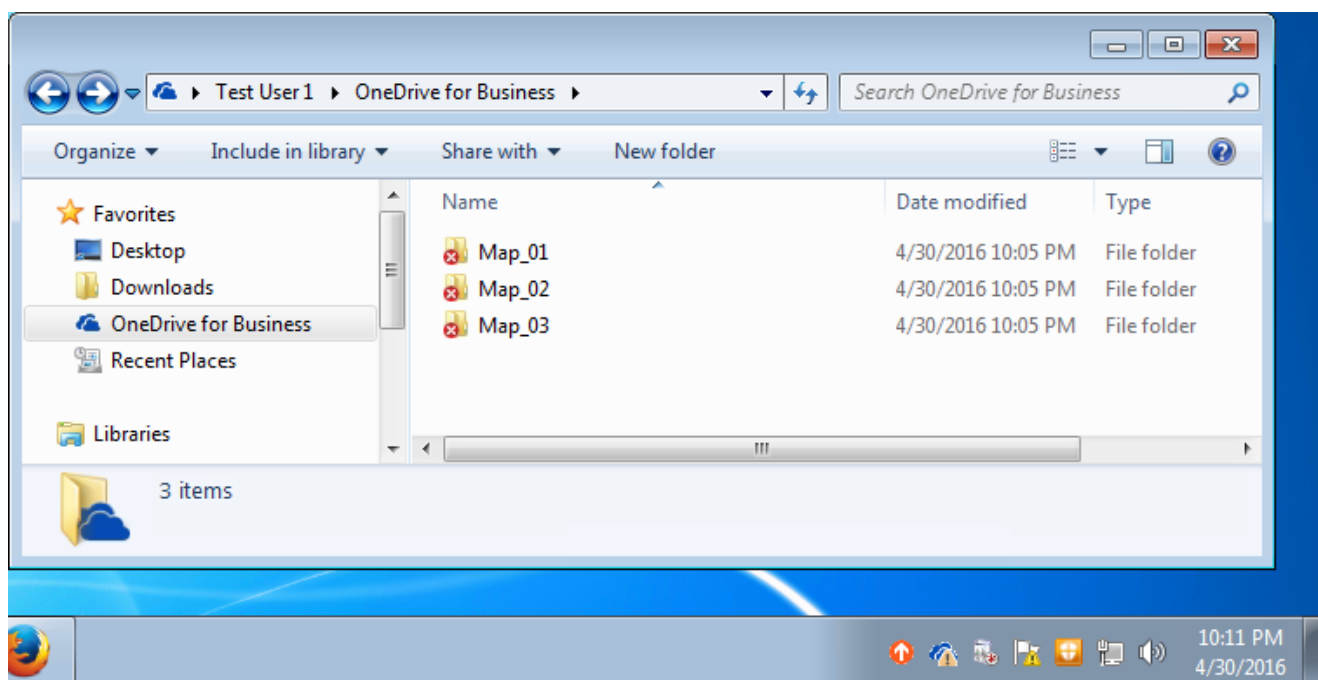


Installing OneDrive NextGen Synchronisation Client

The OneDrive for Business client can give some difficulties, like unable to synchronize:



Unable to sync files with OneDrive for Business.

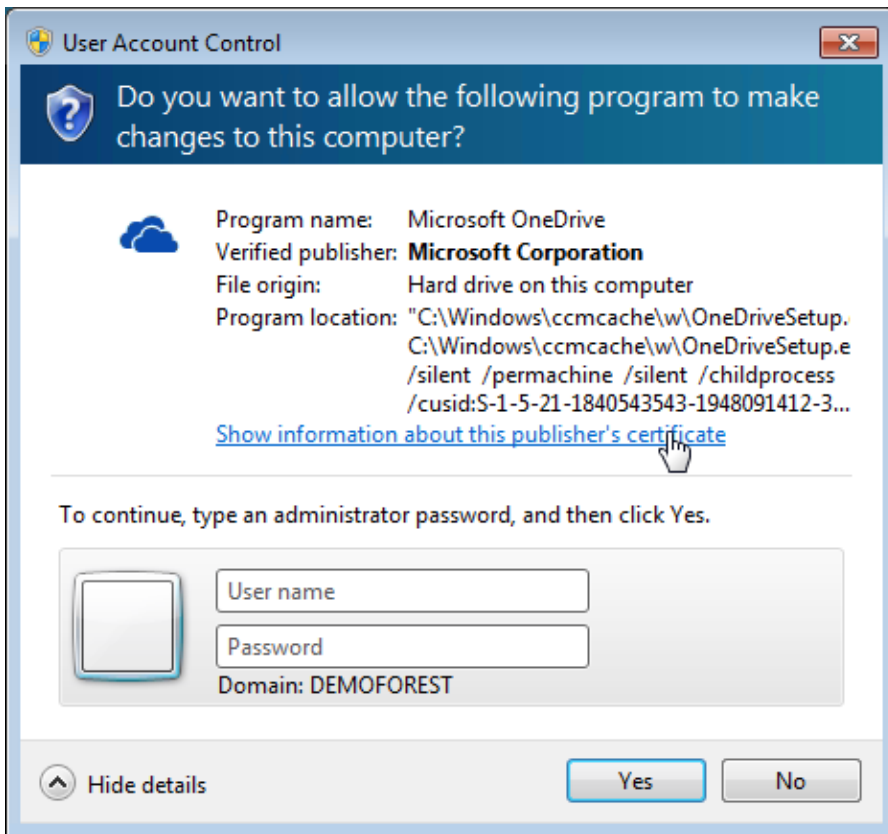
This can be solved by installing the new OneDrive Next Gen Sync Client. In this article I will describe how this application can be installed via SCCM 2012.

Please be informed that you still need the OneDrive for Business client to synchronize with Sharepoint lists.

Background information

The OneDrive NextGen Sync Client is installed to the users' %LOCALAPPDATA% directory. As the application is installed to the %LOCALAPPDATA% directory, you can assume that you can install the application with user rights in SCCM 2012. If you do so,

the installation will fail:



Failure as admin rights are needed to install.

As you can see, there are 'hidden' admin rights needed.

The solution

- Created a vscript that installs the application and sets a detection rule after a successful installation. The detection rule can be used in SCCM 2012 to detect a successful installation.

'

=====
=====

' Installs Microsoft OneDriveNextGenSyncClient
17.3.6386.0412

' Created by Willem-Jan Vroom

' Version history:

'

' 0.0.1

' Initial version

'

' 1.0.0

' Final version

'

'

=====
=====

'
.....

' Declare the most variables.

'
.....

Option Explicit

```
Dim objShell : set objShell
= WScript.CreateObject("WScript.Shell")
Dim objLogFileFS0 : Set
objLogFileFS0 =
CreateObject("Scripting.FileSystemObject")
Dim objFS0 : Set objFS0
= CreateObject("Scripting.FileSystemObject")
Dim objProcessEnv : Set
objProcessEnv =
objShell.Environment("PROCESS")
Dim objWMIService : Set
objWMIService =
GetObject("winmgmts:\\.\root\cimv2")
Dim fn_objWMIService : Set
fn_objWMIService =
GetObject("winmgmts:\\.\root\cimv2")
```

```

Dim objReg : Set objReg
= GetObject("winmgmts:\\.\root\default:StdRegProv")
Dim objLogFile
Dim CurrentDir : CurrentDir
= Left(Wscript.ScriptFullname,
InstrRev(Wscript.ScriptFullname, "\"))
CurrentDir
= Left(CurrentDir, len(CurrentDir)-1)
Dim strcomputerName :
strcomputerName =
objProcessEnv("COMPUTERNAME")
Dim strLogLocation : strLogLocation
= "C:\WINDOWS\system32\Logfiles"
Dim strOutputFile : strOutputFile
= strLogLocation & "\" & strcomputerName &
"_Installation_OneDriveSyncClient_" &
Replace(FormatDateTime(Now(), 2),"/","-") & ".log"
Dim strArchitecture :
strArchitecture = "x86"
Dim strCurrentUser : strCurrentUser
= ""
Dim strCurrentUserSID :
strCurrentUserSID = ""
Dim strOS : strOS
= ""
Dim strCommand : strCommand
= ""
Dim strLine : strLine
= ""
Dim strQuery : strQuery
= ""
Dim strCommonDesktop :
strCommonDesktop = ""
Dim strArray : strArray
= ""
Dim strProcess : strProcess
= ""

