How to Manage Deployment of the Barracuda Content Shield Agent for Windows

https://campus.barracuda.com/doc/79463711/

The Barracuda Content Shield agent for Windows supports Windows Server 2012 R2, 2016 and 2019. For endpoint computers using Firefox, Barracuda Content Shield agent requires Firefox 68.0 or later.

The agent is installed on endpoint machines with one of the following methods:

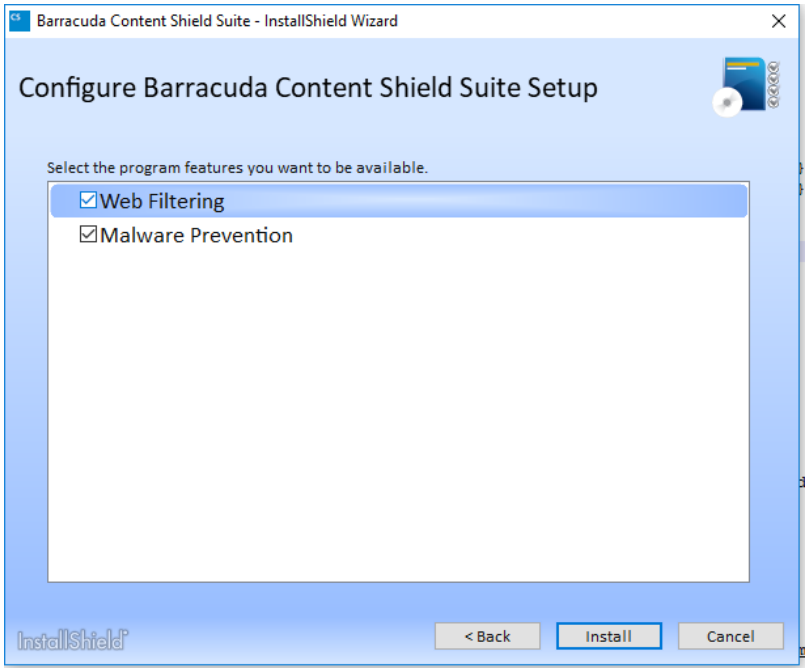
- Manually, using the command line
- RMM (see How to Deploy the Barracuda Content Shield Agent Using RMM Solutions)
- Microsoft Intune (an MDM)
- GPO (see How to Deploy the Barracuda Content Shield Agent via GPO)

The agent includes a web filtering component (WFC) for content protection and control of web traffic. After installing the agent, see How to Use the Barracuda Content Shield Agent for Windows for details about the user interface on the endpoint.

Windows Installation

For manual installation on one of the endpoint machines:

1. Log into your Barracuda Content Shield account.
2. Go to the **Downloads** page.
3. Click **Download** to download the Windows Agent installer. The installer file is named `BarracudaContentShieldSetup-[version].exe`.
4. Click **Account Configuration** File to download the configuration file. This file is named `bcs.key`. The configuration file is picked up by the installer when you run it. Put this file and the installer file in the same folder on the endpoint computer. Note that network drives may present permissions issues.
5. Run the `BarracudaContentShieldSetup-[version].exe` executable on the endpoint, accept the license agreement, and follow the prompts in the wizard to install.
6. After browsing for the `bcs.key` file on the **Locate Account Configuration** screen, the wizard presents a **Setup Type** dialog with options for **Complete** or **Custom** installations:
   - Selecting **Complete** installs only the Web Filtering component (WFC) – OR –
   - *Selecting **Custom** installs only the Web Filtering component by default,* but you can check the **Malware Prevention** box if you want to install the MPC as well.



When you uninstall the agent from the endpoint machine, you will be prompted to either reboot the machine or choose to reboot later. Barracuda Networks recommends rebooting if you want to reinstall the agent at a later time.

Managed Deployment Using RMM or GPO

You can either use an RMM or GPO to install and distribute to endpoint machines if you don't want to do manual installations. If you use an RMM or GPO, make sure to indicate that the installer (`BarracudaContentShieldSetup-[version].exe`) and the configuration file (`bcs.key`) must be in the same folder. See How to Deploy the Barracuda Content Shield Agent Using RMM Solutions and How to Deploy the Barracuda Content Shield Agent via GPO for details.

