2015

SCCM Application Packaging

Part 3 – PowerShell Examples

A ZEN GUIDE TO SCCM APPLICATION PACKAGING WITH POWERSHELL

JERRY "ZEN" SENFF

Contents

Documentation

The documentation has been broken into three components to make both reading and downloading from GitHub easier:

- Part 1 covers the application packaging process and the SCCM interface.
- Part 2 covers the PowerShell programming tips and tricks used with SCCM.
- Part 3 is a collection of example PowerShell install and uninstall programs demonstrating the PowerShell tips and tricks used with SCCM for the successful installation of complex software packages.

Example SCCM Application Packages

Some actual packaged applications to demonstrate processes described in ***SCCM Application Packaging – Part 2 – PowerShell***.

7-Zip

Description
7-Zip is an open source, file archiver with a high compression ratio.

Package Notes/Comments – Copy to all **Notes/Comments** sections in SCCM creating package.
Free software license.

Source file location (share or weblink):
http://www.7-zip.org/download.html

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\7zip

Import *7z920-x64.msi* into SCCM Applications as an *MSI Installer*.

Installation program:
msiexec /i "7z920-x64.msi" /qn /norestart

Uninstall program:
msiexec /x {23170F69-40C1-2702-0920-000001000000} /qn /norestart

Detection Method:
MSI Product Code: {23170F69-40C1-2702-0920-000001000000}

User Experience:
Behavior: Install for system
Logon: Whether or not a user is logged on
Visibility: Hidden
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):

Notes
GUID: {23170F69-40C1-2702-0920-000001000000} (64-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{23170F69-40C1-2702-0920-000001000000}

UninstallString (determines uninstall method to use):
MsiExec.exe /I{23170F69-40C1-2702-0920-000001000000}

DisplayName: 7-Zip 9.20 (x64 edition)
DisplayVersion: 9.20.00.0
Publisher: Igor Pavlov

Agent Ransack – Manual

This application was used for the MSI Installation screenshots in the **2. How to Package MSI Installers** section of the ***SCCM Application Packaging – Part 1 – SCCM*** document. The final SCCM application package version ended up being a **Script Installer** that required user interaction to accept the EULA and desktop interaction to create shortcuts on the Desktop and in the Start Menu.

The Process:
Initially, the `install64.msi` was extracted from the downloaded executable and used to create an **MSI Installer** using the /qn switch to silently install. However, after the SCCM silent MSI install, no shortcuts were available on the Desktop or in the Start **Menu**. Starting the application from the install folder worked, but the Office 2010 Filter Pack integration did not work. Some registry entries were missing or not mapped correctly. Switched the SCCM package type to a **Script Installer** using the downloaded executable as the **Installation program**. Changed the **Visibility** to **Normal** since the .EXE had no switches available for silently installing. Now, the user must accept the EULA, the setup executable gets to interact with the Desktop to create shortcuts, and all the correct registry magic happens so the Office 2010 Filter Pack integration works. Problem solved.

Description
Finding files that other search engines miss. Agent Ransack is a free 'lite' version of FileLocator Pro.

Package Notes/Comments – Copy to all **Notes**/**Comments** sections in SCCM creating package.
Free software license.

Source file location (share or weblink):
http://www.mythicsoft.com/agentransack/download

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\Agent Ransack

Import *AgentRansack_822.exe* into SCCM Applications as a *Script Installer*.

Installation program:
"AgentRansack_822.exe"

Uninstall program:
msiexec /x {D7DDA334-FF1D-4A04-B056-22AB301026C8} /qn /norestart

Detection Method:
MSI Product Code: {D7DDA334-FF1D-4A04-B056-22AB301026C8}

User Experience:
Behavior: Install for user
Logon: Only when a user is logged on
Visibility: Normal
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
Office 2010 Filter Packs (64-bit)

Notes
GUID: {D7DDA334-FF1D-4A04-B056-22AB301026C8} (64-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{D7DDA334-FF1D-4A04-B056-22AB301026C8}

UninstallString (determines uninstall method to use):
MsiExec.exe /X{D7DDA334-FF1D-4A04-B056-22AB301026C8}

DisplayName: Agent Ransack x64
DisplayVersion: 7.0.822.1
Publisher: Mythicsoft Ltd

Alpha Five v6 - Manual

<u>Description</u>
Develop custom desktop and web applications quickly and easily with Alpha Five V6. With no programming skills required, Version 6's application scripting and web components allow you to take control of your information helping you build applications for your particular business or organization. Alpha Five V6 works with a built-in DBF engine and MySQL, Oracle, SQL Server, DB2 and ODBC sources.

Category: Software Development.

<u>Package Notes/Comments</u> – Copy to all **Notes/Comments** sections in SCCM creating package.
Single license package. License key required to activate product. For first time run, choose "Run as Administrator".

Source file location (share or weblink):
\\server\share\installs\Alpha 5 V6

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\Alpha 5 V6

Create a PowerShell program, *AlphaFive_v6_Manual_Uninstall.ps1*, to uninstall AlphaFive v6.

Import *setup.exe* into SCCM Applications as a *Script Installer*.

Installation program:
"setup.exe"

Uninstall program:
Powershell.exe –executionpolicy Bypass –file "AlphaFive_v6_Manual_Uninstall.ps1"

Detection Method:
Registry key exists.
Registry hive: HKLM
Registry key: SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Alpha Five V6

User Experience:
Behavior: Install for user
Logon: Only when a user is logged on
Visibility: Normal
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
None

<u>Notes</u>
GUID: Alpha Five V6 (32-bit)

Registry key or file path (used to check if application is installed):

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Alpha Five V6

UninstallString (determines uninstall method to use):
C:\PROGRA~2\A5V6\UNWISE.EXE C:\PROGRA~2\A5V6\INSTALL.LOG

DisplayName: Alpha Five V6
DisplayVersion: 5.1 build 1532
Publisher: Alpha Software Inc.

AlphaFive_v6_Manual_Uninstall.ps1

```
<#
    .SYNOPSIS
      SCCM Uninstall program for Alpha Five v6 - Manual

    .DESCRIPTION
      Remove the following component(s) from Window 7 x64 systems:
        - Alpha Five v6 - Manual

    .NOTES
      FileName: AlphaFive_v6_Manual_Uninstall.ps1
      Author: Jerry Senff
      Created: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file
"AlphaFive_v6_Manual_Uninstall.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\AlphaFive_v6_Manual_Uninstall.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Uninstall Alpha Five v6
$AlphaFive = 'Alpha Five v6'
$AlphaFiveUninstall = ${env:SystemDrive} + '\PROGRA~2\A5V6\UNWISE.EXE'
$AlphaFiveParams = ${env:SystemDrive} + '\PROGRA~2\A5V6\INSTALL.LOG'
$AlphaFiveRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Alpha Five V6'

Write-Host ""
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "$AlphaFive Uninstaller"
Write-Host ""
Write-Host "Purpose: Remove the following components:"
Write-Host "          - Uninstall $AlphaFive"
Write-Host ""

# Uninstall Alpha Five v6
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $AlphaFive installation..."
Write-Host ""

If ((Test-Path -Path $AlphaFiveRegkey) -eq 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $AlphaFive is installed."
    Write-Host ""
    Write-Host "Uninstalling $AlphaFive..."
    Write-Host ""
    Write-Host "Command:  $AlphaFiveUninstall $AlphaFiveParams"
    Write-Host ""
    Start-Process -FilePath $AlphaFiveUninstall -ArgumentList $AlphaFiveParams -
ErrorVariable +err -Verb Open -Wait
    Write-Host ""
    Write-Host "    $AlphaFive uninstall complete."
    Write-Host ""
}
Else
{
    Write-Host "    $AlphaFive is not installed."
    Write-Host ""
}

Start-Sleep 5

$err | Out-File $errFileLocation
Write-Host "*** Log file location = " $errFileLocation " ***"

Start-Sleep 5
```

Cyberlink PowerDVD - Manual

Description
A media player for Microsoft Windows providing DVD playback, with Blu-ray playback available in higher editions.

Package Notes/Comments – Copy to all **Notes**/**Comments** sections in SCCM creating package.
Free software license.

Source file location (share or weblink):
\\server\share\installs\Cyberlink PowerDVD

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\Cyberlink PowerDVD

Create a PowerShell program, *CyberlinkPowerDVD_Manual_Uninstall.ps1*, to uninstall Cyberlink PowerDVD.

Import *Setup.exe* into SCCM Applications as a *Script Installer*.

Installation program:
"Setup.exe"

Uninstall program:
Powershell.exe –executionpolicy Bypass –file "CyberlinkPowerDVD_Manual_Uninstall.ps1"

Detection Method:
Registry key exists.
Registry hive: HKLM
Registry key: SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{6811CAA0-BF12-11D4-9EA1-0050BAE317E1}

User Experience:
Behavior: Install for user
Logon: Only when a user is logged on
Visibility: Normal
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
None

Notes
GUID: {6811CAA0-BF12-11D4-9EA1-0050BAE317E1} (32-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{6811CAA0-BF12-11D4-9EA1-0050BAE317E1}

UninstallString (determines uninstall method to use):
RunDll32
C:\PROGRA~2\COMMON~1\INSTAL~1\PROFES~1\RunTime\11\00\Intel32\Ctor.dll,LaunchSetup "C:\Program Files (x86)\InstallShield Installation Information\{6811CAA0-BF12-11D4-9EA1-0050BAE317E1}\Setup.exe" -l0x9 -cluninstall

DisplayName: PowerDVD DX
DisplayVersion: 8.3.5424
Publisher: CyberLink Corp.