```

```

    Dim strLanguage : strLanguage
= ""
    Dim strInstallLanguage :
strInstallLanguage = ""
    Dim strDetectUserUILanguage :
strDetectUserUILanguage = ""
    Dim strValue : strValue
= ""
    Dim strKeyPath : strKeyPath
= ""
    Dim strValueName : strValueName
= ""
    Dim strCommonPrograms :
strCommonPrograms = ""
    Dim strFileToDelete :
strFileToDelete = ""
    dim strProgramFilesFolder :
strProgramFilesFolder =
objProcessEnv("ProgramFiles")
    Dim strDotNetInstallPath :
strDotNetInstallPath = ""
    Dim strWindir : strWinDir
= objShell.ExpandEnvironmentStrings("%WinDir%")
    Dim dwValue : dwValue
= 0

    Dim arrFileNamesToKill :
arrFileNamesToKill =
Array("doesnotexist.exe")
    Dim objFileNameToKill

    Dim arrArguments
    Dim arrLanguage
    Dim strDNSDomain : strDNSDomain
= CreateObject("ADSystemInfo").DomainShortName
    Dim valCounter
    Dim valResult

```

```
Dim valReturnCode
```

```
Dim colProcess, objProcess
```

```
Dim valOSBuildNumber
```

```
Dim objSubfolder
```

```
Dim colItems, objItem, Subfolder
```

```
Dim colSoftware, objSoftware
```

```
Dim arrValues, arrSubKeys
```

```
Const ForWriting          = 2
```

```
Const ForReading         = 1
```

```
Const ForAppending       = 8
```

```
Const OverWriteFiles     = True
```

```
const HKEY_CURRENT_USER  = &H80000001
```

```
const HKEY_LOCAL_MACHINE = &H80000002
```

```
Const HKEY_USERS          = &H80000003
```

```
Const DEBUGMODE          = False
```

```
' -----
```

```
-----
```

```
' Create the log location (if not exists)
```

```
' Open the logfile.
```

```
' -----
```

```
-----
```

```
CreateFolderStructure(strLogLocation)
```

```
OpenLogFile()
```

```
WriteToLog "- ACTION: script started."
```

```
WriteToLog " "
```

```
' -----
```

```
-----
```

```
' Detect the current OS.
```

```
' -----
```

```
-----
```

```
    Set colItems = objWMIService.ExecQuery("Select  
Caption,BuildNumber from Win32_OperatingSystem")  
    For Each objItem in colItems  
        strOS = objItem.Caption  
        valOSBuildNumber = objItem.BuildNumber  
    next
```

```
' -----  
-----  
' Find the currently logged on userid.  
' And translate it to the users SID.  
' -----  
-----
```

```
    Set colProcess = objWMIService.ExecQuery("Select *  
from Win32_Process where Name = 'explorer.exe'")  
    For Each objProcess in colProcess  
        valReturnCode =  
objProcess.GetOwner(strCurrentUser)  
        if valReturnCode <> 0 Then strCurrentUser =  
"Unable to find the username!"  
    Next
```

```
    strCurrentUser = Ucase(strDNSDomain) & "\" &  
strCurrentUser
```

```
    strCurrentUserSID = GetSIDFromUser(strCurrentUser)
```

```
' -----  
-----  
' Detect the current processor architecture.  
' -----  
-----
```

```
if objFS0.FolderExists(strWindir & "\syswow64") Then  
    strArchitecture = "x64"  
    strProgramFilesFolder =
```

```

objProcessEnv("ProgramFiles(x86)")
  end if

' -----
-----
' Detect the current User Interface MUI.
' -----
-----

  strDetectUserUILanguage = DetectUserUILanguage

  WriteToLog("##### Details regarding operating
system and logged on user      #####")
  WriteToLog("Found Operating System:      " & strOS)
  WriteToLog("Found architecture:          " &
strArchitecture)
  WriteToLog("Logged on userid:           " &
strCurrentUser)
  WriteToLog("Logged on user SID:         " &
strCurrentUserSID)
  WriteToLog("Language:                   " &
strDetectUserUILanguage)
  WriteToLog("##### End details regarding operating
system and logged on user #####")
  WriteToLog(" ")
      WriteToLog("##### Installation
#####")

' -----
-----
' Close applications that might screw up the
installation.
' -----
-----

For Each objFileNameToKill in arrFileNamesToKill
  fnKillProcess(objFileNameToKill)

```


Next

```
' -----  
-----  
' Sets SEE_MASK_NOZONECHECKS to 1.  
' That will avoid a screen that asks for permission to  
run an application from  
' a share or mapped drive.  
' -----  
-----  
  
objProcessEnv("SEE_MASK_NOZONECHECKS") = 1  
  
' -----  
-----  
' Uninstall other Samsung Kies installations.  
' if the command line option '/uninstall' is given, then  
only a uninstall and quit.  
' -----  
-----  
  
if wscript.arguments.count > 0 Then  
    if lcase(wscript.arguments(0)) = "/uninstall" or  
lcase(wscript.arguments(0)) = "/remove" then  
        WriteToLog("The command line option '" &  
wscript.arguments(0) & "' was found, so only a silent  
uninstall will be performed.")  
        RemoveOneDriveSyncClient()  
        WriteToLog("##### End Uninstall  
#####")  
        WriteToLog(" ")  
        CloseLogFile()  
        wscript.quit  
    end if  
end if  
  
' -----  
-----
```

```

-----
' Start the base installation
' -----
-----

        strCommand = chr(34) & CurrentDir &
"\OneDriveSetup.exe" & chr(34)
        strCommand = strCommand & " /silent"
        WriteToLog ("Running command: " & strCommand)
        valResult = objShell.Run (strCommand,6,True)
        WriteToLog("Result: " & valResult)
        if valResult = 0 Then
            GenereteSCCM2012DetectionRule
        end if
        WriteToLog ("##### End Installation
#####")
        WriteToLog(" ")
        CloseLogFile()
        wscript.quit valReturnCode

Sub OpenLogFile()

' -----
-----
' Subroutine: OpenLogFile()
' The name of the logfile is mentinoed in the variabele
strOutputFile.
' -----
-----

        If objLogFileFS0.FileExists(strOutputFile) Then
                                Set      objLogFile      =
objLogFileFS0.OpenTextFile(strOutputFile, ForAppending)
        Else
                                Set      objLogFile      =
objLogFileFS0.CreateTextFile(strOutputFile)
        End If

```

End Sub

Sub CloseLogFile()

```
' -----  
-----  
' Subroutine: CloseLogFile()  
' Close the log file.  
' -----  
-----
```

```
WriteToLog "- ACTION: script ended."  
objLogFile.Close  
Set objLogFileFS0 = Nothing
```

End Sub

Function WriteToLog(sLogMessage)

```
' -----  
-----  
' Function: WriteToLog(sLogMessage)  
' Writes an entry 'sLogMessage' in the logfile.  
' -----  
-----
```

```
    if instr(sLogMessage, "- ACTION: ") = 0 then  
sLogMessage = "                " & sLogMessage  
    objLogFile.WriteLine("Time: " & now & "    " &  
sLogMessage)
```

