

Central Deployment of OpenLab CDS Client

DISCLAIMER: The procedures described in this document require knowledge and experience with deploying products using Microsoft SCCM in a production environment.

Please contact Microsoft directly for any questions related to SCCM. Agilent does not provide support for Microsoft SCCM.

Overview

Enterprises with a large number of systems manage software installations using professional software configuration management (SCM) tools such as Microsoft SCCM. SCM tools make it easier to know what software is installed on which machines and eliminates chances for errors during installation or during applying updates and patches.

Software installers with all the required parameters can be defined as “Applications” in SCM tools. Such predefined applications can then easily be pulled or pushed onto target machines. SCCM provide many benefits such as:

- Make installations accessible to end users, so they are able to search for and pull down any product onto their machine themselves
- Reduce delays and expenses associated with deploying new systems.
- Keeping track of what is installed on which machines.
- Notify users when a patch is available and needs to be applied on their system.
- Provide a time window when users should apply the patch.
- Allow users to schedule the patching to happen during off hours.
- Allow IT administrators to push an update.

This document provides the information needed to create applications for OpenLab CDS Clients (with OpenLab Server backend) in Microsoft SCCM. This setup can be used to install CDS Client on a new system as well as to upgrade an older version of CDS Client.

Setting up OpenLab CDS Client

In order to setup “OpenLab CDS Client” in SCCM an application needs to be defined. Figure below shows an example setup for CDS Client version 2.5.0.927. Fill in the “General Information” tab as applicable to your environment.

OpenLab CDS Client Application for server 66 Properties ✕

Security

General Information Application Catalog References Distribution Settings Deployment Types Content Locations Supersedence

Name:

Administrator comments:

Publisher: Software version:

Optional reference:

Administrative categories:

Date published:

Allow this application to be installed from the Install Application task sequence action without being deployed

Specify the administrative users who are responsible for this application.

Owners:

Support contacts:

Created date:	12/25/2019 5:17 AM	Revision:	7
Created by:	IOWA\administrator	Status:	Active
Modified date:	12/25/2019 7:59 AM	Superseded:	No
Modified by:	IOWA\administrator		

While defining an application a “Deployment Type” also needs to be defined for “OpenLab CDS Client”. Deployment Types provides information about the installation method and source file location for an application.

Defining Deployment Type content

The deployment type for “OpenLab CDS Client” needs to be manually specified. First the OpenLab CDS installation media needs to be hosted on a shared network location that the client systems can access. This location then needs to be provided as the content location for the deployment type. Additionally the following settings should be used to specify the installation program, uninstallation program and start location.

General Information

Content

Detection Method

User Experience

Requirements

Dependencies

Summary

Progress

Completion

Specify information about the content to be delivered to target devices

Specify the location of the deployment type's content and other settings that control how content is delivered to target devices. All the contents in the path specified will be delivered.

Content location:

Persist content in the client cache

Specify the command used to install this content.

Installation program:

Installation start in:

Configuration Manager can remove installations of this content if an uninstall program is specified below.

Uninstall program:

Uninstall start in:

Run installation and uninstall program as 32-bit process on 64-bit clients.

Install program:

```
CDSInstaller.exe -s Client=True OlssHostName=<<OpenLabServer>> LicenseAccepted=True -norestart
```

Installation start in:

```
\Setup
```

Uninstall program:

```
CDSInstaller.exe -s -uninstall -norestart
```

```
Installation start in:  
\\Setup
```