CyberlinkPowerDVD_Manual_Uninstall.ps1

```
<#
    .SYNOPSIS
      SCCM Uninstall program for CyberLink PowerDVD DX

    .DESCRIPTION
      Remove the following component(s) from Window 7 x64 systems:
        - CyberLink PowerDVD DX

    .NOTES
      FileName: CyberlinkPowerDVD_Manual_Uninstall.ps1
      Author: Jerry Senff
      Created: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file 
"CyberlinkPowerDVD_Manual_Uninstall.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\CyberlinkPowerDVD_Manual_Uninstalltxt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Uninstall CyberLink PowerDVD DX
$PowerDVD = 'CyberLink PowerDVD DX'
$PowerDVDUninstall = 'RunDll32'
$PowerDVDParams = (${env:SystemDrive} + 
'\PROGRA~2\COMMON~1\INSTAL~1\PROFES~1\RunTime\11\00\Intel32\Ctor.dll,LaunchSetup "' + 
${env:ProgramFiles(x86)} + '\InstallShield Installation Information\{6811CAA0-BF12-
11D4-9EA1-0050BAE317E1}\Setup.exe" -l0x9  -cluninstall')
$PowerDVDRegkey = 
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{6811CAA0-BF12-
11D4-9EA1-0050BAE317E1}'

Write-Host ""
Write-Host "******************************************************************"
Write-Host ""
Write-Host "$PowerDVD Uninstaller"
Write-Host ""
Write-Host "Purpose: Remove the following components:"
Write-Host "          - Uninstall $PowerDVD"
Write-Host ""

# Uninstall CyberLink PowerDVD DX
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking for $PowerDVD installation..."
Write-Host ""

If ((Test-Path -Path $PowerDVDRegkey) -eq 'True')  # Check uninstall registry key or 
install path
```

```
{
    Write-Host "    $PowerDVD is installed."
    Write-Host ""
    Write-Host "Uninstalling $PowerDVD..."
    Write-Host ""
    Write-Host "Command:  $PowerDVDUninstall $PowerDVDParams"
    Write-Host ""
    Start-Process -FilePath $PowerDVDUninstall -ArgumentList $PowerDVDParams -
ErrorVariable +err -Verb Open -Wait
    Write-Host ""
    Write-Host "    $PowerDVD uninstall complete."
    Write-Host ""
}
Else
{
    Write-Host "    $PowerDVD is not installed."
    Write-Host ""
}

Start-Sleep 5

$err | Out-File $errFileLocation
Write-Host "*** Log file location = " $errFileLocation " ***"

Start-Sleep 5
```

EasyLobby 10 - Manual

EasyLobby 10 was an application installation with many configuration tasks. Everyone said it could not be packaged for distribution through SCCM. The development time covered several days over a two-week period. The setup instructions forwarded by the team requesting the packaging provided a framework for the PowerShell install and uninstall programs. The system file and folder ACL permission changes required some research since the initial efforts threw errors and the ACLs were never changed. Some Internet searching led to the *Adjusting Token Privileges with PowerShell* snippet by Precision Computing. Once that snippet was included at the beginning of the install and uninstall PowerShell programs, the `Set-Acl` commands worked on the System files and folders. No more errors. After solving that issue, the application package was fine-tuned as the technicians installing the package provided feedback.

Description
HID Global EasyLobby® Secure Visitor Management (SVM™) software is the main application for processing visitors, including ID scanning, record creation, badge printing, watch list screening, check-in and check-out, and email notification, among other features. The solution improves security by enabling organizations to identify exactly who is in their facilities, while enhancing the professionalism of an organization by streamlining the visitor check-in process and providing high quality visitor badges

Package
Single License package. License key required to complete setup. Scanner model required for ScanSnap driver and SDK installations.

Source file *EasyLobby100ISO.iso* location:
\\server\image$\EasyLobby

Use *Virtual CloneDrive* to open the ISO image and copy all the files to the location below.

Extract ISO source files to:
\\server\share\Sources\Software Vault\EasyLobby10\Install

Create an PowerShell program, *EasyLobby10_Manual_Install.ps1*, to install EasyLobby 10 components and make environmental changes to the desktop system.

Create a second PowerShell program, *EasyLobby10_Manual_Uninstall.ps1*, to uninstall all the EasyLobby 10 components and back out the environmental changes made on the desktop system during the installation process.

Import *EasyLobby10_Manual_Install.ps1* into SCCM Applications as a *Script Installer*.

Installation program:
Powershell.exe –executionpolicy Bypass –file "./EasyLobby10_Manual_Install.ps1"

Uninstall program:
Powershell.exe –executionpolicy Bypass –file "./EasyLobby10_Manual_Uninstall.ps1"

Detection Method:

Registry key exists.
Registry hive: HKLM
Registry key: SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EasyLobby SVM
Value: DisplayVersion
Type: String
Equals: 10.0

User Experience:
Behavior: Install for user
Logon: Only when a user is logged on
Visibility: Normal
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
None

<u>**Notes**</u>

EasyLobby SVM 10.0
Installs the following software:

- EasyLobby SVM
- SCCN SDK Version 9.50.19

Install string: EasyLobbySVM100Setup.exe

GUID: EasyLobby SVM (32-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EasyLobby SVM

UninstallString (determines uninstall method to use):
%ProgramFiles(x86)%\EasyLobby\EasyLobby SVM 10.0\Uninstall.EXE

DisplayName: EasyLobby SVM
DisplayVersion: 10.0
Publisher: HID Global

ScanSnap Drivers 9.50
No entries appear in Programs and Features for any of the scanner drivers selected during setup.

Note: Uninstalls as part of EasyLobby SVM 10 uninstall which deletes the subfolder containing the ScanShell drivers.

Driver folders are located at:
C:\Program Files (x86)\EasyLobby\EasyLobby SVM 10.0\ScanShell

- SnapShell (default) installs two folders:
  - 64Bit
  - Backup
  - Snapshell Windows 7 64bit
- ScanShell 800 installs three folders:
  - Backup
  - ScanShell800 Windows 7 64bit
  - ScanShell800R Windows 7 64bit
- ScanShell 1000 installs one folder:
  - Backup
- ScanShell 1000A installs three folders:
  - Backup
  - ScanShell1000A Windows 7 64bit
  - ScanShell1000AN Windows 7 64bit

Install string: ScanShellDriversSetup950.exe

GUID: CSSN SDK Version 9.50 (32-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\CSSN SDK Version 9.50

UninstallString (determines uninstall method to use):
C:\PROGRA~2\EASYLO~1\EASYLO~1.0\SCANSH~1\UNWISE.EXE
C:\PROGRA~2\EASYLO~1\EASYLO~1.0\SCANSH~1\INSTALL.LOG
- Note: Uninstalls as part of EasyLobby10-Uninstall

DisplayName: CSSN SDK
DisplayVersion: 9.50
Publisher: HID Global

CSSN SDK Version 9.50.19
Installs CSSN SDK Version 9.50.19 again, but with more options.

Install string: sdk_setup.exe

GUID: CSSN SDK Version 9.50.19 (32-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\CSSN SDK Version 9.50.19

UninstallString (determines uninstall method to use):
C:\PROGRA~2\CARDSC~1\SDK\UNWISE.EXE C:\PROGRA~2\CARDSC~1\SDK\INSTALL.LOG