Upgrade Using Windows

When you run the Windows installer, if there is a newer version of the Barracuda Content Shield Agent, you will see the **Upgrade** button in the installer Welcome screen. Click **Upgrade** and follow the wizard prompts.

Command Line Installation / Upgrade / Uninstallation (silent)

See command line syntax and examples of each process in the table below.

Important note on uninstalling: If *uninstalling* using the Windows installer, you will be prompted for a password to uninstall (also with command line). **If you don't have the password you cannot uninstall the agent.** See Agent Settings for details on the Tamper Proof feature (**Allow Agent Removal**) and setting this password.

To install from the command line on one of the endpoint machines:

1. Follow steps 1-4 above.
2. From the command line, run the appropriate command from the examples in the table below. Note: For KEYPATH = "*path/to*/bcs.key" , replace the example in italics with the actual path to the `bcs.key` file. For USER_PASS, enter the password you created on the **AGENT SETTINGS** page for **Allow Agent Removal.**

Install / uninstall complete suite	**Syntax**
Install	`BarracudaContentShieldSetup-[version].exe KEYPATH="path/to/bcs.key" /silent`
Uninstall	`BarracudaContentShieldSetup-[version].exe USER_PASS="my_password" /remove /silent`
Uninstall, and then reboot the endpoint machine	`BarracudaContentShieldSetup-[version].exe USER_PASS="my_password" /remove /silent & shutdown /r /t 0`
Upgrade the suite	
Upgrade the suite to the latest version	`BarracudaContentShieldSetup-[version].exe /silent`

Sample Script For Use With RMM, GPO, Intune and Other Deployment Solutions

Use an elevated command prompt (run cmd.exe as administrator) for deployment. Apply the following parameter values to suit your target environment.

Parameters

Parameter	Type	Description
action	string	**options: [install | remove]** Use 'install' for installation or upgrading your installation to the next version. Use 'remove' to uninstall the agent and clean up the machine.
workDir	string	If the workDir does not yet exist, it will be created as specified. In that case the executable as mentioned in 'setupName' parameter as well as the .key as mentioned in 'keyName' parameter are assumed in same dir as the ConfigBCSPlus.ps1 script. This is useful, if you want to use this script for deployment via Intune or another tool where the contents are zipped up and saved to an undisclosed folder. This way you have more control over where the files will end up - as well as the log file to check in on the result.
setupName	string	**BarracudaContentShieldSetup-[VERSION].exe** - make sure the version is part of the string - or if renaming this exe make sure the -version parameter is passed with the version to install. Only needed for action 'install'.
keyName	string	**bcs.key** - the name of the account configuration file. Only needed for action 'install'.
feature	string	**options: [WebFiltering | MalwarePrevention]**. **OPTIONAL**. Only needed if installation is customized to install also / only MalwarePrevention. (Default: install only WebFiltering), example for override: *-feature "WebFiltering,MalwarePrevention"* (use this if you want to install MalwarePrevention on top of default WebFiltering component)
version	string	1.2.3.4 - the version of the new installer. **OPTIONAL.** Only needed if original installer executable is renamed.
logEventLog	string	*-logEventLog '1'* . **OPTIONAL**. Only needed if you want to log to the Application Event log which is part of the System Event logs. All tool output is redirected to this event log.