End Function

Function GetSIDFromUser(UserName)

```
' -----  
-----
```

```
' Function: GetSIDFromUser(Username)
' Gets the SID from the give username.
' We use the registry to avoid an empty result if run
under the SYSTEM account.
```

```
' -----
-----
```

```
Dim DomainName, Result, WMIUser
```

```
If InStr(Username, "\") > 0 Then
    DomainName = Mid(Username, 1, InStr(Username, "\") -
1)
    Username = Mid(Username, InStr(Username, "\") + 1)
Else
    DomainName =
CreateObject("WScript.Network").UserDomain
End If
```

```
On Error Resume Next
    Set WMIUser =
GetObject("winmgmts:{impersonationlevel=impersonate}!" &
    & "/root/cimv2:Win32_UserAccount.Domain='" &
DomainName & "' _
    & ",Name='" & Username & "'")
If Err.Number = 0 Then
    Result = WMIUser.SID
Else
    Result = ""
End If
On Error GoTo 0
```

```
GetSIDFromUser = Result
```

```
End Function
```

```
Sub CreateFolderStructure(strFolderNameToBeCreated)
```

```
' -----  
-----  
' Subroutine:  
CreateFolderStructure(strFolderNameToBeCreated)  
' Creates the map as mentioned in  
strFolderNameToBeCreated.  
' -----  
-----
```

```
Dim arrFoldersTMP : arrFoldersTMP = split  
(strFolderNameToBeCreated, "\")  
Dim strFolder : strFolder = ""  
Dim objFolderTMP
```

```
For Each objFolderTMP in arrFoldersTMP  
    strFolder = strFolder & objFolderTMP  
    If NOT objFS0.FolderExists(strFolder) Then  
        objFS0.CreateFolder(strFolder)  
    end If  
    strFolder = strFolder & "\"  
Next
```

```
End Sub
```

```
Function DetectUserUILanguage
```

```
' -----  
-----  
' Function: DetectUserUILanguage  
' Detects the User Interface Language  
' Modified by Willem-Jan Vroom d.d 12 Feb 14: also  
detects Czech, Finnish, Hungarian, Polish,  
' Russian, Slovak, Slovenian and Turkish languages on  
Windows 7.  
' -----  
-----  
if instr(lcase(strOS),"windows xp") > 0 then
```

```

        strKeyPath                = strCurrentUserSID &
"\Control Panel\Desktop"
        strValueName              = "MultiUILanguageId"
        strLanguage               = "eng"
                                objReg.GetStringvalue
HKEY_USERS, strKeyPath, strValueName, strValue
    if NOT IsEmpty(strValue) Then
        strValue = UCase(right(strValue,4))
        if strValue = "040C" then strLanguage = "fra"
        if strValue = "0410" then strLanguage = "ita"
        if strValue = "0407" then strLanguage = "ger"
        if strValue = "040A" then strLanguage = "spa"
        if strValue = "0C0A" then strLanguage = "spa"
    end if
    else
        strKeyPath                = strCurrentUserSID &
"\Control Panel\Desktop"
        strValueName              = "PreferredUILanguages"
        strLanguage               = "eng"
                                objReg.GetMultiStringValue
HKEY_USERS, strKeyPath, strValueName, arrValues
    if not IsNull(arrValues) Then
        strValue=arrValues(0)
        if NOT IsEmpty(strValue) then
            strValue = lcase(left(strValue,2))
            if strValue = "fr" then strLanguage = "fra"
            if strValue = "it" then strLanguage = "ita"
            if strValue = "de" then strLanguage = "ger"
            if strValue = "es" then strLanguage = "spa"
            if strValue = "cs" then strLanguage = "cze"
            if strValue = "fi" then strLanguage = "fin"
            if strValue = "hu" then strLanguage = "hun"
            if strValue = "pl" then strLanguage = "pol"
            if strValue = "ru" then strLanguage = "rus"
            if strValue = "sk" then strLanguage = "sky"
            if strValue = "sl" then strLanguage = "slv"
            if strValue = "sv" then strLanguage = "swe"

```

```

        if strValue = "tr" then strLanguage = "tur"
        if strValue = "nb" then strLanguage = "nor"
    end if
end if
if strLanguage = "eng" then
    strKeyPath                = ".DEFAULT\Control
Panel\Desktop\MuiCached"
    strValueName              =
"MachinePreferredUILanguages"
    objReg.GetMultiStringValue
HKEY_USERS, strKeyPath, strValueName, arrValues
    if not IsNull(arrValues) Then
        strValue=arrValues(0)
        if NOT IsEmpty(strValue) then
            strValue = lcase(left(strValue,2))
            if strValue = "fr" then strLanguage =
"fra"
            if strValue = "it" then strLanguage =
"ita"
            if strValue = "de" then strLanguage =
"ger"
            if strValue = "es" then strLanguage =
"spa"
            if strValue = "cs" then strLanguage =
"cze"
            if strValue = "fi" then strLanguage =
"fin"
            if strValue = "hu" then strLanguage =
"hun"
            if strValue = "pl" then strLanguage =
"pol"
            if strValue = "ru" then strLanguage =
"rus"
            if strValue = "sk" then strLanguage =
"sky"
            if strValue = "sl" then strLanguage =
"slv"

```

```

        if strValue = "sv" then strLanguage =
"swe"
        if strValue = "tr" then strLanguage =
"tur"
        if strValue = "nb" then strLanguage =
"nor"
            end if
        end if
    end if
end if

```

```

    DetectUserUILanguage = strLanguage
End Function

```

```

Function fnKillProcess(strProcessName)

```

```

' -----
-----
' Function: fnKillProcess(strProcessName)
' Terminates the given processname.
' -----
-----

```

```

    Set colProcess = fn_objWMIService.ExecQuery ("Select *
From Win32_Process")
    For Each objProcess In colProcess
                                                If
Instr(LCase(objProcess.Name),LCase(strProcessName)) > 0
Then
        objShell.Run "TASKKILL /F /T /IM " &
objProcess.Name, 0, False
        objProcess.Terminate()
        WriteToLog("Terminating application: " &
objProcess.Name)
    End If
Next

```


End Function

Function GenereteSCCM2012DetectionRule

```
' -----  
-----  
' Function: GenereteSCCM2012DetectionRule  
' Creates the SCCM 2012 Detection Rule to detect a  
successfull installation.  
' -----  
-----  
  
    WriteToLog ("Creates the SCCM 2012 Detection Rule.")  
                strKeyPath          =  
"SOFTWARE\VroomSoft\SCCM2012DetectionRules"  
                strValueName        =  
"Install_OneDriveSyncClient_17.3.6386.0412"  
    strValue      = "true"  
    objReg.CreateKey      HKEY_LOCAL_MACHINE, strKeyPath  
    objReg.SetStringValue HKEY_LOCAL_MACHINE, strKeyPath,  
strValueName, strValue  
End Function
```

Function RemoveOneDriveSyncClient()

```
' -----  
-----  
' Function: RemoveOneDriveSyncClient()  
' Removes OneDrive Next Gen Sync Client.  
' -----  
-----  
  
    strCommand = chr(34) & CurrentDir &  
"\OneDriveSetup.exe" & chr(34)  
    strCommand = strCommand & " /uninstall"  
    WriteToLog ("Running command: " & strCommand)  
    valResult = objShell.Run (strCommand,6,True)
```

```

WriteToLog("Result: " & valResult)
if valResult = 0 Then
    strKeyPath =
"SOFTWARE\VroomSoft\SCCM2012DetectionRules"
    strValueName =
"Install_OneDriveSyncClient_17.3.6386.0412"
    objReg.DeleteValue HKEY_LOCAL_MACHINE, strKeyPath,
strValueName
end if
End Function

```

- This is the detection rule that will be used in SCCM 2012:

```
Function GenereteSCCM2012DetectionRule
```

```

' -----
' Function: GenereteSCCM2012DetectionRule
' Creates the SCCM 2012 Detection Rule to detect a successfull installation.
' -----