DisplayName: CSSN SDK Version 9.50.19

DisplayVersion: 9.50.19
Publisher: HID Global

EasyLobby10_Manual_Install.ps1

```
<#
    .SYNOPSIS
      SCCM install program for EasyLobby 10 - Manual

    .DESCRIPTION
      Install the following component(s) on Window 7 x64 systems:
        - EasyLobby SVM 10.0 and CSSN SDK 9.50
        - ScanShell Drivers 9.50
        - CSSN SDK 9.50.19  with more options

      Additional configuration tasks:
        - Copies  two files from a share to the local drive
        - Add Security domain group to local Power Users group
        - Grant Power Users Full Control of three folders
        - Enable "Show hidden files, etc" in Folder Options
        - Copy three shortcuts to the Default User's Desktop

    .NOTES
      FileName: EasyLobby10_Manual_Install.ps1
      Author: Jerry Senff
      Created: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file
"EasyLobby10_Manual_Install.ps1"
#>

<#######################################################################################
#######

    BEGIN SUBROUTINE

      Enable 'SeTakeOwnershipPrivilege' to set system folder ACLs without errors
        - Privilege token only good for current PowerShell session

      "Adjusting Token Privileges with PowerShell" by Precision Computing, copyright
2010

      Retrieved from:

      http://www.leeholmes.com/blog/2010/09/24/adjusting-token-privileges-in-
powershell/
#>

function enable-privilege {
 param(
  ## The privilege to adjust. This set is taken from
  ## http://msdn.microsoft.com/en-us/library/bb530716(VS.85).aspx
  [ValidateSet(
   "SeAssignPrimaryTokenPrivilege", "SeAuditPrivilege", "SeBackupPrivilege",
   "SeChangeNotifyPrivilege", "SeCreateGlobalPrivilege", "SeCreatePagefilePrivilege",
   "SeCreatePermanentPrivilege", "SeCreateSymbolicLinkPrivilege",
"SeCreateTokenPrivilege",
   "SeDebugPrivilege", "SeEnableDelegationPrivilege", "SeImpersonatePrivilege",
"SeIncreaseBasePriorityPrivilege",
   "SeIncreaseQuotaPrivilege", "SeIncreaseWorkingSetPrivilege",
"SeLoadDriverPrivilege",
   "SeLockMemoryPrivilege", "SeMachineAccountPrivilege", "SeManageVolumePrivilege",
   "SeProfileSingleProcessPrivilege", "SeRelabelPrivilege",
"SeRemoteShutdownPrivilege",
   "SeRestorePrivilege", "SeSecurityPrivilege", "SeShutdownPrivilege",
"SeSyncAgentPrivilege",
   "SeSystemEnvironmentPrivilege", "SeSystemProfilePrivilege",
"SeSystemtimePrivilege",
   "SeTakeOwnershipPrivilege", "SeTcbPrivilege", "SeTimeZonePrivilege",
"SeTrustedCredManAccessPrivilege",
   "SeUndockPrivilege", "SeUnsolicitedInputPrivilege")]
  $Privilege,
  ## The process on which to adjust the privilege. Defaults to the current process.
  $ProcessId = $pid,
  ## Switch to disable the privilege, rather than enable it.
  [Switch] $Disable
 )

 ## Taken from P/Invoke.NET with minor adjustments.
 $definition = @'
 using System;
 using System.Runtime.InteropServices;

 public class AdjPriv
 {
  [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
  internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disall,
   ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);

  [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
  internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);
  [DllImport("advapi32.dll", SetLastError = true)]
  internal static extern bool LookupPrivilegeValue(string host, string name, ref long
pluid);
  [StructLayout(LayoutKind.Sequential, Pack = 1)]
  internal struct TokPriv1Luid
  {
   public int Count;
   public long Luid;
   public int Attr;
  }

  internal const int SE_PRIVILEGE_ENABLED = 0x00000002;
  internal const int SE_PRIVILEGE_DISABLED = 0x00000000;
  internal const int TOKEN_QUERY = 0x00000008;
  internal const int TOKEN_ADJUST_PRIVILEGES = 0x00000020;
  public static bool EnablePrivilege(long processHandle, string privilege, bool
disable)
  {
   bool retVal;
   TokPriv1Luid tp;
   IntPtr hproc = new IntPtr(processHandle);
   IntPtr htok = IntPtr.Zero;
   retVal = OpenProcessToken(hproc, TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, ref htok);
   tp.Count = 1;
   tp.Luid = 0;
   if(disable)
   {
    tp.Attr = SE_PRIVILEGE_DISABLED;
   }
   else
   {
    tp.Attr = SE_PRIVILEGE_ENABLED;
   }
   retVal = LookupPrivilegeValue(null, privilege, ref tp.Luid);
   retVal = AdjustTokenPrivileges(htok, false, ref tp, 0, IntPtr.Zero, IntPtr.Zero);
   return retVal;
  }
 }
'@

 $processHandle = (Get-Process -id $ProcessId).Handle
 $type = Add-Type $definition -PassThru
 $type[0]::EnablePrivilege($processHandle, $Privilege, $Disable)
}

# Enable 'SeTakeOwnershipPrivilege' to set ACLs without errors
enable-privilege SeTakeOwnershipPrivilege |out-null

<#
```

```
    END SUBROUTINE

#####################################################################################
######>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\EasyLobby10_Manual_Install.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Install EasyLobby SVM 10.0 and CSSN SDK 9.50
$EasyLobbyName = 'EasyLobby 10 and CSSN SDK 9.50'
$EasyLobbySvmInstall = '.\EasyLobbySVM100Setup.exe'
$EasyLobbyRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EasyLobby SVM'

# Install ScanShell Drivers 9.50
$ScanSnapDriversName = 'ScanShell Drivers 9.50'
$ScanSnapDriversInstall = '.\ScanShellDriversSetup950.exe'
$ScanSnapInstallCheck = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\ScanShell\Backup')

# Install CSSN SDK Version 9.50 with more options
$CssnSdkName = 'CSSN SDK Version 9.50.19 with additional options'
$CssnSdkInstall = '.\sdk_setup.exe'
$CssnSdkRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\CSSN SDK Version
9.50.19'

# Copies two files from a share to the local drive
$CopyFile1name = '.\_options.el'
$CopyFile2name = '.\New logo.bmp'
$FileCopyDestination = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM 10.0')

# Add Security Domain Group to the local Power Users group
$DomainGroup = 'Security'
$Domain = 'YOURDOMAIN'
$LocalGroup = 'Power Users'

# Grant Power Users Full Control of three folders
$folder1 = (${env:ProgramFiles(x86)} + '\EasyLobby')
$folder2 = (${env:windir} + '\Temp')
$folder3 = (${env:windir} + '\twain_32')

$User = 'BUILTIN\Power Users'
$FolderRights = 'FullControl'
$Flags = 'ContainerInherit,ObjectInherit'
$AllowDeny = 'Allow'

$objUser = New-Object System.Security.Principal.NTAccount("$User")
$colRights = [system.Security.AccessControl.fileSystemRights]$FolderRights
$InheritanceFlag = [System.Security.AccessControl.InheritanceFlags]$Flags
$PropagationFlag = [System.Security.AccessControl.PropagationFlags]::None
$objType = [system.Security.AccessControl.AccessControlType]::$AllowDeny
$objAr = New-Object  System.Security.AccessControl.FileSystemAccessRule($objUser,
$colRights, $InheritanceFlag, $PropagationFlag, $objType)

# Enable "Show hidden files, etc" in Folder Options
$ExplorerHiddenRegkey =
'HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced'

# Copy three shortcuts to the Default User's Desktop
$WshShell = New-Object -comObject WScript.Shell
$SourceExe1 = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\Utilities\EmployeeImport.exe')
$SourceExe2 = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\PhotoExport.exe')
$SourceExe3 = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\EasyLobbySVM.EXE')
$DestinationPath1 = (${env:SystemDrive} + '\Users\Default\Desktop\EmployeeImport.lnk')
$DestinationPath2 = (${env:SystemDrive} + '\Users\Default\Desktop\PhotoExport.lnk')
```

```
$DestinationPath3 = (${env:SystemDrive} + '\Users\Default\Desktop\EasyLobbySVM.lnk')

Write-Host ""
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "$EasyLobbyName Install"
Write-Host ""
Write-Host "Purpose: Install the following components:"
Write-Host "         - $EasyLobbyName"
Write-Host "         - $ScanSnapDriversName"
Write-Host "         - $CssnSdkName"
Write-Host ""
Write-Host "Additional configuration tasks:"
Write-Host "         - Copies  two files from a share to the local drive"
Write-Host "         - Adds two users to the Power Users group"
Write-Host "         - Grants Power Users Full Control of three folders"
Write-Host "         - Enables 'Show hidden files, etc' in Folder Options"
Write-Host "         - Copies three shortcuts to the Default User's Desktop"

# Install EasyLobby SVM 10.0 and CSSN SDK 9.50.19
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $EasyLobbyName installation..."
Write-Host ""

If ((Test-Path -Path $EasyLobbyRegkey) -ne 'True')
{
    Write-Host "    $EasyLobbyName is not installed."
    Write-Host ""
    Write-Host "Installing $EasyLobbyName..."
    Write-Host ""
    Write-Host "Command: " $EasyLobbySvmInstall
    Start-Process -FilePath $EasyLobbySvmInstall -ErrorVariable +err -Verb Open -Wait
    Write-Host ""
    Write-Host "    $EasyLobbyName install complete."
    Write-Host ""
}
Else
{
    Write-Host "    $EasyLobbyName already installed."
    Write-Host ""
}

Start-Sleep 5

# Install ScanShell Drivers 9.50
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $ScanSnapDriversName installation..."
Write-Host ""

If ((Test-Path -Path $ScanSnapInstallCheck) -ne 'True')
{
    Write-Host "    $ScanSnapDriversName is not installed."