Deployment script source code

This script is provided as an example only.

```
param(
   [Parameter(Mandatory=$true,HelpMessage='Specify "install" or "remove"')]
   [string]$action='',
   [Parameter(Mandatory=$true,HelpMessage='Specify working directory. Gets
```

```
created if does not exist and setup files are copied here.')]
    [string]$workDir='',
    [Parameter(HelpMessage='Required for "install" action only. BCS setup
executable file name')]
    [string]$setupName='',
    [Parameter(HelpMessage='Required for "install" action only. BCS account
configuration (*.key) file name')]
    [string]$keyName='',
    [Parameter(HelpMessage='Required for customized "install" only. Set to
"WebFiltering" or "MalwarePrevention". Not setting this defaults to install
both.')]
    [string]$feature='',
    [Parameter(HelpMessage='Optional. Pass version, i.e. "2.0.0.0", if
original installer filename is renamed.')]
    [string]$version='',
    [Parameter(HelpMessage='Optional. Pass this flag and set to "1" if
logging to Event log Application.evtx')]
    [string]$logEventLog='',
    [Parameter(HelpMessage='User password for uninstall (obsolete, since
tamperproof gets disabled in this script)')]
    [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
elevated) that invokes the powershell script - or any other script that you
created to install / remove the agent suite installer (executable).
 Return codes:
  # exitcode 3010 currently returned as 0. Remove condition from ExitWithCode
function in order to receive this return code.
  # 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.
  1601 - Install aborted. SETUP.EXE not found.
  1602 - Install aborted. Check KEYPATH value.
  1603 - Uninstall canceled. Most likely because either Tamper-proof is
```

```
disabled or the wrong password was provided.
  For any of the 16xx return codes, please check the component MSI logs that
can be found in the %temp% folder of the process owner.
================================================================================
=======
#>
# ACTION: set the action to do in 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: 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 default complete install - to install
ONLY EITHER WebFiltering OR Malware Protection - possible values:
"WebFiltering" | "MalwarePrevention"
$FEATURE        =$feature
# creates an operational log file in workDir if logEventLog not set.
$LOG_FILE       =$(Join-Path $workDir 'install.log')
#### SETUP DONE
################################################################################
if($VERSION -eq "" -and $action -ne "remove"){
    #try to get it from our setup name. Of course, this can only work if the
name has not been 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.
}
$actions = @("install","remove")
$ErrorActionPreference="Stop"
$level=@{INFO="[INFO]";WARNING="[WARN]";ERROR="[ERROR]"}
$date=(Get-Date).ToString('MMddyy_HHmmss')
$MMddyyyy=(Get-Date).ToString('MM-dd-yyyy');
# creates an operational event source for Application.evtx
If($logEventLog) {
    $eventSource = "$($MyInvocation.MyCommand.Name)"
    if($eventSource -eq "") { $eventSource = "ConfigureBCS.ps1"}
    if(-not([System.Diagnostics.EventLog]::SourceExists("$eventSource"))){
        New-EventLog -LogName Application -Source "$eventSource"
```

```
    }
}
function Bcs-Log
{
    Param(
        [Parameter(Mandatory=$true)]
        [string]$LogMessage,
        [Parameter(Mandatory=$true)]
        [string]$eventType
    )
    If($logEventLog) {
        Try{
            Write-EventLog -LogName Application -Source "$eventSource" -
EntryType $eventType -EventId 1 -Message $LogMessage
        }
        Catch{
            $ErrorMessage = $_.Exception.Message
            Write-Host "$LogMessage, Error: $ErrorMessage"
        }
    } Else {
        Write-Output "$(Get-Date) : [$($eventType)] $($LogMessage)" | Tee-
Object -FilePath $LOG_FILE -Append
    }
}
# make sure workDir exists
mkdir $BASE_DIR -ErrorAction SilentlyContinue
# Starting script execution
Bcs-Log -LogMessage "*** START $action
**********************************************`nParams: ACTION=$action,
WORKDIR=$workDir, SETUP_NAME=$setupName, KEY_NAME=$keyName, VERSION=$version,
FEATURE=$feature" -eventType Information
