Windows Client Management AG
Software Deployment Solutions

Scripting Framework PowerShell Toolkit

Benutzerhandbuch – Version 6.0

1 Über Scripting Framework

1.1 Wie wird Scripting Framework eingesetzt?

Das Windows PowerShell Toolkit bietet Ihnen zahlreiche Funktionen, die für die Softwarepaketierung, allgemeines Scripting und die schlussendliche Bereitstellung benötigt werden. Es vereinfacht in Ihrem Unternehmen die komplexen Herausforderungen, welche Sie immer mehr in diesem Bereich antreffen. Mit diesem Toolkit können Sie ihre VBScripte, WiseScripte, Batches, etc. mit einem vielseitigen und erweiterbaren Werkzeug ersetzen und die Qualität erhöhen. Durch die Vielzahl der vorhandenen und speziell abgestimmten Funktionen sowie der Struktur, erreichen Sie eine einheitliche Paketierung und Standardisierung. Die gesamte Logik (z.B. Sprachumschaltungen, Einstellungen wie Servernamen, etc.) können dynamisch und zentral im Scripting Framework Paket hinterlegt werden. Die zur Verfügung stehenden Funktionen sind einfach und verständlich aufgebaut und können ohne PowerShell Kenntnisse erfolgreich eingesetzt werden. Diese Flexibilität ermöglicht es Ihnen, einheitliche Pakete für verschiedene Mandate oder Standorte mit abweichenden Einstellungen anzufertigen. Hybrid Pakete welche auf Server (auch Citrix) und Clients mit unterschiedlichen Einstellungen und Architekturen (x86 / x64) eingesetzt werden, sind problemlos zu realisieren. Somit können Sie in Zukunft nur noch ein einziges Paket erstellen, welches die gesamten Anforderungen abdeckt. Einzigartig ist auch die Einfachheit, wie Sie vorhandene Benutzereinstellungen innerhalb von Sekunden in einem Paket implementieren.
Alle vorhandenen Funktionen sind ebenfalls ausserhalb von einem Scripting Framework Paket im sogenannten „Scripting Modus“ verfügbar. Dadurch kann das Toolkit in verschiedenen Szenarien sehr flexibel eingesetzt werden, z.B. für dynamische App-V Pakete, welche je nach Standort andere Einstellungen benötigen.

1.2 Facts

- **Packager -** Sie wollen eine Einstellung innerhalb von einer Applikation setzen? Kein Problem, unser Packager hilft Ihnen die entsprechenden Registry Keys oder Dateien zu finden.
- **Einfach zu verwenden -** Sie benötigen keine PowerShell Kenntnisse. Eine grosse Anzahl von Funktionen, speziell für den Bereich der Paketierung und Softwarebereitstellung.
- **Qualität -** Sie erarbeiten Pakete und Skripte mit einem einheitlichen Erscheinungsbild, unabhängig der Komplexität.
- **Zukunft -** MS PowerShell ist etabliert und die Zukunft im Scripting Bereich, verwenden Sie es auch für die Softwarepaketierung!
- **Leistungsstark -** Über 60 Funktionen für Installation und Deinstallation von Anwendungen, Löschen von Dateien und Ordnern inklusiv das automatische Schliessen der aktiven Executables und offenen Handels, Schreiben von Registry Werten (32-Bit und 64-Bit), Kopieren von Dateien, Installation und De-Installation von AppV-Paketen, Shortcuts, Fonts, Services, etc. Selbstverständlich ist es auch möglich zusätzlich eigenen PowerShell Code innerhalb der Scripting Framework Pakete zu verwenden.
- **Benutzereinstellungen** - Mit dem optionalen Benutzerscript, bewältigen Sie komplexe Einstellungen sehr einfach. Schnell und mühelos können Sie Registry Key's schreiben, Dateien in das Benutzerprofil kopieren, Shortcuts erstellen, INI-File Einträge schreiben, etc. Zudem werden die Benutzereinstellungen nach der Paket Installation ohne Benutzer Log Off / Log On automatisch durch Scripting Framework auf allen auf dem Gerät angemeldeten Benutzern angewendet.
- **Dynamisch -** Das Erstellen eines einzigen Paketes, welches für verschiedene Standorte, z.B. mit unterschiedlichen Servernamen einzusetzen ist.

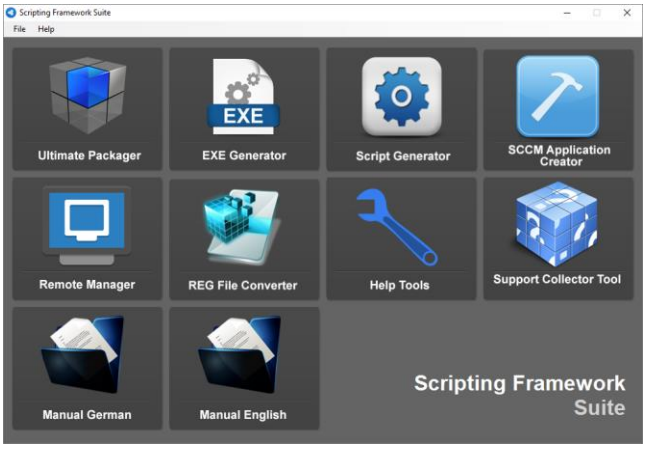
- **Usability -** Die gleichzeitige Verwendung auf Citrix und Clients, egal ob es sich um ein 32-Bit oder 64-Bit Betriebssystem handelt.
- **Unabhängigkeit -** Alle Software Pakete, die mittels Scripting Framework erstellt sind, können mit der von Ihnen eingesetzten Softwareverteilung (Hersteller unabhängig) problemlos bereitgestellt werden.
- **Integrierbar -** Scripting Framework wird unkompliziert in Ihre bestehende Umgebung integriert.
- **Einsatzerprobt -** Dieses Toolkit wird von unseren Kunden erfolgreich eingesetzt.

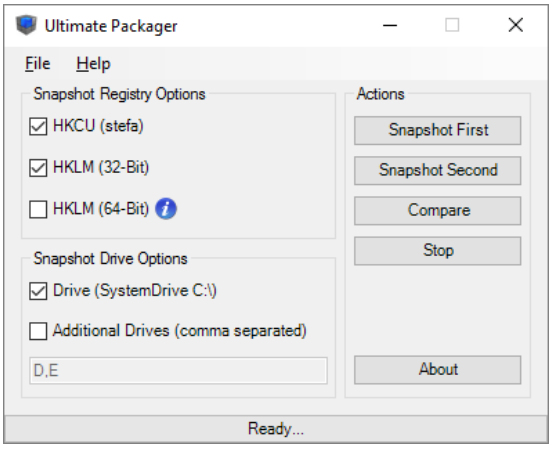
1.3 Environment Integration

- Scripting Framework ist mit allen von uns bekannten Software Deployment Lösungen kompatibel. Eine Portierung von bestehenden Scripting Framework Paketen auf ein neues Verteilsystem ist problemlos möglich und bietet Ihnen somit höchste Flexibilität.
- Optimiert für den Einsatz mit dem Microsoft System Center Configuration Manager (SCCM).
- Es werden Standard Return Codes (analog Windows Installer) zurückgegeben
- Integrierbar in App-V Pakete mittels Scripting Framework "Scripting Mode" (Scripting and Embedded Scripting for AppV 5.0)

1.4 Scripting Framework Suite

Die Scripting Framework Suite beinhaltet alle Werkzeuge, welche nebst der Script Engine zur Verfügung stehen und Sie optimal unterstützen. Die Suite beinhaltet zum Beispiel den Ultimate Packager und den Script Generator mit welchem Sie schnell und einfach Software-Pakete auf Basis der Scripting Framework Funktionen erstellen.





1.5 Technische Anforderungen

- Microsoft .Net Framework 4.8
- Microsoft PowerShell 5.x

1.6 Optionale Anforderungen

- Ein Share für zentrale Konfigurationsdateien von Paketen, sofern diese Funktionalität von Scripting Framework verwendet wird.
- Ein Benutzer Account welcher auf den Share mit den Konfigurationsdateien über „Read“ Berechtigungen verfügt.

2 Verfügbare Funktionen

2.1 SF-AppvInstall

Übersicht:
Installation von einem Microsoft App-V Package.

Beschreibung:
Voraussetzung für die Funktion ist der Microsoft App-V Client.

Syntax:
```
SF-AppvInstall "Path" "DynamicDeploymentConfiguration" "DynamicUserConfiguration" -Switches
```

Switches:

-Global	Das App-V Paket wird global hinzugefügt und veröffentlicht
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Installs an App-V package without a dynamic deployment configuration (xml)
SF-AppvInstall "%_PkgSource%\Setup\Example.appv" "" ""

# Installs an App-V package with a dynamic deployment configuration (xml)
SF-AppvInstall "%_PkgSource%\Setup\Example.appv" "%_PkgSource%\Setup\DynamicDeploymentConfiguration.xml" ""
```

2.2 SF-AppvUninstall

Übersicht:
Deinstallation von einem App-V Package.

Beschreibung:
Voraussetzung für die Funktion ist der Microsoft App-V Client.

Syntax:

```
SF-AppvUninstall "PackageId" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:

```
# Uninstall an App-V package
SF-AppvUninstall "03c5ed5d-3c1c-4518-92c9-d1175c5454fb"
```

2.3 SF-AppxInstall

Übersicht:
Installation von einem App-Paket (.appx).

Beschreibung:
Fügt einem Windows-Image ein App-Paket (.appx) hinzu, dass es für jeden neuen Benutzer installiert wird.

Syntax:

```
SF-AppxInstall "PackagePath" "DependencyPackagePath" -Switches
```

Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:

```
# Installs an AppX package
SF-AppXInstall "%_PkgSource%\Setup\Microsoft.Whiteboard_20.10615.5289.0_x64__8wekyb3d8bbwe.appx"

# Installs an AppX package with Dependency Packages
SF-AppXInstall "%_PkgSource%\Setup\E046963F.LenovoCompanion_10.2009.18.0_neutral___k1h2ywk1493x8.AppxBundle" -
DependencyPackagePath "%_PkgSource%\Setup\Microsoft.NET.Native.Framework.1.3_1.3.24211.0_x64__8wekyb3d8bbwe.Appx",
"%_PkgSource%\Setup\Microsoft.NET.Native.Framework.1.3_1.3.24211.0_x86__8wekyb3d8bbwe.Appx",
"%_PkgSource%\Setup\Microsoft.NET.Native.Runtime.1.4_1.4.24201.0_x64__8wekyb3d8bbwe.Appx",
"%_PkgSource%\Setup\Microsoft.NET.Native.Runtime.1.4_1.4.24201.0_x86__8wekyb3d8bbwe.Appx"
```

2.4 SF-AppxUninstall

Übersicht:
Entfernt ein App-Paket (.appx).

Beschreibung:
Entfernt App-Pakete (.appx) aus einem Windows-Image.

Syntax:
```
SF-AppxUninstall "DisplayName" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Uninstalls an AppX package
SF-AppxUninstall "Microsoft.Whiteboard"
```

2.5 SF-CD

Übersicht:
Wechseln des Verzeichnisses innerhalb von PowerShell.

Beschreibung:
Keine

Syntax:
```
SF-CD "Path"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Change directory
SF-CD "%_ProgramFiles32%"
```

2.6 SF-Copy

Übersicht:
Kopieren von Dateien und Ordnern.

Beschreibung:
Die Dateien welche in Verwendung sind, werden mittels einer Windows API mit dem nächsten Reboot automatisch ersetzt. Die Funktion verwendet RoboCopy im Hintergrund, die entsprechende Log Datei des letzten Kopiervorgangs befindet sich im Scripting Framework Log Ordner.

Syntax:
```
SF-Copy "Source" "Destination" –Switches
```

Verfügbare Switches:

-Retries	Gibt die Anzahl der Wiederholungen für fehlgeschlagene Kopien an. Der Standardwert ist 3.
-RetriesTime	Gibt die Wartezeit zwischen den Wiederholungsversuchen in Sekunden an. Der Standardwert ist 10.
-Changed	Überschreibt alle Dateien, ob nun neuer oder älter.
-LocalUserProfiles	Kopiere eine Datei in alle lokalen Benutzerprofile
-Newer	Überschreibt nur die Dateien welche neuer sind.
-NoErrors	Fehlermeldungen werden ignoriert und der Script fortgesetzt.
-SelfRegister	Verwenden Sie diese Aktion, um alle kopierten .OCX- und .DLL-Dateien selbst zu registrieren.
-UseVersion	Verwendet die Dateiversion anstelle des Datums, sofern das Attribut zur Verfügung steht und nicht 0.0.0.0 ist.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Copy file
SF-Copy "%_Windows%\Notepad.exe" "%_SystemDrive%\Example" -Newer

# Copy file (with versions comparison)
SF-Copy "%_PkgSource%\Setup\WindowsSystem32" "%_WindowsSystem32%" -Newer -UseVersion

# Copy folder (changed)
SF-Copy "%_PkgSource%\Setup\ProgramFiles32" "%_ProgramFiles32%" -Changed

# Copy folder only for a 32-bit OS
SF-Copy "%_PkgSource%\Setup\ProgramFiles32" "%_ProgramFiles32%" -Changed –x86
```

```
# Copy the file Example.txt to all local user profiles
SF-Copy "%_PkgSource%\Setup\Example.txt" "AppData\Roaming" -Changed -LocalUserProfiles
```

ACHTUNG: Ein Rename von einer Datei ist mit dieser Funktion nicht möglich, verwenden Sie dazu die Funktion SF-Rename. Nicht funktionierendes Beispiel:

```
SF-Copy "%_Windows%\Notepad.exe" "%_SystemDrive%\Example\New.exe" -Newer
```

2.7 SF-Crypt

Übersicht:

Verschlüsseln von einem String.

Beschreibung:

Um den verschlüsselten String zu entschlüsseln, steht der Befehl SF-Decrypt zur Verfügung.

Syntax:

```
SF-Crypt "String"
```

Verfügbare Switches:

Keine	

Beispiele:

```
# Stores the crypted string in the PowerShell variable $CryptString
$CryptString = SF-Crypt "SecurePassword"
```

Achtung: Die Funktionen SF-Crypt und SF-Decrypt sind dafür gedacht, um Strings wie eine Seriennummern zu crypten, damit solche Informationen nicht sofort als Klartext im Paket sichtbar sind. Verschlüsseln Sie aus Sicherheitsgründen jedoch keine Kennwörter von Benutzern, usw., da diese Strings schlussendlich als Klartext über PowerShell ausgegeben werden können.

2.8 SF-Decrypt

Übersicht:
Entschlüsselt einen String.

Beschreibung:
Entschlüsselt einen String, welcher zuvor mit SF-Crypt verschlüsselt wurde.

Syntax:
```
SF-Decrypt "String"
```

Wird die PowerShell Variable mit dem Namen $DecryptString deklariert, so ist in der Log Datei der entschlüsselte String nur mit Sternchen (*******) sichtbar.

Achtung: Die Variabel muss mit *$DecryptString* benannt werden, ansonsten wird das Passwort in Klartext dargestellt!

Verfügbare Switches:

Keine	

Beispiele:
```
# Decrypt the string and stores it in a PowerShell variable. Five simultaneous variables are currently supported
$DecryptString = SF-Decrypt "VtDlxXqGsib8oZ6ErSFgJw=="
$DecryptString2 = SF-Decrypt "VtDlxXqGsib8oZ6ErSFgJw=="
$DecryptString3 = SF-Decrypt "VtDlxXqGsib8oZ6ErSFgJw=="
$DecryptString4 = SF-Decrypt "VtDlxXqGsib8oZ6ErSFgJw=="
$DecryptString5 = SF-Decrypt "VtDlxXqGsib8oZ6ErSFgJw=="

# Decrypt the string, if input is a Scripting Framework variable
$DecryptString = SF-Decrypt (SF-Variables "%ExamplePassword%")

# Create the local user ScriptingFramework with the password in $DecryptString
SF-Run "%_WindowsSystem32%\net.exe" "user ""ScriptingFramework"" $DecryptString /Add" -Wait

# Write textfile with password and open in notepad
SF-Textfile "%_Temp%\Decrypt.txt" "$DecryptString" -Add -IfMissing
SF-Run '%_Windows%\Notepad.exe' '"%_Temp%\Decrypt.txt"' -Show
```

Achtung: Die Funktionen SF-Crypt und SF-Decrypt sind dafür gedacht, um Strings wie eine Seriennummern zu crypten, damit solche Informationen nicht sofort als Klartext im Paket sichtbar sind. Verschlüsseln Sie aus Sicherheitsgründen jedoch keine Kennwörter von Benutzern, usw., da diese Strings schlussendlich als Klartext über PowerShell ausgegeben werden können.

2.9 SF-Delete

Übersicht:
Löschen von einer Datei.

Beschreibung:
Gegebenenfalls werden automatisch die offenen Handles geschlossen.

Syntax:
```
SF-Delete "Path" -Switches
```

Verfügbare Switches:

-DisableTaskkill	Es erfolgt kein automatischer Taskkill auf den Prozess.
-LocalUserProfiles	Löscht eine Datei rekursiv aus allen lokalen Benutzerprofilen (siehe Beispiele)
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Remove file
SF-Delete "%_SystemDrive%\Example\File.txt"

# Remove example.log from all user profiles
SF-Delete "AppData\Roaming\Example.txt" -LocalUserProfiles

Resultat Beispiel (LocalUserProfiles):
20:03:2017-08:52:57    INFO: ===== Delete ================================================
20:03:2017-08:52:57    INFO:      Path: C:\Users\Public\AppData\Roaming\Example.txt
20:03:2017-08:52:57    INFO:      Path: C:\Users\Username1\AppData\Roaming\Example.txt
20:03:2017-08:52:57    INFO:      Path: C:\Users\Username2\AppData\Roaming\Example.txt
20:03:2017-08:52:57    INFO:      Path: C:\Users\Username3\AppData\Roaming\Example.txt
```

2.10 SF-DriverInstall

Übersicht:
Installiert einen Gerätetreiber.

Beschreibung:
Installiert einen einzelnen Gerätetreiber oder alle Gerätetreiber aus einem Ordner inklusive der Unterordner.

Syntax:
```
SF-DriverInstall "Path" –Switches
```

Verfügbare Switches:

-NoErrors	Die Exit Codes werden ignoriert und es wird mit dem Script fortgefahren.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Installs a single driver
SF-DriverInstall "%_PkgSource%\Setup\Driver\Sp40_7_x64.inf"

# Installs all device drivers from a folder including subfolders
SF-DriverInstall "%_PkgSource%\Setup\Drivers"
```

2.11 SF-DriverUninstall

Übersicht:
Deinstalliert einen Gerätetreiber.

Beschreibung:
Deinstalliert einen Treiber aus dem Windows Driver Store.

Syntax:
```
SF-DriverUninstall "Path" "SearchString" -Switches
```

Verfügbare Switches:

-NoErrors	Die Exit Codes werden ignoriert und es wird mit dem Script fortgefahren.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Uninstalls a specific driver from the DriverStore (oem*.inf), if the search string was found
SF-DriverUninstall "%_Windows%\INF" "MPC3504/C3004E-23C/MPC6004/C5504/C4504E-23C PCL 6 Printer Driver"

# Uninstalls a single driver by path
SF-DriverUninstall "%_Windows%\INF\oem43.inf"

# Uninstalls a single driver by name
SF-DriverUninstall "oem43.inf"
```

2.12 SF-Exit

Übersicht:
Zum vorzeitigen Beenden eines Paketes (Fehler).