    Write-Host ""
    Write-Host "Installing $ScanSnapDriversName..."
    Write-Host ""
    Write-Host "Command: " $ScanSnapDriversInstall
    Start-Process -FilePath $ScanSnapDriversInstall -ErrorVariable +err -Verb Open -
Wait
    Write-Host ""
    Write-Host "    $ScanSnapDriversName install complete."
    Write-Host ""
}
Else
{
    Write-Host "    $ScanSnapDriversName already installed."
    Write-Host ""
}
```

```
Start-Sleep 5

# Install CSSN SDK Version 9.50.19 with more options
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $CssnSdkName installation..."
Write-Host ""

If ((Test-Path -Path $CssnSdkRegkey) -ne 'True')
{
    Write-Host "    $CssnSdkName is not installed."
    Write-Host ""
    Write-Host "Installing $CssnSdkName..."
    Write-Host ""
    Write-Host "Command: " $CssnSdkInstall
    Start-Process -FilePath $CssnSdkInstall -ErrorVariable +err -Verb Open -Wait
    Write-Host ""
    Write-Host "    $CssnSdkName install complete."
    Write-Host ""
}
Else
{
    Write-Host "    $CssnSdkName already installed."
    Write-Host ""
}

Start-Sleep 5

# Copies two files from a share to the local drive
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Copying two files to local drive..."
Write-Host ""
Write-Host "Command: Copy-Item $CopyFile1name $FileCopyDestination -Force"
Copy-Item $CopyFile1name $FileCopyDestination -Force
Write-Host ""
Write-Host "Command: Copy-Item $CopyFile2name $FileCopyDestination -Force"
Copy-Item $CopyFile2name $FileCopyDestination -Force
Write-Host ""
Write-Host "    File copy complete."
Write-Host ""

Start-Sleep 5

# Add Security Domain Group to the local Power Users group
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for Security group membership in local Power Users group..."
Write-Host ""

$r = (([ADSI]"WinNT://./$LocalGroup").Invoke("IsMember",
"WinNT://$Domain/$DomainGroup"))

$string = '[ADSI]"WinNT://.' + $LocalGroup + ',group").Add("WinNT://' + $Domain + '/'
+ $DomainGroup + '")'

If ($r -match 'False')
{
    Write-Host "Adding Security Group to Power Users Group..."
    Write-Host ""
    Write-Host 'Command: ' $string
    Write-Host ""
    ([ADSI]"WinNT://./$LocalGroup,group").Add("WinNT://$Domain/$DomainGroup")
    Write-Host ""
    Write-Host "    Adding Security Group to Power Users complete."
    Write-Host ""
}
Else
{
    Write-Host "    Security group is already a member of local Power Users group"
    Write-Host ""
}

Start-Sleep 5

# Grant Power Users Full Control of three folders
# Set ACL on EasyLobby folder
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking ACL on $folder1..."
Write-Host ""

$r = (Get-Acl $folder1).Access | where {$_.IdentityReference -eq $User -and
$_.FileSystemRights -eq $FolderRights -and $_.AccessControlType -eq $AllowDeny}

If ($r -eq $null)
{
    write-host "    $User group does not have $FolderRights permissions on $folder1"
    Write-Host ""

    $objAcl1 = Get-Acl $folder1
    $string = ('' + $objAcl1 + '.SetAccessRule(' + $objAr + ')')

    Write-Host ""
    Write-Host "Granting $User group $FolderRights for $folder1"
    Write-Host ""
    Write-Host "Command: " $string
    Write-Host ""
    $objAcl1.SetAccessRule($objAr)
    Write-Host ""
    Write-Host "Setting new access list on" $folder1
    Write-Host ""
    Write-Host "Command: Set-Acl" $folder1 $objAcl1
    Write-Host ""
    Set-Acl $folder1 $objAcl1
    Write-Host ""
    Write-Host "    New" $folder1 "access list set complete."
    Write-Host ""
}
Else
{
    Write-Host "    $User group already have $FolderRights for $folder1"
    Write-Host ""
}

Start-Sleep 2

# Set ACL on Temp folder
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking ACL on $folder2..."
Write-Host ""

$r = (Get-Acl $folder2).Access | where {$_.IdentityReference -eq $User -and
$_.FileSystemRights -eq $FolderRights -and $_.AccessControlType -eq $AllowDeny}

If ($r -eq $null)
{
    write-host "    $User group does not have $FolderRights permissions on $folder2"
    Write-Host ""

    $objAcl2 = Get-Acl $folder2
    $string = ('' + $objAcl2 + '.SetAccessRule(' + $objAr + ')')

    Write-Host ""
    Write-Host "Granting $User group $FolderRights for $folder2"
    Write-Host ""
    Write-Host "Command: " $string
    Write-Host ""
    $objAcl2.SetAccessRule($objAr)
    Write-Host ""
    Write-Host "Setting new access list for" $folder2
    Write-Host ""
    Write-Host "Command: Set-Acl" $folder2 $objAcl2
    Write-Host ""
    Set-Acl $folder2 $objAcl2
    Write-Host ""
    Write-Host "    New $folder2 access list set complete."
    Write-Host ""
}
Else
{
    Write-Host "    $User group already have $FolderRights for $folder2"
    Write-Host ""
}

Start-Sleep 2

# Set ACL on twain_32 folder
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking ACL on $folder3..."
Write-Host ""

$r = (Get-Acl $folder3).Access | where {$_.IdentityReference -eq $User -and
$_.FileSystemRights -eq $FolderRights -and $_.AccessControlType -eq $AllowDeny}

If ($r -eq $null)
{
    write-host "    $User group does not have $FolderRights permissions on $folder3"
    Write-Host ""

    $objAcl3 = Get-Acl $folder3
    $string = ('' + $objAcl3 + '.SetAccessRule(' + $objAr + ')')

    Write-Host ""
    Write-Host "Granting $User group $FolderRights for $folder3"
    Write-Host ""
    Write-Host "Command: " $string
    Write-Host ""
    $objAcl3.SetAccessRule($objAr)
    Write-Host ""
    Write-Host "Setting new access list on" $folder3
    Write-Host ""
    Write-Host "Command: Set-Acl" $folder3 $objAcl3
    Write-Host ""
    Set-Acl $folder3 $objAcl3
    Write-Host ""
    Write-Host "    New" $folder3 "access list set complete."
    Write-Host ""
}
Else
{
    Write-Host "    $User group already have $FolderRights for $folder3"
    Write-Host ""
}

Start-Sleep 5

# Enable "Show hidden files, etc" in Folder Options
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Enabling 'Show hidden files, etc' in Folder Options..."
Write-Host ""
Write-Host "Command: Set-ItemProperty $ExplorerHiddenRegkey Hidden 1"
Write-Host ""
Set-ItemProperty $ExplorerHiddenRegkey Hidden 1

Sleep 2
```

```
Write-Host ""
Write-Host "Stopping and restarting the Explorer process..."
Write-Host ""
Write-Host "Command: Stop-Process -processname explorer"
Write-Host ""
Stop-Process -processname explorer
Write-Host ""
Write-Host "    'Show hidden files...' enable complete."
Write-Host ""

Start-Sleep 5

# Copy three shortcuts to the Default User's Desktop
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking for 'EmployeeImport.exe' shortcut on Default User's Desktop..."
Write-Host ""

If ((Test-Path -Path $DestinationPath1) -ne 'True')
{
    Write-Host "    'EmployeeImport.exe' shortcut on Default User's Desktop does not
exist."
    Write-Host ""
    Write-Host "Copying 'EmployeeImport.exe' shortcut Default User's Desktop..."
    Write-Host ""
    $Shortcut = $WshShell.CreateShortcut($DestinationPath1)
    $Shortcut.TargetPath = $SourceExe1
    $Shortcut.Save()
    Write-Host ""
    Write-Host "    Copying 'EmployeeImport.exe' shortcut complete."
    Write-Host ""
}
Else
{
    Write-Host "    'EmployeeImport.exe' shortcut already exists."
    Write-Host ""
}

Start-Sleep 2

set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking for 'PhotoExport.exe' shortcut on Default User's Desktop..."
Write-Host ""

If ((Test-Path -Path $DestinationPath2) -ne 'True')
{
    Write-Host "    'PhotoExport.exe' shortcut on Default User's Desktop does not
exist."
    Write-Host ""
    Write-Host "Copying 'PhotoExport.exe' shortcut Default User's Desktop..."
    Write-Host ""
    $Shortcut = $WshShell.CreateShortcut($DestinationPath2)
    $Shortcut.TargetPath = $SourceExe2
    $Shortcut.Save()
    Write-Host ""
    Write-Host "    Copying 'PhotoExport.exe' shortcut complete."
    Write-Host ""
}
Else
{
    Write-Host "    'PhotoExport.exe' shortcut already exists."
    Write-Host ""
}

Start-Sleep 2

set-location $startLocation
Write-Host "******************************************************************"
```

```
Write-Host ""
Write-Host "Checking for 'EasyLobbySVM.EXE' shortcut on Default User's Desktop..."
Write-Host ""

If ((Test-Path -Path $DestinationPath3) -ne 'True')
{
    Write-Host "    'EasyLobbySVM.EXE' shortcut on Default User's Desktop does not
exist."
    Write-Host ""
    Write-Host "Copying 'EasyLobbySVM.EXE' shortcut Default User's Desktop..."
    Write-Host ""
    $Shortcut = $WshShell.CreateShortcut($DestinationPath3)
    $Shortcut.TargetPath = $SourceExe3
    $Shortcut.Save()
    Write-Host ""
    Write-Host "    Copying 'EasyLobbySVM.EXE' shortcut complete."
    Write-Host ""
}
Else
{
    Write-Host "    'EasyLobbySVM.EXE' shortcut already exists."
    Write-Host ""
}

Start-Sleep 5

Write-Host "*******************************************************************"
Write-Host ""
Write-Host "    EasyLobby SVM 10.0 installation complete."
Write-Host ""
Write-Host "*******************************************************************"
Write-Host ""

# Indicate location of error log file
$err | Out-File $errFileLocation
Write-Host "*** Log file location = " $errFileLocation " ***"
Write-Host ""

Start-Sleep 5
```