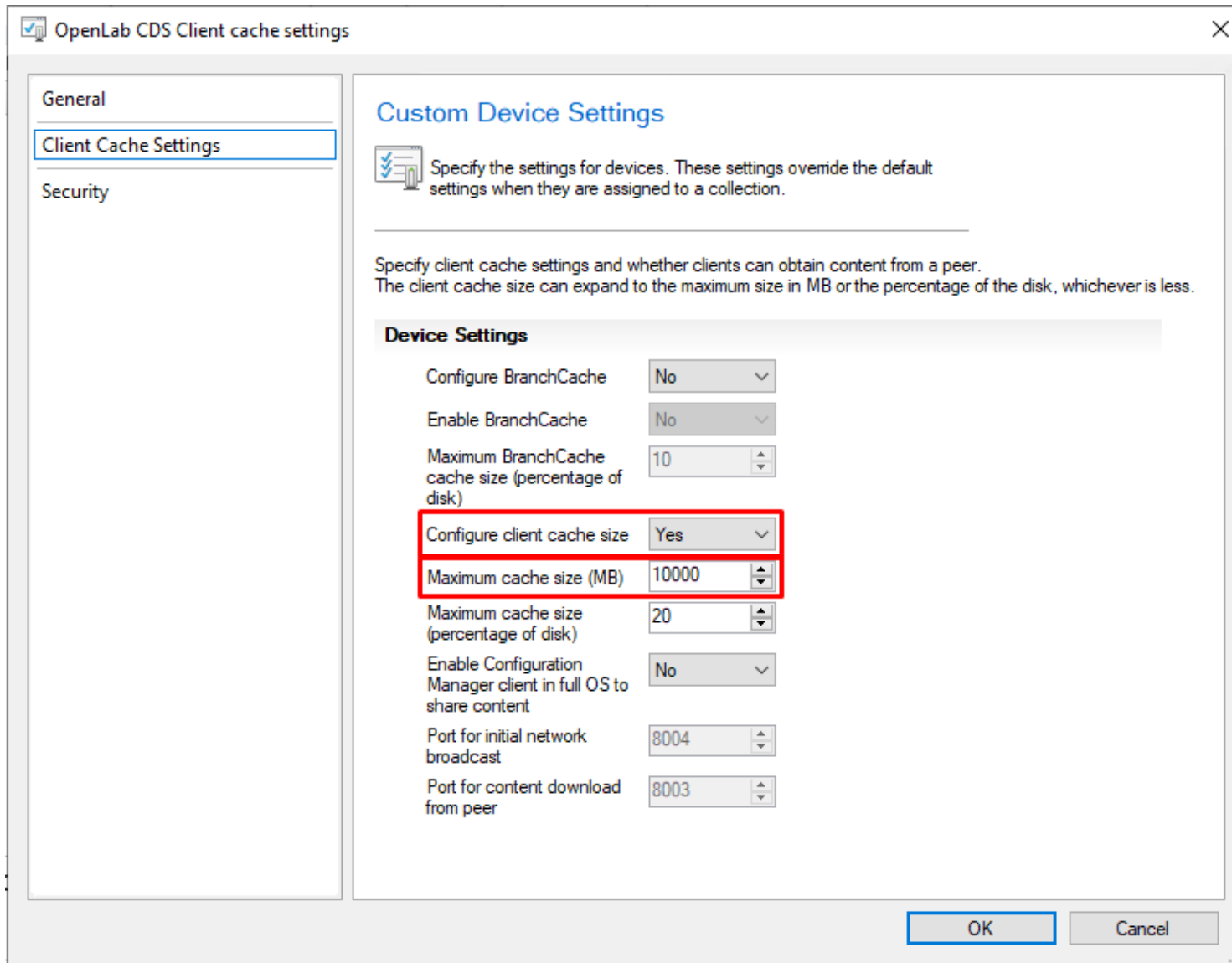
Replace “<<OpenLabServer>>” with the name of your OpenLab Server that the client system can use to connect to it.

Client cache settings

SCCM Client cache is a temporary location (%windir%\ccmcache) for downloading software, applications and updates. By default, cache size is 5 GB. However, OpenLab CDS Client installation may require up to 10 GB of cache size. So, the cache size needs to be increased.

This setting can be updated in **Client Settings** on SCCM server and pushed to client machines:

- In the Configuration Manager Console navigate to **Administration > Overview > Client Settings**.
- On the **Home** tab, in the **Create** group, select **Create Custom Client Device Settings** and the **Create Custom Client Device Settings** –popup will show.
- On the **General** page, fill in with **Name**<aName> and select **Client Cache Settings**.
- On the **Client Cache Settings** page, set *Configure client cache size* to "Yes" and set *Maximum cache size (MB)* to "10000" and click **OK**.
- Select the new policy <aName> and on the **Home** tab, in the **Client Settings** group, select **Deploy**.
- Select <aDeviceCollection> and click **OK**.



Defining the Detection Method

SCCM requires detection methods for applications. Detection methods are used to determine whether the application is already installed or not. If an application is deployed to a system where it is already present, then a properly configured detection method will find that application and the application reinstall will be avoided. If the application is not present it will be installed and the configured detection method will be used to verify the installation once it is complete. An install that can be verified by the detection method will be viewed as successful while an install that cannot be

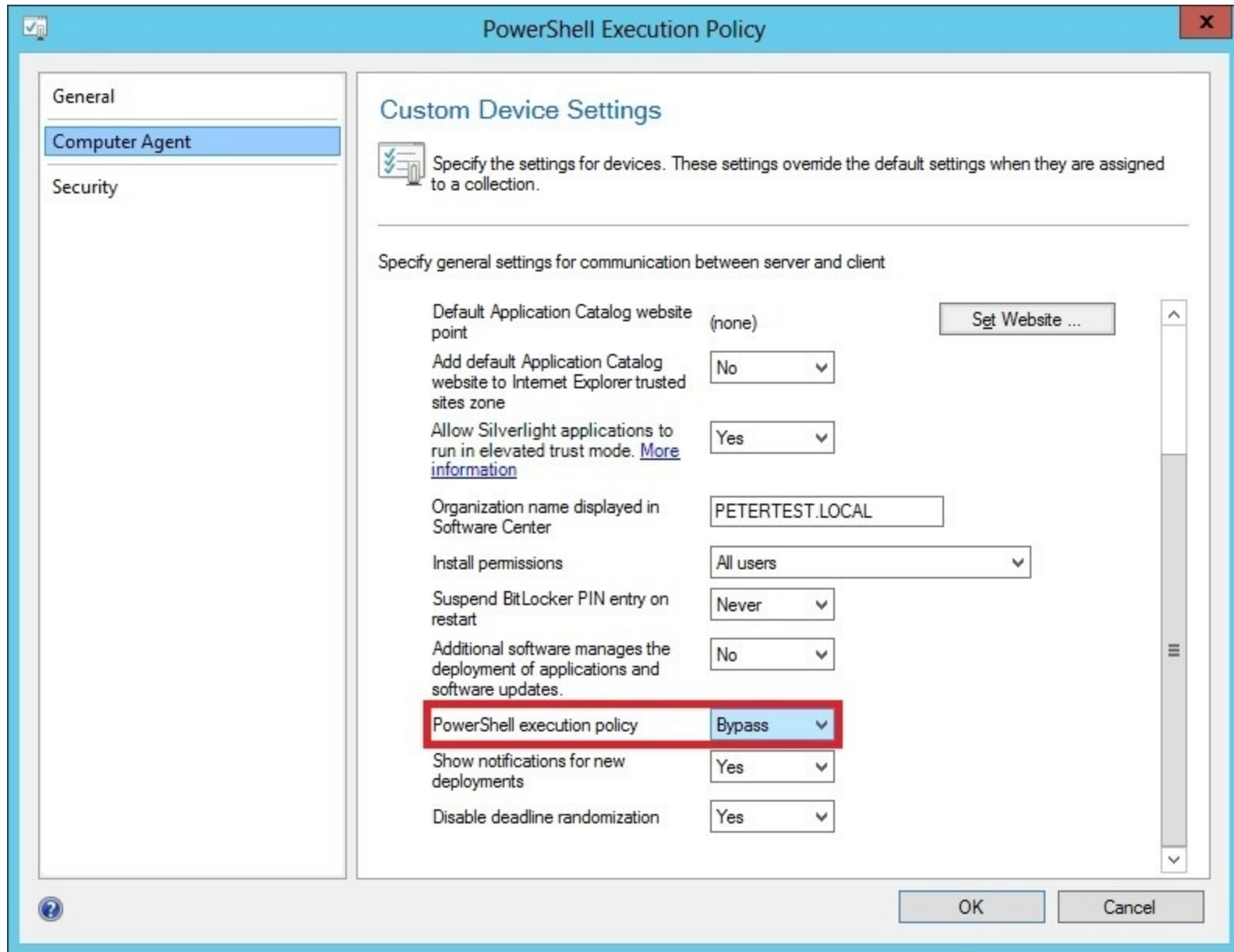
verified with the configured detection method will be seen as a failure. The SCCM client can also detect whether an application is installed using these methods, even if the user did not use SCCM to install the application.