Beschreibung:
Erstellt einen entsprechenden Scripting Framework Log Eintrag und beendet danach das Script mit dem Standard Return Code 1603. Es sind benutzerdefinierte Return Codes möglich.

Syntax:
```
SF-Exit "Message" "ReturnCode" -Switches
```

Verfügbare Switches:

-Severity	Definiert den Schweregrad der Nachricht. Folgende Werte sind gültig: Custom, Error, Info, Warning und Title
-ShowBalloonTip	Zeigt einen BalloonTip in der unteren rechten Ecke des Bildschirms an. Der Switch muss im aktuellen Benutzerkontext verwendet werden (es wird kein BalloonTip angezeigt, sofern der Befehl in einem anderen Kontext ausgeführt wird).

Beispiele:
```
# Sample 1 - Exit Scripting Framework with 1603
SF-Exit "My Example Message"

# Sample 2 - Exit Scripting Framework (Check Condition)
SF-LoadVariables "%c_ConfigPath%\%_PkgCFGName%.cfg" "CommonClientSettings"
If ((SF-Variables "%_VarPrefix-Example%") -eq "") {SF-Exit "Value for Variable <Example> must be set in %_PkgCFGName%.cfg"}

# Sample 3 - Exit Scripting Framework with a custom return code
SF-Exit "My Example Message" "1610" -Severity Warning
```

2.13 SF-ExtractArchive

Übersicht:
Extrahiert eine Zip Datei.

Beschreibung:
Extrahiert ein Zip Archiv an die angegebene Destination.

Syntax:
```
SF-ExtractArchive "Path" "Destination" -Switches
```

Verfügbare Switches:

-CacheLocal	Vor dem entpacken findet eine automatische Überprüfung statt, ob sich die Datei auf einem Netz- oder auf einem lokalen Laufwerk befindet. Sollte es sich nicht um ein lokales Laufwerk handeln, wird die Datei vor dem entpacken automatisch lokal gecached und aus diesem entpackt.
-Log	Das Entpacken wird detailliert im Scripting Framework Log angezeigt.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Extract archive file
SF-ExtractArchive "%_PkgSource%\Setup\ProgramFiles64.zip" "%_ProgramFiles64%\Example" -x64 -CacheLocal
```

2.14 SF-File

Übersicht:
Überprüft ob eine Datei oder ein Ordner existiert.

Beschreibung:
Die Funktion gibt den Wert true oder false zurück.

Syntax:
```
SF-File "Path"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Check file
If ((SF-File "%_Windows%\Notepad.exe") -eq $True) {
    SF-Log "File found"
    SF-Run "%_Windows%\Notepad.exe" -Show
}
Else {
    SF-Log "File not found"
}

# Check folder
If ((SF-File "%_Windows%") -eq $True) {
    SF-Log "Folder found"
}
```

2.15 SF-FileSearch

Übersicht:
Suche nach einer Datei.

Beschreibung:
Die Funktion sucht nach einer Datei und gibt das Ergebnis als Array zurück.

Syntax:
```
SF-FileSearch "Path" "Filename"
```

Verfügbare Switches:

-Depth x	Kontrollieren Sie die Tiefe der Rekursion.

Beispiele:
```
# Example 1: Search the drive C:\ for the file Example.txt and write the path of the found files into the log
$SearchResult = SF-FileSearch "%_SystemDrive%" "Example.txt"

ForEach ($Result in $SearchResult) {
    SF-Log $Result
}

# Example 2: Search the drive C:\ for the file Example.txt and write the path of the found files into the log - With the
maximum folder depth of 2
$SearchResult = SF-FileSearch "%_SystemDrive%" "Example.txt" -Depth 2

ForEach ($Result in $SearchResult) {
    SF-Log $Result
}
```

2.16 SF-FileVersion

Übersicht:
Liest die Versionsnummer von einer Datei aus und gibt diese zurück. Mit der Funktion lassen sich zusammen mit den PowerShell Vergleichsoperatoren (-eq, -ne, -lt, -le, -gt und -ge) entsprechende Abfragen erstellen.

Beschreibung:
Die Funktion gibt als Wert die Dateiversion zurück.

Syntax:
```
SF-FileVersion "Path" –Switches
```

Verfügbare Switches:

-AsString	Gibt die Version mit dem Datentyp [String] anstelle von [Version] zurück.

Beispiele:
```
# Checks the file version of notepad
If ((SF-FileVersion "%_Windows%\Notepad.exe") -gt "6.1.7600.16300") {
    SF-Run "%_Windows%\Notepad.exe" -Show
}
Else {
    SF-Log "Version is less than indicated" -Severity Error
}
```

2.17 SF-FontInstall

Übersicht:
Installiert eine einzelne Schriftart oder alle in einem Folder.

Beschreibung:
Folgende Typen werden unterstützt (.fon, .fnt, .ttf, .ttc, .otf, .pfm).

Syntax:

```
SF-FontInstall "Path"
```

Verfügbare Switches:

-CurrentSessionLoad	Mit diesem Switch wird die Schriftart nur temporär für den ausführenden Benutzer geladen. Nach einem Logoff ist die Schriftart nicht mehr vorhanden. Dieser Switch kann auf Terminal Servern verwendet werden, um lizenzpflichtige Schriften zu managen. Normalerweise wird die Funktion im Scripting Framework «Scripting Mode» zusammen mit einem Share und aktiviertem Access Based Enumeration verwendet. Bei Fragen steht Ihnen die Windows Client Management AG gerne zur Verfügung.

Beispiele:

```
# Install a single font
SF-FontInstall "%_PkgSource%\Setup\Example.ttf"

# Install all fonts in folder
SF-FontInstall "%_PkgSource%\Setup"
```

2.18 SF-FontUninstall

Übersicht:
Deinstallation von einer Schriftart.

Beschreibung:
Die Schrift wird über den Dateinamen deinstalliert. Anstelle des Dateinamen besteht die Möglichkeit ein Ordner anzugeben, somit werden alle in diesem Ordner gefunden Schriften deinstalliert.

Syntax:

```
SF-FontUninstall "Path"
```

Verfügbare Switches:

Keine	

Beispiele:

```
# Uninstall a single font
SF-FontUninstall "Example.ttf"

# Uninstall all fonts in folder
SF-FontUninstall "%_PkgSource%\Setup"
```

2.19 SF-GroupMembership

Übersicht:
Überprüfen ob ein Benutzer Mitglied von einer spezifischen Active Directory Gruppe ist, oder ob eine AD Gruppe einen Benutzer enthält.

Beschreibung:
Die Funktion gibt den Wert true oder false zurück. Bei einem Funktionsaufruf ohne den Parameter *MemberOf*, wird nur der Array *$GroupMembership* erstellt. Durch den Array können mehrere Abfragen, ohne mehrmaligen AD Zugriff realisiert werden. Siehe Beispiele:

Syntax:

```
SF-GroupMembership "Object" "MemberOf"
```

Verfügbare Switches:

Keine	

Beispiele:

```
# Check if the user ExampleUser a member from the group ExampleGroup
# Variables can be used such as %_User%
If ((SF-GroupMembership "ExampleUser" "ExampleGroup") -eq $True) {
    SF-Log "The user is a member"
}

# Create the array $GroupMembership and check for group membership
$GroupMembership = SF-GroupMembership "%_User%"

If ($GroupMembership -contains 'ExampleGroup') {
    SF-Log "The user is a member of ExampleGroup"
}
ElseIf ($GroupMembership -contains 'ExampleGroup2') {
    SF-Log "The user is a member of ExampleGroup2"
}
Else {
    SF-Log "The user is not a member"
}
```

2.20 SF-INIRead

Übersicht:
Lesen von einer Datei im INI-Format.

Beschreibung:
Der ausgelesene Wert wird entsprechend in der Registry als Scripting Framework Variabel gespeichert.

Syntax:
```
SF-INIRead "Path" "Section" "Name" "VariableName"
```

Verfügbare Switches:

-Machine	Speichert die Machine bezogene Variabel unter HKLM\Software\ScriptingFramework\Variables
-Script	Erstellt eine temporäre PowerShell Variabel
-User	Speichert die User bezogene Variabel unter HKCU\Software\ScriptingFramework\Variables
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
ExampleFile.ini
[Section]
Name=Example

```
# Example 1 - Generate the Scripting Framework variable INIRead_Var with the value „Example“
SF-INIRead "%_SystemDrive%\Example\ExampleFile.ini" "Section" "Name" "INIRead_Var"

# Example 2 - Read INI File and Check the Value
SF-INIRead "%_Windows%\win.ini" "Mail" "CMCDLLNAME32" "New_IniReadVar"
If ((SF-Variables "%New_IniReadVar%") -ne "mapi32.dll") {
    SF-Exit "Error occures, the value is not equal mapi32.dll"
}
```

2.21 SF-INIWrite

Übersicht:
Erstellen, editieren und löschen von Werten in einer Datei im INI Format.

Beschreibung:
Keine

Syntax:
```
SF-INIWrite "Path" "Section" "Name" "Value"
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Write a value to the file ExampleFile.ini
SF-INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Section" "Name" "Example"
```

Output ExampleFile.ini:
[Section]
Name=Example

```
# Delete value
SF-INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Section" "Name" "{nil}"

# Delete section
SF-INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Section" "{nil}" "{nil}"
```

2.22 SF-Installed

Übersicht:
Prüfen ob ein Produkt auf dem System installiert ist.

Beschreibung:
Für die Suche kann der DisplayName oder ein MSI ProductCode verwendet werden. Zusätzlich wird auch die Suche nach einem SF Package Identifier unterstützt. Die Funktion gibt true zurück sofern die Software gefunden wurden, ansonsten false.

Syntax:

```
SF-Installed "Product"
```

Verfügbare Switches:

Keine	

Beispiele:

```
#  Checks if the product is installed with the specified MSI ProductCode
If ((SF-Installed "{02382870-19C7-3ACD-BBAE-F6E3760947DC}") -eq $True) {
    SF-Log "The product is installed"
}

# Checks if the product is installed with the specified DisplayName
If ((SF-Installed "Citrix Receiver 4.6") -eq $True) {
    SF-Log "The product is installed"
}


# Checks if a product with the DisplayName like Citrix is installed (Using Wildcards)
If ((SF-Installed "Citrix*") -eq $True) {
    SF-Log "The product is installed"
}

# Check for a Scripting Framework package
If ((SF-Installed "Free_PDF_Writer_414_MUI_UNV_01") -eq $True)
{
    SF-Log "The package is installed"
}
```

2.23 SF-Language

Übersicht:
Für das Ermitteln der Sprache innerhalb von einem MUI Paket.

Beschreibung:
Die Funktion gibt den entsprechenden Sprachcode in der Variabel *_PkgLang* zurück, welcher anschliessend für die Installation oder Konfiguration verwendet werden kann.

Sprachcodes Beispiele:

Sprache	Code
Deutsch	1031
Englisch	1033
Französisch	1036
Italienisch	1040

Case 1 – Die Verwendung der Funktion in einem Paket:
Beim Aufruf der Funktion wird als erstes überprüft, in welchem Kontext die Installation durchgeführt wird. Bei einer Installation (Install.ps1, Uninstall.ps1), wird die gesetzte Default Sprache des Clients (OS) ermittelt. Anschliessend findet die Prüfung statt, ob es sich dabei um eine verfügbare Sprache handelt. Falls die ausgelesene OS Sprache mit den verfügbaren Sprachen übereinstimmt, so wird die Variabel *_PkgLang* mit dem entsprechenden Wert erstellt. Sollte die Client Sprache nicht mit einer unterstützten Sprache übereinstimmen, wird die *FallbackLanguage* verwendet.

Bei einem Aufruf der Funktion im User Script „InstallUser.ps1“, wird anstelle der Default Sprache des Clients (OS) die aktuelle Sprache des angemeldeten Benutzers ermittelt.

Case 2 - Übersteuern der Sprache per Command Line Parameter:
Die Funktion kann per Command Line Parameter über die Install.exe des Paketes übersteuert werden:

```
Install.exe -StartLanguage "1031"
```

In diesem Fall führt die Funktion SF-Language keine Erkennung der Sprache aus und in der Variabel _PkgLang befindet sich automatisch der als Command Line Parameter übergebene Sprachcode. Eine weitere Überprüfung findet nicht statt.

Case 3 - Sprache wird per Konfiguration fixiert:
Sofern für die Überprüfung nicht die Default Sprache des OS ermittelt werden soll, besteht die Möglichkeit die Sprache per Registry Eintrag zu fixieren. Somit wird jeweils ermittelt, ob die fixierte Sprache zur Verfügung steht, ansonsten wird die *FallbackLanguage* verwendet. Die Konfiguration wird in der Regel nur im Zusammenhang mit einem Termin Server verwendet. Konfiguration:

Schlüssel: HKEY_LOCAL_MACHINE\SOFTWARE*Wow6432Node*\ScriptingFramework\Config
Name: ForceMachineLanguage
Value: 1031
Typ: REG_SZ

Syntax:

```
SF-Language "AvailableLanguages" "FallbackLanguage"
```

Verfügbare Switches:

Keine	

Beispiele:

```
# Language (Example 1)
SF-Language "1031,1033,1036,1040" "1033"

# Installation Reader
SF-MSIInstall "%_PkgSource%\Setup\AcroRead.msi" "TRANSFORMS='%_PkgSource%\Setup\%_PkgLang%.mst' ALLUSERS='1'
EULA_ACCEPT='YES' SUPPRESS_APP_LAUNCH='YES' /qn /norestart"


# Language (Example 2)
SF-Language "1031,1033,1036" "1033"

# DEU
If ((SF-Variables "%_PkgLang%") -eq "1031") {
   SF-Register32 "HKey_Local_Machine" "Software\Example" "Language" "de-DE" "REG_SZ"
}

 # ENU
ElseIf ((SF-Variables "%_PkgLang%") -eq "1033") {
    SF-Register32 "HKey_Local_Machine" "Software\Example" "Language" "en-US" "REG_SZ"
 }

 # FRA
ElseIf ((SF-Variables "%_PkgLang%") -eq "1036") {
   SF-Register32 "HKey_Local_Machine" "Software\Example" "Language" "fr-FR" "REG_SZ"
}
```

2.24 SF-LoadVariables

Übersicht:
Laden von Scripting Framework Variablen aus einer Datei im INI-Format.

Beschreibung:
Die geladenen Variablen werden in die Registry als Scripting Framework Variablen gespeichert und können anschliessend im Script verwendet werden. Die CFG Datei wird automatisch zweistufig gesucht, für Details siehe Laden von Paketvariablen für dynamische Pakete.

Syntax:
```
SF-LoadVariables "Path" "Section" -Switches
```

Verfügbare Switches:

-NoErrors	Fehlermeldungen werden ignoriert und der Script fortgesetzt. Dieser Switch wird in der Regel im „Scripting Mode“ verwendet. Dadurch läuft der Script weiter, sollte sich der Client nicht im Firmennetzwerk befinden.
-NoVarPrefix	Per Default wird dem geladenen Variablenamen automatisch ein Prefix *a_PkgIdentifier*-VariabelName hinzugefügt. Mit diesem Parameter lässt sich das automatische hinzufügen von einem Prefix (Angaben aus dem Package.xml) unterbinden.
-ShowBalloonTip	Zeigt einen BalloonTip in der unteren rechten Ecke des Bildschirms an, sofern ein Fehler auftritt. Der Switch muss im aktuellen Benutzerkontext verwendet werden (es wird kein BalloonTip angezeigt, sofern der Befehl in einem anderen Kontext ausgeführt wird). Dieser Switch wird üblicherweise nur im Scripting Mode verwendet.
-User	Die Variablen in den Benutzerkontext laden (HKCU).

Beispiele:
Inhalt der CFG Datei:
; === Client Settings ==
[CommonClientSettings]
ServerLocation=https://xen.citrix.net

```
# Load Variables
SF-LoadVariables "%c_ConfigPath%\%_PkgCFGName%.cfg" "CommonClientSettings"
```

In der Registry werden die geladenen Scripting Framework Variablen aus dem CFG automatisch mit dem entsprechenden Prefix (a_ + Identifier aus dem Package.xml) abgespeichert (das *a_* steht für Application):
a_Citrix_Receiver_Enterprise_13_MUI_UNV-ServerLocation

Um die Variabel „ServerLocation“ ohne den kompletten Variablennamen inkl. Prefix ausschreiben zu müssen, kann folgende Variante für den Zugriff verwendet werden:

```
SF-Run "%_PkgSource%\Setup\CitrixReceiverEnterprise.exe" "/silent /noreboot SERVER_LOCATION=%_VarPrefix-ServerLocation%" -Wait
```

Durch die Verwendung von *_VarPrefix-* sind im Falle einer Anpassung im Package.xml, keine Änderungen im Script nötig (dies ist die empfohlene Variante). Das Ansprechen ist jedoch auch über den kompletten Variablenamen möglich:

```
SF-Run "%_PkgSource%\Setup\CitrixReceiverEnterprise.exe" "/silent /noreboot
SERVER_LOCATION=%a_Citrix_Receiver_Enterprise_13_MUI_UNV-ServerLocation%" -Wait
```

2.25 SF-Lockscreen

Übersicht:

Sperrt den Bildschirm des angemeldeten Benutzers, während der Installation (dies funktioniert auch unter dem System Account).

Beschreibung:

Der Bildschirm des Benutzers wird gesperrt, wodurch Benutzereingaben verhindert werden. Der Lock kann per Passworteingabe aufgehoben werden: Mit der Maus in den Lockscreen klicken und das Passwort auf der Tastatur eingeben. Das Standard Passwort ist unlock, ausser es wurde ein anderes per Parameter übergeben.

Syntax:

```
SF-Lockscreen "Title" "Description" "Logo" -Switches
```

Verfügbare Switches:

-Timeout x	Timeout in Sekunden, bis zur automatischen Aufhebung des Locks. Ohne diesen Parameter ist kein Timeout gesetzt (Wert 0). Ohne den Parameter DisableAutoUnlock wird der Lock am Ende des Scripts aufgehoben, auch wenn der Timeout nicht erreicht wurde.
-DisableAutoUnlock	Nach erfolgreicher Installation des Paketes wird der Lockscreen per Default aufgehoben. Mit diesem Switch lässt sich die automatische Aufhebung des Locks unterdrücken. Der Lockscreen bleibt solange aktiv, bis dieser mit dem Switch Stop aufgehoben wird.
-DisableErrorUnlock	Sofern Scripting Framework ein Error innerhalb eines Scripts feststellt (1603), wird der Lockscreen per Default aufgehoben. Mit diesem Switch bleibt der Lock des Bildschirms bei einem erkannten Fehler weiterhin aktiv.
-Password x	Das Default Password (unlock) wird mit diesem Parameter übersteuert.
-Permanent	Der Lockscreen wird in der Registry registriert, damit dieser nach einem Reboot bei einer Benutzeranmeldung erneut gestartet wird. Dieser bleibt solange aktiv, bis dieser mit dem Switch Stop aufgehoben wird.
-Stop	Aufhebung der Sperrung bei allen Benutzern. Sofern der Switch Permanent verwendet wurde, wird der automatische Start aus der Registry entfernt.

Beispiele:

```
# Lockscreen without logo and timeout and with default
SF-Lockscreen "Example" "This is an example"

# Lockscreen with two lines and a custom unlock password. Use `n for a new line
SF-Lockscreen "Example" "Line1 `nLine2" -Password "secret"

# Lockscreen without logo and timeout and with a custom unlock password
SF-Lockscreen "Example" "This is an example" -DisableAutoUnlock -Timeout 10 -Password "secret"