```

```
WriteToLog ("Creates the SCCM 2012 Detection Rule.")
```

```

strKeyPath = "SOFTWARE\VroomSoft\SCCM2012DetectionRules"
strValueName = "Install_OneDriveSyncClient_17.3.6386.0412"
strValue = "true"
objReg.CreateKey HKEY_LOCAL_MACHINE, strKeyPath
objReg.SetStringValue HKEY_LOCAL_MACHINE, strKeyPath, strValueName, strValue

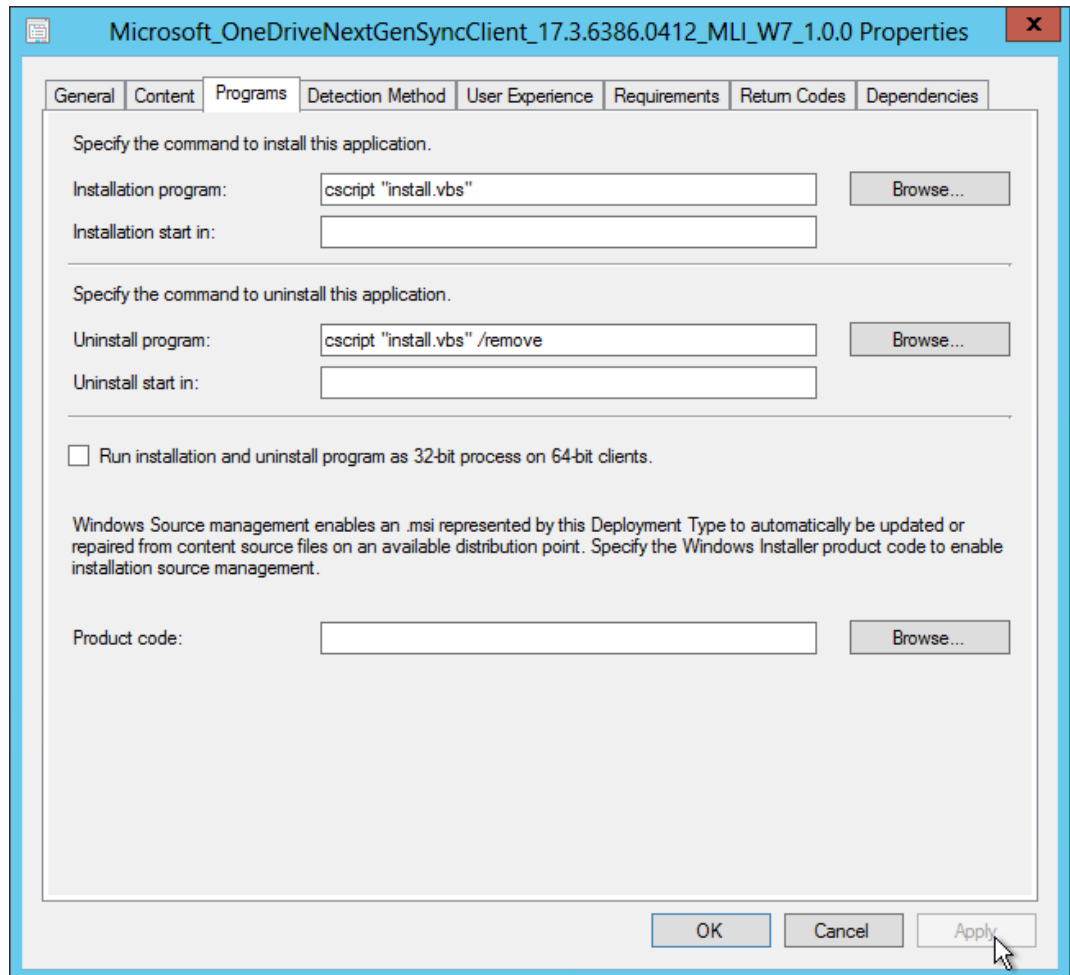
```

```
End Function
```

Detection rule set in the vbscript.

Implementation in SCCM 2012

1. Create a 'installation program' and 'uninstall program':



New application.

2. The detection rule:

Detection Rule

Create a rule that indicates the presence of this application.

Setting Type: Registry

Specify the registry key or value to detect this application.

Hive: HKEY_LOCAL_MACHINE

Key: SOFTWARE\VroomSoft\SCCM2012DetectionRules

Value: Install_OneDriveSyncClient_17.3.6386.0412

Use (Default) registry key value for detection

This registry key is associated with a 32-bit application on 64-bit systems

Data Type: String

This registry setting must exist on the target system to indicate presence of this application

This registry setting must satisfy the following rule to indicate the presence of this application

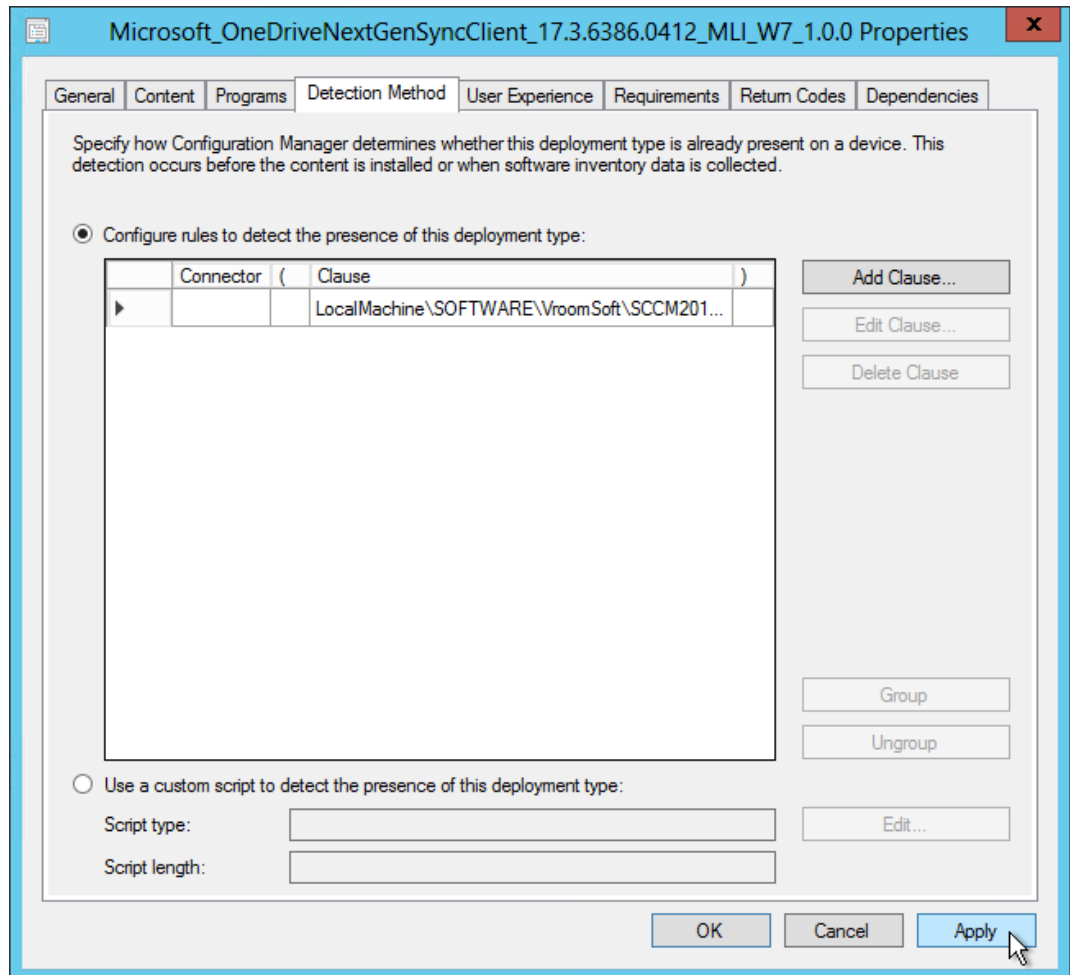
Operator: Equals

Value: true

OK Cancel

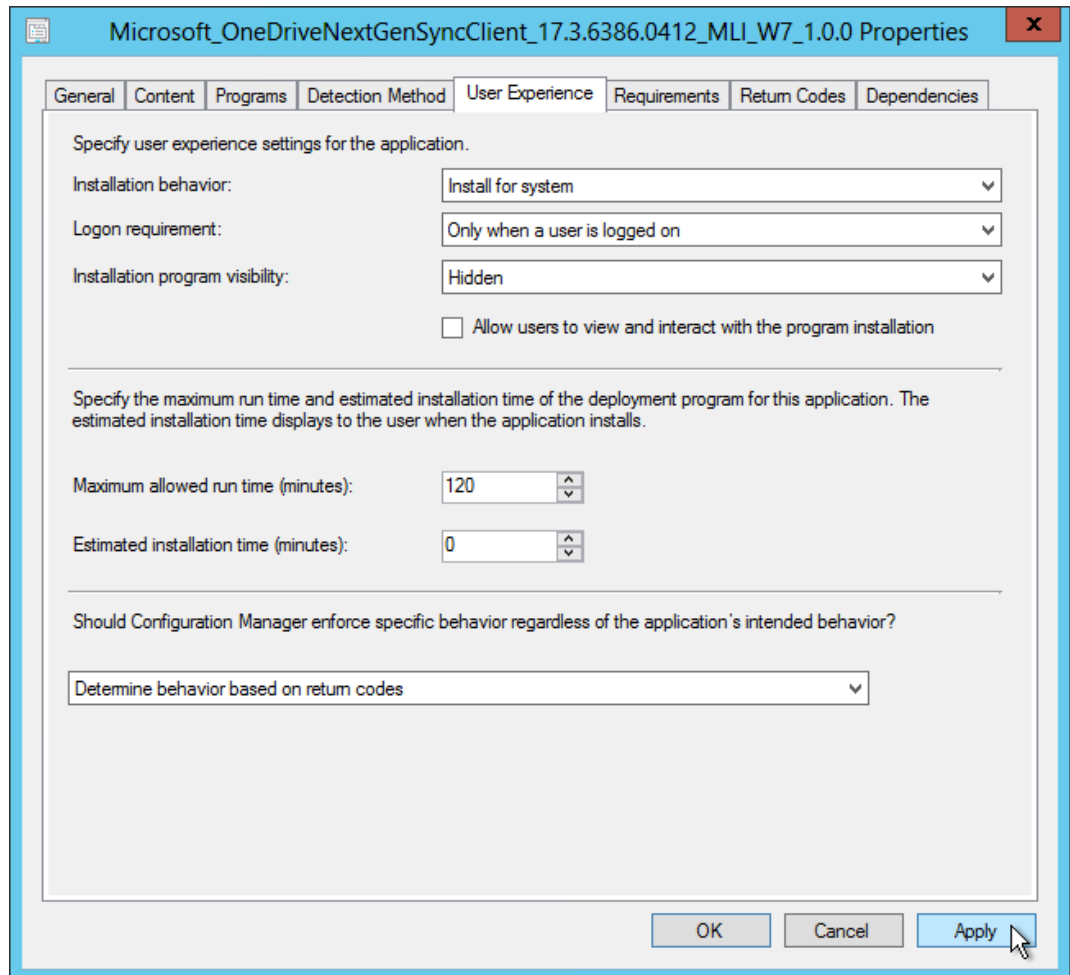
Detection rule.