For detecting "OpenLab CDS Client" a custom script should be used.

Setting PowerShell Execution Policy

PowerShell execution policies on target client systems usually will not allow script execution. SCCM server should be set up so that it instructs configuration manager clients to bypass the execution policy.

1. In the Configuration Manager Console navigate to **Administration > Overview > Client Settings**.
2. On the **Home** tab, in the **Create** group, select **Create Custom Client Device Settings**.
3. On the **General** page, fill in with **Name**: *OpenLabCDS* and select **Computer Agent**.
4. On the **Computer Agent** page, select the option **ByPass** for **PowerShell execution policy** and click Ok.
5. Select the new policy: *OpenLabCDS* and on the **Home** tab, in the **Client Settings** group, select **Deploy**.
6. Select your device collection and click Ok.



PowerShell Example

Script Parameters:

1. `$productUpgradeCode = {5447F66A-1B10-4DF5-AB1B-D880A1F298FA}`

2. \$productVersion = 2.5
3. \$serverAddress = net.tcp://<<OpenLabServer>>:6577/Agilent/OpenLAB/

Replace “<<OpenLabServer>>” with the name or IP address of your OpenLab Server that the client system can use to connect to it.

The following example script can be used after updating the `serverAddress` variable to the correct value.

Note: PowerShell execution policy must be set in SCCM as described earlier also.

```
#####
# OpenLab Installers: centralized install & update
#
# Version: 2.5
# Build: <OLINST_BUILD_NUMBER_INSERT_HERE>
#
# (c) Agilent Technologies 2020
#
#####

<#
.DESCRIPTION
    Detection method to check already installed CDS (Client, AIC, Workstation)

.PARAMETER $productUpgradeCode
    Upgrade code of CDS bundle to be installed
.PARAMETER $productVersion
    Current version of CDS bundle to be installed
.PARAMETER $serverAddress
    Current address of Agilent OpenLab CDS Server to connect

.OUTPUTS
    "Installed" will be written to the output, if CDS is already installed and configured with provided server address,
    otherwise an empty string.
#>

#####
$productUpgradeCode='{5447F66A-1B10-4DF5-AB1B-D880A1F298FA}' # OpenLab CDS Upgrade code
$productVersion=[version]'2.5' # OpenLab CDS Client Version
$serverAddress = 'net.tcp://<<OpenLabServer>>:6577/Agilent/OpenLAB/' # OpenLab Server endpoint
#####

$keys="HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*",
      "HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"

```



```

$product = Get-ItemProperty $keys |
    Where-Object BundleUpgradeCode -eq $productUpgradeCode |
    Select-Object DisplayName, BundleUpgradeCode, BundleVersion

if (-not($product)) {
    Exit # Not installed
}

if ([version]$product.BundleVersion -lt $productVersion) {
    Exit # Not installed
}

$olssConfig = Resolve-Path C:\ProgramData\Agilent\Agilent.OpenLab.Configuration.xml -ErrorAction SilentlyContinue

if (-not($olssConfig)) {
    Exit # Not installed
} else {
    $xml = New-Object System.Xml.XmlDocument
    $xml.PreserveWhitespace = $true
    $xml.Load($olssConfig)
    $serverId = $xml.config.ServersConfiguration.ConnectionId
    $currentServerAddress = $xml.DocumentElement.SelectSingleNode("descendant::ServerAddress[Id='" + $serverId + "']")

    if ($currentServerAddress.Address -eq $serverAddress) {
        Write-Host "Installed"
        Exit
    }
    Exit
}