EasyLobby10_Manual_Uninstall.ps1

```
<#
    .SYNOPSIS
      SCCM uninstall program for EasyLobby 10 - Manual

    .DESCRIPTION
      Uninstall the following component(s) on Window 7 x64 systems:
        - CSSN SDK Version 9.50.19
        - ScanShell Drivers 9.50
        - EasyLobby SVM 10.0

      Additional configuration tasks:
        - Delete leftover install folders
        - Delete three shortcuts from the Default User's Desktop
        - Remove Power Users Full Control of two folders
        - Remove Security group from local Power Users group

    .NOTES
      FileName: EasyLobby10_Manual_Uninstall.ps1
      Author: Jerry Senff
      Created: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file
"EasyLobby10_Manual_Uninstall.ps1"
#>

<#####################################################################################
#######
```

```
    BEGIN SUBROUTINE

      Enable 'SeTakeOwnershipPrivilege' to set system folder ACLs without errors
        - Privilege token only good for current PowerShell session

      "Adjusting Token Privileges with PowerShell" by Precision Computing, copyright
2010

      Retrieved from:

      http://www.leeholmes.com/blog/2010/09/24/adjusting-token-privileges-in-
powershell/
#>

function enable-privilege {
 param(
  ## The privilege to adjust. This set is taken from
  ## http://msdn.microsoft.com/en-us/library/bb530716(VS.85).aspx
  [ValidateSet(
   "SeAssignPrimaryTokenPrivilege", "SeAuditPrivilege", "SeBackupPrivilege",
   "SeChangeNotifyPrivilege", "SeCreateGlobalPrivilege", "SeCreatePagefilePrivilege",
   "SeCreatePermanentPrivilege", "SeCreateSymbolicLinkPrivilege",
"SeCreateTokenPrivilege",
   "SeDebugPrivilege", "SeEnableDelegationPrivilege", "SeImpersonatePrivilege",
"SeIncreaseBasePriorityPrivilege",
   "SeIncreaseQuotaPrivilege", "SeIncreaseWorkingSetPrivilege",
"SeLoadDriverPrivilege",
   "SeLockMemoryPrivilege", "SeMachineAccountPrivilege", "SeManageVolumePrivilege",
   "SeProfileSingleProcessPrivilege", "SeRelabelPrivilege",
"SeRemoteShutdownPrivilege",
   "SeRestorePrivilege", "SeSecurityPrivilege", "SeShutdownPrivilege",
"SeSyncAgentPrivilege",
   "SeSystemEnvironmentPrivilege", "SeSystemProfilePrivilege",
"SeSystemtimePrivilege",
   "SeTakeOwnershipPrivilege", "SeTcbPrivilege", "SeTimeZonePrivilege",
"SeTrustedCredManAccessPrivilege",
   "SeUndockPrivilege", "SeUnsolicitedInputPrivilege")]
  $Privilege,
  ## The process on which to adjust the privilege. Defaults to the current process.
  $ProcessId = $pid,
  ## Switch to disable the privilege, rather than enable it.
  [Switch] $Disable
 )

 ## Taken from P/Invoke.NET with minor adjustments.
 $definition = @'
 using System;
 using System.Runtime.InteropServices;

 public class AdjPriv
 {
  [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
  internal static extern bool AdjustTokenPrivileges(IntPtr htok, bool disall,
   ref TokPriv1Luid newst, int len, IntPtr prev, IntPtr relen);

  [DllImport("advapi32.dll", ExactSpelling = true, SetLastError = true)]
  internal static extern bool OpenProcessToken(IntPtr h, int acc, ref IntPtr phtok);
  [DllImport("advapi32.dll", SetLastError = true)]
  internal static extern bool LookupPrivilegeValue(string host, string name, ref long
pluid);
  [StructLayout(LayoutKind.Sequential, Pack = 1)]
  internal struct TokPriv1Luid
  {
   public int Count;
   public long Luid;
   public int Attr;
  }

  internal const int SE_PRIVILEGE_ENABLED = 0x00000002;
  internal const int SE_PRIVILEGE_DISABLED = 0x00000000;
  internal const int TOKEN_QUERY = 0x00000008;
  internal const int TOKEN_ADJUST_PRIVILEGES = 0x00000020;
  public static bool EnablePrivilege(long processHandle, string privilege, bool
disable)
  {
   bool retVal;
   TokPriv1Luid tp;
   IntPtr hproc = new IntPtr(processHandle);
   IntPtr htok = IntPtr.Zero;
   retVal = OpenProcessToken(hproc, TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, ref htok);
   tp.Count = 1;
   tp.Luid = 0;
   if(disable)
   {
    tp.Attr = SE_PRIVILEGE_DISABLED;
   }
   else
   {
    tp.Attr = SE_PRIVILEGE_ENABLED;
   }
   retVal = LookupPrivilegeValue(null, privilege, ref tp.Luid);
   retVal = AdjustTokenPrivileges(htok, false, ref tp, 0, IntPtr.Zero, IntPtr.Zero);
   return retVal;
  }
 }
'@

 $processHandle = (Get-Process -id $ProcessId).Handle
 $type = Add-Type $definition -PassThru
 $type[0]::EnablePrivilege($processHandle, $Privilege, $Disable)
}

# Enable 'SeTakeOwnershipPrivilege' to set ACLs without errors
enable-privilege SeTakeOwnershipPrivilege |out-null

<#
    END SUBROUTINE
#####################################################################################
######>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\EasyLobby10_Manual_Uninstall.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Uninstall variables
$CssnSdkName = 'CSSN SDK Version 9.50.19'
$CssnSdkRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\CSSN SDK Version
9.50.19'
$CssnSdk2Uninstall = (${env:ProgramFiles(x86)} + '\Card Scanning
Solutions\SDK\UNWISE.EXE')

$CssnSdk1Name = 'ScanShell Drivers 9.50'
$CssnSdk1Uninstall = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\ScanShell\UNWISE.EXE')
$CardScanningFolder = (${env:ProgramFiles(x86)} + '\Card Scanning Solutions')

$EasyLobbyName = 'EasyLobby SVM 10.0'
$EasyLobbyRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EasyLobby SVM'
$EasyLobbyUninstall = (${env:ProgramFiles(x86)} + '\EasyLobby\EasyLobby SVM
10.0\Uninstall.EXE')
$EasyLobbyFolder = (${env:ProgramFiles(x86)} + '\EasyLobby')

$EmployeeImportShortcut = (${env:SystemDrive} +
'\Users\Default\Desktop\EmployeeImport.lnk')
$PhotoExportShortcut  = (${env:SystemDrive} +
'\Users\Default\Desktop\PhotoExport.lnk')
$EasyLobbyShortcut = (${env:SystemDrive} + '\Users\Default\Desktop\EasyLobbySVM.lnk')

$User = 'BUILTIN\Power Users'
```

```
$FolderRights = 'FullControl'
$Flags = 'ContainerInherit,ObjectInherit'
$AllowDeny = 'Allow'

$objUser = New-Object System.Security.Principal.NTAccount("$User")
$colRights = [system.Security.AccessControl.fileSystemRights]$FolderRights
$InheritanceFlag = [System.Security.AccessControl.InheritanceFlags]$Flags
$PropagationFlag = [System.Security.AccessControl.PropagationFlags]::None
$objType = [system.Security.AccessControl.AccessControlType]::$AllowDeny
$objAr = New-Object  System.Security.AccessControl.FileSystemAccessRule($objUser,
$colRights, $InheritanceFlag, $PropagationFlag, $objType)

$DomainGroup = 'Security'
$Domain = 'YOURDOMAIN'
$LocalGroup = 'Power Users'

$folder2 = (${env:windir} + '\Temp')
$folder3 = (${env:windir} + '\twain_32')

Write-Host ""
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "$EasyLobbyName Uninstall"
Write-Host ""
Write-Host "Purpose: Uninstall the following components:"
Write-Host "         - $CssnSdkName"
Write-Host "         - $CssnSdk1Name"
Write-Host "         - $EasyLobbyName"
Write-Host ""
Write-Host "Additional configuration tasks:"
Write-Host "         - Delete leftover install folders"
Write-Host "         - Delete three shortcuts from the Default User's Desktop"
Write-Host "         - Remove Power Users Full Control of two folders"
Write-Host "         - Remove Security group from local Power Users group"

# Uninstall CSSN SDKs Instance
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $CssnSdkName installation..."
Write-Host ""

If ((Test-Path -Path $CssnSdkRegkey) -eq 'True')
{
    Write-Host "    $CssnSdkName is installed."
    Write-Host ""
    $title =  "$CssnSdkName Uninstall"
    $message = "Do you want to uninstall the $CssnSdkName ?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Uninstall $CssnSdkName"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$CssnSdkName"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Uninstalling $CssnSdkName..."
        Write-Host ""
        Write-Host "Command: " $CssnSdk2Uninstall
        Write-Host ""
        Start-Process -FilePath $CssnSdk2Uninstall -ErrorVariable +err -Verb Open -
Wait
```

```
        Write-Host ""
        Write-Host "    $CssnSdkName uninstall complete."
        Write-Host ""

        Start-Sleep 2

        Write-Host ""
        Write-Host "Uninstalling $CssnSdk1Name..."
        Write-Host ""
        Write-Host "Command: " $CssnSdk1Uninstall
        Write-Host ""
        Start-Process -FilePath $CssnSdk1Uninstall -ErrorVariable +err -Verb Open -
Wait
        Write-Host ""
        Write-Host "    $CssnSdk1Name uninstall complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping $CssnSdk1Name uninstall."
        Write-Host ""
    }
}
Else
{
    Write-Host "    $CssnSdk1Name not installed."
    Write-Host ""
}

Start-Sleep 5

# Uninstall EasyLobby SVM
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking for $EasyLobbyName installation..."
Write-Host ""

If ((Test-Path -Path $EasyLobbyRegkey) -eq 'True')
{
    Write-Host "    $EasyLobbyName is installed."
    Write-Host ""
    $title =  "$EasyLobbyName Uninstall"
    $message = "Do you want to uninstall $EasyLobbyName ?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Uninstall $EasyLobbyName"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$EasyLobbyName"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Uninstalling $EasyLobbyName..."
        Write-Host ""
        Write-Host "Command: " $EasyLobbyUninstall
        Write-Host ""
        Start-Process -FilePath $EasyLobbyUninstall -ErrorVariable +err -Verb Open -
Wait
        Write-Host ""
        Write-Host "    $EasyLobbyName uninstall complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping $EasyLobbyName uninstall."
        Write-Host ""
    }
}
Else
{
    Write-Host "    $EasyLobbyName not installed."
    Write-Host ""
}

Start-Sleep 5

# Delete Card Scanning Solutions folder in Program Files (x86)
set-location $startLocation
Write-Host "*******************************************************************"
Write-Host ""
Write-Host "Checking for Card Scanning Solutions install folder..."
Write-Host ""

if ((Test-Path -Path $CardScanningFolder) -eq 'True')
{
    Write-Host "    Card Scanning Solutions install folder exists."
    Write-Host ""
    $title =  "Card Scanning Solutions Install Folder"
    $message = "Do you want to delete the Card Scanning Solutions install folder?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Delete Card Scanning Solutions install folder"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
Card Scanning Solutions install folder"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Deleting Card Scanning Solutions install folder..."
        Write-Host ""
        Write-Host "Command:  Remove-Item -Path $CardScanningFolder -Recurse -Force"
        Write-Host ""
        Remove-Item -Path $CardScanningFolder -Recurse -Force
        Write-Host ""
        Write-Host "    Card Scanning Solutions install folder deletion complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping Card Scanning Solutions install folder deletion."
        Write-Host ""
    }
}
else
{
    Write-Host "    Card Scanning Solutions install folder does not exist."
    Write-Host ""
}

Start-Sleep 5

# Delete EasyLobby 10 install folder in Program Files (x86)
```

```
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for EasyLobby 10 install folder..."
Write-Host ""

if ((Test-Path -Path $EasyLobbyFolder) -eq 'True')
{
    Write-Host "    EasyLobby 10 install folder exists."
    Write-Host ""
    $title =  "EasyLobby 10 Install Folder"
    $message = "Do you want to delete the EasyLobby 10 install folder?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Delete EasyLobby 10 install folder"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
EasyLobby 10 install folder"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Deleting EasyLobby 10 install folder..."
        Write-Host ""
        Write-Host "Command:  Remove-Item -Path $EasyLobbyFolder -Recurse -Force"
        Write-Host ""
        Remove-Item -Path $EasyLobbyFolder -Recurse -Force
        Write-Host ""
        Write-Host "    EasyLobby 10 install folder deletion complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping EasyLobby 10 install folder deletion."
        Write-Host ""
    }
}
else
{
    Write-Host "    EasyLobby 10 install folder does not exist."
    Write-Host ""
}

Start-Sleep 5

# Delete three shortcuts from the Default User's Desktop
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for Employee Import shortcut on Default Desktop..."
Write-Host ""

if ((Test-Path -Path $EmployeeImportShortcut) -eq 'True')
{
    Write-Host ""
    Write-Host "Removing Employee Import shortcut.."
    Write-Host ""
    Write-Host "Command:  Remove-Item -Path $EmployeeImportShortcut -Force"
    Write-Host ""
    Remove-Item -Path $EmployeeImportShortcut -Force
    Write-Host ""
    Write-Host "    Employee Import shortcut removal complete."
    Write-Host ""
```

```
}
else
{
    Write-Host "    Employee Import shortcut does not exist."
    Write-Host ""
}

Start-Sleep 2

Write-Host "*******************************************************************"
Write-Host ""
Write-Host "Checking for Photo Export shortcut on Default Desktop..."
Write-Host ""

if ((Test-Path -Path $PhotoExportShortcut) -eq 'True')
{
    Write-Host ""
    Write-Host "Removing Photo Export shortcut.."
    Write-Host ""
    Write-Host "Command:  Remove-Item -Path $PhotoExportShortcut -Force"
    Write-Host ""
    Remove-Item -Path $PhotoExportShortcut -Force
    Write-Host ""
    Write-Host "    Photo Export shortcut removal complete."
    Write-Host ""
}
else
{
    Write-Host "    Photo Export shortcut does not exist."
    Write-Host ""
}

Start-Sleep 2

Write-Host "*******************************************************************"
Write-Host ""
Write-Host "Checking for Easy Lobby shortcut on Default Desktop..."
Write-Host ""

if ((Test-Path -Path $EasyLobbyShortcut) -eq 'True')
{
    Write-Host ""
    Write-Host "Removing Easy Lobby shortcut.."
    Write-Host ""
    Write-Host "Command:  Remove-Item -Path $EasyLobbyShortcut -Force"
    Write-Host ""
    Remove-Item -Path $EasyLobbyShortcut -Force
    Write-Host ""
    Write-Host "    Easy Lobby shortcut removal complete."
    Write-Host ""
}
else
{
    Write-Host "    Easy Lobby shortcut does not exist."
    Write-Host ""
}

Start-Sleep 5

# Check Power User access on temp folder
set-location $startLocation
Write-Host "*******************************************************************"
Write-Host ""
Write-Host "Checking ACL on $folder2..."
Write-Host ""

$r = (Get-Acl $folder2).Access | where {$_.IdentityReference -eq $User -and
$_.FileSystemRights -eq $FolderRights -and $_.AccessControlType -eq $AllowDeny}

If ($r -ne $null)
{
    Write-Host "    $User group has $FolderRights permissions for $folder2"
    Write-Host ""
    $title =  "Remove $User Permissions"
    $message = "Do you want to remove the $User from the $folder2 folder permissions?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Remove $User group"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$User group"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        $objAcl2 = Get-Acl $folder2
        $string = ('' + $objAcl2 + '.RemoveAccessRule(' + $objAr + ')')
        Write-Host ""
        Write-Host "Removing $User group from $folder2 folder access list..."
        Write-Host ""
        Write-Host 'Command: ' $string
        Write-Host ""
        $objAcl2.RemoveAccessRule($objAr)
        Write-Host ""
        Write-Host "Setting new access list on $folder2 folder..."
        Write-Host ""
        Write-Host "Command: Set-Acl $folder2 $objAcl2"
        Write-Host ""
        Set-Acl $folder2 $objAcl2
        Write-Host ""
        Write-Host "    New $folder2 folder access list set complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping ACL check on $folder2 folder."
        Write-Host ""
    }
}
Else
{
    Write-Host "    $User group does not have $FolderRights permissions for $folder2
folder."
    Write-Host ""
}

Start-Sleep 5

# Check Power User access on twain_32 folder
set-location $startLocation
Write-Host "******************************************************************"
Write-Host ""
Write-Host "Checking ACL on $folder3..."
Write-Host ""

$r = (Get-Acl $folder3).Access | where {$_.IdentityReference -eq $User -and
$_.FileSystemRights -eq $FolderRights -and $_.AccessControlType -eq $AllowDeny}

If ($r -ne $null)
{
    Write-Host "    $User group has $FolderRights permissions for $folder3"
    Write-Host ""
    $title =  "Remove $User Permissions"
    $message = "Do you want to remove the $User from the folder $folder3 permissions?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Remove $User group"
```

```
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$User group"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        $objAcl3 = Get-Acl $folder3
        $string = ('' + $objAcl3 + '.RemoveAccessRule(' + $objAr + ')')
        Write-Host ""
        Write-Host "Removing $User group from $folder3 folder access list..."
        Write-Host ""
        Write-Host 'Command: ' $string
        Write-Host ""
        $objAcl3.RemoveAccessRule($objAr)
        Write-Host ""
        Write-Host "Setting new access list on $folder3 folder..."
        Write-Host ""
        Write-Host "Command: Set-Acl $folder3 $objAcl3"
        Write-Host ""
        Set-Acl $folder3 $objAcl3
        Write-Host ""
        Write-Host "    New $folder3 folder access list set complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping ACL check on $folder3 folder."
        Write-Host ""
    }
}
Else
{
    Write-Host "    $User group does not have $FolderRights permissions for $folder3
folder."
    Write-Host ""
}

Start-Sleep 5

# Remove Security domain group from the Power Users group
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for Security group membership in local Power Users group..."
Write-Host ""

$r = (([ADSI]"WinNT://./$LocalGroup").Invoke("IsMember",
"WinNT://$Domain/$DomainGroup"))

$string = '[ADSI]"WinNT://.' + $LocalGroup + ',group").Remove("WinNT://' + $Domain +
'/' + $DomainGroup + '")'

If ($r -match 'True')
{
    Write-Host "    Security group is member of local Power Users group."
    Write-Host ""
    $title =  "Security Group Removal"
    $message = "Do you want to remove the Security group from the local Power Users
group?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Remove Security group"
```

```
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
Security group"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Removing Security group from local Power Users group..."
        Write-Host ""
        Write-Host 'Command: ' $string
        Write-Host ""
        ([ADSI]"WinNT://./$LocalGroup,group").Remove("WinNT://$Domain/$DomainGroup")
        Write-Host ""
        Write-Host "    Security group removal complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping Security group removal."
        Write-Host ""
    }
}
Else
{
    Write-Host "    Security group is not member of local Power Users group."
    Write-Host ""
}

Start-Sleep 5

Write-Host "*****************************************************************"
Write-Host ""
Write-Host "    EasyLobby SVM 10.0 removal complete."
Write-Host ""
Write-Host "*****************************************************************"
Write-Host ""

$err | Out-File $errFileLocation
Write-Host "*** Log file location = " $errFileLocation " ***"

Start-Sleep 5
```