3. The detection method:



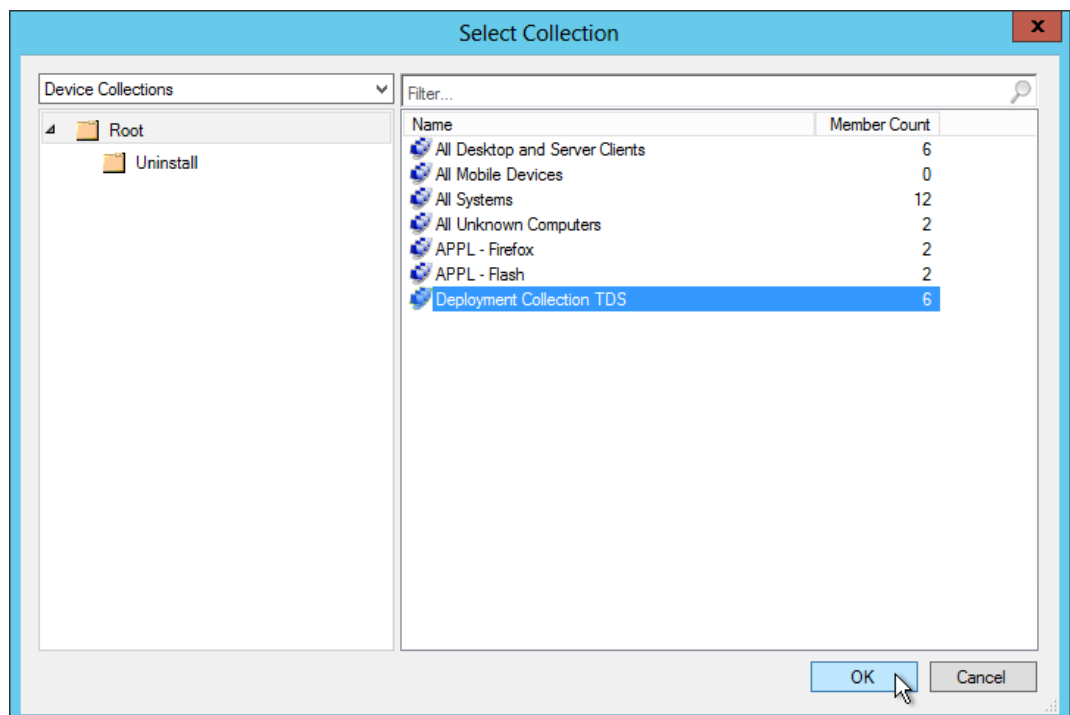
Detection method.

4. The application is installed with admin rights:

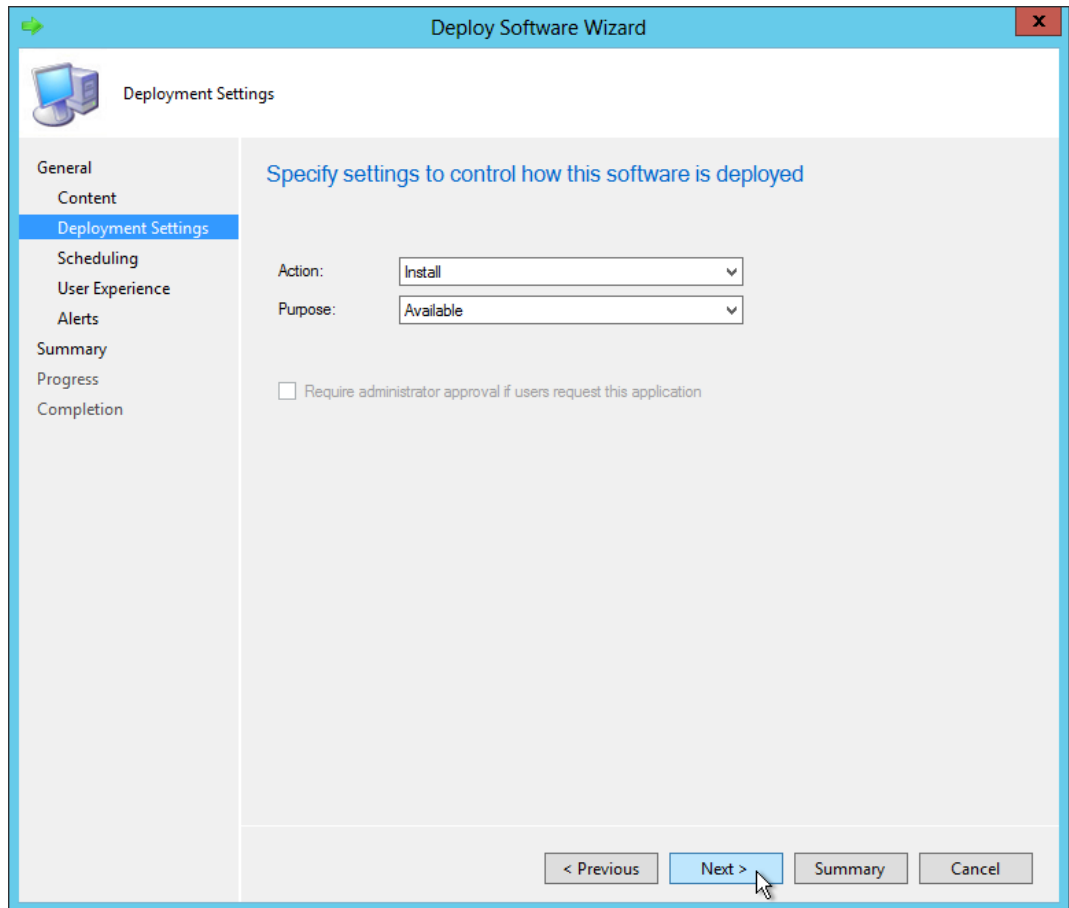


Installation with admin rights.