# Lockscreen with logo, disable auto unlock and disable automatic unlock on error
SF-Lockscreen "Example" "This is an example" "%_PkgSource%\Setup\Logo.jpg" -DisableAutoUnlock -DisableErrorUnlock

# Stops the Lockscreen
SF-Lockscreen -Stop
```

Lockscreen mit Logo:

2.26 SF-Log

Übersicht:
Schreibt einen Eintrag in die Log Datei, per Default wird als Severity CUSTOM ausgegeben.

Beschreibung:
Keine

Syntax:
```
SF-Log "Message" -Switches
```

Verfügbare Switches:

-Severity	Definiert den Schweregrad der Nachricht. Folgende Werte sind gültig: Custom, Error, Info, Warning und Title

Beispiele:
```
# Write log (CUSTOM)
SF-Log "Example log entry"

# Write log (Info)
SF-Log "ProgramFiles32 folder: %_ProgramFiles32%" -Severity Info

# Write log (ERROR)
SF-Log "Example log entry" -Severity Error
```

2.27 SF-Logoff

Übersicht:
Abmelden von allen oder einem bestimmten Windows Benutzer.

Beschreibung:
Keine

Syntax:
```
SF-Logoff "User" -Switches
```

Verfügbare Switches:

-Delay x	Verzögert den Logoff um x Sekunden.

Beispiele:
```
# Log out all logged in users
SF-Logoff

# Log out all logged in users with a delay of 10 seconds
SF-Logoff -Delay "10"

# Log off a specific user
SF-Logoff "User1"
```

2.28 SF-MD

Übersicht:
Erstellt einen leeren Ordner inklusive Unterordner.

Beschreibung:
Keine

Syntax:
```
SF-MD "Path" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Create folder
SF-MD "%_SystemDrive%\Example\Subfolder"
```

2.29 SF-MessageBox

Übersicht:
Zeigt dem angemeldeten Benutzer eine Message Box mit Windows Forms an. Dies funktioniert auch, wenn es unter dem SYSTEM Account Kontext ausgeführt wird.

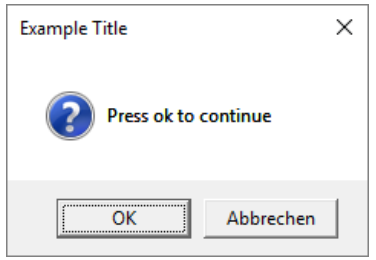
Beschreibung:
Die Funktion zeigt eine benutzerdefinierte MessageBox an. Über die Parameter wird definiert, welche Icons und Schaltflächen verwendet werden sollen. Als Rückgabewert wird der geklickte Button zurückgegeben, die möglichen Rückgabewerte sind Abort, Cancel, Ignore, No, OK, Retry oder Yes.

Syntax:
```
SF-MessageBox "Title" "Text" -Switches
```

Verfügbare Switches:

-Button	Definiert welche Schaltflächen angezeigt werden. Folgende Werte sind gültig: AbortRetryIgnore, OK, OKCancel, RetryCancel, YesNo und YesNoCancel
-Icon	Definiert welches Icon angezeigt wird. Folgende Werte sind gültig: Information, Asterisk, Question, Stop und Warning
-Wait	Warten bis der Benutzer einen Button klickt. Achtung: Ohne den Wait Parameter kann das Resultat nicht ausgewertet werden, in diesem Fall ist es immer OK!

Beispiele:
```
# This example creates a msgbox with the Ok and Cancel button and the Question Icon. Once the message box is displayed it creates the Result variable with the message box selection choosen. Once the message box is done you can use an if statement to finish the script.

$Result = SF-MessageBox "Example" "Press ok to continue" -Button OKCancel -Icon Question -Wait

If ($Result -eq "OK" )
{
	SF-Log "The user has pressed 'OK'"
}
ElseIf ($Result -eq "Cancel")
{
	SF-Log "The user has pressed 'Cancel'"
}
```



2.30 SF-MSIGetProperty

Übersicht:
Liest den Wert einer Property aus einer MSI-Datei.

Beschreibung:
Der Befehl liest den Wert der Property aus der angegebenen MSI-Datei und gibt den Wert zurück.

Syntax:
```
SF-MSIGetProperty "Path" "PropertyName" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Reads the property ARPHELPLINK and returns it.
$Value = SF-MSIGetProperty "%_PkgSource%\Setup\Google Earth.msi" "ARPHELPLINK"
SF-Log $Value -Severity Info
```

2.31 SF-MSIInstall

Übersicht:
Installiert eine MSI Datei und wertet automatisch die Return Codes (0, 1605, 1641 und 3010) aus.

Beschreibung:
Im Falle, dass bei der Installation der MSI ein Reboot erkannt wurde, beendet sich Scripting Framework am Ende des Scripts mit dem Return Code 3010. Sollte das Produkt bereits installiert sein, führt die Funktion automatisch eine Reparatur mit dem Parameter /foums aus, ausser der Switch „RemoveFirstOnRepair“ wird verwendet.

Syntax:
```
SF-MSIInstall "Path" "Arguments" -Switches
```

Verfügbare Switches:

-Log	Das MSI Log wird im Ordner „C:\Windows_ScriptingFramework_Logs\Software“ abgelegt.
-NoErrors	Die MSI Exit Codes werden ignoriert und es wird mit dem Script fortgefahren.
-RemoveFirstOnRepair	Sollte das Produkt bereits auf dem System installiert sein, wird anstelle der Reparatur einen Remove und anschliessend die Installation durchgeführt.
-SkipIfInstalled	Die Installation wird übersprungen, sofern das Programm bereits installiert ist.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# MSI installation (x86 and x64)
SF-MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn /norestart"

# MSI installation (only x86 and with log)
SF-MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn /norestart" -x86 -Log

# MSI installation (only x86, with log, and remove first if already installed)
SF-MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn /norestart" -x86 -Log -RemoveFirstonRepair
```

2.32 SF-MSIPatchInstall

Übersicht:
Installiert eine MSI Patch Datei (.msp) und wertet automatisch die Return Codes (0, 1605, 1641, 1642 und 3010) aus.

Beschreibung:
Keine

Syntax:
```
SF-MSIPatchInstall "Path" "Arguments" -Switches
```

Verfügbare Switches:

-Log	Das MSI Log wird im Ordner „C:\Windows_ScriptingFramework_Logs\Software“ abgelegt.
-NoErrors	Fehlermeldungen werden ignoriert und der Script fortgesetzt.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Install a MSI patch
SF-MSIPatchInstall "%_PkgSource%\Setup\Patch.msp" "/qn /norestart"

# Install a MSI patch (only x86 and write log)
SF-MSIPatchInstall "%_PkgSource%\Setup\Patch.msp" "/qn /norestart" -x86 -Log

# Installs all MSI patches from the specified folder
SF-MSIPatchInstall "%_PkgSource%\Setup" "/qn /norestart"
```

2.33 SF-MSIRepair

Übersicht:
Ausführen von einer MSI Reparatur.

Beschreibung:
Die Funktion wird für Installationen verwendet, welche über einen Bootstrapper (z.B. setup.exe) installiert werden, jedoch eine MSI Reparatur unterstützen. Der Befehl *SF-MSIInstall* führt automatisch beim vorhanden sein der Anwendung eine Reparatur aus, wodurch der Befehl *SF-MSIRepair* bei einer reinen MSI Installation ohne Bootstrapper nicht benötigt wird.

Syntax:
```
SF-MSIRepair "ProductCode" "Arguments" -Switches
```

Verfügbare Switches:

-Log	Das MSI Log wird im Ordner „C:\Windows_ScriptingFramework_Logs\Software“ abgelegt.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Repair a MSI installation
SF-MSIRepair "{F27DC4B4-42BC-43AF-BF5F-18ED9A7867BD}" "/qn /norestart"
```

2.34 SF-MSIUninstall

Übersicht:
Deinstalliert ein MSI Produkt und wertet automatisch die Return Codes (0, 1605, 1641 und 3010) aus.

Beschreibung:
Keine

Syntax:
```
SF-MSIUninstall "ProductCodeOrFile" "Arguments" -Switches
```

Verfügbare Switches:

-NoErrors	Die MSI Exit Codes werden ignoriert und es wird mit dem Script fortgefahren.
-Log	Das MSI Log wird im Ordner „C:\Windows_ScriptingFramework_Logs\Software“ abgelegt.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# MSI uninstall (ProductCode)
SF-MSIUninstall "{F27DC4B4-42BC-43AF-BF5F-18ED9A7867BD}" "/qn /norestart"

# MSI Uninstall (MSI file)
SF-MSIUninstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "/qn /norestart"
```

2.35 SF-MSIUninstallByDisplayName

Übersicht:
Deinstallation von einer Applikation anhand des Namens.

Beschreibung:
Die Funktion sucht nach dem angegebenen DisplayName (rekursiv) und deinstalliert die Programme, welche diesem Namen entsprechen. Voraussetzung dafür ist eine MSI Installation, andere Setuproutinen werden nicht unterstützt. Eine Suche mit Wildcards ist ebenfalls möglich.

Achtung: Der Befehl birgt bei nicht korrekter Verwendung Gefahren und kann unter Umständen bei einem „falschen“ DisplayNamen ungewollt Applikationen deinstallieren! Deshalb sollte dieser Befehl mit entsprechender Vorsicht angewendet werden.

Syntax:
```
SF-MSIUninstallByDisplayName "DisplayName" "Arguments" -Switches
```

Verfügbare Switches:

-Log	Das MSI Log wird im Ordner „C:\Windows_ScriptingFramework_Logs\Software“ abgelegt.
-NoErrors	Die MSI Exit Codes werden ignoriert und es wird mit dem Script fortgefahren.

Beispiele:
```
# Uninstall
SF-MSIUninstallByDisplayName "Microsoft Visual C++ 2008"

# Uninstall with wildcards
SF-MSIUninstallByDisplayName "*Microsoft Visual*"

# By default the parameters /qn /norestart are given, these can be customized if properties are needed
SF-MSIUninstallByDisplayName "*Microsoft Visual*" "REMOVESETTINGS=true /qn /norestart"
```

2.36 SF-MSIXInstall

Übersicht:
Installiert ein MSIX Paket.

Beschreibung:
Der Befehl installiert ein MSIX-Paket oder ein MSIX-Bundle.

Syntax:
```
SF-MSIXInstall "Path" "DependencyPackagePath" -Switches
```

Verfügbare Switches:

-NoErrors	Wenn während der Installation Fehler auftreten, werden diese ignoriert.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiel:
```
# MSIX installation
SF-MSIXInstall "%_PkgSource%\Setup\MSIX.Commander_1.0.7.2-x64.msix"

# Installs an MSIX package with Dependency Packages
SF-MSIXInstall "%_PkgSource%\Setup\Barcode_1.2.18.0_x64.msixbundle" -DependencyPackagePath
"%_PkgSource%\Setup\Dependencies\x64\Microsoft.NET.Native.Framework.2.2.appx","%_PkgSource%\Setup\Dependencies\x64\Microsoft.NET.Native.Runtime.2.2.appx","%_PkgSource%\Setup\Dependencies\x64\Microsoft.VCLibs.x64.14.00.appx"
```

2.37 SF-MSIXUninstall

Übersicht:
Deinstalliert eine MSIX Paket.

Beschreibung:
Der Befehl deinstalliert ein MSIX-Paket anhand des Namens oder PackageFullName.

Syntax:
```
SF-MSIXUninstall "Package" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# MSIX uninstallation by Name
SF-MSIXUninstall "PascalBerger.MSIXCommander"

# MSIX uninstallation by PackageFullName
SF-MSIXUninstall "PascalBerger.MSIXCommander_1.0.7.2_x64__ajjbhed1xnq88"

# MSIX uninstallation by PackageFullName with wildcard
SF-MSIXUninstall "PascalBerger.MSIXCommander_1.0.7.2_*"
```

2.38 SF-NTFSPerm

Übersicht:
Setzt NTFS Berechtigungen auf eine Datei oder auf einen Ordner.

Beschreibung:
Die drei Berechtigungstypen Full, Modify und Read sind möglich.

Syntax:
```
SF-NTFSPerm "Path" "User" "Permission" -Switches
```

Verfügbare Switches:

-InheritsFromParent	Vererbt die Berechtigung von dem übergeordneten Objekt, wodurch Änderungen in den Unterordnern zurückgesetzt werden.
-Remove	Entfernt den entsprechenden Benutzer aus den Berechtigungen.

Beispiele:
```
# Set full permission for the users group
SF-NTFSPerm "%_ProgramFiles32%\Example\Example.ini" "%_GroupUsers%" "Full"

# Set modify permission for the users group
SF-NTFSPerm "%_ProgramFiles32%\Example" "%_GroupUsers%" "Modify"

# Set read permission for the users group
SF-NTFSPerm "%_ProgramFiles32%\Example" "%_GroupUsers%" "Read"

# Remove the users group from permissions
SF-NTFSPerm "%_ProgramFiles32%\Example" "%_GroupUsers%" -Remove
```

2.39 SF-NuGetInstall

Übersicht:
Installiert ein NuGet-Softwarepaket.

Beschreibung:
Installiert ein NuGet-Paket über das PackageManagement-Modul.

Syntax:
```
SF-NuGetInstall "Name" -Switches
```

Verfügbare Switches:

-Details	Displays detailed information about the operation done by the command.
-SkipDependencies	Überspringt die Installation von Softwareabhängigkeiten.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Installs an NuGet package from a local source
SF-NuGetInstall "%_PkgSource%\Setup\notepad2.4.2.25.3.nupkg"

# Installs a Nuget Package via the NuGet PackageProvider (Internet)
SF-NuGetUninstall "notepad2"
```

2.40 SF-NuGetUninstall

Übersicht:
Deinstalliert ein NuGet-Softwarepaket.

Beschreibung:
Deinstalliert ein NuGet-Paket über das PackageManagement-Modul.

Syntax:
```
SF-NuGetInstall "Name" -Switches
```

Verfügbare Switches:

-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Uninstalls the NuGet package named notepad2
SF-NuGetUninstall "notepad2"
```

2.41 SF-Path

Übersicht:
Erweitert die Environment Variabel *Path* um den angegebenen Pfad.

Beschreibung:
Keine

Syntax:
```
SF-Path "Path" -Switches
```

Verfügbare Switches:

-AtFirst	Fügt den Pfad am Anfang ein
-Force	Falls der Path Eintrag im Zusammenspiel mit AtFirst bereits existiert, wird der Eintrag auf die erste Position verschoben
-Remove	Entfernt den Pfad aus der Path Environment Variabel
-User	Setzt oder entfernt den Pfad für den aktuellen User Kontext

Beispiele:
```
# Add a path
SF-Path "%_SystemDrive%\Example"

# Remove a path
SF-Path "%_SystemDrive%\Example" -Remove
```

2.42 SF-PinnedApplication

Übersicht:
Zum Anheften von einer Applikation an die Taskleiste oder an das Startmenü. Zusätzlich kann die Applikation auch wieder „gelöst“ werden.

Beschreibung:
Die Funktion benötigt mindestens Windows Vista oder Windows Server 2008.

Syntax:
```
SF-PinnedApplication "Path" -Switches
```

Verfügbare Switches:

-PinToTaskbar	Heftet eine Applikation an die Taskbar
-UnPinFromTaskbar	Löst eine Applikation von der Taskbar
-PinToStartMenu	Heftet eine Applikation an das Startmenü
-UnPinFromStartMenu	Löst eine Applikation vom Startmenü

Beispiele:
```
# Pin to taskbar
SF-PinnedApplication "%_Windows%\explorer.exe" -PinToTaskbar

# Unpin from taskbar
SF-PinnedApplication "%_Windows%\explorer.exe" -UnPinFromTaskbar

# Pin to startmenu
SF-PinnedApplication "%_CommonPrograms%\Accessories\WordPad.lnk" -PinToStartMenu

# Unpin from startmenu
SF-PinnedApplication "%_CommonPrograms%\Accessories\WordPad.lnk" -UnPinFromStartMenu
```

2.43 SF-ProgramsAndFeatures

Übersicht:
Erstellt einen Eintrag unter "Programme und Funktionen". Es werden standardmässig die Werte aus der Package.xml verwendet, sofern kein Parameter angegeben wird.

Beschreibung:
Erstellt einen Deinstallationseintrag, um die Software unter "Programme und Funktionen" anzuzeigen. Wird ein Paket deinstalliert, wird dieser Eintrag automatisch entfernt. Dieser Eintrag wird zudem auch entfernt, wenn der Befehl SF-Supersedence angewendet wird.

Syntax:
```
SF-ProgramsAndFeatures -Switches
```

Verfügbare Switches:

-DisplayIcon	Das Symbol, das angezeigt wird. Als Standard wird das Scripting Framework Icon verwendet.
-DisplayName	Der Name der Applikation.
-DisplayVersion	Version der Applikation.
-Publisher	Publisher/Hersteller der Applikation.
-Remove	Entfernt den entsprechenden Eintrag.
-UninstallString	Befehl zum Deinstallieren der Anwendung.

Beispiele:
```
# Creates a standard entry from the Package.xml informations
SF-ProgramsAndFeatures

# Creates a entry with customized values
SF-ProgramsAndFeatures -DisplayIcon "%_ProgramFiles32%\CDBurnerXP\cdbxpcmd.exe,0" -DisplayName "Example Titel"

# Removes the entry
SF-ProgramsAndFeatures -Remove
```

2.44 SF-RD

Übersicht:
Entfernt ein Verzeichnis.

Beschreibung:
Alle Prozesse und offenen Handles von diesem Verzeichnis werden automatisch geschlossen.

Syntax:
```
SF-RD "Path" -Switches
```

Verfügbare Switches:

-DisableTaskkill	Die Prozesse im entsprechenden Ordner werden nicht automatisch beendet.
-Empty	Löscht den Ordner nur sofern dieser leer ist.
-LocalUserProfiles	Löscht ein Ordner rekursiv aus allen lokalen Benutzerprofilen (siehe Beispiele)
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Remove folder
SF-RD "%_SystemDrive%\Example"

# Remove folder if empty
SF-RD "%_SystemDrive%\Example" -Empty

# Remove example.log from all user profiles
SF-RD "AppData\Roaming\Example" -LocalUserProfiles

Resultat Beispiel (LocalUserProfiles):
20:03:2017-08:52:57    INFO: ===== RD ================================================
20:03:2017-08:52:57    INFO:     Path: C:\Users\Public\AppData\Roaming\Example
20:03:2017-08:52:57    INFO:     Path: C:\Users\Username1\AppData\Roaming\Example
20:03:2017-08:52:57    INFO:     Path: C:\Users\Username2\AppData\Roaming\Example
20:03:2017-08:52:57    INFO:     Path: C:\Users\Username3\AppData\Roaming\Example
```

2.45 SF-ReloadVariables

Übersicht:
Der Befehl schreibt die Standardvariablen von Scripting Framework neu in die Registry.

Beschreibung:
Scripting Framework prüft bei jeder Ausführung die letzte Bootzeit. Die verfügbaren Standardvariablen werden nur neu geladen, sofern die Bootzeit abweichend ist. Dies garantiert ein sehr schnelles Laden der Engine. Mit dieser Funktion lassen sich die Variablen innerhalb eines Scripts forciert laden.

Syntax:
```
SF-RemoveVariables -Switches
```

Verfügbare Switches:

-Machine	Lädt nur die Computer Variablen neu (HKLM).
-User	Lädt nur die Benutzer Variablen neu (HKCU).

Beispiele:
```
# Reload machine and user variables
SF-ReloadVariables

# Reload only machine variables
SF-ReloadVariables -Machine

# Reload only user variables
SF-ReloadVariables -User
```

2.46 SF-RemoveVariables

Übersicht:
Die Funktion entfernt automatisch die Variablen, welche mit der Funktion SF-LoadVariables (ohne Parameter -NoVarPrefix) geladen wurden.