$bcs_suite_installed=''
$bcs_cpa_installed=''
$bcs_wca_installed=''
function ExitWithCode
{
    param
    (
        [Int32]$exitcode
    )
    if($exitcode-eq 3010){
        # we expect exitcode 3010, as this just states pending reboot.
        # you can check the component logs in %temp% folder (MSI*.log) for
further details on the installation of each component.
        Bcs-Log -LogMessage "Requested action '$ACTION' finished with return
code: $exitcode. The action was successful. A Reboot is pending." -eventType
```

```
Information
        exit 0   #handle as success
    }
    else{
        Bcs-Log -LogMessage "Action '$ACTION' finished with return code:
$exitcode" -eventType Information
    }
    $host.SetShouldExit($exitcode)
    exit $exitcode
}
function InstallOrUpgradeBCSPlus
{
    <#
        .SYNOPSIS
        Installs or upgrades BCS Suite using given parameters.
        .DESCRIPTION
        Depending on state this will either install or upgrade BCS Suite if
installer detects
        a lower version of the BCS Suite is already installed. The command is
the same.
        Once installer finishes this script exits.
    #>
    $bcsExePath = $(Join-Path $BASE_DIR $SETUP_FILENAME)
    $bcsKeyPath = $(Join-Path $BASE_DIR $KEY_FILENAME)
    $installShieldLogPath = $(Join-Path $BASE_DIR InstallShield.log)
    # copy setup files to workDir
    if($PSScriptRoot -ne $BASE_DIR){
        $msg = "$PSScriptRoot != $BASE_DIR"
        # copy over files from script dir. If none found here, assume setup
files are already in specified workDir
        if(Test-Path "$(Join-Path $PSScriptRoot $SETUP_FILENAME)"){
            $msg += "`nCopy Setup Files to $($BASE_DIR): $SETUP_FILENAME and
$KEY_FILENAME"
            Copy-Item -Path "$(Join-Path $PSScriptRoot $SETUP_FILENAME)" -
Destination $bcsExePath -Recurse -Force
            Copy-Item -Path "$(Join-Path $PSScriptRoot $KEY_FILENAME)" -
Destination $bcsKeyPath -Recurse -Force
        }
        Bcs-Log -LogMessage $msg -eventType Information
    }
        if(-not (Test-Path $bcsExePath)) {
                Bcs-Log -LogMessage "$($bcsExePath) does not exist" -
eventType Error
                ExitWithCode -exitcode 1601
        }
        if(-not (Test-Path $bcsKeyPath)) {
```

```
                Bcs-Log -LogMessage "$($bcsKeyPath) does not exist" -
eventType Error
                ExitWithCode -exitcode 1602
        }
    [string[]]$arguments = @()
    If ($KEY_FILENAME -ne ""){
        $arguments += "KEYPATH=`"$($bcsKeyPath)`""
    }
    If ($feature -ne ""){
        $arguments += "ISFeatureInstall=$($($feature -split "," |
foreach({"$($_.Trim())"})) -join ',')"
    }
    Bcs-Log -LogMessage "Install: $bcsExePath $arguments
/debuglog`"$($installShieldLogPath)`" /silent" -eventType Information
    $process = (Start-Process -FilePath "$($bcsExePath)" -ArgumentList
"$arguments /debuglog`"$($installShieldLogPath)`" /silent" -PassThru -Wait -
NoNewWindow)
    ExitWithCode -exitcode $process.ExitCode
}
function Remove-Pass {
     <#
        .SYNOPSIS
        Remove tamper proof protection from installation.
        .DESCRIPTION
        In order to be able to uninstall the tamper proof protection needs to
be disabled.
        This function removes the protection so that uninstall command does
not require the password to be checked against.
    #>
    Bcs-Log -LogMessage "Try stopping BCS Services if installed.." -eventType
Information
    Stop-Service -Name "bcs_cpa" -ErrorAction SilentlyContinue
    Stop-Service -Name "bcs_wca" -ErrorAction SilentlyContinue
        Bcs-Log -LogMessage "Remove pass" -eventType Information
        $registryPath = "HKLM:\SOFTWARE\Barracuda\Content Shield"
    If(Test-Path $registryPath)
    {
                $value1 = (Get-ItemProperty $registryPath).BCS_Admin -eq
$null
                If ($value1 -eq $False) {Remove-ItemProperty -Path
"$registryPath" -name BCS_Admin -Force | Out-Null}
        $value1 = (Get-ItemProperty $registryPath).BCS_InstallingSuite -eq
$null
                If ($value1 -eq $False) {Remove-ItemProperty -Path
"$registryPath" -name BCS_InstallingSuite -Force | Out-Null}
                $value1 = (Get-ItemProperty $registryPath).BCS_UpgradingSuite
```

```
-eq $null
                If ($value1 -eq $False) {Remove-ItemProperty -Path