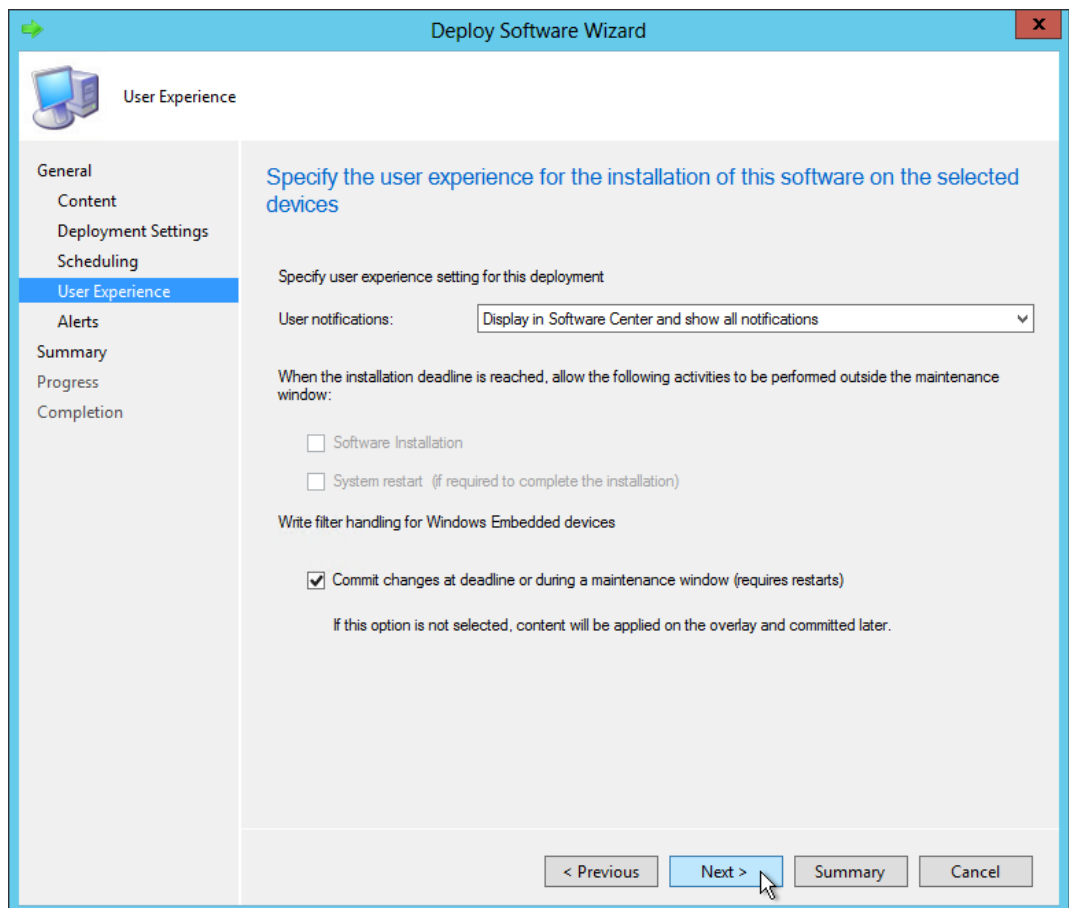
5. And deploy the application to a device based collection:



Deployment to a device based collection.



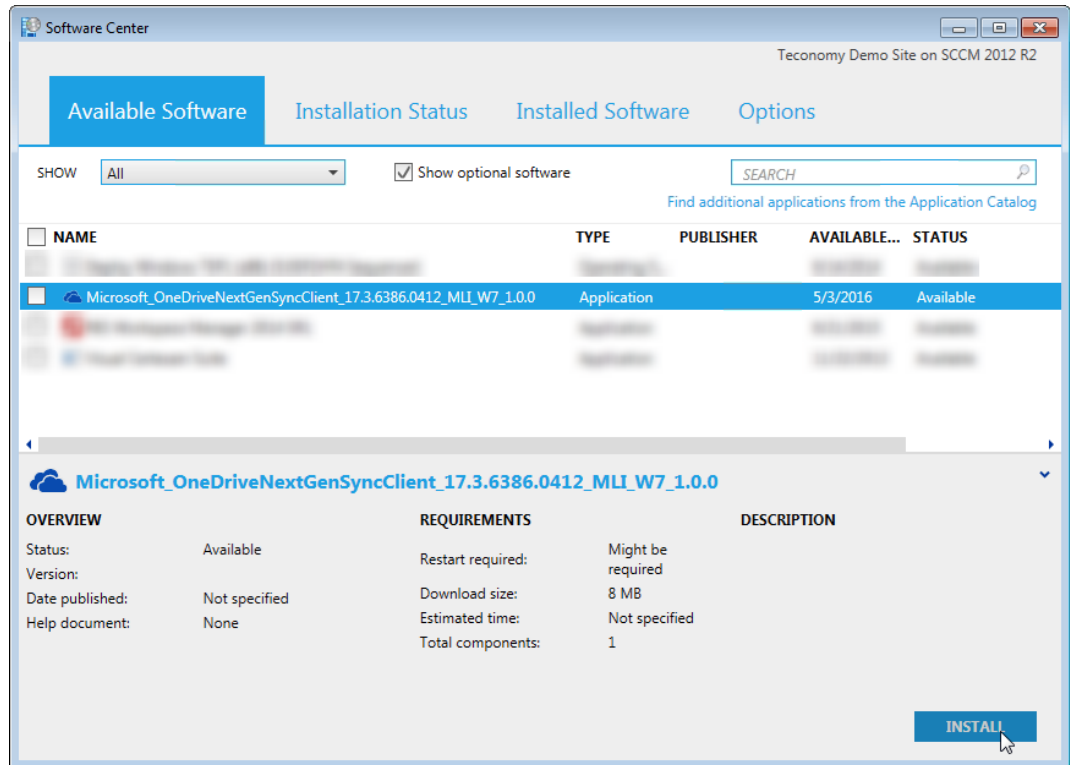
Deploy as available.



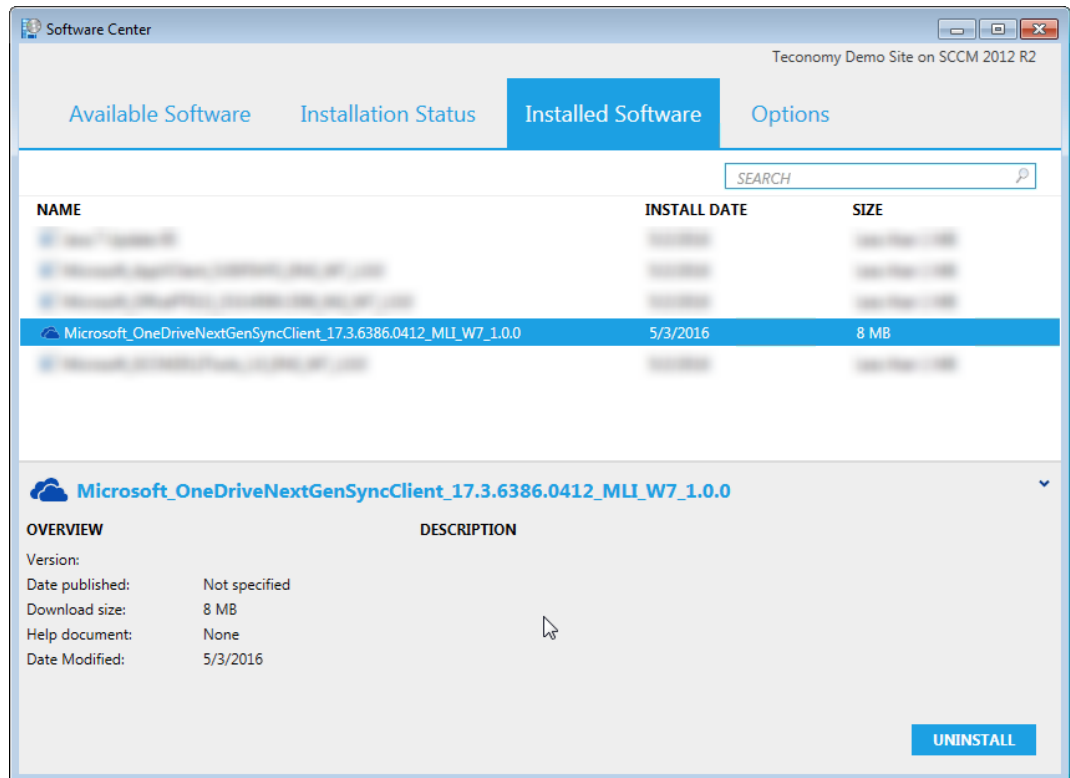
Show all notifications.