Beschreibung:
Die Funktion wird in der Regel nur im Uninstall.ps1 Script verwendet.

Syntax:
```
SF-RemoveVariables -Switches
```

Verfügbare Switches:

-User	Löscht die Variablen aus dem User Kontext, in welchem der Script ausgeführt wird.

Beispiele:
```
# Remove Scripting Framework variables
SF-RemoveVariables
```

2.47 SF-Register32

Übersicht:
Schreibt und löscht Registry Keys unter dem 32-Bit Hive.

Beschreibung:
Es sind folgende Registry Typen gültig: REG_SZ, REG_DWORD, REG_QWORD, REG_BINARY, REG_EXPAND_SZ und REG_MULTI_SZ.

Syntax:
```
SF-Register32 "RegHive" "SubKey" "ValueName" "ValueKind" -Switches
```

Verfügbare Switches:

-Add	Erweitert einen REG_MULTI_SZ um den angegeben Wert. Wird der Switch nicht angegeben, wird der Wert überschrieben.
-Remove	Entfernt den angegebenen Wert aus einem REG_MULTI_SZ Key.
-ConvertToHex	Für das Schreiben von einem String in einen Binary Key. Mit dem Parameter wird der String in das korrekte Format konvertiert.
-LocalUserProfiles	Führt den Befehl auf allen verfügbaren lokalen Benutzerprofilen aus.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Registry (32-bit)
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_String" "%_ProgramFiles32%" "REG_SZ"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_Dword" "32" "REG_DWORD"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_Qword" "32" "REG_QWORD"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_EXPAND_SZ" "Expand_32" "REG_EXPAND_SZ"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_MULTI_SZ" "Value1_32' "REG_MULTI_SZ" -Add
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_Binary" "H#32698347997fddff" "REG_BINARY"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_Binary" "32,69,83,47,99,7f,dd,ff" "REG_BINARY"
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_Binary" "Example" "REG_BINARY" -ConvertToHex

# Writes the value to all local user profiles
SF-Register32 "HKEY_CURRENT_USER" "SOFTWARE\Example" "Hello" "World" "REG_SZ" -LocalUserProfiles

# Remove a value (32-Bit)
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "32_String" "{nil}" "REG_SZ"

# Remove a key (32-Bit)
SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "{nil}" "{nil}" "REG_SZ"
```

2.48 SF-Register64

Übersicht:
Schreibt und löscht Registry Keys unter dem 32-Bit Hive.

Beschreibung:
Es sind folgende Registry Typen gültig: REG_SZ, REG_DWORD, REG_QWORD, REG_BINARY, REG_EXPAND_SZ und REG_MULTI_SZ.

ACHTUNG: Auf einem 32-Bit OS wird keine Aktion ausgeführt und es wird auch keine Info ausgegeben.

Syntax:
```
SF-Register64 "RegHive" "SubKey" "ValueName" "ValueKind" -Switches
```

Verfügbare Switches:

-Add	Erweitert einen REG_MULTI_SZ um den angegeben Wert. Wird der Switch nicht angegeben, wird der Wert überschrieben.
-Remove	Entfernt den angegebenen Wert aus einem REG_MULTI_SZ Key.
-ConvertToHex	Für das Schreiben von einem String in einen Binary Key. Mit dem Parameter wird der String in das korrekte Format konvertiert.

Beispiele:
```
# Registry (64-bit)
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_String" "%_ProgramFiles64%" "REG_SZ"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_Dword" "64" "REG_DWORD"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_Qword" "64" "REG_QWORD"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_EXPAND_SZ" "Expand_64" "REG_EXPAND_SZ"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_MULTI_SZ" "Value1_64" "REG_MULTI_SZ" -Add
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_Binary" "H#32698347997fddff" "REG_BINARY"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_Binary" "32,69,83,47,99,7f,dd,ff" "REG_BINARY"
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_Binary" "Example" "REG_BINARY" -ConvertToHex

# Remove a value (64-Bit)
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "64_String" "{nil}" "REG_SZ"

# Remove a key (64-Bit)
SF-Register64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "{nil}" "{nil}" "REG_SZ"
```

2.49 SF-RegisterFile32

Übersicht:
Registriert eine Datei.

Beschreibung:
Registriert eine 32-Bit Datei.

Syntax:
```
SF-RegisterFile32 "Path"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Register a file (32-bit)
SF-RegisterFile32 "%_WindowsSystem32%\Example.dll"
```

2.50 SF-RegisterFile64

Übersicht:
Registriert eine Datei.

Beschreibung:
Registriert eine 64-Bit Datei.

Syntax:
```
SF-RegisterFile64 "Path"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Register a file (64-bit)
SF-RegisterFile64 "%_WindowsSystem64%\Example.dll"
```

2.51 SF-RegPerm32

Übersicht:
Modifiziert die 32-Bit Registry Berechtigungen.

Beschreibung:
Keine

Syntax:
```
SF-RegPerm32 "RegHive" "SubKey" "User" -Switches
```

Verfügbare Switches:

-InheritsFromParent	Vererbt die Berechtigung vom übergeordneten Objekt, womit Änderungen wieder zurückgesetzt werden.
-Remove	Entfernt den entsprechenden Benutzer komplett aus den Berechtigungen.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Set the permission "Full" for the user group (32-bit hive)
SF-RegPerm32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%"

# Set the permission "Full" for the user group (32-bit hive)
SF-RegPerm32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%" "Read"

# Remove permission (32-bit hive)
SF-RegPerm32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%" -Remove

# Reset permissions (32-bit hive)
SF-RegPerm32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" -InheritsFromParent
```

2.52 SF-RegPerm64

Übersicht:
Modifiziert die 64-Bit Registry Berechtigungen.

Beschreibung:
Keine

Syntax:
```
SF-RegPerm64 "RegHive" "SubKey" "User" -Switches
```

Verfügbare Switches:

-InheritsFromParent	Vererbt die Berechtigung vom übergeordneten Objekt, womit Änderungen wieder zurückgesetzt werden.
-Remove	Entfernt den entsprechenden Benutzer komplett aus den Berechtigungen.

Beispiele:
```
# Set the permission "Full" for the user group (64-bit hive)
SF-RegPerm64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%"

# Set the permission "Full" for the user group (64-bit hive)
SF-RegPerm64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%" "Read"

# Remove permission (64-bit hive)
SF-RegPerm64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" "%_GroupUsers%" -Remove

# Reset permissions (64-bit hive)
SF-RegPerm64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Example" -InheritsFromParent
```

2.53 SF-RegRead32

Übersicht:
Liest einen Registierungswert aus dem 32-Bit Hive.

Beschreibung:
Der ausgelesene Wert wird als Scripting Framework Variable gespeichert, welche anschliessend im Script verwendet werden kann.

Syntax:
```
SF-RegRead32 "RegHive" "SubKey" "ValueName" "VariabelName" -Switches
```

Verfügbare Switches:

-Machine	Speichert die Machine bezogene Variabel unter HKLM\Software\ScriptingFramework\Variables
-Script	Erstellt eine temporäre PowerShell Variabel
-User	Speichert die User bezogene Variabel unter HKCU\Software\ScriptingFramework\Variables

Beispiele:
```
# Example 1 - Read a registry value (32-bit) to Scripting Framework variable %RegRead_Example%
SF-RegRead32 "HKey_Local_Machine" "SOFTWARE\Example" "ValueName" "RegRead_Example"

# Example 2 - Create New Scripting Framework Variable <New_VariableName>
SF-RegRead32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Microsoft\Windows\CurrentVersion" "CommonFilesDir" "New_SFVariable"
If ((SF-Variables "%New_SFVariable%") -eq "C:\Program Files (x86)\Common Files")
{
    SF-Log "Value is equal <C:\Program Files (x86)\Common Files>"
}
Else {
    SF-Log "Value is NOT equal <C:\Program Files (x86)\Common Files>"
}

# Example 3 - Create New PowerShell Variable $New_PSVariable
SF-RegRead32 "HKEY_LOCAL_MACHINE" "SOFTWARE\Microsoft\Windows\CurrentVersion" "CommonFilesDir" "New_PSVariable" -Script
SF-Log "PSVariable: $($New_PSVariable)"
```

2.54 SF-RegRead64

Übersicht:
Liest einen Registierungswert aus dem 64-Bit Hive.

Beschreibung:
Der ausgelesene Wert wird als Scripting Framework Variable gespeichert, welche anschliessend im Script verwendet werden kann.

Syntax:
```
SF-RegRead64 "RegHive" "SubKey" "ValueName" "VariabelName" -Switches
```

Verfügbare Switches:

-Machine	Speichert die Machine bezogene Variabel unter HKLM\Software\ScriptingFramework\Variables
-Script	Erstellt eine temporäre PowerShell Variabel
-User	Speichert die User bezogene Variabel unter HKCU\Software\ScriptingFramework\Variables

Beispiele:
```
# Example 1 - Read a registry value (32-bit) to Scripting Framework variable %RegRead_Example%
SF-RegRead64 "HKey_Local_Machine" "SOFTWARE\Example" "ValueName" "RegRead_Example"

# Example 2 - Create New Scripting Framework Variable <New_VariableName>
SF-RegRead64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Microsoft\Windows\CurrentVersion" "CommonFilesDir" "New_SFVariable"
If ((SF-Variables "%New_SFVariable%") -eq "C:\Program Files (x86)\Common Files")
{
    SF-Log "Value is equal <C:\Program Files (x86)\Common Files>"
}
Else {
    SF-Log "Value is NOT equal <C:\Program Files (x86)\Common Files>"
}

# Example 3 - Create New PowerShell Variable $New_PSVariable
SF-RegRead64 "HKEY_LOCAL_MACHINE" "SOFTWARE\Microsoft\Windows\CurrentVersion" "CommonFilesDir" "New_PSVariable" -Script
SF-Log "PSVariable: $($New_PSVariable)"
```

2.55 SF-RegSearch32

Übersicht:
Die Funktion durchsucht die Registry (32-Bit Hive) nach einem Wert und kann optional dessen Wert oder Namen überprüfen. Die Suche erfolgt rekursiv und die Funktion gibt true oder false als Wert zurück. Sofern jede glich der RegHive und ein SubKey der Funktion übergeben wird, findet eine Prüfung auf die Existenz des Schlüssels statt. Das Resultat wird zusätzlich in einem mehrdimensionalen Array (SearchList) gespeichert, wodurch weitere Auswertungen möglich sind.

Beschreibung:
Keine

Syntax:
```
SF-RegSearch32 "RegHive" "SubKey" "SearchText" "ValueName" -Switches
```

Verfügbare Switches:

-ExcludeKeys	Per Default wird auch in dem Schlüssel Name nachdem Text gesucht. Mit diesem Parameter kann der Schlüssel Name von der Suche ausgeschlossen werden.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Searches for "SearchString" under the Key "Software\Example"
If ((SF-RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SearchString") -eq $True) {
	SF-Log "Value found"
}

# Searches for "SearchString" under the Key "Software\Example" the value name must be "DisplayName". The function returns
true if the text was found
If ((SF-RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SearchString" "DisplayName") -eq $True) {
	SF-Log "Value found"
}

# Search with wildcards
If ((SF-RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "*SuchText*") -eq $True) {
	SF-Log "Value found"
}

# Search and subsequently evaluating the array
SF-RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText"

# First Array Entry
If ($SearchList -ne $null)
{
```

```
    SF-Log $SearchList[0][0] # RootKey
    SF-Log $SearchList[0][1] # SubKey
    SF-Log $SearchList[0][2] # ValueName
    SF-Log $SearchList[0][3] # Value
}

# All Array Entries
ForEach($Search in $SearchList)
{
    SF-Log $Search
}
```

2.56 SF-RegSearch64

Übersicht:
Die Funktion durchsucht die Registry (64-Bit Hive) nach einem Wert und kann optional dessen Wert oder Namen überprüfen. Die Suche erfolgt rekursiv und die Funktion gibt true oder false als Wert zurück. Sofern jede glich der RegHive und ein SubKey der Funktion übergeben wird, findet eine Prüfung auf die Existenz des Schlüssels statt. Das Resultat wird zusätzlich in einem mehrdimensionalen Array (SearchList) gespeichert, wodurch weitere Auswertungen möglich sind.

Beschreibung:
Keine

Syntax:
```
SF-RegSearch64 'RegHive' 'SubKey' 'SearchText' 'ValueName' -Switches
```

Verfügbare Switches:

-ExcludeKeys	Per Default wird auch in dem Schlüssel Name nachdem Text gesucht. Mit diesem Parameter kann der Schlüssel Name von der Suche ausgeschlossen werden.

Beispiele:
```
# Searches for "SearchString" under the Key "Software\Example"
If ((SF-RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SearchString") -eq $True) {
        SF-Log "Value found"
}
```

```
# Searches for "SearchString" under the Key "Software\Example" the value name must be "DisplayName". The function returns
true if the text was found
If ((SF-RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SearchString" "DisplayName") -eq $True) {
      SF-Log "Value found"
}

# Search with wildcards
If ((SF-RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "*SuchText*") -eq $True) {
      SF-Log "Value found"
}

# Search and subsequently evaluating the array
SF-RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText"

# First Array Entry
If ($SearchList -ne $null)
{
    SF-Log $SearchList[0][0] # RootKey
    SF-Log $SearchList[0][1] # SubKey
    SF-Log $SearchList[0][2] # ValueName
    SF-Log $SearchList[0][3] # Value
}

# All Array Entries
ForEach($Search in $SearchList)
{
      SF-Log $Search
}
```

2.57 SF-Rename

Übersicht:
Zum Umbenennen von einem Ordner oder einer Datei.

Beschreibung:
Keine

Syntax:
```
SF-Rename "Source" "Target" -Switches
```

Verfügbare Switches:

-Force	Falls der TargetPath existiert, wird er vor der Umbenennung entfernt
-LocalUserProfiles	Benennt eine Datei rekursiv in allen lokalen Benutzerprofilen um

Beispiele:
```
# Rename folder to Example1
SF-Rename "%_SystemDrive%\Example" "%_SystemDrive%\Example1"

# Rename file to Example1.ini
SF-Rename "%_SystemDrive%\Example\Example.ini" "%_SystemDrive%\Example\Example1.ini"

# Renames the file Example.txt to Example2.txt in all local user profiles
SF-Rename "AppData\Roaming\Example.txt" "AppData\Roaming\Example2.txt" -LocalUserProfiles
```

2.58 SF-Replace

Übersicht:
Sucht und ersetzt einen Wert und schreibt den neuen Wert als Scripting Framework Variabel in die Registrierung.

Beschreibung:
Keine

Syntax:
```
SF-Replace "Value" "Search" "Replace" "VariableName" -Switches
```

Verfügbare Switches:

-User	Die Variabel wird für den User definiert HKCU\Software\ScriptingFramework\Variables

Beispiele:
```
# Replace the word input with output in the variable Example
SF-Replace "InputText" "Input" "Output" "Example"

# Writes the value C:/Windows in the variable Example
SF-Replace "%_Windows%" "\" "/" "Example"
```

2.59 SF-Requirements

Übersicht:
Prüft die Systemanforderungen.

Beschreibung:
Anhand der angegebenen Parameter wird überprüft, ob das System die Bedingungen erfüllt. Wenn eine Bedingung nicht erfüllt ist, wird das Skript beendet.

Syntax:
```
SF-Requirements -Switches
```

Verfügbare Switches:

-CPUSpeed	Prüft den minimalen CPU Speed in MHz.
-Date "mm/dd/yyyy H:mm"	Prüft ob das definierte Datum und Uhrzeit erreicht ist.
-FreeDiskSpaceSystemDrive	Prüft den minimalen freien Speicherplatz auf dem System Drive (C:\) in MB.
-TotalPhysicalMemory	Prüft den minimalen physikalischen Arbeitsspeicher in MB.
-x86	Prüft ob es sich um ein 32-Bit Betriebssystem handelt.
-x64	Prüft ob es sich um ein 64-Bit Betriebssystem handelt.

Beispiele:
```
# Example 1 - Checks only the free disk space of the system drive
SF-Requirements -FreeDiskSpaceSystemDrive "2000"

# Example 2 - Checks the free disk space of the system drive and the physical memory
SF-Requirements -FreeDiskSpaceSystemDrive "2000" -TotalPhysicalMemory "4000"

# Example 3 - Checks the free disk space of the system drive, the physical memory and OS architecture
SF-Requirements -FreeDiskSpaceSystemDrive "2000" -TotalPhysicalMemory "4000" -x64

# Example 4 - Checks the date
SF-Requirements -Date "06/22/2019 12:00"
```

2.60 SF-Run

Übersicht:
Startet eine gewünschte Datei und wertet den Return Code automatisch aus.

Beschreibung:
Per Default läuft der Run Befehl auf einen Fehler, sofern nicht einer der folgenden ExitCodes zurückgegeben wird (0, 1605, 1641 und 3010). Bei einem 1641 oder 3010 wird analog zum SF-MSIInstall Befehl automatisch ein Reboot registriert (z.B. Installshield Installationen). Ein Error kann mit dem Parameter -NoErrors unterdrückt werden. Somit ist es möglich den ReturnCode über die PowerShell Variabel $ExitCode selbst auszuwerten → siehe Beispiel. Zusätzlich besteht die Möglichkeit über den Parameter ExcludedExitCodes weitere Codes zu ignorieren, diese müssen Komma getrennt angegeben werden.

Syntax:
```
SF-Run "Path" "Arguments" "WorkingDirectory" -Switches
```

Verfügbare Switches:

-ExcludedExitCodes	Mit diesem Parameter können zusätzliche Retrun Codes angegeben werden, welche nicht als Fehler gelten.
-LoggedOnUsers	Dieser Parameter ermöglicht es, dass der Prozess automatisch auf allen angemeldeten Benutzer mit dessen Credentials startet. Dadurch kann ein Prozess für den Benutzer gestartet und angezeigt werden, selbst wenn die Installation des Paketes unter dem SYSTEM Account abläuft. Falls kein Benutzer angemeldet ist, wird der Run Befehl übersprungen.
-NoErrors	Fehlermeldungen werden ignoriert und der Script fortgesetzt.
-Show	Das Fenster wird angezeigt und nicht als Hidden gestartet.
-SkipIfNotExist	Falls die Datei nicht existiert, wird der Script ohne Fehler fortgeführt.
-Timeout x	Im Zusammenspiel mit dem Switch Wait kann ein Timeout (maximale Wartezeit) in Sekunden angegeben werden. Sofern der Prozess länger als der definierte Timeout läuft, so wird der Exit Code 6010 zurückgegeben.
-Wait	Warten bis der Prozess beendet wurde.
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Run Notepad without wait and show the interface
SF-Run "%_Windows%\notepad.exe" "" -Show

# Run Notepad on a x86 OS
SF-Run "%_Windows%\notepad.exe" "" -Show -x86
```

```
# Run Notepad and waiting until it is closed, and check the return code manually
SF-Run "%_Windows%\notepad.exe" "" -Show -Wait -NoErrors
If (($ExitCode -eq 3) -or ($ExitCode -eq 4)) {
	SF-Exit "Error detected"
}

# Run Notepad and excluding other return codes
SF-Run "%_Windows%\notepad.exe" "" -Show -Wait -ExcludedExitCodes "256,1024"


# Start notepad and wait 10 seconds maximum
SF-Run "%_Windows%\notepad.exe" "" -Show -Wait -Timeout 10
```

2.61 SF-Service

Übersicht:
Dieser Befehl kontrolliert einen Windows Service.