"$registryPath" -name BCS_UpgradingSuite -Force | Out-Null}
        }
}
function Override-Pass {
        <#
        .SYNOPSIS
        Override tamper proof password with a preset password in order to be
able to uninstall suite versions  < 2.0.
        .DESCRIPTION
        In order to be able to uninstall the tamper proof protection needs to
be disabled.
        This function disables the protection so that uninstall command can
use the standard USER_PASS parameter 'password'.
    #>
        Bcs-Log -LogMessage "Try stopping BCS Services if installed.." -
eventType Information
    Stop-Service -Name "bcs_cpa" -ErrorAction SilentlyContinue
    Stop-Service -Name "bcs_wca" -ErrorAction SilentlyContinue
        Bcs-Log -LogMessage "Override pass" -eventType Information
        $registryPath = "HKLM:\SOFTWARE\Barracuda\Content Shield"
        $name = "BCS_Admin"
        $value =
"`$2a`$12`$CGmJgrD1dY3ZiUroiCLvDeozBM0eRYDpkL03d3q10kreJwndFEM1a"
    If(!(Test-Path $registryPath))
    {
                # create new
                New-Item -Path $registryPath -Force | Out-Null
                New-ItemProperty -Path $registryPath -Name $name -Value
$value -PropertyType String -Force | Out-Null
        }
        Else{ # update
                New-ItemProperty -Path $registryPath -Name $name -Value
$value -PropertyType String -Force | Out-Null
                $value1 = (Get-ItemProperty
$registryPath).BCS_InstallingSuite -eq $null
                If ($value1 -eq $False) {Remove-ItemProperty -Path
"$registryPath" -name BCS_InstallingSuite -Force | Out-Null}
                $value1 = (Get-ItemProperty $registryPath).BCS_UpgradingSuite
-eq $null
                If ($value1 -eq $False) {Remove-ItemProperty -Path
"$registryPath" -name BCS_UpgradingSuite -Force | Out-Null}
        }
}
function Reset-Acls {
```

```
    # Reset folder  permissions on locked down support folder
    $resetAcl = "icacls `"$($env:PROGRAMDATA)\Barracuda\Content Shield`"
/RESET /T /Q"
    $process = (Start-Process 'cmd' -ArgumentList "/c","$($resetAcl)" -
PassThru -Wait -NoNewWindow)
    Bcs-Log -LogMessage "Reset ACLs finished. Result: $($process.ExitCode)" -
eventType Information
}
function Stop-BCSProcesses {
     Param(
        [Parameter(Mandatory=$true)]
        [array]$processes
     )
    <#
        .SYNOPSIS
        Stop all processes in given list.
        .DESCRIPTION
        Terminates given list of processes.
        .PARAMETER processes
        Specifies the list of processes to terminate.
    #>
    $msg = 'Stop BCS processes if running'
    ForEach($pName in $processes){
        $proc = Get-Process $pName -ErrorAction SilentlyContinue
        if ($proc) {
            Try{
                # try gracefully first
                $proc.CloseMainWindow()
                # kill after five seconds
                Sleep 5
                if (!$proc.HasExited) {
                    $proc | Stop-Process -Force
                }
                Sleep 2
                if (!$proc.HasExited) {
                    taskkill /IM "$($pName).exe" /F
                }
            } Catch {
                $ErrorMessage = $_.Exception.Message
                $msg += "`nFailed to stop $pName - $ErrorMessage"
            }
        }
        else {
            $msg += "`nFailed to stop $pName. Not running."
        }
    }
```

```
    Bcs-Log -LogMessage $msg -eventType Information
}
function UninstallBCSPlus {
    <#
        .SYNOPSIS
        Uninstall Suite incl. clean up of left over components and files.
        .DESCRIPTION
        Remove and clean up BCS Suite installation. The machine should be
left in a clean state after this. Make sure to reboot before re-installing
the BCS Suite.
    #>
    #reset any acls
    Reset-Acls
    #remove tamperproof protection
        Remove-Pass
    $resultCode = 0
    # Let's make sure we get the latest info about what is installed
    $installation_info = Get-BCSInstallation-Info
    $suite = $($installation_info['Suite'])
        Bcs-Log -LogMessage "Found $($suite.DisplayName) -
$($suite.DisplayVersion)." -eventType Information
        If ($suite.UninstallString) {
        $installShieldLogPath = $(Join-Path $BASE_DIR InstallShield.log)
                $uninst = "$($suite.UninstallString) -silent -