Check on a client

1. Log on to a client and go to the Software Catalog:

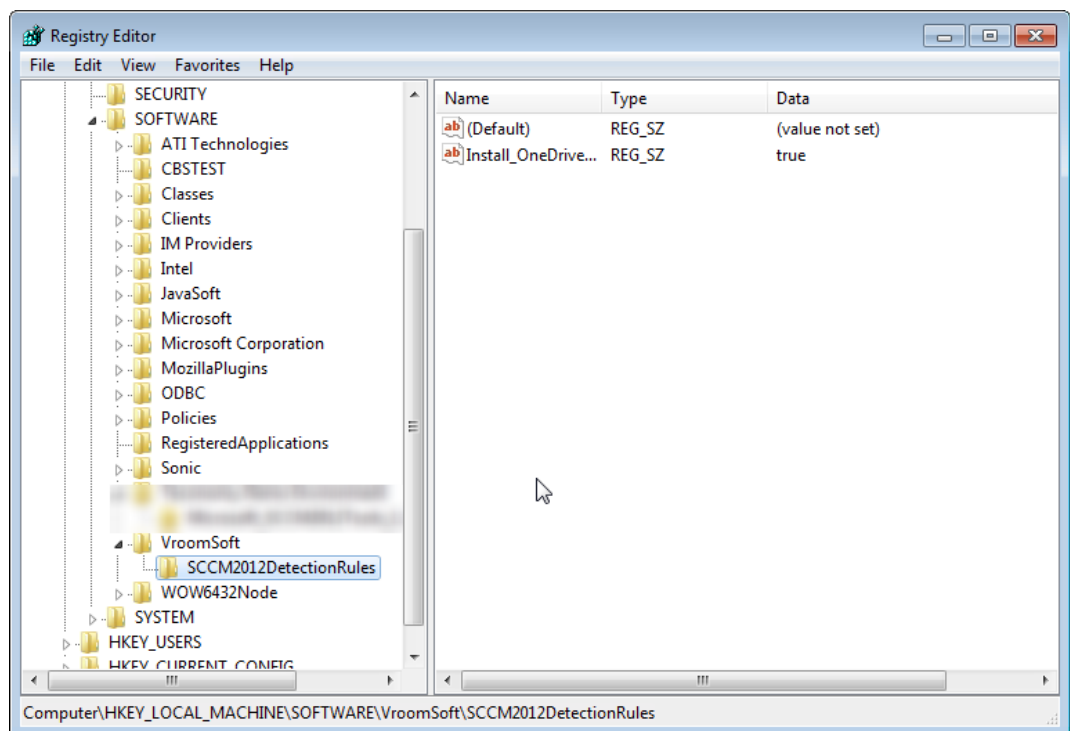


The application is visible in the software center



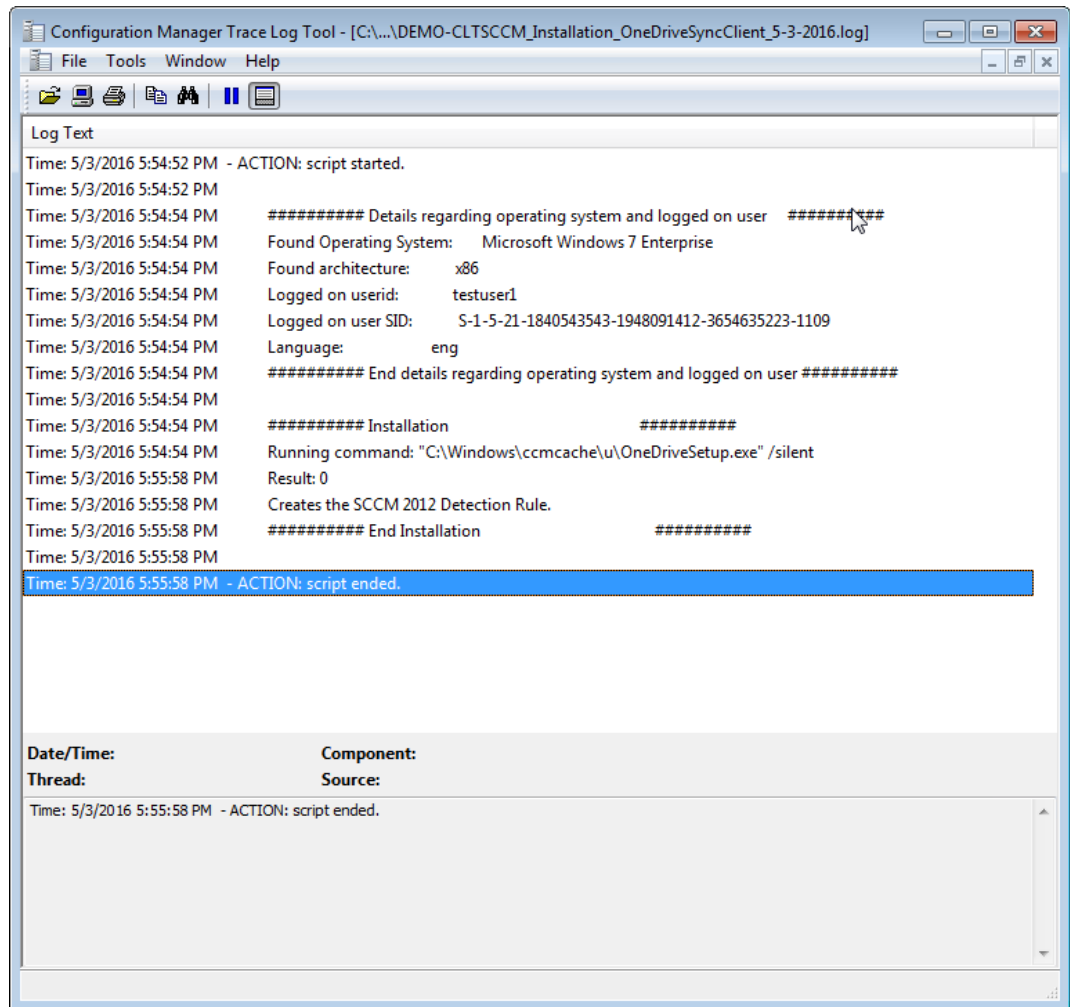
You can find the new application under installed applications.

2. You can use regedit to find the detection rule which is set via vbscript:



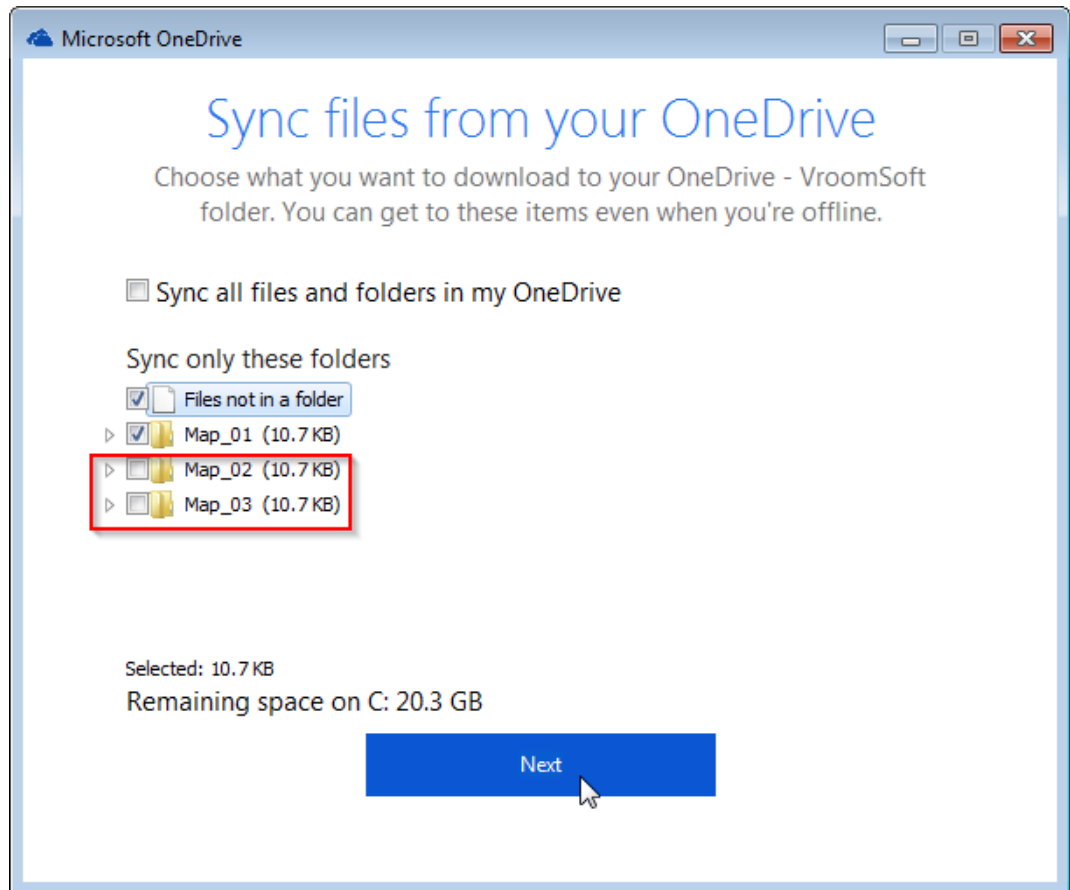
The detection rule is found in the registry