```

VBScript Example

The following example script can be used after updating the `ServerAddress` variable to the correct value.

```

'#####
'# OpenLab Installers: centralized install & update
'#
'# Version: 2.5
'# Build: <OLINST_BUILD_NUMBER_INSERT_HERE>
'#

```

```

'# (c) Agilent Technologies 2020
'#
#####
' "Installed" will be written to the output, if CDS is already installed
' and configured with the provided server address, otherwise an empty string.

#####
ProductUpgradeCode = "{5447F66A-1B10-4DF5-AB1B-D880A1F298FA}"           ' OpenLab CDS Upgrade code
ProductVersion = 2.5                                                    ' OpenLab CDS Client Version
ServerAddress = "net.tcp://<<OpenLabServer>>:6577/Agilent/OpenLAB/"      ' OpenLab Server endpoint
#####

Main()

'WScript.Echo "Installed"
WScript.Stdout.Write "Installed"
WScript.Quit(0)

Function Main()
    OlssConfig = "C:\ProgramData\Agilent\Agilent.OpenLab.Configuration.xml" 'Olss config file
    RegistryHive = "HKEY_LOCAL_MACHINE\"
    RegLocation1 = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\"
    RegLocation2 = "SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\"

    On Error Resume Next

    If GetBundleVersion(RegistryHive, RegLocation1, RegLocation2, ProductUpgradeCode) <> ProductVersion Then
        'WScript.Echo "Not installed"
        WScript.Quit(0)
    End If

    If FileExists(OlssConfig) = false Then
        'WScript.Echo "Not installed"
        WScript.Quit(0)
    Else
        If GetServerAddress(OlssConfig) <> ServerAddress Then
            'WScript.Echo "Not installed"
            WScript.Quit(0)
        End If
    End If
End Function

```

```

    End If
End If
End Function

Function GetBundleVersion(Hive, location1, location2, upgradeCode)
    On Error Resume Next
    BundleVersion = TryGetVersionInBranch(Hive, location1, upgradeCode)
    If BundleVersion <> 0 Then
        GetBundleVersion = BundleVersion
        Exit Function
    End If
    GetBundleVersion = TryGetVersionInBranch(Hive, location2, upgradeCode)
End Function

Function TryGetVersionInBranch(Hive, location, upgradeCode)
    On Error Resume Next
    Dim BundleVersion
    Dim WSHShell
    Set WSHShell = CreateObject("WScript.Shell")

    keys = GetSubKeys(location)
    For Each RegistryKey In keys
        'BundleUpgradeCode - is a Multi-string value and WSHShell.RegRead returns it as an array
        bundleUpgradeCodesValues = WSHShell.RegRead(Hive + location + RegistryKey + "\BundleUpgradeCode")
        If Err.Number = 0 Then
            For i = 0 To UBound(bundleUpgradeCodesValues)
                If LCase(bundleUpgradeCodesValues(i)) = LCase(upgradeCode) Then
                    BundleVersion = WSHShell.RegRead(Hive + location + RegistryKey + "\BundleVersion")
                    If Err.Number = 0 Then
                        TryGetVersionInBranch = CSng(Left(BundleVersion, 3))
                        Set WSHShell = Nothing
                        Exit Function
                    End If
                Else
                    Err.Clear
                End If
            End If
        End If
    Next
    Else
        Err.Clear
    End If
Next

```

```

Set WSHShell = Nothing
TryGetVersionInBranch = 0
End Function

Function GetSubKeys(location)
Const HKEY_LOCAL_MACHINE = &H80000002
strComputer = "."

Set oReg=GetObject("winmgmts:{impersonationLevel=impersonate}!\" & _
strComputer & "\root\default:StdRegProv")

oReg.EnumKey HKEY_LOCAL_MACHINE, location, arrSubKeys
Set oReg = Nothing
GetSubKeys = arrSubKeys
End Function

Function FileExists(FilePath)
Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(FilePath) Then
FileExists=CBool(1)
Else
FileExists=CBool(0)
End If
Set fso = Nothing
End Function

Function GetServerAddress(conigName)
On Error Resume Next
Set objXMLDoc = CreateObject("Microsoft.XMLDOM")
objXMLDoc.async = False
objXMLDoc.load(conigName)

Set ConnectionId = objXMLDoc.documentElement.selectSingleNode("ServersConfiguration/ConnectionId")
Set AvailableServers = objXMLDoc.documentElement.selectNodes("ServersConfiguration/AvailableServers/ServerAddress")

For Each ServerAddress In AvailableServers
Set CurrentId = ServerAddress.selectSingleNode("Id")
If CurrentId.text = ConnectionId.text Then
GetServerAddress = ServerAddress.selectSingleNode("Address").text
Set objXMLDoc = Nothing
Exit Function