debuglog`"$($installShieldLogPath)`""
        Bcs-Log -LogMessage " Uninstall: `"$($uninst)`"" -eventType
Information
        $process = (Start-Process 'cmd' -ArgumentList "/c", "`"$($uninst)`""
-PassThru -Wait -NoNewWindow)
        $resultCode = $process.ExitCode
        Bcs-Log -LogMessage "Uninstall finished. Result: $($resultCode)" -
eventType Information
        }
    # Check second time if we need a cleanup (retry)
    $installation_info = Get-BCSInstallation-Info
    $suite = $($installation_info['Suite'])
    If ($suite.UninstallString) {
        # For some reason the first uninstall attempt of the Suite did not
succeed, so we try here again
                $uninst = "$($suite.UninstallString) -silent"
        $process = (Start-Process 'cmd' -ArgumentList "/c", $uninst -PassThru
-Wait -NoNewWindow)
        $resultCode = $process.ExitCode
        Bcs-Log -LogMessage "2nd Try: Uninstall finished. Result:
$($resultCode)" -eventType Information
        }
```

```
    # The following measures are brute force and only serve to clean up
anything that a broken installatation cannot clean up.
################################################################################
##############################################
    # REMARK: If this cleanup does not help to successfully reinstall (reboot
before reinstall and also run tool at least twice before giving up),
    #         it is worth checking for the Microsoft clean-up tool for broken
Windows Installers.
    #         The registry management within Windows installers is
complicated and should not be lightheartedly muddled with.
################################################################################
##############################################
    # Search for all components that are managed by Suite installer and
remove them using their uninstall string
    ForEach($leftover in $installation_info.Keys){
        $component = $($installation_info[$leftover])
        Bcs-Log -LogMessage "Found $($component.DisplayName) -
$($component.DisplayVersion)." -eventType Information
            If ($component.UninstallString) {
            # remove the single suite component leftovers
                    $uninst = "$($component.UninstallString) /quiet
/norestart"
            $process = (Start-Process 'cmd' -ArgumentList "/c","$uninst" -
PassThru -Wait -NoNewWindow)
            Bcs-Log -LogMessage "Uninstall: $($uninst) finished with
$($process.ExitCode)" -eventType Information
            }
    }
    # If the installation got really broken, make sure we check if there are
any left over services. Remove these as well.
    $bcsServices = @("bcs_iot", "bcs_cpa", "bcs_wca", "bcs_fb")
    ForEach( $bcsServiceName in $bcsServices){
        Try{
            $service = Get-Service -Name $bcsServiceName -ErrorAction
SilentlyContinue
            if ($service.Length -gt 0) {
               sc.exe delete "$bcsServiceName"
            }
        } Catch {
            $ErrorMessage = $_.Exception.Message
            Bcs-Log -LogMessage "Failed to delete $bcsServiceName -
$ErrorMessage" -eventType Information
        }
    }
    # Only after services are removed from service controller, we can now
force stop them without fearing they may be restarted by SVC
```

```
    # All processes in below list are force stopped, should they still be
running.
    $bcsProcesses = @( "avupdate", "bcs-wca", "bcs-cpa", "filebeat",
"bcs_iot", "BCSGui")
    Stop-BCSProcesses -processes $bcsProcesses
    # Finally, also check for any left over folders.
    $bcsFolders =  @("$env:Programfiles\Barracuda\Content Shield",
"$env:Programdata\Barracuda\Content Shield")
    ForEach( $folder in $bcsFolders){
        If((Test-Path $folder)){
            Try{
                Remove-Item -Recurse -Force $folder
            } Catch {
                $ErrorMessage = $_.Exception.Message
                Bcs-Log -LogMessage "Failed to remove $folder -
$ErrorMessage" -eventType Information
            }
        }
    }
    ExitWithCode -exitcode $resultCode
}
function Get-BCSInstallation-Info {
    <#
        .SYNOPSIS
        Get all installation state of BCS Suite components.
        .DESCRIPTION
        Returns the installation state of all BCS Suite components by
iterating the registry.