Beschreibung:
Es ist möglich diesen zu starten, stoppen, restarten oder auch zu löschen. Zudem kann der Starttyp auf Automatisch, Verzögert, Deaktiviert oder Manuell gesetzt werden.

Syntax:
```
SF-Service "Name" -Switches
```

Verfügbare Switches:

-Start	Startet einen Dienst
-Stop	Stoppt einen Dienst
-Restart	Startet einen Dienst neu
-Automatic	Setzt den Starttyp auf Automatisch
-Delayed	Setzt den Starttyp auf Automatisch (Verzögerter Start)
-Disabled	Setzt den Starttyp auf Deaktiviert
-Manual	Setzt den Starttyp auf Manuell
-Delete	Löscht den Service beim nächsten Neustart

Beispiele:
```
# Set the startup type to automatic
SF-Service "spooler" –Automatic -Start

# Stop the service
SF-Service "spooler" –Stop

# Restart the service
SF-Service "spooler" -Restart

# Set the startup type to Automatic and restart the service
SF-Service "spooler" –Automatic -Restart
```

2.62 SF-ServiceInstall

Übersicht:
Dieser Befehl installiert einen Windows Service.

Beschreibung:
Keine

Syntax:
```
SF-ServiceInstall "Path" "ServiceName" "DisplayName" "Description" "Username" "Password" -Switches
```

Verfügbare Switches:

Username	Der Parameter Username ist optional
Password	Der Parameter Password ist optional
-Automatic	Setzt den Starttyp auf Automatisch
-Delayed	Setzt den Starttyp auf Delayed
-Disabled	Setzt den Starttyp auf Deaktiviert
-Restart	Startet einen Dienst neu
-Manual	Setzt den Starttyp auf Manual

Beispiele:
```
# Creates a service with the startup type Automatic (runs with the system account)
SF-ServiceInstall "%_SystemDrive%\Example\MyService.exe" "ServiceExample" "My Example Service" "This is a sample service"
"" "" -Automatic

# Creates a service with the startup type Automatic (runs with the user ExampleUser1)
SF-ServiceInstall "%_SystemDrive%\Example\MyService.exe" "ServiceExample" "My Example Service" "This is a sample service"
"ExampleUser1" "Password" -Automatic
```

2.63 SF-ServiceStatus

Übersicht:
Dieser Befehl überprüft den Status eines Windows-Dienstes.

Beschreibung:
Es ist möglich zu überprüfen, ob der Dienst existiert, oder ob der Status oder StartType einem bestimmten Wert entspricht. Der Befehl gibt den Wert true oder false zurück. Zudem kann der Befehl als Wert auch den StartType oder Status als Text zurückgeben.

Syntax:
```
SF-ServiceStatus "ServiceName" -Switches
```

Verfügbare Switches:

-StartType	Gibt an welcher StartType des Dienstes abgefragt wird. Die zulässigen Werte für diesen Parameter sind: Automatic, AutomaticDelayedStart, Disabled, Manual
-Status	Gibt an welcher Status des Dienstes abgefragt wird. Die zulässigen Werte für diesen Parameter sind: Running, Stopped, Paused
-ReturnStartType	Gibt an, dass vom angegebenen Dienst der StartType als Text zurückgegeben wird.
-ReturnStatus	Gibt an, dass vom angegebenen Dienst der Status als Text zurückgegeben wird.
-NoErrors	Falls der Dienst nicht existiert, wird der Fehler ignoriert.

Beispiele:
```
# Checks if the service is exists
If ((SF-ServiceStatus "spooler") -eq $true) {
    SF-Log "The Service exists"
}
Else {
    SF-Log "The service does not exist"
}

# Checks the service StartType
If ((SF-ServiceStatus "spooler" -StartType Automatic) -eq $true) {
    SF-Log "The Service exists and the StartType is Automatic"
}
Else {
    SF-Log "The service does not exist or the StartType is not Automatic."
}
```

```powershell
# Checks the service StartType and Status
If ((SF-ServiceStatus "spooler" -StartType Automatic -Status Running) -eq $true) {
    SF-Log "The Service exists, the StartType is Automatic and the Status is Running"
}
Else {
    SF-Log "The service does not exist, the StartType is not automatic or the status is not running."
}

# Returns the status as value
$Result = SF-ServiceStatus "spooler" -ReturnStatus
If ($Result -eq "Running") {
    SF-Log "Service is Running"
}
```

2.64 SF-Set

Übersicht:
Setzt eine Scripting Framework Variabel, welche in der Registry gespeichert wird.

Beschreibung:
Die gesetzte Variabel kann anschliessend im Script mit *%Name%* angesprochen werden.

Syntax:
```
SF-Set "Variabelname" "Value" -Switches
```

Verfügbare Switches:

-Delete	Löscht die Variabel
-Environment	Setzt eine Environment Variable (System oder User). Um eine User Environment Variable zu setzen muss zusätzlich der Parameter User mitgegeben werden.
-Script	Setzt eine temporäre Variable, welche nur innerhalb des Scripts gültig ist
-User	Die Variabel wird für den User definiert HKCU\Software\ScriptingFramework\Variables
-UseVarPrefix	Beim Erstellen der Variable wird automatisch ein Präfix für den Variablename verwendet (a_PkgIdentifier-NAME). Der Wert der Variablen kann im Skript mit %_VarPrefix-NAME% angesprochen werden.

Beispiele:
```
# Sets the machine variable Example with a value of C:\Windows
SF-Set "Example" "%_Windows%"

# Set the system environment variable Example with the value C:\Windows
SF-Set "Example" "%_Windows%" -Environment

# Sets the user variable Example with the value %AppData%
SF-Set "Example" "%_ApplicationData%" -User

# Removes the machine variable
SF-Set "Example" -Delete

# Removes the system environment variable
SF-Set "Example" -Environment -Delete

# Removes the user variable
SF-Set "Example" -Delete -User
```

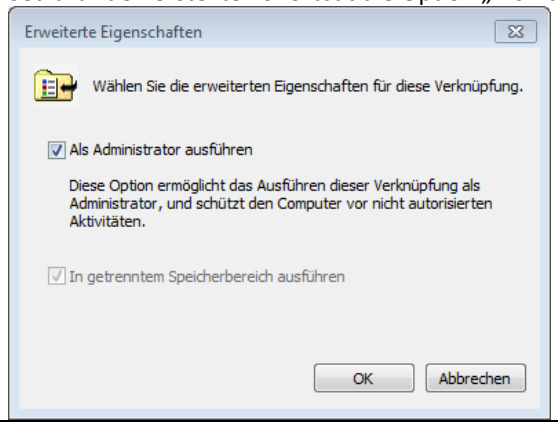
2.65 SF-Shortcut

Übersicht:
Erstellt einen Shortcut mit den entsprechenden Angaben.

Beschreibung:
Sollte zusätzlich zu dem Target ein Parameter notwendig sein, wird dies mit einem Komma abgetrennt (siehe Beispiel).

Syntax:
```
SF-Shortcut "Shortcut" "Target" "Start in" "Icon" "Comment" -Switches
```

Verfügbare Switches:

-Elevated	Setzt für den erstellten Shortcut die Option „Als Administrator ausführen“. 
-Hotkey	Einen Hotkey (Tastenkomination) für den Shortcut definieren. Zum Beispiel "CTRL+SHIFT+F1"
-Maximized	Definiert den Window Style als Maximized
-Minimized	Definiert den Window Style als Minimized

Beispiele:
```
# Create a shortcut for notepad
SF-Shortcut "%_CommonPrograms%\Notepad.lnk" "%_Windows%\notepad.exe" "%_Windows%" "%_Windows%\notepad.exe,0" "Notepad
Example"

# Create a shortcut with parameters
SF-Shortcut "%_CommonPrograms%\Notepad with Parameter.lnk" "%_Windows%\notepad.exe,%_Windows%\WindowsUpdate.log"
"%_Windows%" "%_Windows%\notepad.exe,0" "Notepad Example" -Maximized -Hotkey "CTRL+SHIFT+F1"
```

2.66 SF-ShowBalloonTip

Übersicht:
Zeigt einen BalloonTip in der unteren rechten Ecke des Bildschirms an. Der Befehl muss im aktuellen Benutzerkontext verwendet werden (es wird kein BalloonTip angezeigt, sofern der Befehl unter einem anderen Kontext verwendet wird).

Beschreibung:
Zeigt einen BalloonTip in der unteren rechten Ecke des Bildschirms an. Titel, Text und Schweregrad können angepasst werden.

Syntax:
```
SF-ShowBalloonTip -Switches
```

Verfügbare Switches:

-Severity	Definiert den Schweregrad der Nachricht. Folgende Werte sind gültig: Error, Info, Warning und None

Beispiele:
```
# Displays a balloon tip
SF-ShowBalloonTip "Scripting Framework" "This is a example balloon tip" -Severity Info
```

2.67 SF-ShowCustomNotification

Übersicht:
Zeigt dem Benutzer eine benutzerdefinierte Nachricht an (dies funktioniert auch unter dem Systemkonto).

Beschreibung:
Die Funktion zeigt eine benutzerdefinierte Scripting Framework Notification an. Über die Parameter sind die Textelemente und Buttons konfigurierbar. Die Anordnung der Buttons ist von rechts nach links. Als Rückgabewert wird der geklickte Button zurückgegeben, die möglichen Rückgabewerte sind Button1, Button2 oder Button3.

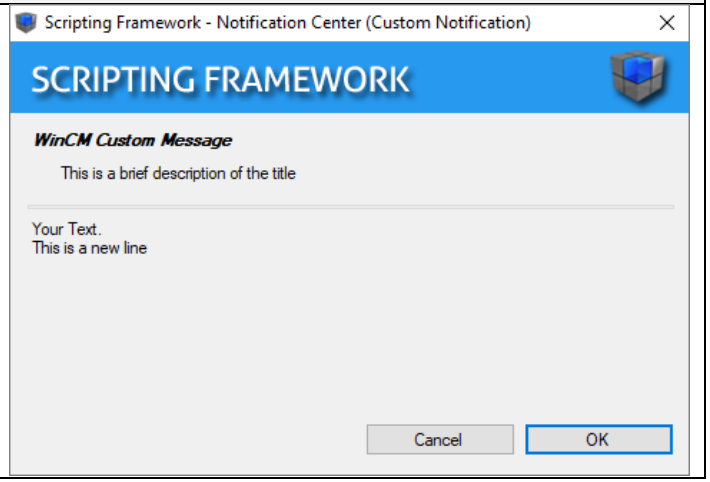
Syntax:
```
SF-ShowCustomNotification "Titel" "TitelDescription" "Text" -Switches
```

Verfügbare Switches:

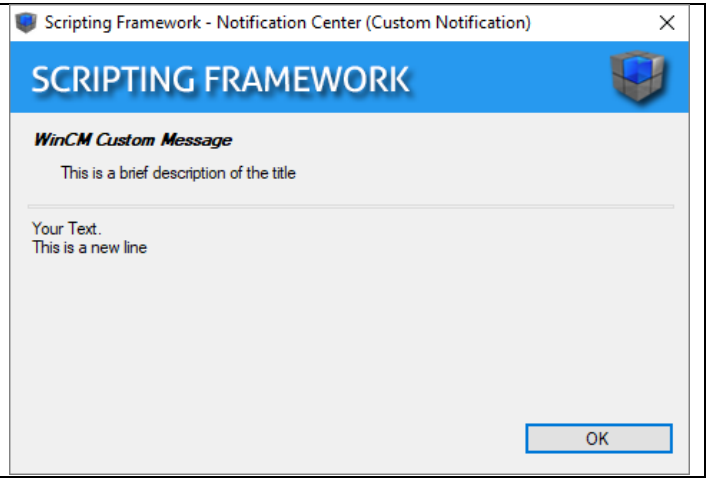
-Button1 x	Zeigt den Button1 mit dem definierten Text an.
-Button2 x	Zeigt den Button2 mit dem definierten Text an.
-Button3 x	Zeigt den Button3 mit dem definierten Text an.
-Wait	Warten bis der Benutzer einen Button klickt. Achtung: Ohne den Wait Parameter kann das Resultat nicht ausgewertet werden, in diesem Fall ist es immer OK!

Beispiele:
```
# Displays a notification with two buttons, then the result is
evaluated.
$Result = SF-ShowCustomNotification "WinCM Custom Message" "This is a
brief description of the title" "Your Text. `nThis is a new line" -
Button1 "OK" -Button2 "Cancel" -Wait
```

Notification:



```powershell
# Displays a notification with one button without waiting for the result
SF-ShowCustomNotification "WinCM Custom Message" "This is a brief
description of the title" "Your Text. `nThis is a new line" -Button1
"OK"
```



2.68 SF-ShowInstallNotification

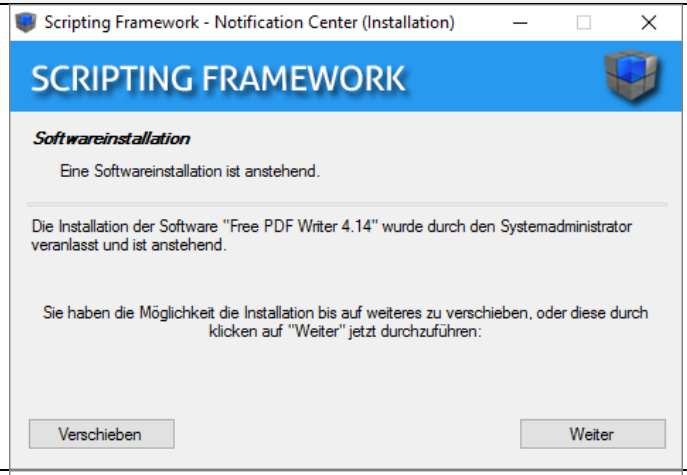
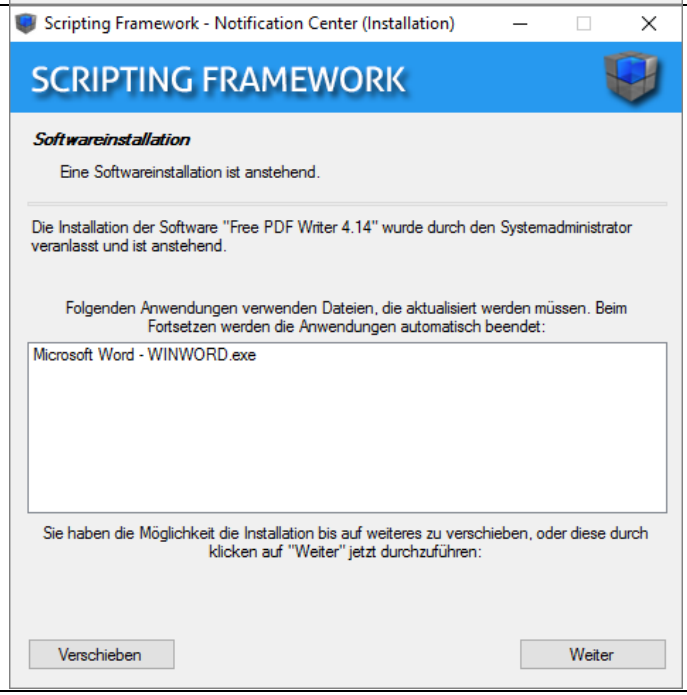
Übersicht:
Zeigt dem Benutzer eine Notification an, welche die Installation des Paketes ankündigt (dies funktioniert auch unter dem System Account). Siehe auch Scripting Framework Notification Center.

Beschreibung:
Der angemeldete Benutzer wird per Notification informiert, dass die Installation von einem Softwarepaket ansteht. Der Benutzer kann entscheiden, ob die Installation zu einem späteren Zeitpunkt oder sofort ausgeführt werden soll. Dabei ist es optional möglich die Anzahl Verschiebungen und/oder das Stichtagdatum festzulegen. Zusätzlich besteht die Möglichkeit den Benutzer über die zu schliessenden Tasks zu informieren. Sofern kein Benutzer am Gerät angemeldet ist, wird die Installation ohne eine Meldung automatisch fortgeführt. Bei einer Verschiebung der Installation, gibt Scripting Framework den Return Code 70100 zurück.

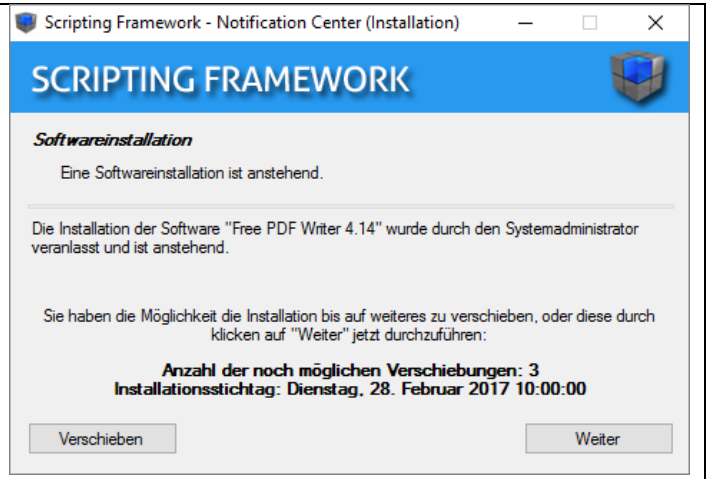
Syntax:
```
SF-ShowInstallNotification -Switches
```

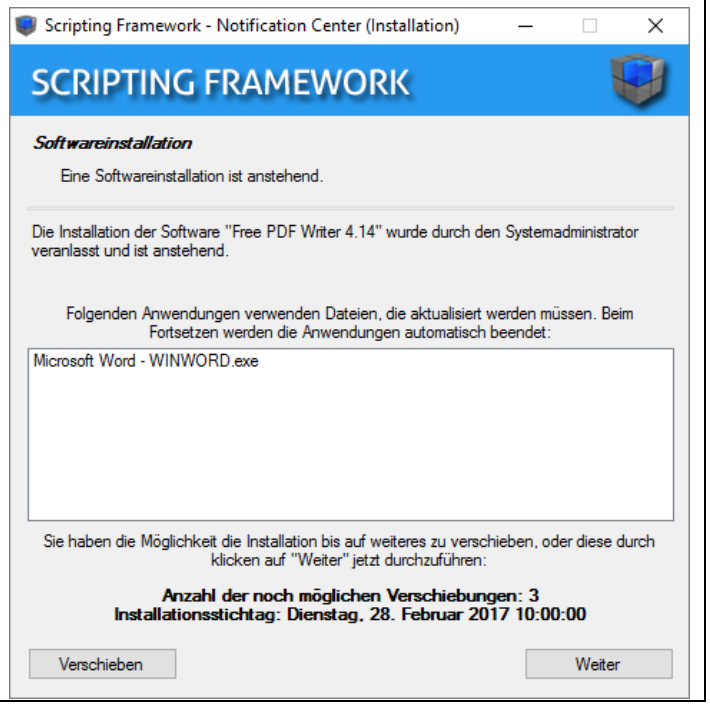
Verfügbare Switches:

-Description	Optional kann anstelle der Description (aus der Language.xml) ein eigener Text definiert werden. Mit ##NAME## kann der Paketname ausgegeben werden, siehe Beispiele.
-DefaultActionDefer x	Timeout in Sekunden, bis die Installation automatisch verschoben wird, sofern der Benutzer keine Auswahl trifft. Ohne diesen Parameter ist kein Timeout gesetzt (Wert 0). Der Timeout Countdown ist auf dem entsprechenden Button sichtbar.
-DefaultActionContinue x	Timeout in Sekunden, bis die Installation automatisch durchgeführt wird, sofern der Benutzer keine Auswahl trifft. Ohne diesen Parameter ist kein Timeout gesetzt (Wert 0). Der Timeout Countdown ist auf dem entsprechenden Button sichtbar.
-DeferCounter x	Um die Anzahl der möglichen Verschiebungen einzuschränken, kann mit diesem Parameter die gewünschte Zahl definiert werden. Eine Kombination mit dem Parameter DeferDeadline ist möglich.
-DeferDeadline "mm/dd/yyyy H:mm"	Das Datum und Uhrzeit, ab welchem keine weitere Verschiebung der Installation mehr möglich ist.
-DeferDisabled	Verbirgt auf dem GUI den Button „Verschieben”.
-Reappear x	Wenn die Installation verschoben wird, wird das Notification Center minimiert und nicht geschlossen. Nach x Sekunden wird es automatisch wieder angezeigt.
-Taskkill x	Definiert die zu schliessenden Tasks für die Installation. Analog dem Befehl SF-Taskkill besteht die Möglichkeit mehrere Prozesse oder Verzeichnisse mit dem Trennungszeichen | anzugeben. Das Notification Center überprüft, ob einer der angegebenen Prozesse aktiv ist. Sollte kein aktiver Prozess gefunden werden, wird eine Notification ohne die geöffneten Anwendungen angezeigt.
-ShowOnlyIfTaskkill	Die Notification wird nur angezeigt, sofern aktive Prozesse vorhanden sind, welche mit dem Parameter Taskkill definiert wurden. Ansonsten wird die Installation des Paketes, ohne den Benutzer zu informieren fortgeführt.