EndoScan-V 4 SP1

This application was used for the screenshots in the **3. How to Package Script Installers** section. The application setup supported a /SILENT switch as does the uninstall executable. Because the **UninstallString** is correctly formatted with double quotes around the executable followed by the switch separated with a space, the **UninstallString** can be read from the application's Uninstall registry key using the `(Get-ItemProperty $Regkey).UninstallString` command passed directly into the `cmd` uninstall method without any modifications.

Description
EndoScan-V™, is endotoxin measuring software that is compatible with a variety of plate readers. It has been verified and validated to be consistent with FDA requirements and performs requisite calculations and batch reports for product release.

Package Notes/Comments – Copy to all **Notes/Comments** sections in SCCM creating package.
Single license package. Registration Code required to activate software. Choose **No** at the system restart popup message if one appears.

Source file location (share or weblink):
\\server\share\installs\EndoScan\ESV 4 SP1 (D)

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\EndoScan\ESV 4 SP1 (D)

Created a PowerShell program, *EndoScanV_Uninstall.ps1*, to remove EndoScan-V 4 SP1.

Import *EndoScanVSetup.exe* into SCCM Applications as a *Script Installer*.

Installation program:
"EndoScanVSetup.exe" /SILENT

Uninstall program:
Powershell.exe -executionpolicy Bypass -file "EndoScanV_Uninstall.ps1"

Detection Method:
Registry key exists.
Registry hive: HKLM
Registry key: SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EndoScan-V_is1

User Experience:
Behavior: Install for system
Logon: Whether or not a user is logged on
Visibility: Hidden
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
None

Notes
GUID: EndoScan-V_is1 (32-bit)

Registry key or file path (used to check if application is installed):
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EndoScan-V_is1

UninstallString (determines uninstall method to use):
"C:\Program Files (x86)\EndoScanV\unins000.exe" /SILENT

DisplayName: EndoScan-V 4 SP 1
DisplayVersion: 4.1
Publisher: Charles River

EndoScanV_Uninstall.ps1

```
  <#
     .SYNOPSIS
       SCCM Uninstall program for EndoScan-V

     .DESCRIPTION
       Remove the following component(s) from Window 7 x64 systems:
         - EndoScan-V

     .NOTES
       FileName: EndoScanV_Uninstall.ps1
       Author: Jerry Senff
       Updated: MM/DD/YYYY
       Comments: Powershell.exe -executionpolicy bypass -file "EndoScanV_Uninstall.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\EndoScanV_Uninstall.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Uninstall variables -- Do NOT modify
$Cmd = ${env:SystemRoot} + '\System32\cmd.exe'

# Uninstall variables -- Modify
$EndoScanV= 'EndoScan-V 4 SP 1'
$EndoScanVRegkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\EndoScan-V_is1'

Write-Host ""
Write-Host
"**************************************************************************"
Write-Host ""
Write-Host "$EndoScanV Uninstaller"
Write-Host ""
Write-Host "Purpose: Remove the following components:"
Write-Host "         - Uninstall $EndoScanV"
Write-Host ""

# Uninstall EndoScanV
set-location $startLocation
Write-Host
"**************************************************************************"
```

```
Write-Host ""
Write-Host "Checking for $EndoScanV installation..."
Write-Host ""

If ((Test-Path -Path $EndoScanVRegkey) -eq 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $EndoScanV is installed."
    Write-Host ""

    $CmdParams = '/c ' + (Get-ItemProperty $EndoScanVRegkey).UninstallString

    Write-Host "Uninstalling $EndoScanV..."
    Write-Host ""
    Write-Host "Command:  $Cmd $CmdParams"
    Write-Host ""
    Start-Process -FilePath $Cmd -ArgumentList $CmdParams -ErrorVariable +err -Verb
Open -Wait
    Write-Host ""
    Write-Host "    $EndoScanV uninstall complete."
    Write-Host ""
}
Else
{
    Write-Host "    $EndoScanV is not installed."
    Write-Host ""
}

Start-Sleep 5

# Indicate location of log file.
$err | Out-File $errFileLocation
Write-Host "*** Log file location = $errFileLocation ***"

Start-Sleep 5
```

MSOW - Manual

Description
MSO for the Web (MSOW) is a comprehensive, web-based credentialing and privileging system by Morrisey Associates. MSOW combines the speed and flexibility of the Internet with Morrisey's advanced technology to automate physician and allied health professional credentialing.

Package Notes/Comments – Copy to all **Notes/Comments** sections in SCCM creating package.
Enterprise license. **Office 2010 Professional must be uninstalled first!** Install Office 2003 Word 2003, then Office 2003 Access 2003 SP2. Program copies files to local hard drive folder, creates shortcut on All Users desktop, and creates the MSOW_PRD database 32-bit ODBC connection.

Source file location (share or weblink):
\\server\MSOW

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\MSOW

Create a PowerShell program, *MSOW_Manual_Install.ps1*, to install MSOW.

Create a second PowerShell program, *MSOW_Manual_Uninstall.ps1*, to remove MSOW.

Import *MSOW_Manual_Install.ps1* into SCCM Applications as a *Script Installer*.

Installation program:
Powershell.exe –executionpolicy Bypass –file "MSOW_Manual_Install.ps1"

Uninstall program:
Powershell.exe –executionpolicy Bypass –file "MSOW_Manual_Uninstall.ps1"

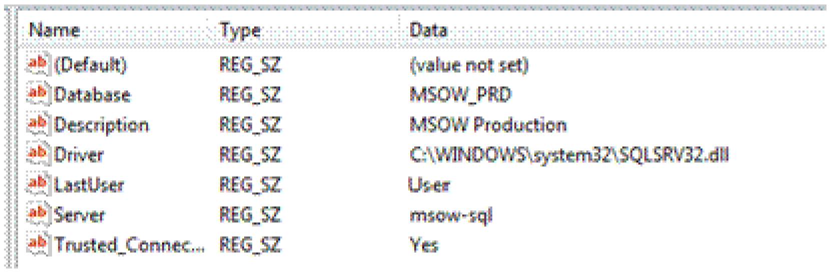
Detection Method:
File exists.
Path: %SystemDrive%\Morrisey\MSOW
Target: MSOWPrivs_PRD.mdb

User Experience:
Behavior: Install for user
Logon: Only when a user is logged on
Visibility: Normal
Enforce: No specific action

Dependencies (list redistributables, runtime, etc,. available for install with SCCM):
None

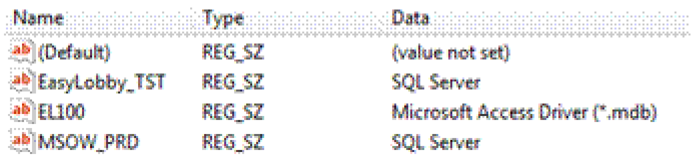
Setup Notes

MSOW_PRD Regkey:



Name	Type	Data
(Default)	REG_SZ	(value not set)
Database	REG_SZ	MSOW_PRD
Description	REG_SZ	MSOW Production
Driver	REG_SZ	C:\WINDOWS\system32\SQLSRV32.dll
LastUser	REG_SZ	User
Server	REG_SZ	msow-sql
Trusted_Connec...	REG_SZ	Yes

ODBC Data Sources Regkey:



Name	Type	Data
(Default)	REG_SZ	(value not set)
EasyLobby_TST	REG_SZ	SQL Server
EL100	REG_SZ	Microsoft Access Driver (*.mdb)
MSOW_PRD	REG_SZ	SQL Server

MSOW_Manual_Install.ps1

```
<#
    .SYNOPSIS
      SCCM install program for MSOW

    .DESCRIPTION
      Install the following component(s) on Window 7 x64 systems:
        - Verify Word 2003 is installed
        - Verify Access 2003 is installed
        - Create folder on the local hard drive root
        - Copy two files to the local drive
        - Copy shortcut to ALL USERS desktop
        - Create a 32-bit ODBC connection

    .NOTES
      FileName: MSOW_Manual_Install.ps1
      Author: Jerry Senff
      Updated: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file "MSOW_Manual_Install.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\MSOW_Manual_Install.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

$MSOW = 'MSOW'

# Verify Word 2003 is installed
$Word2003 = 'Microsoft Office Word 2003'
$Word2003Regkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{901B0409-6000-
11D3-8CFE-0150048383C9}'

# Verify Access 2003 is installed
$Access2003 = 'Microsoft Office Access 2003 SP2'
$Access2003Regkey =
'HKLM:SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\{90150409-6000-
11D3-8CFE-0150048383C9}'

# Create folder on the local hard drive root
```

```
$RootFolder = ${env:SystemDrive} + '\Morrisey'
$InstallFolder = ${env:SystemDrive} + '\Morrisey\MSOW'

# Copy two files to the local drive
$CopyFileName1 = '.\MSOW_PRD.mdb'
$CopyFileName2 = '.\MSOW_PRD.ICO'
$FileCopyDestination = (${env:SystemDrive} + '\Morrisey\MSOW\')

# Copy shortcut to ALL USERS desktop
$SourcePath1 = ('.\MSOW.lnk')
$DestinationPath1 = (${env:SystemDrive} + '\Users\Public\Desktop\MSOW.lnk')

# Created a 32-bit ODBC connection
$ConnectionName = 'MSOW_PRD'
$ConnectionDescription = 'MSOW PRODUCTION'
$SqlServer = 'MSOW-SQL'
$SqlDatabase = 'MSOW_PRD'
$SqlDriver = '"C:\WINDOWS\System32\sqlsrv32.dll"'
$HKLMPath1 = “HKLM:SOFTWARE\Wow6432Node\ODBC\ODBC.INI\” + $ConnectionName
$HKLMPath2 = “HKLM:SOFTWARE\Wow6432Node\ODBC\ODBC.INI\ODBC Data Sources”

Write-Host ""
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "$MSOW Setup Program"
Write-Host ""
Write-Host "Purpose: Performs the following tasks:"
Write-Host "         - Check for $Word2003 installation"
Write-Host "         - Check for $Access2003 installation"
Write-Host "         - Copy two files to the local drive"
Write-Host "         - Copy shortcut to ALL USERS desktop"
Write-Host "         - Create a $ConnectionName 32-bit ODBC connection"
Write-Host ""

# Verify Word 2003 is installed
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $Word2003 installation..."
Write-Host ""

If ((Test-Path -Path $Word2003Regkey) -ne 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $Word2003 is not installed!"
    Write-Host ""
    Write-Host "Please install $Word2003 before continuing..."
    Write-Host ""
    Write-Host "    Press any key to quit..."
    Pause
    Break
}
Else
{
    Write-Host "    $Word2003 already installed."
    Write-Host ""

    # Get Word 2003 InstallLocation from registry
    $Word2003Location = (Get-ItemProperty -Path $Word2003Regkey -Name
InstallLocation).InstallLocation

    Write-Host "$Word2003 located at:  $Word2003Location"
    Write-Host ""
}

Start-Sleep 5

# Verify Access 2003 is installed
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $Access2003 installation..."
```

```
Write-Host ""

If ((Test-Path -Path $Access2003Regkey) -ne 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $Access2003 is not installed!"
    Write-Host ""
    Write-Host "Please install $Access2003 before continuing..."
    Write-Host ""
    Write-Host "    Press any key to quit..."
    Pause
    Break
}
Else
{
    Write-Host "    $Access2003 already installed."
    Write-Host ""

    # Get Word 2003 InstallLocation from registry
    $Access2003Location = (Get-ItemProperty -Path $Access2003Regkey -Name
InstallLocation).InstallLocation

    Write-Host "$Access2003 located at:  $Access2003Location"
    Write-Host ""
}

Start-Sleep 5

# Create folder on the local hard drive root
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $RootFolder on local drive..."
Write-Host ""

If ((Test-Path -Path $RootFolder) -ne 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $RootFolder does not exist."
    Write-Host ""
    Write-Host "Creating $RootFolder on local drive..."
    Write-Host ""
    Write-Host "Command: md $RootFolder"
    md $RootFolder
    Write-Host ""
    Write-Host "    $RootFolder creation complete."
    Write-Host ""
}
Else
{
    Write-Host "    $RootFolder already exists."
    Write-Host ""
}

Start-Sleep 2

# Create folder on the local hard drive root
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $InstallFolder on local drive..."
Write-Host ""

If ((Test-Path -Path $InstallFolder) -ne 'True')  # Check uninstall registry key or
install path
{
    Write-Host "    $InstallFolder does not exist."
    Write-Host ""
    Write-Host "Creating $InstallFolder on local drive..."
    Write-Host ""
    Write-Host "Command: md $InstallFolder"
    md $InstallFolder
    Write-Host ""
    Write-Host "    $InstallFolder creation complete."
    Write-Host ""
}
Else
{
    Write-Host "    $InstallFolder already exists."
    Write-Host ""
}

Start-Sleep 2

# Copy two files to the local drive
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Copying two files to local drive..."
Write-Host ""
Write-Host "Command: Copy-Item $CopyFileName1 $FileCopyDestination -Force"
Copy-Item $CopyFileName1 $FileCopyDestination -Force
Write-Host ""
Write-Host "Command: Copy-Item $CopyFileName2 $FileCopyDestination -Force"
Copy-Item $CopyFileName2 $FileCopyDestination -Force
Write-Host ""
Write-Host "    File copy complete."
Write-Host ""

Start-Sleep 2

# Copy shortcut to ALL USERS desktop
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Copying shortcut to ALL USERS desktop..."
Write-Host ""
Write-Host "Command: Copy-Item $SourcePath1 $DestinationPath1 -Force"
Copy-Item $SourcePath1 $DestinationPath1 -Force
Write-Host ""
Write-Host "    Shortcut copy complete."
Write-Host ""

Start-Sleep 2

# Created a 32-bit ODBC connection
set-location $startLocation
Write-Host "*****************************************************************"
Write-Host ""
Write-Host "Checking for $ConnectionName 32-bit ODBC connection..."
Write-Host ""

If ((Test-Path -Path $HKLMPath1) -ne 'True')
{
    Write-Host "    $ConnectionName 32-bit ODBC connection does not exist."
    Write-Host ""
    Write-Host "Creating $ConnectionName 32-bit ODBC connection..."
    Write-Host ""

    md $HKLMPath1 -ErrorAction silentlycontinue

    set-itemproperty -path $HKLMPath1 -name   Driver -value $SqlDriver
    set-itemproperty -path $HKLMPath1 -name   Description -value 
$ConnectionDescription
    set-itemproperty -path $HKLMPath1 -name   Server -value $SqlServer
    set-itemproperty -path $HKLMPath1 -name LastUser -value “”
    set-itemproperty -path $HKLMPath1 -name   Trusted_Connection -value “Yes”
    set-itemproperty -path $HKLMPath1 -name   Database -value $SqlDatabase

    ## This is required to allow the ODBC connection to show up in the ODBC 
Administrator application.
    md $HKLMPath2 -ErrorAction silentlycontinue

    set-itemproperty -path $HKLMPath2 -name “$ConnectionName” -value 'SQL Server'
    Write-Host ""
    Write-Host "    Creating $ConnectionName 32-bit ODBC connection complete."
    Write-Host ""
    Write-Host "To confirm 32-bit ODBC connection creation, run odbcad32.exe from the
C:\Windows\SysWOW64 folder"
    Write-Host ""
}
Else
{
    Write-Host "    $ConnectionName 32-bit ODBC connection already exists."
    Write-Host ""
}

Start-Sleep 5

# Indicate location of error log file
$err | Out-File $errFileLocation
Write-Host "*** Log file location = $errFileLocation ***"
Write-Host ""

Start-Sleep 5
```