3. The installation log file (found in c:\windows\system32\logfiles):

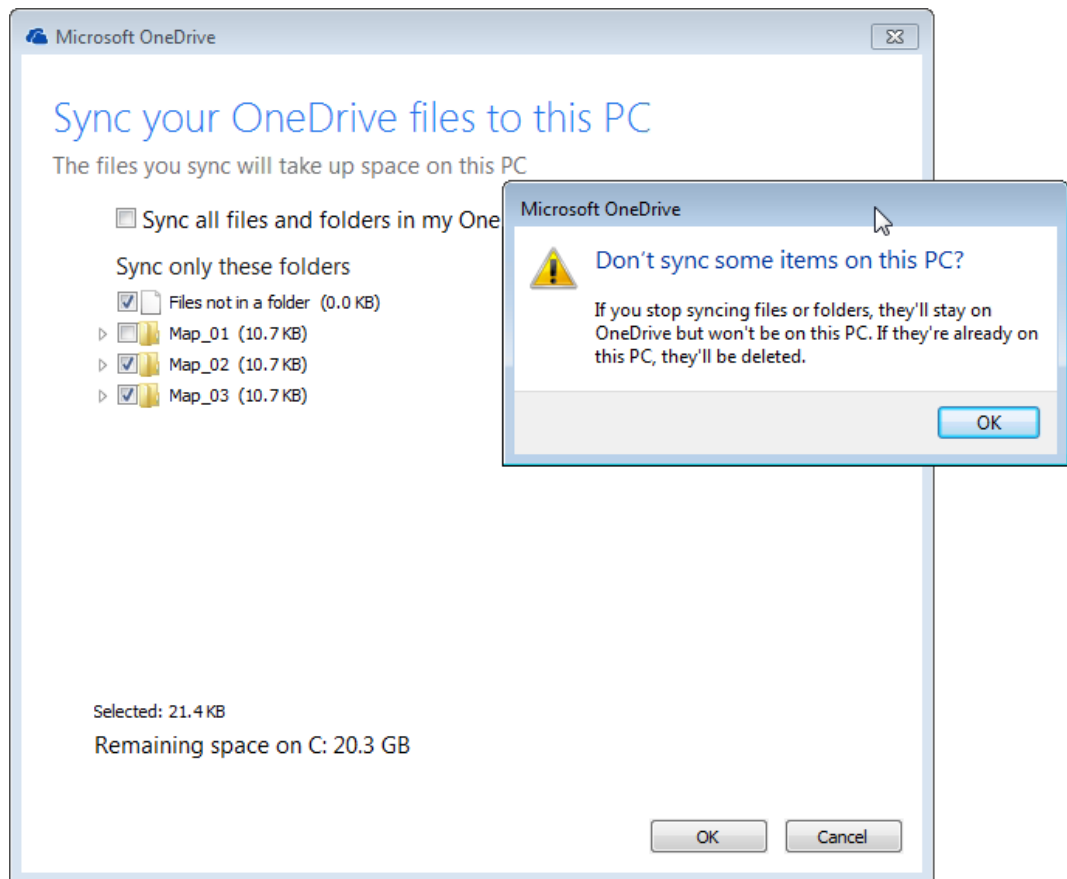


Installation log file

4. Now you can select the folders you want to synchronize or want to exclude from synchronization:



You can select the folders you want so synchronize.



Unselect the folders you do not want to synchronize.

Uninstall the application

You can uninstall the OneDrive NextGen Sync Client if you do not want to use it anymore. All the data will remain on the computer:

Software Center Teconomy Demo Site on SCCM 2012 R2

Available Software Installation Status **Installed Software** Options

SEARCH

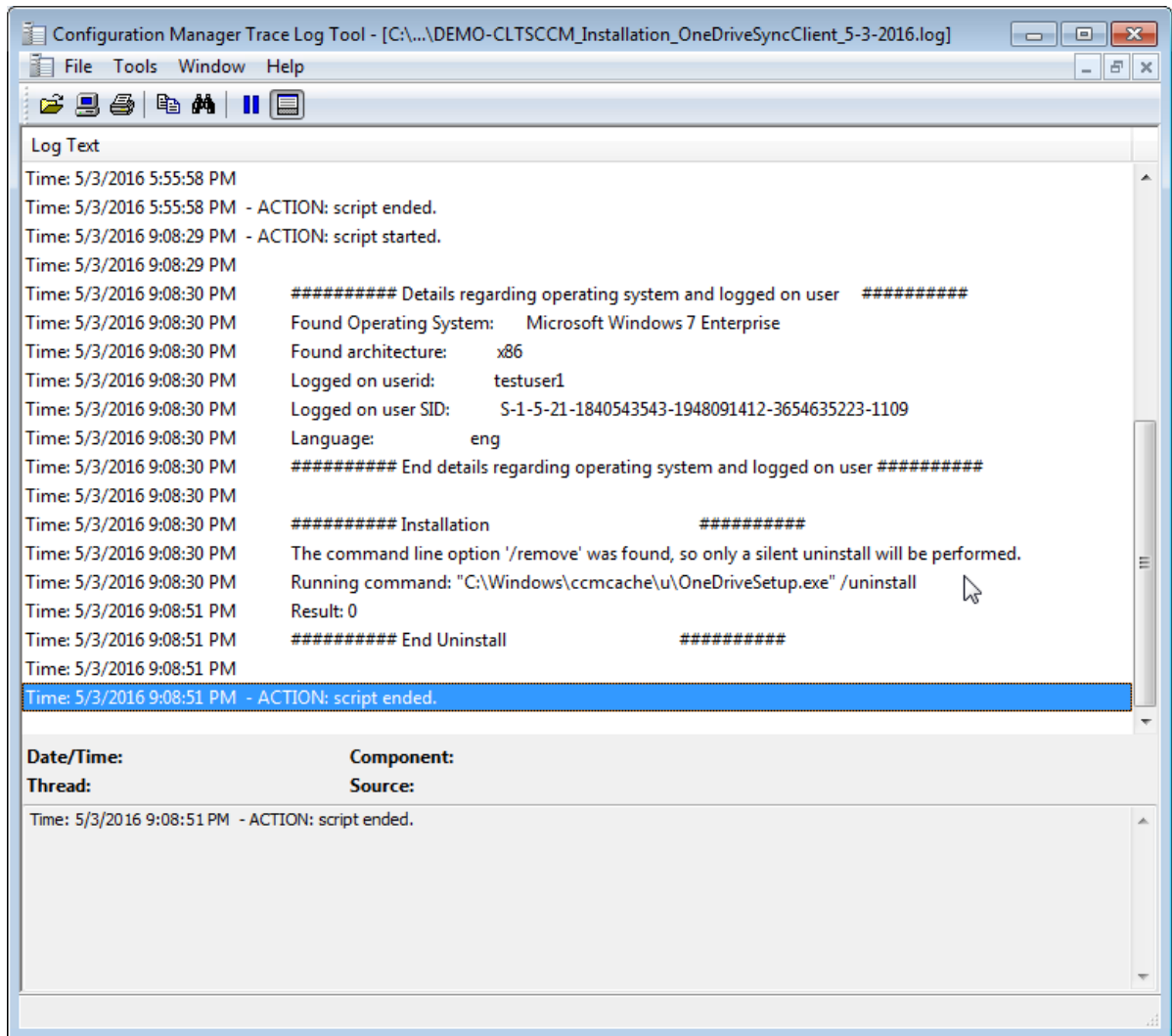
NAME	INSTALL DATE	SIZE
Microsoft_OneDriveNextGenSyncClient_17.3.6386.0412_MLI_W7_1.0.0	5/3/2016	8 MB

Microsoft_OneDriveNextGenSyncClient_17.3.6386.0412_MLI_W7_1.0.0

OVERVIEW	DESCRIPTION
Version:	
Date published:	Not specified
Download size:	8 MB
Help document:	None
Date Modified:	5/3/2016

UNINSTALL

You can uninstall the client.



The uninstall in the log file.