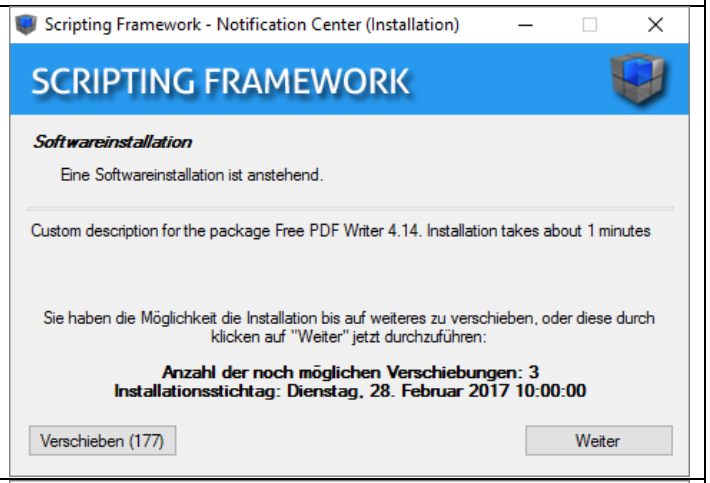
Beispiele:	Notification:
`# Default notification without parameters` `SF-ShowInstallNotification`	
`# Default notification with killing tasks (only the active tasks are displayed to the user). In this example, Microsoft Word was found` `SF-ShowInstallNotification -Taskkill "winword.exe|excel.exe"`	

```
# Notification with a taskkill (all processes in a folder and two
individual tasks), maximum defer 3 times, and installation deadline is
2.01.2016. The first screenshot shows the notification, if there is no
active process has been found. On the second screenshot was Microsoft
Word active.
SF-ShowInstallNotification -Taskkill "%_ProgramFiles32%\Internet
Explorer|winword.exe|excel.exe" -DeferCounter 3 -DeferDeadline
"02/28/2017 10:00"
```

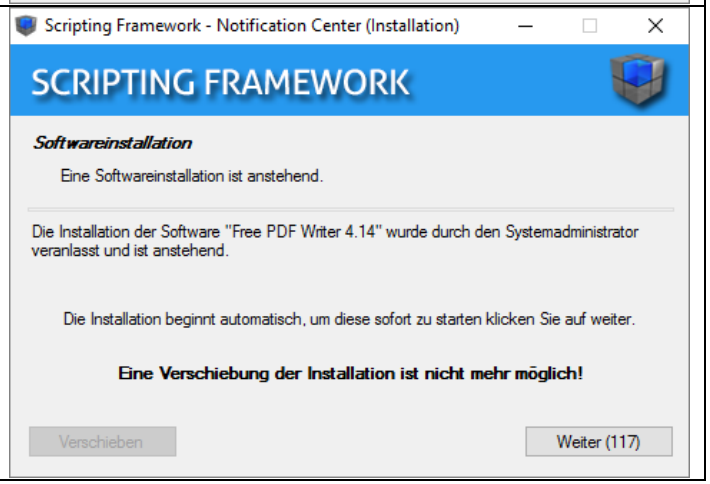




```
# Notification with a custom description. With ##NAME## you can show the package installation name.
SF-ShowInstallNotification -Description "Custom description for the package ##NAME##. Installation takes about 1 minutes" -DeferCounter 3 -DeferDeadline "02/28/2017 10:00" -DefaultActionDefer 180
```




```
# Notification when deadline is reached (no task to kill)
```



2.69 SF-ShowProgressBar

Übersicht:
Zeigt dem Benutzer während der Installation oder Deinstallation eine ProgressBar an (dies funktioniert auch unter dem System Account).

Beschreibung:
Mit dem Befehl kann dem Benutzer eine ProgressBar angezeigt werden, um dem Benutzer anzuzeigen, dass noch eine Installation oder Deinstallation im Hintergrund läuft. Dies kann besonders bei langen Installationsdauern hilfreich sein. Nach Abschluss des Paketes wird die ProgressBar automatisch beendet.

Syntax:
```
SF-ShowProgressBar -Switches
```

Verfügbare Switches:

-ExpectedTime x	Gibt die erwartete Zeit in Sekunden an, wie lange der Vorgang in etwa dauert. Dies hat Einfluss darauf, wie schnell der Fortschrittbalken auf 100% ist. Sind die 100% erreicht, wird die Textanzeige automatisch angepasst, dass der Vorgang demnächst abgeschlossen ist. Wird der Parameter nicht angegeben, wird ein sich wiederholender Ladebalken angezeigt.
-IfSYSTEM	Der Befehl wird nur ausgeführt, sofern das Paket unter dem SYSTEM Account Kontext ausgeführt wird, ansonsten wird der Befehl übersprungen.
-Stop	Beendet die ProgressBar bevor das Paket abgeschlossen ist.

Beispiele:
```
# Displays a ProgressBar to the user
SF-ShowProgressBar

# Shows the user a ProgressBar with the progress bar reaching 100 after 60 seconds
SF-ShowProgressBar -ExpectedTime 60

# Stop displaying the ProgressBar early
SF-ShowProgressBar -Stop
```

2.70 SF-ShowSCCM

Übersicht:
Zeigt dem Benutzer das Microsoft System Center Client an (dies funktioniert auch unter dem System Account).

Beschreibung:
Bei allen angemeldeten Benutzer wird der System Center Client (SCClient.exe) gestartet, um den Installation Status anzuzeigen.

Syntax:
```
SF-ShowSCCM
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Shows the System Center client to the user
SF-ShowSCCM
```

2.71 SF-Supersedence

Übersicht:
Die Funktion wird verwendet, um ein bereits installiertes Scripting Framework Paket ohne dessen Deinstallation, innerhalb von einem neuen Paket abzulösen. Durch den Aufruf der Funktion, werden alle Scripting Framework relevanten Einträge (Variablen, etc.) des abzulösenden Paketes bereinigt. Als Rückgabewert wird bei vorhandener Installation true anderenfalls false zurückgegeben. Durch den Rückgabewert sind entsprechende Abfragen und Aktionen möglich.

Beschreibung:
Als Parameter wird der Funktion der Identifier des Paketes übergeben, welches abgelöst werden soll. Hierbei werden auch Wildcards unterstützt (siehe Beispiele). Anhand des PkgIdentifier wird überprüft, ob das entsprechende Scripting Framework Paket auf dem Client installiert ist. Wird das abzulösende Paket gefunden, werden alle Scripting Framework relevanten Einträge bereinigt und als Rückgabewert wird true zurückgegeben. Durch den Rückgabewert besteht die Möglichkeit mit einer einfachen Abfrage entsprechend im Script zu reagieren und Aktionen auszulösen. Beispielsweise kann anstelle von einer kompletten Softwareinstallation nur ein Patch installiert werden, sofern ein abzulösendes Paket erkannt wurde (siehe Beispiele).

Syntax:
```
SF-Supersedence "Identifier" -Switches
```

Verfügbare Switches:

-Cleanup	Mit diesem Switch wird der entsprechende Application Key unter „HKLM\SOFTWARE*WOW6432Node*\ScriptingFramework\Inventory\Applications*PkgIdentifier*“ komplett entfernt. Ohne Verwendung des Parameters, wird der Wert InstallState auf Uninstalled und das UninstallDate gesetzt. Dies entspricht dem Verhalten, wenn ein Paket über die Uninstall.exe deinstalliert wird.

Beispiele:
```
# Sample 1 – Clears the entries of the package and sets the InstallState in the application inventory to “uninstalled”
SF-Supersedence "Adobe_Acrobat_Reader_DC_1501020060_MUI_UNV_01"

# Sample 2- Clears the entries of all version 15 packages (wildcard *) and removes the package from the application
inventory
SF-Supersedence "Adobe_Acrobat_Reader_DC_15*" -Cleanup
```

```
# Sample 3 - Replacement of Adobe Reader 15.010.20060 with version 15.020.20039 with the use of a query
If ((SF-Supersedence "Adobe_Acrobat_Reader_DC_1501020060_MUI_UNV_01")) {
    # Only Patch Installation
    SF-MSIPatchInstall "%_PkgSource%\Setup\Patches\AcroRdrDCUpd1502020039_MUI.msp" "ALLUSERS='1' /qn /norestart"
}
Else {
    # Full Installation with Patch
    SF-MSIInstall "%_PkgSource%\Setup\AcroRead.msi" "ALLUSERS='1' EULA_ACCEPT='YES' /qn /norestart"
    SF-MSIPatchInstall "%_PkgSource%\Setup\Patches\AcroRdrDCUpd1502020039_MUI.msp" "ALLUSERS='1' /qn /norestart"
}
```

2.72 SF-SymbolicLink

Übersicht:
Erstellt eine symbolische Verknüpfung zu einer Datei oder einem Verzeichnis.

Beschreibung:
Erstellt eine symbolische Verknüpfung zu einer Datei oder einem Verzeichnis als Alternative zu mklink.exe.

Syntax:
```
SF-SymbolicLink "Path" "Target"
```

Verfügbare Switches:

-Keine	

Beispiele:
```
# Creates a symbolic link to Example folder that resides on C:\Program Files (x86)\Example.
SF-SymbolicLink "%_SystemDrive%\Example" "%_ProgramFiles32%\Example"

# Creates a symbolic link to Document.txt file under the directory C:\Program Files (x86) called SomeDocument.
SF-SymbolicLink "%_SystemDrive%\Example\Document.txt" "%_ProgramFiles32%\SomeDocument"

# Creates a symbolic link to Document.txt file under the current directory called SomeDocument.
SF-SymbolicLink "%_SystemDrive%\Example\Document.txt" "SomeDocument"
```

2.73 SF-SystemReboot

Übersicht:
Der Befehl registriert einen Neustart innerhalb des Scripts, wodurch bei einem erfolgreichen Ablauf der Return Code 3010 (Reboot) zurückgegeben wird. Der Neustart selbst wird nicht durch Scripting Framework ausgeführt! Siehe Details.

Beschreibung:
Keine

Syntax:
```
SF-SystemReboot -Switches
```

Verfügbare Switches:

-DuringInstallation	Der Script beendet sich ohne weiteren Code auszuführen, mit dem Return Code 1641. Im Zusammenspiel mit SCCM wird ein Reboot ausgelöst, bevor weitere Installation ablaufen. Sofern die Kriterien der Erkennungsmethoden nicht erfüllt sind, läuft die Installation erneut an. In diesem Fall muss im Script mit einem Flag gearbeitet werden, damit der Reboot nur einmalig ausgeführt wird.
-Immediate	Am Ende der Paket Installation wird ein Return Code 1641 zurückgegeben.

Beispiele:
```
# Register a reboot
SF-SystemReboot

# Register a reboot immediate
SF-SystemReboot -Immediate

# Reboot during installation
If ((SF-Variables "%Reboot_Flag%") -ne "1") {
        SF-Set "Reboot_Flag" "1"
        SF-SystemReboot -DuringInstallation
}
```

2.74 SF-Taskkill

Übersicht:
Mit der Taskkill Funktion kann ein Prozess beendet werden.

Beschreibung:
Anstelle eines einzelnen Prozesses kann auch ein Verzeichnis angegeben werden, um alle darin vorhanden Prozesse und Handles automatisch zu schliessen. Mehrere Prozesse oder Ordner sind mit dem Trennungszeichen | möglich.

Syntax:
```
SF-Taskkill "Prozess oder Ordner"
```

Verfügbare Switches:

-Notification	Sofern die Anwendung läuft, wird dem Benutzer vor dem fortführen der Installation eine Meldung über das Scripting Framework Notification Center angezeigt.
-Test	Bei der Verwendung des Switches, gibt die Funktion den Wert true oder false zurück, aktive Prozesse werden dabei nicht geschlossen.

Beispiele:
```
# Close Excel without a notification
SF-Taskkill "EXCEL.EXE"

# Close Excel with a notification
SF-Taskkill "EXCEL.EXE" -Notification

# Close Excel and Winword with a notification
SF-Taskkill "EXCEL.EXE|WINWORD.EXE" -Notification

# Close all processes in the Office14 and Reader 11.0 folder
SF-Taskkill "%_ProgramFiles32%\Microsoft Office\Office14|%_ProgramFiles32%\Adobe\Reader 11.0" -Notification

# Close specified process
SF-Taskkill "%_ProgramFiles64%\Java\jre1.8.0_77\bin\javaw.exe"

# Close Winword and all processes in Reader 11.0 folder
SF-Taskkill "WINWORD.EXE|%_ProgramFiles32%\Adobe\Reader 11.0"
```

2.75 SF-TestMSUpdate

Übersicht:
Prüfen ob ein bestimmter Microsoft Update installiert ist.

Beschreibung:
Die Funktion gibt den Wert true oder false zurück.

Syntax:

```
SF-TestMSUpdate "Id"
```

Verfügbare Switches:

Keine	

Beispiele:

```
# Check if Microsoft Update installed
If ((SF-TestMSUpdate "KB2719306") -eq $True) {
    SF-Log "The update is installed"
}
```

2.76 SF-Textfile

Übersicht:
Mit dem Textfile Befehl können Textdateien manipuliert/editiert werden.

Beschreibung:
Dazu gehört das Suchen und Ersetzen von einem Text, das Ersetzen von einer ganzen Linie oder das Hinzufügen von Text am Anfang oder am Ende der Datei. Zudem kann definiert werden, dass der Text nur hinzugefügt wird, sofern dieser noch nicht existiert.

Syntax:
```
SF-Textfile "Path" "Text" "ReplaceWith" -Switches
```

Verfügbare Switches:

-Add	Fügt den gewünschten Text am Ende der Datei ein.
-AtTop	Zusätzliche Parameter zum -Add. Damit wird der Text am Anfang und nicht am Ende der Datei hinzugefügt.
-DeleteLine	Löscht die Linie in welcher der angegebene Text gefunden wird.
-IfMissing	Den Text nur hinzufügen, falls dieser noch nicht existiert.
-Replace	Sucht und ersetzt einen Text.
-ReplaceLine	Ersetzt die ganze Linie mit einem Text, sofern das Wort gefunden wurde.
-TextFromFile	Mit diesem Parameter kann eine Textdatei anstelle eines Textes angegeben werden. In dieser Datei können auch Variablen (z.B. %_ProgramFiles32%) verwendet werden. Die Quelldatei wird dann Linie für Linie ausgelesen und die gewünschte Datei entsprechend geschrieben. Alle anderen Parameter sind für diese Funktion auch gültig!
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Adds "Example Line" at the end of the file
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" –Add

# Adds "Example Line" to the end of the file if it doesn't exist
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" –Add –IfMissing

# Adds "Example Line" at the beginning of the file
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" –Add –AtTop

# Adds the entire contents of "Source.text" file to Example.txt
SF-Textfile "%_SystemDrive%\Example\Example.txt" "%_SystemDrive%\Source.txt" –Add –IfMissing -TextFromFile
```

```
# Search and replace the word Example with Example_2
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example" "Example_2" -Replace

# Replaces the complete line in which the text is found
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example" "Example_2" -ReplaceLine

# Clears the lines in which the text is found
SF-Textfile "%_SystemDrive%\Example\Example.txt" "Example" -DeleteLine
```

2.77 SF-UnRegisterFile32

Übersicht:
Hebt die Registrierung einer Datei.

Beschreibung:
Hebt die Registrierung einer 32-Bit Datei auf

Syntax:
```
SF-UnRegisterFile32 "Path"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Unregister file (32-Bit)
SF-UnRegisterFile32 "%_WindowsSystem32%\Example.dll"
```

2.78 SF-UnRegisterFile64

Übersicht:
Hebt die Registrierung einer Datei.

Beschreibung:
Hebt die Registrierung einer 64-Bit Datei auf

Syntax:
```
SF-UnRegisterFile64 "Datei"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Unregister file (64-Bit)
SF-UnRegisterFile64 "%_WindowsSystem64%\Example.dll"
```

2.79 SF-Variables

Übersicht:
Löst den Wert von einer Scripting Framework Variabel auf und gibt diesen zurück.

Beschreibung:
Damit kann z.B. auf den Inhalt der Variabel geprüft werden, oder es ist möglich den Wert einer Scripting Framework Variable in eine PowerShell Variable zu schreiben.

Syntax:
```
SF-Variables "Variabel"
```

Verfügbare Switches:

Keine	

Beispiele:
```
# Check variable content
If ((SF-Variables "%_ProgramFiles32%") -eq "C:\Program Files (x86)") {
    SF-Log "String found"
}

# Generate a PowerShell variable
$Value = (SF-Variables "%_ProgramFiles32%")
```

2.80 SF-Wait

Übersicht:
Mit dem Befehl Wait ist es möglich eine gewünschte Zeit (in Sekunden) zu warten. Anstelle der Zeit kann auf das Starten oder auf das Beenden eines Prozesses gewartet werden. Mit der Angabe von einer Datei- oder einem Ordnerpfad, wird solange gewartet bis dieser existiert.

Beschreibung:
Keine

Syntax:
```
SF-Wait "Argument" -Switches
```

Verfügbare Switches:

-NoErrors	Dieser Parameter ist im Zusammenhang mit dem Timeout Parameter relevant. Beim Erreichen des Timeout, wird per Default das Paket mit einem Fehler beendet. Mit dem Parameter -NoErrors wird das Paket mit einer Warnung fortgesetzt.
-Timeout x	Definiert die maximale Wartezeit in Sekunden, wenn ein Prozess oder ein Pfad zu einer Datei angegeben wird.
-TillDelete	Falls als Parameter eine Datei- oder Ordnerpfad angegeben wird, kann mit diesem Parameter definiert werden, dass solange gewartet wird, bis die Datei oder der Ordner nicht mehr vorhanden ist.
-TillKill	Falls als Parameter ein Prozess angegeben wird, kann mit diesem Parameter definiert werden, dass solange gewartet wird, bis der Prozess beendet wurde.

Beispiele:
```
# Wait 5 seconds
SF-Wait "5"

