WINDOWS Powershell Scriting

Programme mit Hilfe von Powershell deinstallieren

Möchte man installierte Programme von der Kommandozeile oder via Script entfernen, dann bietet Microsoft dafür gleich mehrere Möglichkeiten. Leider deckt keine Methode alle Szenarien ab. So beschränkt sich etwa das Package-Management nur auf den lokalen Rechner, WMI hingegen erkennt nicht alle Programme. Bereits bei der Deinstallation von Windows-Komponenten gibt es ein unnötiges Nebeneinander von mehreren Optionen wie Remove-WindowsCapability, Remove-WindowsFeature und Disable-WindowsOptionalFeature. Hinzu kommt noch Remove-WindowsPackage aus dem DISM-Modul.Möchte man installierte Programme von der Kommandozeile oder via Script entfernen, dann bietet Microsoft dafür gleich mehrere Möglichkeiten. Leider deckt keine Methode alle Szenarien ab. So beschränkt sich etwa das Package-Management nur auf den lokalen Rechner, WMI hingegen erkennt nicht alle Programme. Bereits bei der Deinstallation von Windows-Komponenten gibt es ein unnötiges Nebeneinander von mehreren Optionen wie Remove-WindowsCapability, Remove-WindowsFeature und Disable-WindowsOptionalFeature. Hinzu kommt noch Remove-WindowsPackage aus dem DISM-Modul.

CLI-Optionen für das Deinstallieren von Anwendungen Ähnlich inkonsistent sieht es bei den Kommandozeilen-Tools für die Deinstallation von Programmen aus. Vor einigen Jahren führte Microsoft das PowerShell Package Management ein, das primär für das Hinzufügen und Entfernen von PowerShell-Modulen gedacht ist. Es eignet sich aber auch für die Deinstallation von Win32-Programmen. Im Jahr 2020 kann dann mit dem Paket-Manager winget ein weiteres Tool hinzu, das ein Jahr später die Fähigkeit zum Entfernen von Programmen erhielt. Es hat bis heute aber kein PowerShell-Interface und ist so für die Automatisierung des Paket-Managements nur bedingt geeignet. Für Store- bzw. UWP-Apps existieren mit Remove-AppxProvisionedPackage und Remove-AppxPackage zudem eigene Cmdlets. Schließlich besteht noch die Möglichkeit, Anwendungen über WMI zu deinstallieren. Dies ist der einzige der hier genannten Mechanismen, um diese Aufgabe remote zu erledigen. Programme über WMI deinstallieren Zuständig ist dafür die Klasse Win32_Product. Mit ihrer Hilfe kann man die installierte Software erst anzeigen: Get-CimInstance -Class Win32_Product -ComputerName <Remote-PC> | Format-List Hat man das fragliche Programm gesichtet, dann kann man die Liste weiter einschränken: Get-CimInstance -Class Win32_Product -ComputerName <Remote-PC> | where name -like " PowerShell*" Schickt man den Output dieses Kommandos durch eine Pipe an Get-Member, dann wird man vergeblich nach einer Uninstall-Methode suchen. Um diese zu bekommen, muss man stattdessen das ältere Get-WMIObject verwenden: $app = Get-WmiObject -Class Win32_Product -ComputerName <Remote-PC> | where name -Like "PowerShell*" Dieser Aufruf würde zum Beispiel eine veraltete Version von PowerShell 7 erfassen und das Resultat an die Variable $app zuweisen.CLI-Optionen für das Deinstallieren von Anwendungen Ähnlich inkonsistent sieht es bei den Kommandozeilen-Tools für die Deinstallation von Programmen aus. Vor einigen Jahren führte Microsoft das PowerShell Package Management ein, das primär für das Hinzufügen und Entfernen von PowerShell-Modulen gedacht ist. Es eignet sich aber auch für die Deinstallation von Win32-Programmen. Im Jahr 2020 kann dann mit dem Paket-Manager winget ein weiteres Tool hinzu, das ein Jahr später die Fähigkeit zum Entfernen von Programmen erhielt. Es hat bis heute aber kein PowerShell-Interface und ist so für die Automatisierung des Paket-Managements nur bedingt geeignet. Für Store- bzw. UWP-Apps existieren mit Remove-AppxProvisionedPackage und Remove-AppxPackage zudem eigene Cmdlets. Schließlich besteht noch die Möglichkeit, Anwendungen über WMI zu deinstallieren. Dies ist der einzige der hier genannten Mechanismen, um diese Aufgabe remote zu erledigen. Programme über WMI deinstallieren Zuständig ist dafür die Klasse Win32_Product. Mit ihrer Hilfe kann man die installierte Software erst anzeigen: Get-CimInstance -Class Win32_Product -ComputerName <Remote-PC> | Format-List Hat man das fragliche Programm gesichtet, dann kann man die Liste weiter einschränken: Get-CimInstance -Class Win32_Product -ComputerName <Remote-PC> | where name -like " PowerShell*" Schickt man den Output dieses Kommandos durch eine Pipe an Get-Member, dann wird man vergeblich nach einer Uninstall-Methode suchen. Um diese zu bekommen, muss man stattdessen das ältere Get-WMIObject verwenden: $app = Get-WmiObject -Class Win32_Product -ComputerName <Remote-PC> | where name -Like "PowerShell*" Dieser Aufruf würde zum Beispiel eine veraltete Version von PowerShell 7 erfassen und das Resultat an die Variable $app zuweisen.

Anschließend deinstalliert man das Programm mit $app.uinstall() Diese Methode funktioniert

grundsätzlich gut und eben auch remote. Ihr Nachteil besteht darin, dass sich die installierten Programme über WMI meist nicht vollständig anzeigen lassen.Anschließend deinstalliert man das Programm mit $app.uinstall() Diese Methode funktioniert grundsätzlich gut und eben auch remote. Ihr Nachteil besteht darin, dass sich die installierten Programme über WMI meist nicht vollständig anzeigen lassen.

Tools wie 7-zip tauchen gar nicht auf, andere wieder nur mit einer GUID, so dass man nicht weiß, was sich dahinter verbirgt. In diesem Fall kann man auf das Package-Management ausweichen, das aber nur lokal funktioniert. Programme mit Uninstall-Package entfernen Um die installierten Programme aufzulisten, verwendet man das Cmdlet Get-Package. Dieses wirft aber auch Standalone-Updates (msu) oder mit PowerShellGet installierte Module aus. Daher kann man den Output für herkömmliche Win32-Anwendungen so filtern: Get-Package -ProviderName Programs -IncludeWindowsInstaller Hat man das betreffende Programm in der Anzeige gefunden, dann spezifiziert man es über den Name-Parameter und übergibt es an Uninstall-Package: Get-Package -Name "7-zip*" | Uninstall-Package Get-Package bietet zusätzlich die Möglichkeit, Programme über ihre Versionsnummer zu filtern und wahlweise nur bestimmte (zum Beispiel "RequiredVersion") oder alle Versionen ("AllVersions") zu deinstallieren.

Eindeutige ID: #1008
Verfasser: n/a
Letzte Änderung: 2022-03-01 08:46