```

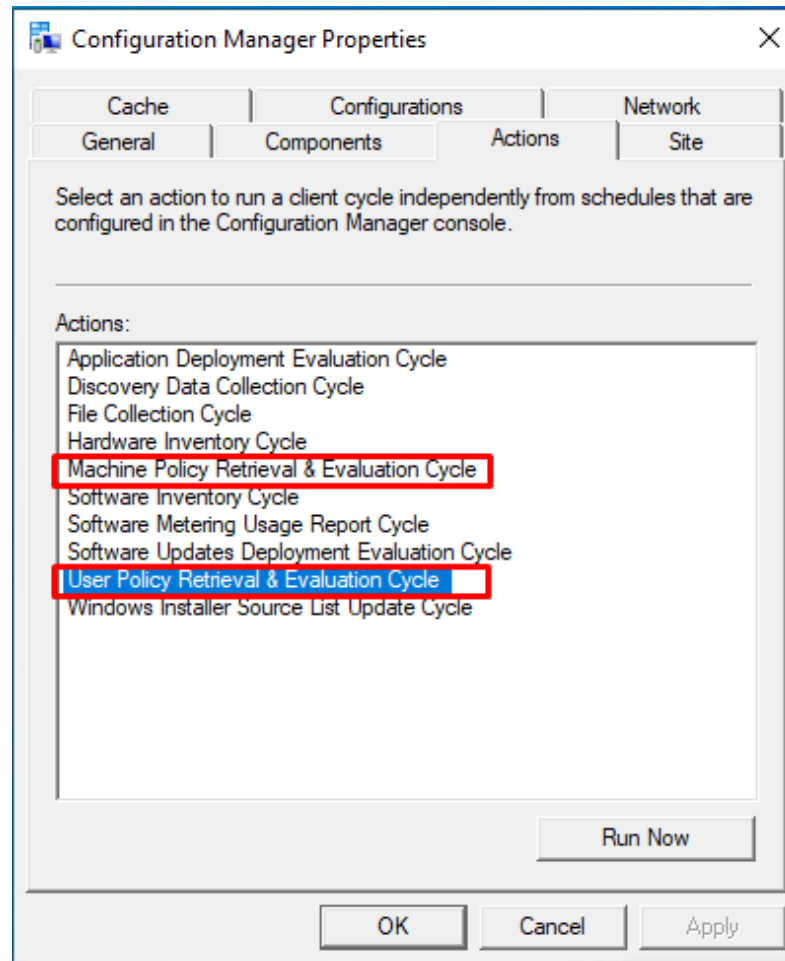
```
End If
Next
Set objXMLDoc = Nothing
End Function
```

Defining User Experience

Take extra care to specify "Install for System" when defining the installation behavior.

SCCM Client Configuration

If the application does not appear in the client console, on the SCCM Client machine navigate to **Control Panel** → **Configuration Manager** → **Actions** and run "**Machine Policy Retrieval & Evaluation Cycle**" and "**User Policy Retrieval & Evaluation Cycle**" actions manually.



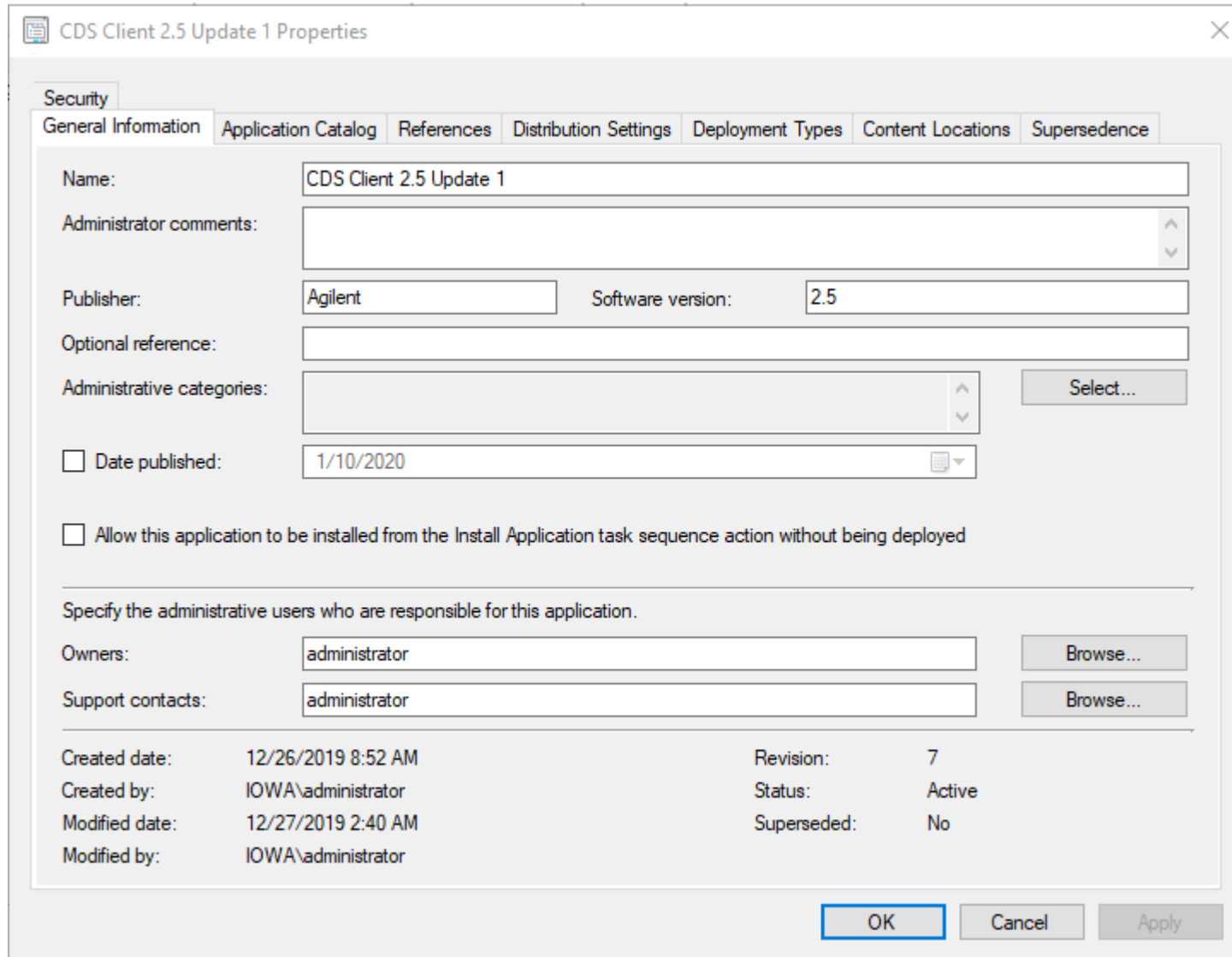
Note: During installation of CDS Client, the system may require to be rebooted. After rebooting, relaunch SCCM client to confirm that CDS Client is fully installed. If SCCM indicates that CDS Client is not yet installed, click on the Install button again.