# Wait until the file exists and then continue
SF-Wait "%_ProgramFiles32%\Example.txt"

# Wait until the file is deleted and then continue
SF-Wait "%_ProgramFiles32%\Example.txt " -TillDelete

# Waits for the Notepad process to start with a maximum wait time of 60 seconds
SF-Wait "Notepad.exe" -Timeout 60

# Wait until the process Notepad is closed and then continue
SF-Wait "Notepad.exe" -TillKill
```

2.81 SF-WUSAInstall

Übersicht:
Installiert ein Microsoft Patch (.msu) und prüft den entsprechenden Return Code (0, 1605, 1641, 3010, 2359302, -2145124329).

Beschreibung:
Keine

Syntax:
```
SF-WUSAInstall "Path" -Switches
```

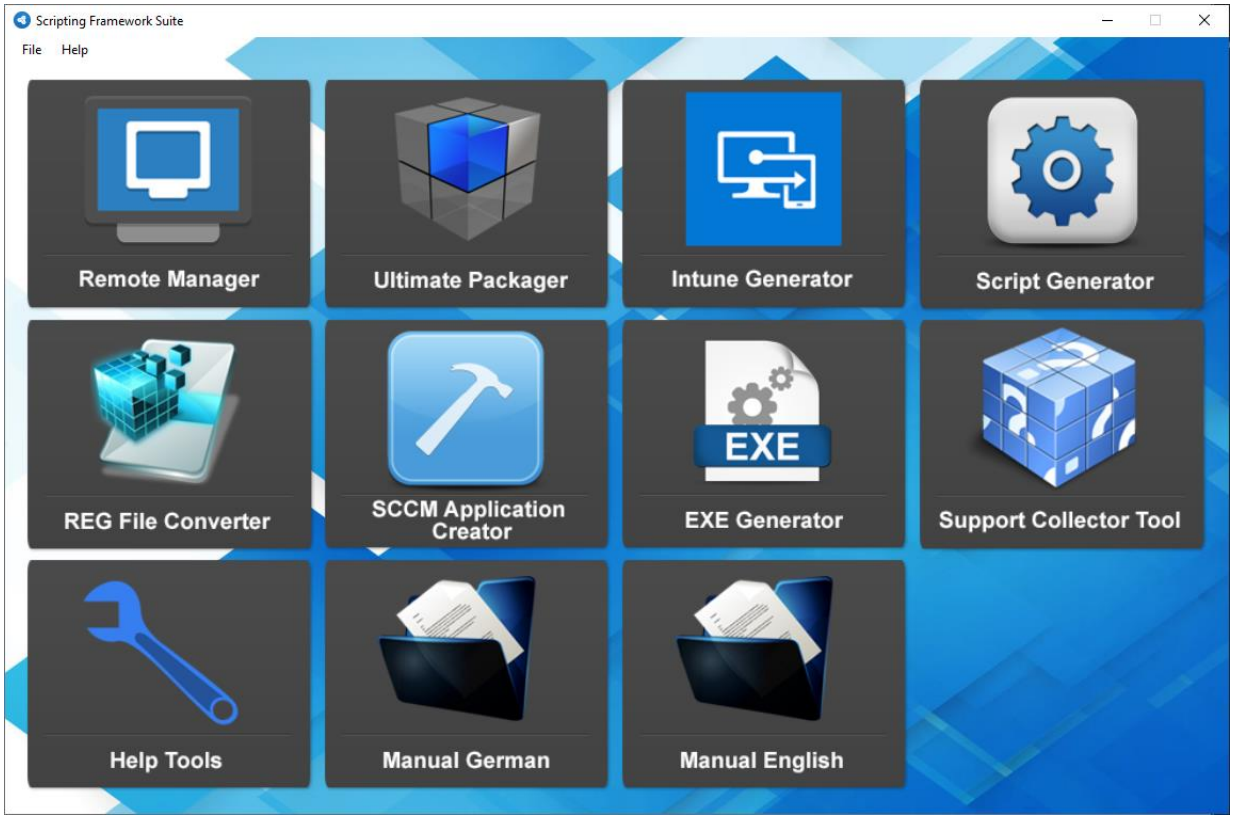
Verfügbare Switches:

-Log	Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software“ geschrieben
-x86	Den Befehl nur auf einem 32-Bit OS ausführen.
-x64	Den Befehl nur auf einem 64-Bit OS ausführen.

Beispiele:
```
# Install a MS hotfix
SF-WUSAInstall "%_PkgSource%\Setup\Windows6.1-KB2756822-x64.msu"

