  UNKNOWN ACTION: $ACTION
(Pick install | remove)" | Tee-Object -FilePath $LOG_FILE -Append
}
Write-Output "*** DONE *** $(Get-Date)
**********************************************" | Tee-Object -FilePath
$LOG_FILE -Append
ExitWithCode -exitcode 0
```