SCCM setup for patch installation

Using a similar procedure, SCCM can be used to manage and distribute patches that have been released for OpenLab CDS also.

Setting up a patch for OpenLab CDS Client

In order to set up a patch in SCCM, an application needs to be defined.



The screenshot shows the 'CDS Client 2.5 Update 1 Properties' dialog box in SCCM. The 'Security' tab is active, and the 'General Information' sub-tab is selected. The dialog contains the following fields and options:

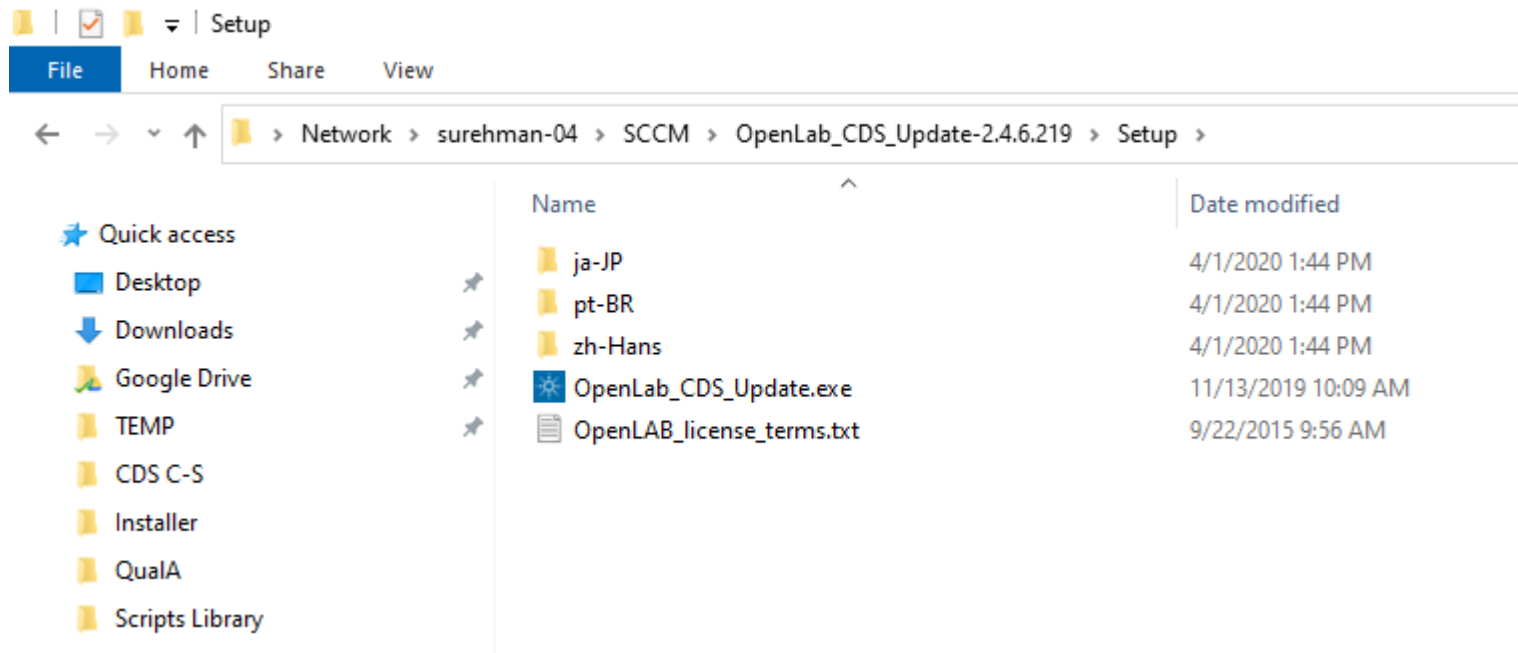
- Name:** CDS Client 2.5 Update 1
- Administrator comments:** (Empty text area)
- Publisher:** Agilent
- Software version:** 2.5
- Optional reference:** (Empty text area)
- Administrative categories:** (Empty list box with 'Select...' button)
- Date published:** 1/10/2020
- Allow this application to be installed from the Install Application task sequence action without being deployed**
- Specify the administrative users who are responsible for this application.**
- Owners:** administrator (with 'Browse...' button)
- Support contacts:** administrator (with 'Browse...' button)
- Created date:** 12/26/2019 8:52 AM
- Created by:** IOWA\administrator
- Modified date:** 12/27/2019 2:40 AM
- Modified by:** IOWA\administrator
- Revision:** 7
- Status:** Active
- Superseded:** No

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Apply'.