MSOW_Manual_Uninstall.ps1

```
<#
    .SYNOPSIS
      SCCM Uninstall program for MSOW

    .DESCRIPTION
      Remove the following component(s) from Window 7 x64 systems:
        - Delete install folder on the local hard drive root
        - Delete shortcut on ALL USERS desktop
        - Delete the 32-bit ODBC connection

    .NOTES
      FileName: MSOW_Manual_Uninstall.ps1
      Author: Jerry Senff
      Updated: MM/DD/YYYY
      Comments: Powershell.exe -executionpolicy bypass -file
"MSOW_Manual_Uninstall.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\MSOW_Manual_Uninstall.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

# Uninstall variables
$MSOW = 'MSOW'
$InstallFolder = ${env:SystemDrive} + '\Morrisey\MSOW'
$DestinationPath1 = (${env:SystemDrive} + '\Users\Public\Desktop\MSOW.lnk')

# Delete the 32-bit ODBC connection
$ConnectionName = 'MSOW_PRD'
$ConnectionDescription = 'MSOW PRODUCTION'
$SqlServer = 'MSOW-SQL'
$SqlDatabase = 'MSOW_PRD'
$SqlDriver = '"C:\WINDOWS\System32\sqlsrv32.dll"'
$HKLMPath1 = “HKLM:SOFTWARE\Wow6432Node\ODBC\ODBC.INI\” + $ConnectionName
$HKLMPath2 = “HKLM:SOFTWARE\Wow6432Node\ODBC\ODBC.INI\ODBC Data Sources”

Write-Host ""
Write-Host "*******************************************************************"
Write-Host ""
Write-Host "$MSOW Uninstaller"
Write-Host ""
Write-Host "Purpose: Performs the following tasks:"
```

```
Write-Host "          - Delete $MSOW install folder off of local hard drive root"
Write-Host "          - Delete $MSOW shortcut on ALL USERS desktop"
Write-Host "          - Delete $ConnectionName 32-bit ODBC connection"
Write-Host ""

# Delete install folder on the local hard drive root
set-location $startLocation
Write-Host "********************************************************************"
Write-Host ""
Write-Host "Checking for $MSOW install folder..."
Write-Host ""

if ((Test-Path -Path $InstallFolder) -eq 'True')
{
    Write-Host "    $MSOW install folder exists."
    Write-Host ""
    $title =  "$MSOW Install Folder"
    $message = "Do you want to delete the $MSOW install folder?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Delete $MSOW install folder"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$MSOW install folder"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Deleting $MSOW install folder..."
        Write-Host ""
        Write-Host "Command:  Remove-Item -Path $InstallFolder -Recurse -Force"
        Write-Host ""
        Remove-Item -Path $InstallFolder -Recurse -Force
        Write-Host ""
        Write-Host "    $MSOW install folder deletion complete."
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping $MSOW install folder deletion."
        Write-Host ""
    }
}
else
{
    Write-Host "    $MSOW install folder does not exist."
    Write-Host ""
}

Start-Sleep 5

# Delete shortcut on ALL USERS desktop
set-location $startLocation
Write-Host "********************************************************************"
Write-Host ""
Write-Host "Checking for $MSOW shortcut on ALL USERS desktop..."
Write-Host ""

if ((Test-Path -Path $DestinationPath1) -eq 'True')
{
    Write-Host ""
    Write-Host "Removing $MSOW shortcut.."
    Write-Host ""
    Write-Host "Command:  Remove-Item -Path $DestinationPath1 -Force"
    Write-Host ""
    Remove-Item -Path $DestinationPath1 -Force
    Write-Host ""
    Write-Host "    $MSOW shortcut removal complete."
    Write-Host ""
}
else
{
    Write-Host "    $MSOW shortcut does not exist."
    Write-Host ""
}

Start-Sleep 5

# Delete the 32-bit ODBC connection
set-location $startLocation
Write-Host "********************************************************************"
Write-Host ""
Write-Host "Checking for $ConnectionName 32-bit ODBC connection..."
Write-Host ""

If ((Test-Path -Path $HKLMPath1) -eq 'True')
{
    Write-Host "    $ConnectionName 32-bit ODBC connection exists."
    Write-Host ""
    $title =  "Remove $ConnectionName ODBC Connection"
    $message = "Do you want to remove the $ConnectionName 32-bit ODBC connection?"
    $yes = New-Object System.Management.Automation.Host.ChoiceDescription "&Yes",
"Remove $ConnectionName connection"
    $no = New-Object System.Management.Automation.Host.ChoiceDescription "&No", "Leave
$ConnectionName connection"
    $options = [System.Management.Automation.Host.ChoiceDescription[]]($yes, $no)
    $result = $host.ui.PromptForChoice($title, $message, $options, 0)
    Write-Host ""

    Switch ($result)
    {
        0 {"    You selected Yes."}

        1 {"    You selected No."}
    }

    If ($result -eq 0)
    {
        Write-Host ""
        Write-Host "Removing $ConnectionName 32-bit ODBC connection..."
        Write-Host ""

        Remove-Item -Path $HKLMPath1 -Force

        ## This is required to remove the ODBC connection from showing up in the ODBC
Administrator application.
        Remove-ItemProperty -path $HKLMPath2 -name $SqlDatabase -Force

        Write-Host ""
        Write-Host "    Removal of $ConnectionName 32-bit ODBC connection complete."
        Write-Host ""
        Write-Host "To confirm 32-bit ODBC connection removal, run odbcad32.exe from
the C:\Windows\SysWOW64 folder"
        Write-Host ""
    }
    Else
    {
        Write-Host ""
        Write-Host "    Skipping $ConnectionName 32-bit ODBC connection removal."
        Write-Host ""
    }
}
Else
{
    Write-Host "    $ConnectionName 32-bit ODBC connection does not exist."
    Write-Host ""
}

Start-Sleep 5

# Indicate location of error log file
$err | Out-File $errFileLocation
Write-Host "*** Log file location = $errFileLocation ***"
Write-Host ""

Start-Sleep 5
```

Surface Pro 3 - November 2014 Driver Package

Description
November 2014 driver release for Surface Pro 3.

Package Notes/Comments – Copy to all **Notes/Comments** sections in SCCM creating package.
Free software license.

Source file location (share or weblink):
Download from Microsoft.com

Copy source files to Software Vault share location:
\\server\share\Sources\Software Vault\Microsoft Surface\OOBNovember18th2014SurfacePro3

Create a PowerShell program, *SurfacePro3_DriverPkg_Install.ps1*, to install the Surface Pro 3 driver package.

Import *SurfacePro3_DriverPkg_Install.ps1* into SCCM Applications as a *Script Installer*.

Installation program:
Powershell.exe –executionpolicy Bypass –file "SurfacePro3_DriverPkg_Install.ps1"

Uninstall program:
N/A

Detection Method:
PowerShell Script

```
$driverlist = @{"Surface Pro System Aggregator Firmware" = "3.9.350.0";
                "Surface Pro Embedded Controller Firmware" = "38.7.50.0";
                "Microsoft LifeCam Front" = "5.20.1034.0";
                "Surface Ethernet Adapter" = "8.14.0704.2014";
                "Surface Pro UEFI" = "3.11.350.0";
                "Surface Pro Touch Controller Firmware" = "426.27.66.0";
                "Intel(R) Serial IO I2C Host Controller - 9C61" = "1.1.165.1";
                "Intel(R) Serial IO I2C Host Controller - 9C62" = "1.1.165.1";
                "Intel(R) Serial IO GPIO Host Controller" = "1.1.165.1";
                "Intel(R) Display Audio"= "6.16.0.3135";
                "Intel(R) Management Engine Interface*" = "9.5.24.1790";
                "Intel(R) HD Graphics Family" = "10.18.10.3496";
                "Intel(R) 8 Series SATA AHCI Controller - 9C03" = "9.4.0.1023";
                 "Intel(R) 8 Series LPC Controller (Premium SKU) - 9C43" =
                    "9.4.0.1023";
                "Intel(R) 8 Series PCI Express Root Port #3 - 9C14" = "9.4.0.1023";
                "Intel(R) 8 Series SMBus Controller - 9C22" = "9.4.0.1023";
                "Marvell AVASTAR Wireless-AC Network Controller" = "15.68.3066.135";
                "Marvell AVASTAR Bluetooth Radio Adapter" = "15.68.3066.135";
                "Realtek High Definition Audio" = "6.0.1.7198";
                "Realtek USB 3.0 Card Reader" = "6.2.9200.30164";
                "Surface Accessory Device" = "2.0.1012.0";
                "Surface Cover Audio" = "2.0.722.0";
                "Surface Cover Click" = "2.0.375.0";
                "Surface Type Cover" = "2.0.364.0";
                "Surface Touch Cover" = "2.0.722.0";
                "Surface Type Cover Fw Update" = "2.0.722.0";
                "Surface Touch Cover Fw Update" = "2.0.722.0";
                "Surface Type Cover 2 Fw Update" = "2.0.722.0";
                "Surface Touch Cover2 FW Update" = "2.0.722.0";
                "Surface Type Cover 3 Firmware Update" = "2.0.1021.0";
                "Surface Display Calibration" = "2.0.1002.0";
                "Surface Intergration" = "2.0.1168.0";
                "Surface Home Button" = "2.0.1174.0";
                "Surface Cover Telemetry" = "2.0.722.0";
                "Surface Pen Driver" = "2.5.14.0";
                "Surface Pen" = "1.0.13.0" }

$LogPath = ${env:SystemRoot} + '\Logs\'
$File = 'SurfaceProDriversNeeded.txt'
$Flag = 0

foreach ($key in $driverlist.GetEnumerator() | Sort-Object Name)
{
    If (Get-WmiObject win32_pnpsigneddriver | where {$_.DeviceName -eq $key.Name})
    {
        If (!(Get-WmiObject win32_pnpsigneddriver | where {$_.DeviceName -eq $key.Name
             -and $_.DriverVersion -eq $key.Value}))
        {
            $m = $key.Name  + ' is not current:  ' + $key.Value
            Add-Content -Path $LogPath$File $m
            $Flag = $Flag + 1
        }
    }
}

If ($Flag -gt 0)
{
    $false
}
Else
{
    if (Test-Path -path $LogPath$File)
    {
        rm $LogPath$File -force
    }
    $true
}
```

NOTE: The driver search key name for the `"Intel(R) Management Engine Interface*"` driver requires a wildcard character at the end of the name due to the driver name in the `.INF` file containing a blank space at the end. Without the wildcard character to account for the blank space in the name, the driver check will fail every time. Every driver `.INF` file should be cracked open to compare against the driver key name string for accuracy.

User Experience:
Behavior: Install for system
Logon: Whether or not a user is logged on
Visibility: Hidden
Enforce: No specific action

Requirements:
Category: Device
Condition: Operating system
One of: All Windows 8.1

Dependencies:
None

SurfacePro3_DriverPkg_Install.ps1

```
<#
    .SYNOPSIS
      SCCM Install program for Surface Pro 3 Driver Package

    .DESCRIPTION
      Install the following component(s) on Surface Pro 3 Windows 8.1 systems:
        - Surface Pro 3 Driver Package

    .NOTES
      FileName: SurfacePro3_DriverPkg_Install.ps1
      Author: Jerry Senff
      Updated: MM/DD/YYYY
      Comments: Powershell.exe –executionpolicy bypass –file
"SurfacePro3_DriverPkg_Install.ps1"
#>

# Error file
$startLocation = Get-Location
$err=@()
$errorpath = '\ErrorLogs\SurfacePro3_DriverPkg_Install.txt'
$errFileLocation =  (${env:SystemDrive} + $errorpath)

$Scriptpath = Split-Path -parent $startLocation
$files = get-childitem -path $Scriptpath -recurse -filter *.inf

foreach ($file in $files)
{
    $PnpUtilCmd = 'pnputil.exe'

    $PnpUtilParams = '-i -a "' + $file.FullName + '"'

    Start-Process -FilePath $PnpUtilCmd -ArgumentList $PnpUtilParams -ErrorVariable
+err -Verb Open -Wait
}

# Indicate location of error log file
$err | Out-File $errFileLocation
Write-Host "*** Log file location = $errFileLocation ***"
Write-Host ""

Start-Sleep 5
```