    #>
    $info = @{}
    $bcsRegistryInfo = @{
        "BCS Web Content Agent" = "WebFiltering"
        "BCS Content Protection Agent" = "MalwarePrevention"
        "BCS Content Shield GUI" = "Gui"
        "BCS Content Shield Support Tool" = "Support"
        "BCSFilebeat" = "Filebeat"
        "bcs_iot" = "iot"
        "Barracuda Content Shield Suite" = "Suite"
    }
    $suiteVer = Get-ChildItem -Path
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,
HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall  |
        Get-ItemProperty |
            Where-Object {($_.DisplayName -in $bcsRegistryInfo.Keys)} |
                Select-Object -Property DisplayName, DisplayVersion,
UninstallString
```

```
    ForEach ($ver in $suiteVer) {
                $info["$($bcsRegistryInfo["$($ver.DisplayName)"])"] = $($ver)
            }
    $compVer = Get-ChildItem -Path
HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,
HKLM:\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall  |
        Get-ItemProperty |
            Where-Object {($_.DisplayName -in $bcsRegistryInfo.Keys)} |
                Select-Object -Property DisplayName, DisplayVersion,
UninstallString
    ForEach ($ver in $compVer) {
            $info["$($bcsRegistryInfo["$($ver.DisplayName)"])"] = $($ver)
        }
    return $info
}
# get versions installed
$installation_info = Get-BCSInstallation-Info
$bcs_suite_installed=$($installation_info['Suite']).DisplayVersion
if($installation_info.Count -gt 0){
    Bcs-Log -LogMessage "Found Content Shield Suite
Components:`n$(($installation_info.Keys | foreach { "$_
v$($($installation_info[$_]).DisplayVersion)" }) -join "`n")" -eventType
Information
}
else {
    Bcs-Log -LogMessage "Found Content Shield Suite Components:`n -- 0
components installed. This machine is clean. --" -eventType Information
}
if ($ACTION -eq "install")
{
    if($installation_info.Count -eq 0)
    {
        Bcs-Log -LogMessage "INSTALL BCS v$VERSION" -eventType Information
        InstallOrUpgradeBCSPlus
    }
    elseif($installation_info['Suite'] -eq ''){
        Bcs-Log -LogMessage "Machine needs cleanup, Suite not installed but
other Content Shield components are. Going to run cleanup now." -eventType
Error
        $ACTION="remove"
        $bcs_suite_installed=''
    }
    else
    {
        if([version]$VERSION -gt [version]$bcs_suite_installed)
        {
```

```
            Bcs-Log -LogMessage "UPGRADE ($VERSION > $bcs_suite_installed)"-
eventType Information
            InstallOrUpgradeBCSPlus
        }
        elseif([version]$VERSION -lt [version]$bcs_suite_installed)
        {
            Bcs-Log -LogMessage "NO ACTION (up to date: $VERSION <
$bcs_suite_installed)" -eventType Information
        }
        elseif([version]$VERSION -eq [version]$bcs_suite_installed)
        {
            # NOTE:
            # If you run 'install' on the same version installed, it will try
to uninstall the installed version.
            # If that is your intention, for that you need to pass the
password parameter.
            Bcs-Log -LogMessage "NO ACTION (same: $VERSION ==
$bcs_suite_installed)" -eventType Information
        }
    }
}
if ($ACTION -eq "remove")
{
    if ($installation_info.Count -gt 0)
    {
        Bcs-Log -LogMessage "UNINSTALL BCS..." -eventType Information
        UninstallBCSPlus
    }
    else{
        Bcs-Log -LogMessage "NO ACTION (No BCS installed. No action
necessary.)" -eventType Information
    }
}
if ($ACTION -inotin $actions)
{
    Bcs-Log -LogMessage "UNKNOWN ACTION: $ACTION (Pick install | remove)" -
eventType Error
}
ExitWithCode -exitcode 0
```

Figures

1. CustomSetupSelectComponent.PNG

© Barracuda Networks Inc., 2024 The information contained within this document is confidential and proprietary to Barracuda Networks Inc. No portion of this document may be copied, distributed, publicized or used for other than internal documentary purposes without the written consent of an official representative of Barracuda Networks Inc. All specifications are subject to change without notice. Barracuda Networks Inc. assumes no responsibility for any inaccuracies in this document. Barracuda Networks Inc. reserves the right to change, modify, transfer, or otherwise revise this publication without notice.