While defining an application a “Deployment Type” also needs to be defined for the patch. Deployment Types provides information about the installation method and source file location for the patch.

Defining content

If you get the patch in a zipped format you will need to unzip it before it can be deployed for SCCM. You may find that the ZIP has more than one patch zips within it. Read the accompanying documentation to determine whether the patch is applicable to a CDS Client. Similar to the OpenLab CDS installation media, you need to host the patch on a shared network location that the client systems can access.



A deployment type for a patch also needs to be manually created. The location then needs to be provided as the content location for the patch deployment type. The following settings should be used to specify the installation program, uninstallation program and start location.

Note: If you want to disallow end users from uninstalling patches from SCCM Client Console, you can leave the uninstall command empty. Doing so will disable the Uninstall button in the SCCM Client Console.

Install program:

```
OpenLab_CDS_Update.exe -s LicenseAccepted=True -norestart
```

Installation start in:

```
\Setup
```

Uninstall program:

```
OpenLab_CDS_Update.exe -s -uninstall -norestart
```


Uninstallation start in:
\\Setup

In the “user experience” tab make sure to set install behavior to **'Install for system'** and set **'Configuration Manager client will force a mandatory restart'**. These settings are required for patches.

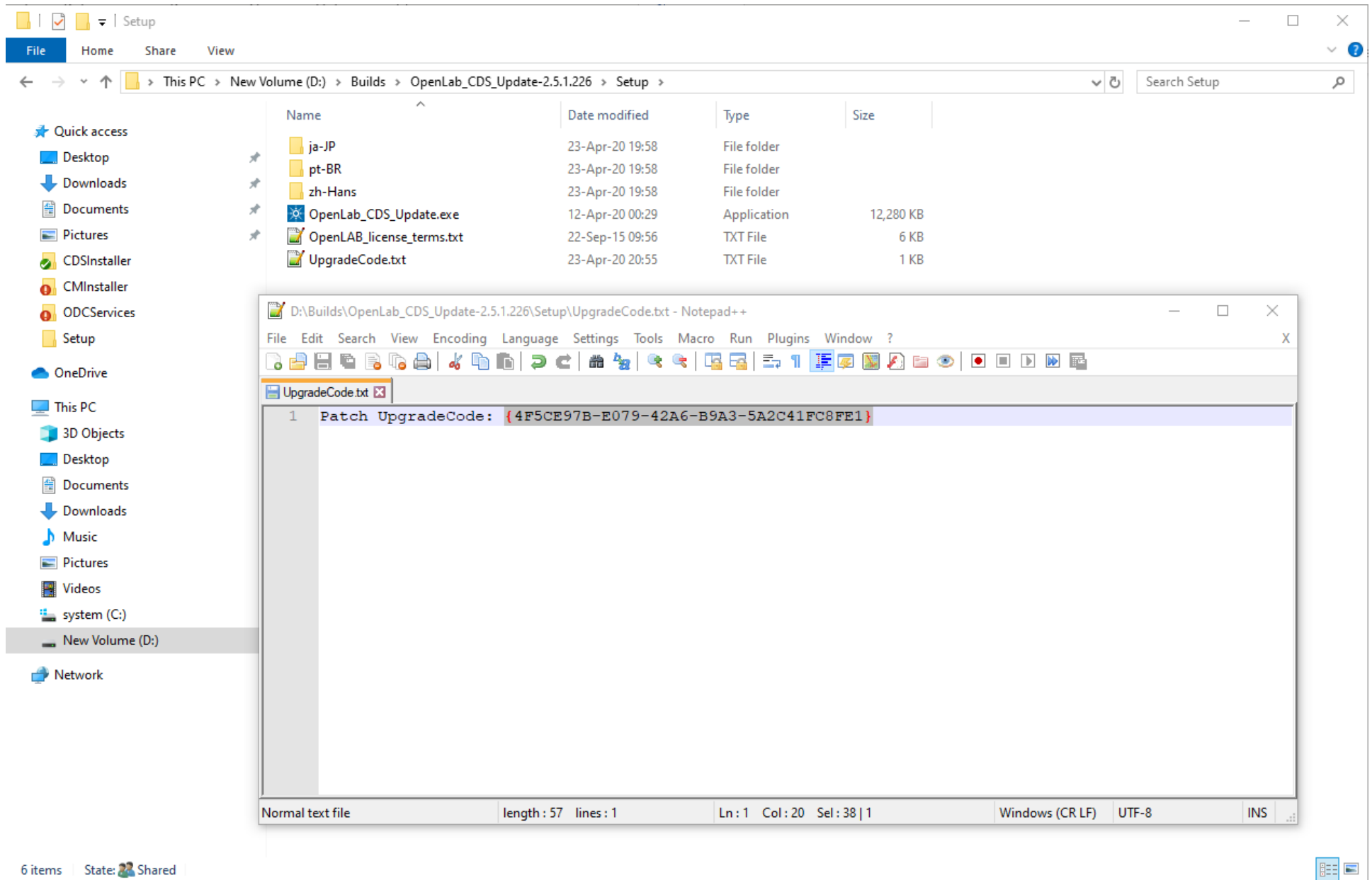
The screenshot shows the 'OpenLab CDS 2.5 Update 1 Properties' dialog box with the 'User Experience' tab selected. The dialog has several tabs: 'General', 'Content', 'Programs', 'Detection Method', 'User Experience', 'Requirements', 'Return Codes', and 'Dependencies'. The 'User Experience' tab contains the following settings:

- Specify user experience settings for the application.
 - Installation behavior: Install for system (dropdown)
 - Logon requirement: Only when a user is logged on (dropdown)
 - Installation program visibility: Normal (dropdown)
 - Allow users to view and interact with the program installation
- Specify the maximum run time and estimated installation time of the deployment program for this application. The estimated installation time displays to the user when the application installs.
 - Maximum allowed run time (minutes): 120 (spin box)
 - Estimated installation time (minutes): 0 (spin box)
- Should Configuration Manager enforce specific behavior regardless of the application's intended behavior?
 - Configuration Manager client will force a mandatory device restart (dropdown)

Buttons at the bottom: OK, Cancel, Apply.

Defining the Detection Method

Before you can define the detection method you need to find the upgrade code for the patch. Each patch has its own upgrade code. The patch upgrade code is in “UpgradeCode.txt” file in the root folder of the patch.



For example, in the image above, the patch upgrade code is `{4F5CE97B-E079-42A6-B9A3-5A2C41FC8FE1}`.

For detecting a patch, a custom script should be used.

Script Parameters:

1. `$PatchUpgradeCode = {4F5CE97B-E079-42A6-B9A3-5A2C41FC8FE1}`

PowerShell Example

The following example script can be used after updating the `PatchUpgradeCode` variable to the correct value.

Note: PowerShell execution policy must be set in SCCM as described earlier also.

```
#####
# OpenLab Installers: centralized install & update
#
# Version: 2.5
# Build: <OLINST_BUILD_NUMBER_INSERT_HERE>
#
# (c) Agilent Technologies 2020
#
#####

<#
.DESCRIPTION
    Detection method to check already installed OpenLab Software Updates

.PARAMETER $PatchUpgradeCode
    Upgrade code of OpenLab Software Update bundle to be installed

.OUTPUTS
    "Installed" will be written to the output, if OpenLab Software Update is already installed and configured,
    otherwise an empty string.
#>

#####
$PatchUpgradeCode = '{4F5CE97B-E079-42A6-B9A3-5A2C41FC8FE1}' # Patch Upgrade Code
#####

$keys="HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*",
      "HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*"

$product = Get-ItemProperty $keys |
            Where-Object BundleUpgradeCode -eq $PatchUpgradeCode |
            Select-Object DisplayName, BundleUpgradeCode

if ($product) {
    Write-Host "Installed"
    Exit
}
```

VBScript Example

The following example script can be used after updating the `patchUpgradeCode` variable to the correct value.

```
'#####  
'# OpenLab Installers: centralized install & update  
'#  
'# Version: 2.5  
'# Build: <OLINST_BUILD_NUMBER_INSERT_HERE>  
'#  
'# (c) Agilent Technologies 2020  
'#  
'#####  
  
' "Installed" will be written to the output, if OpenLab Software Update is already  
' installed and configured, otherwise an empty string.  
  
'#####  
patchUpgradeCode = "{4F5CE97B-E079-42A6-B9A3-5A2C41FC8FE1}" ' Patch Upgrade Code  
'#####  
  
Main()  
  
'WScript.Echo "Installed"  
WScript.StdOut.Write "Installed"  
WScript.Quit(0)  
  
Function Main()  
    RegistryHive = "HKEY_LOCAL_MACHINE\  
    RegLocation1 = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\  
    RegLocation2 = "SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\  
  
    On Error Resume Next  
  
    If (BundleExists(RegistryHive, RegLocation1, patchUpgradeCode) = False) And _  
        (BundleExists(RegistryHive, RegLocation2, patchUpgradeCode) = False) Then  
        'WScript.Echo "Not installed"  
        WScript.Quit(0)  
    End If  
End Function
```

```

Function BundleExists(Hive, location, upgradeCode)
    On Error Resume Next
    Dim BundleVersion
    Dim WSHShell
    Set WSHShell = CreateObject("WScript.Shell")

    keys = GetSubKeys(location)
    For Each RegistryKey In keys
        'BundleUpgradeCode - is a Multi-string value and WSHShell.RegRead returns it as an array
        bundleUpgradeCodesValues = WSHShell.RegRead(Hive + location + RegistryKey + "\BundleUpgradeCode")
        If Err.Number = 0 Then
            For i = 0 To UBound(bundleUpgradeCodesValues)
                If LCase(bundleUpgradeCodesValues(i)) = LCase(upgradeCode) Then
                    BundleExists = CBool(1)
                    Set WSHShell = Nothing
                    Exit Function
                End If
            Next
        Else
            Err.Clear
        End If
    Next
    Set WSHShell = Nothing
    BundleExists = CBool(0)
End Function

Function GetSubKeys(location)
    Const HKEY_LOCAL_MACHINE = &H80000002
    strComputer = "."

    Set oReg=GetObject("winmgmts:{impersonationLevel=impersonate}!\\" & _
    strComputer & "\root\default:StdRegProv")

    oReg.EnumKey HKEY_LOCAL_MACHINE, location, arrSubKeys
    Set oReg = Nothing
    GetSubKeys = arrSubKeys
End Function

```