# Installs all hotfixes from the specified folder
SF-WUSAInstall "%_PkgSource%\Setup"
```

3 Verfügbare Suite Tools

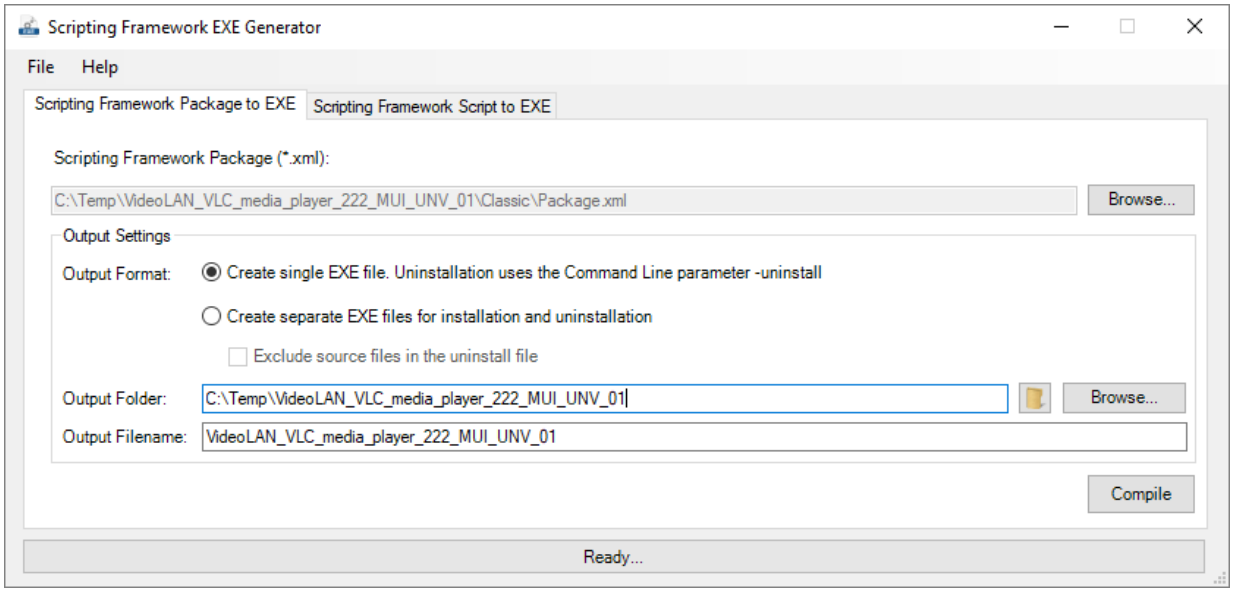
Die Scripting Framework Suite beinhaltet alle Werkzeuge, welche nebst der Script Engine zur Verfügung stehen und Sie optimal unterstützen. Die Suite beinhaltet zum Beispiel den Ultimate Packager und den Script Generator mit welchem Sie schnell und einfach Software-Pakete auf Basis der Scripting Framework Funktionen erstellen.



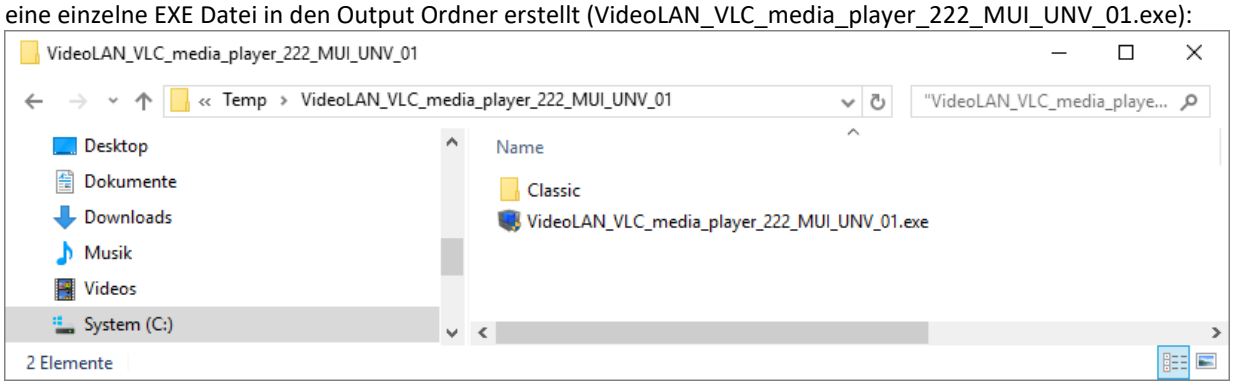
3.1 EXE Generator

Aus einem Scripting Framework Paket, oder aus einer einzelnen Skriptdatei, besteht mit dem EXE Generator die Möglichkeit, eine ausführbare Datei (*.exe) zu erstellen. Auf dem Client auf welchem die Executables generiert werden, ist die Software RARLAB WinRAR mit einer gültigen Lizenz Voraussetzung. Auf dem Ziel Client wird WinRAR nicht benötigt, auf diesem wird nur die Scripting Framework Engine vorausgesetzt.

3.1.1 Beispiel Paket zu EXE

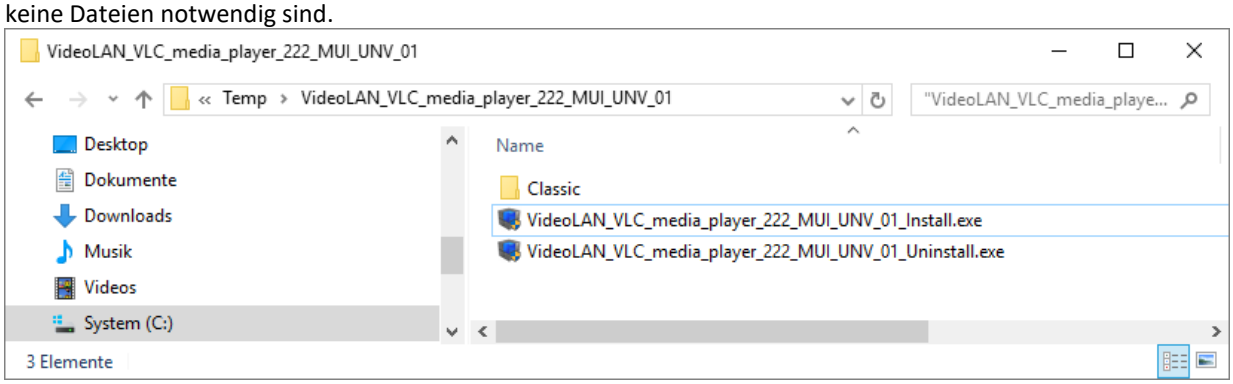


Mit der auf dem Screenshot ausgewählten Option „Create a single EXE file...“ wird aus dem selektierten Paket eine einzelne EXE Datei in den Output Ordner erstellt (VideoLAN_VLC_media_player_222_MUI_UNV_01.exe):

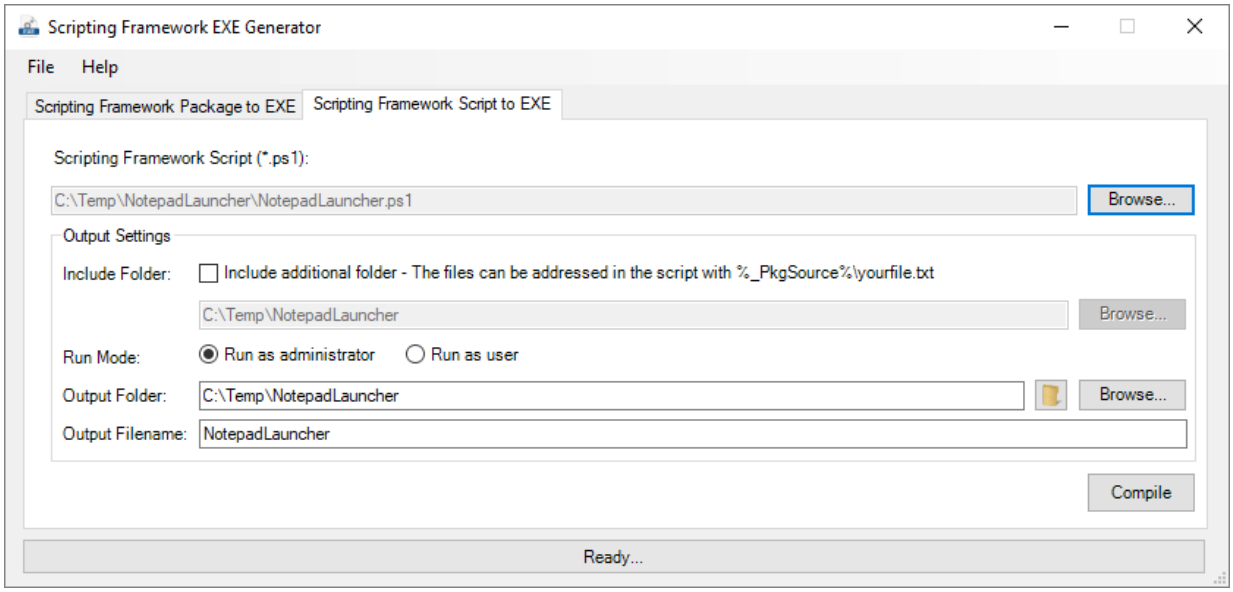


Mit dem Aufruf der EXE Datei startet die Installation im Silent Modus, ein Parameter ist nicht notwendig. Für die Deinstallation steht der Command Line Parameter -uninstall zur Verfügung:
VideoLAN_VLC_media_player_222_MUI_UNV_01.exe -uninstall

Die zweite verfügbare Option „Create separate EXE files for installation and uninstallation“ erstellt zwei separate Dateien für die Installation und Deinstallation. Dabei besteht die Möglichkeit die Setup Source aus der Deinstallation EXE Datei zu exkludieren. Dies wird angewendet, sofern für die Deinstallation nur der Script und keine Dateien notwendig sind.

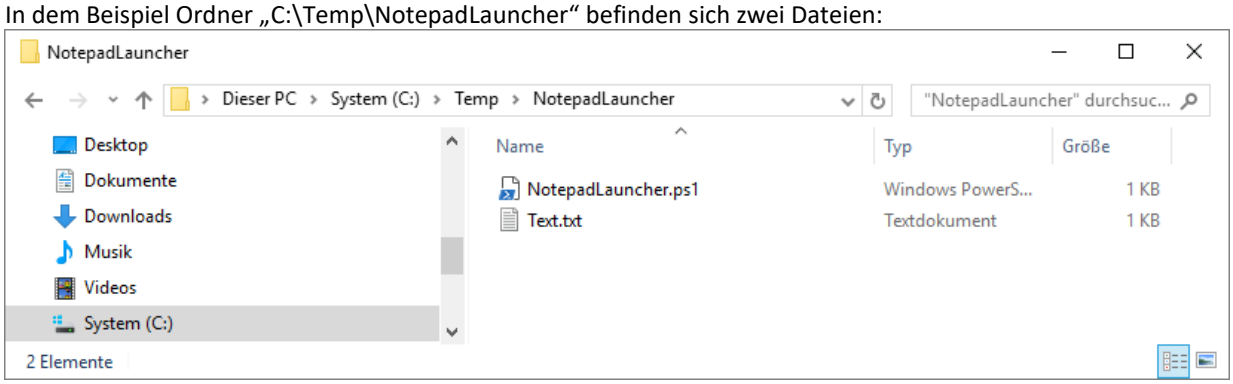


3.1.2 Beispiel Script zu EXE (Scripting Mode)

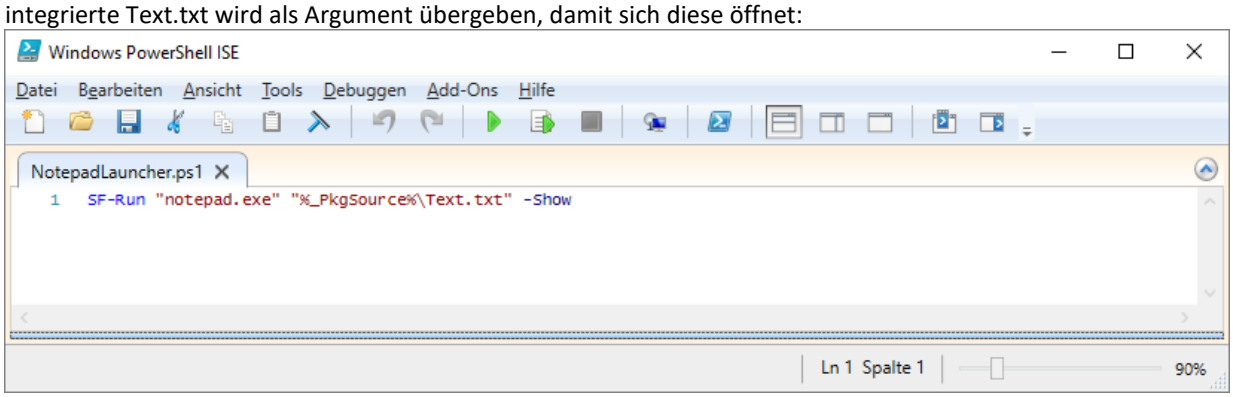


Unter dem Tab „Scripting Framework Scipt to EXE“ wird im Gegensatz zu einem gesamten Paket eine einzelne Skriptdatei (*.ps1) zu einer EXE kompiliert. Zusätzlich zu der Skriptdatei, können bei Bedarf weitere Dateien integriert und direkt aus dem Skript angesprochen werden.

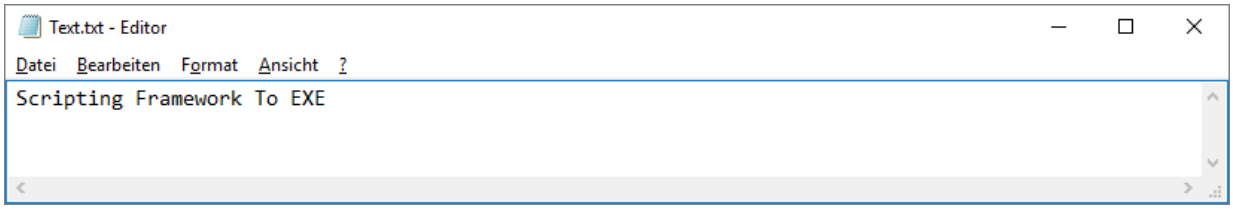
In dem Beispiel Ordner „C:\Temp\NotepadLauncher“ befinden sich zwei Dateien:



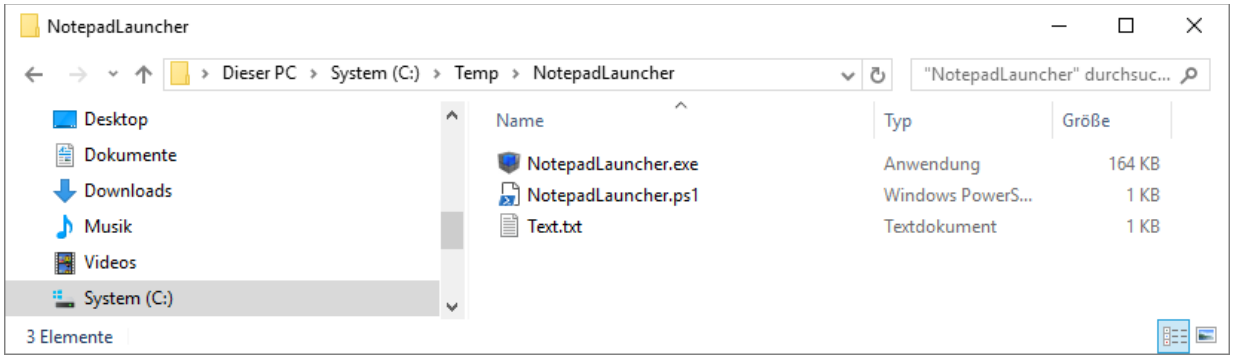
Die Skriptdatei „NotepadLauncher.ps1“ enthält den entsprechenden Code um Notepad zu starten und die integrierte Text.txt wird als Argument übergeben, damit sich diese öffnet:



Inhalt der Text.txt:



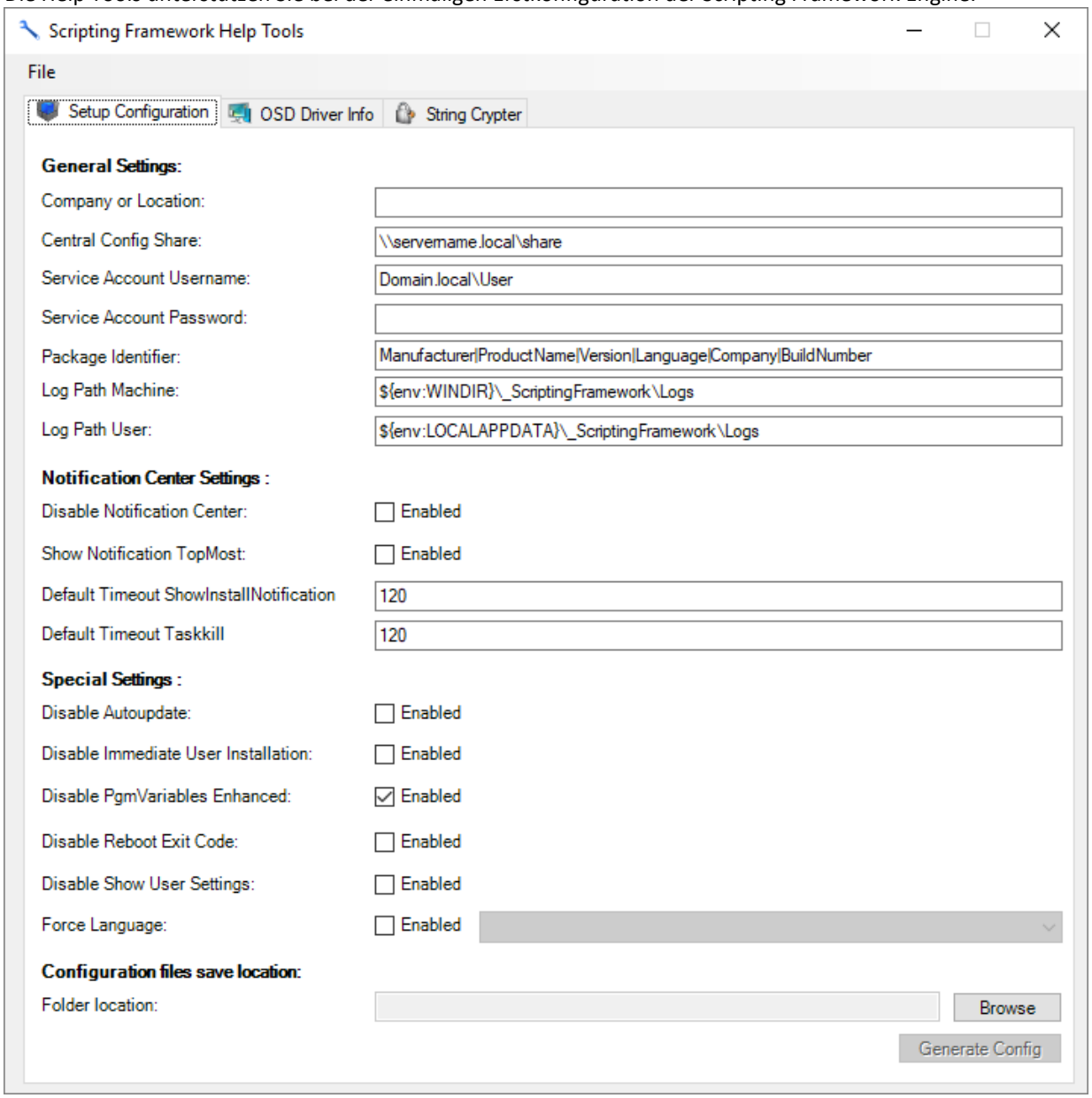
Daraus resultiert eine einzelne ausführbare Datei (NotepadLauncher.exe), welche die zusätzliche Textdatei beinhaltet:



Mit dem Aufruf der NotepadLauncher.exe wird Notepad mit der integrierten Text.txt gestartet. Für die Ausführung wird Scripting Framework auf dem Client vorausgesetzt.

3.2 Help Tools

Die Help Tools unterstützen Sie bei der einmaligen Erstkonfiguration der Scripting Framework Engine.



3.3 Manual English

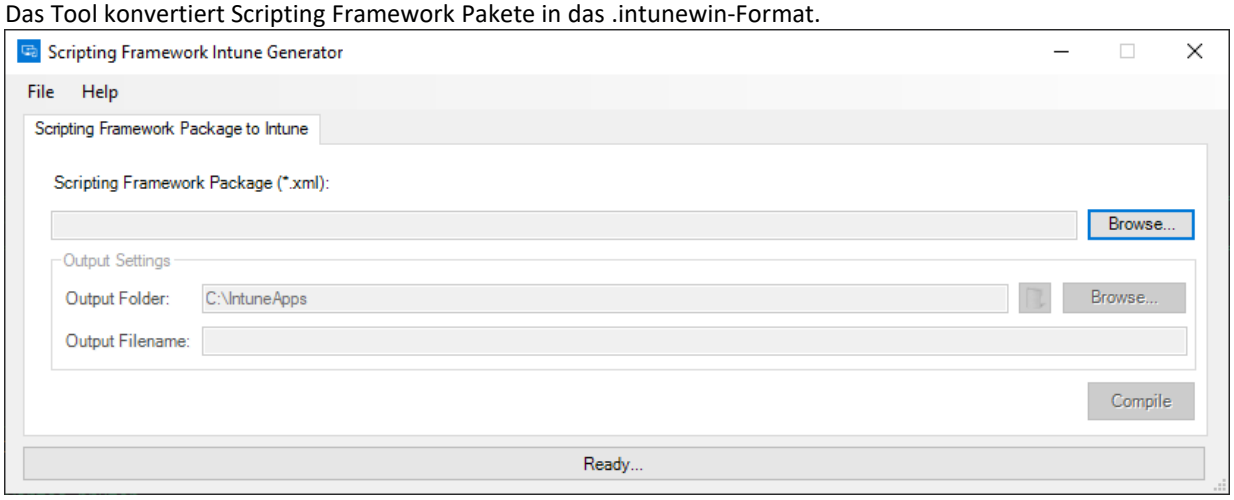
Das Scripting Framework Manual in Englisch als PDF Datei.

3.4 Manual German

Das Scripting Framework Manual in Deutsch als PDF Datei.

3.5 Intune Generator

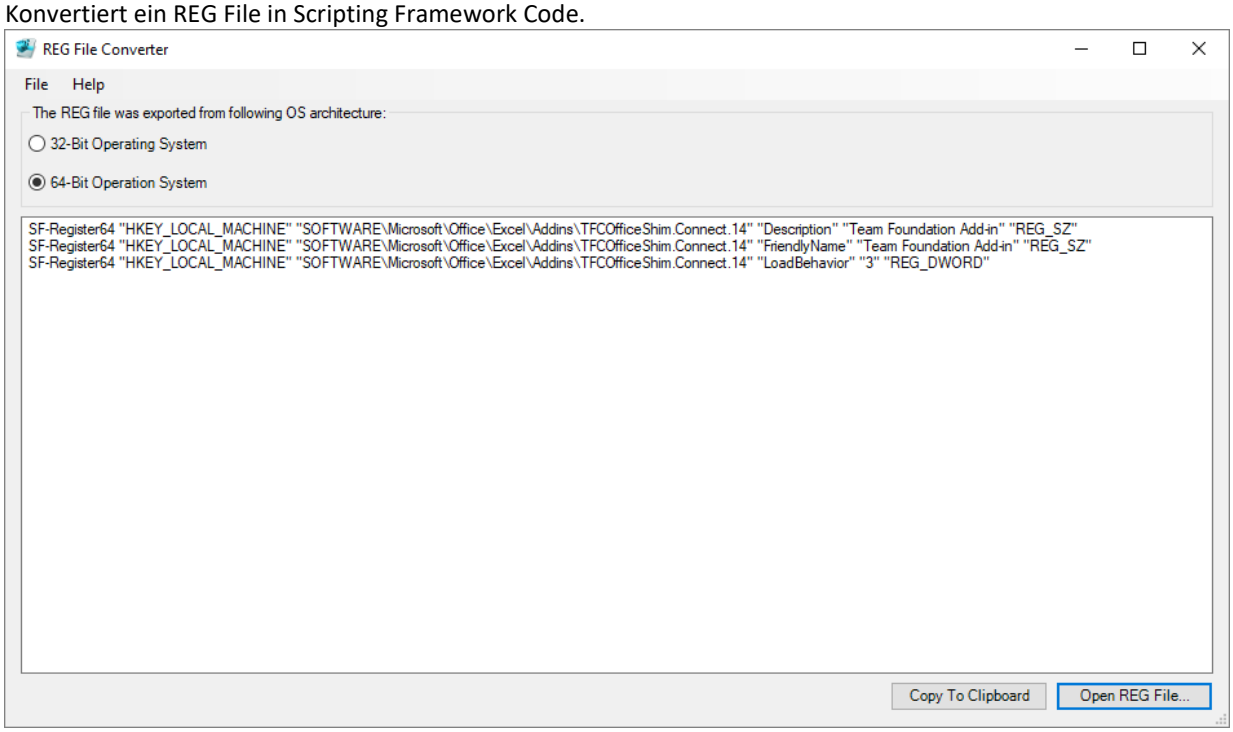
Das Tool konvertiert Scripting Framework Pakete in das .intunewin-Format.



Nach der Konvertierung kann das Paket auf Intune hochgeladen werden.

3.6 REG File Converter

Konvertiert ein REG File in Scripting Framework Code.

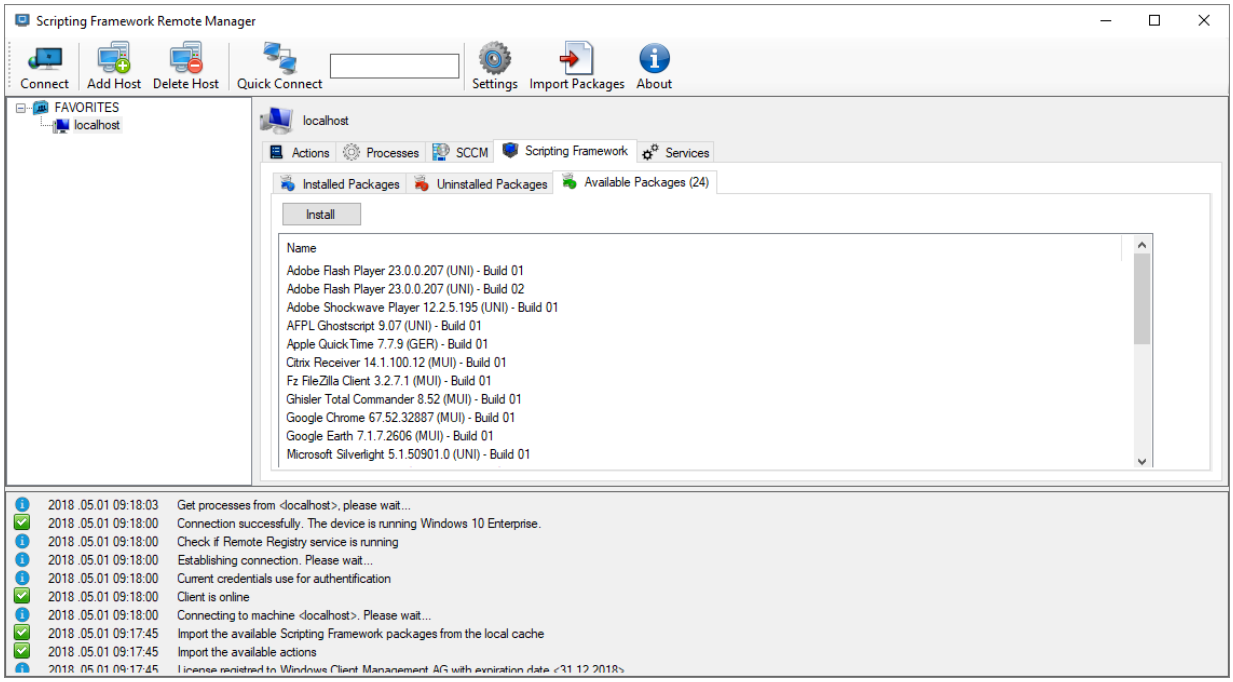


Um eine REG Datei umzuwandeln, selektieren Sie als erstes ob die Datei von einem 32-Bit oder 64-Bit OS exportiert wurde. Danach ziehen Sie per Drag & Drop die gewünschte REG Datei in das Fenster, oder Sie verwenden den Button „Open REG File…“.

Die REG Datei wird danach automatisch umgewandelt. Mit dem Button „Copy To Clipboard“ wird der gesamte Inhalt in die Zwischenablage kopiert. Es können auch einzelne Linien markiert und kopiert werden. Nachdem Kopieren in die Zwischenablage können Sie den Code im gewünschten Paket einfügen.

3.7 Remote Manager

Mit dem Remote Manager lassen sich Scripting Framework Softwarepakete manuell auf einzelne Clients installieren und deinstallieren. Zugleich haben Sie Zugriff auf nützliche Aktionen und Informationen.



Voraussetzungen

- Microsoft .NET Framework 4.5.0 oder höher
- Lokale Administrator Rechte auf dem zu verbindenden Client
- Der Service *Remote Registry* muss gestartet sein
- Die Firewall Einstellungen müssen so eingestellt sein, dass die Verbindung auf den Share Admin$ erlaubt ist

3.7.1 Tool Bar

[Connect]
Verbinden auf einen Favoriten.

[Add Host]
Einen Favoriten hinzufügen. Mit einem Doppelklick auf den entsprechenden Favoriten, verbinden Sie sich direkt auf den Client.

[Delete Host]
Einen Favoriten entfernen.

[Quick Connect]
Durch klicken auf den Button oder drücken der Enter Taste wird eine Verbindung zum Client hergestellt, welcher im nebenstehenden Textfeld eingegeben wurde.

[Settings]
Öffnen der verfügbaren Einstellungen, wie z.B. der Package Share, auf welchem die Pakete abgelegt sind.

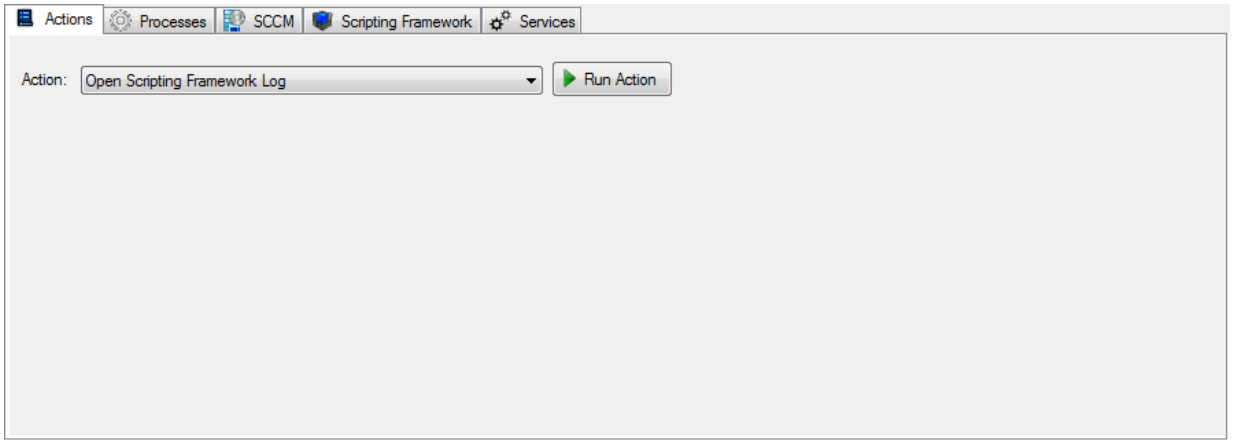
[Import Packages]
Importieren der Scripting Framework Paket Informationen von Ihrem Paket-Share, welcher in den Settings entsprechend hinterlegt werden muss. Nachdem Importvorgang können Sie dem verbundenen Client zusätzliche Pakete installieren oder bereits installierte Pakete deinstallieren.

Die Paketinformationen werden zwischengespeichert, damit diese bei einem Neustart des Remote Managers sofort zur Verfügung steht. Sofern ein Paket neu hinzukommt oder entfernt wird, ist ein erneuter Importvorgang erforderlich.

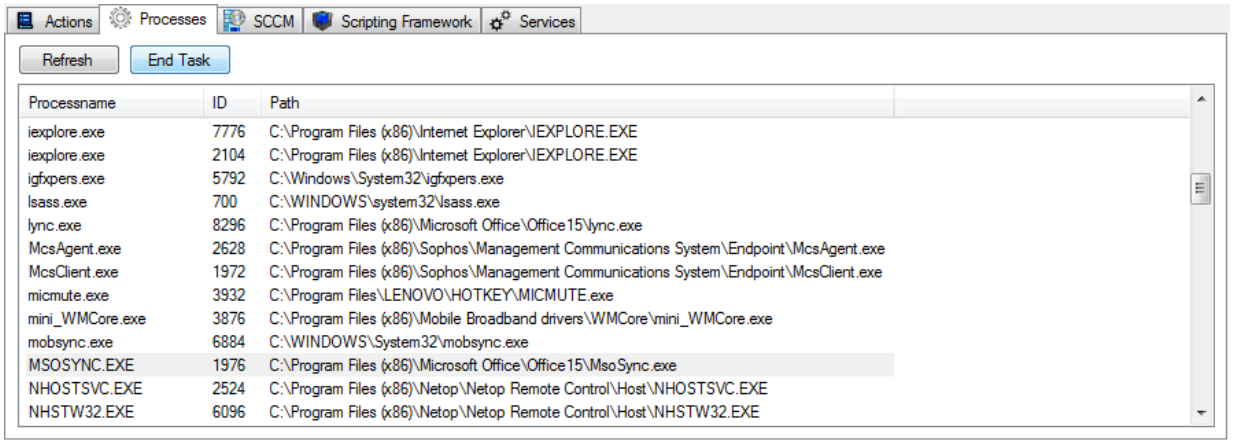
3.7.2 Tab Pages

[Actions]
Unter Actions haben Sie die Möglichkeit häufig benötigte Aktionen auszuführen, wie z.B. das Öffnen des Scripting Framework Logs auf dem entsprechenden Computer. Um eine Aktion auszuführen, müssen Sie sich nicht zwingend mit dem Remote Manager auf den Client verbinden. Es reicht aus, wenn Sie aus den Favoriten einen Client auswählen (kein Doppelklick), oder Sie den Hostnamen im Feld Quick Connect nur eingeben (ohne die Enter Taste zu drücken, oder auf den Button Quick Connect zu klicken).

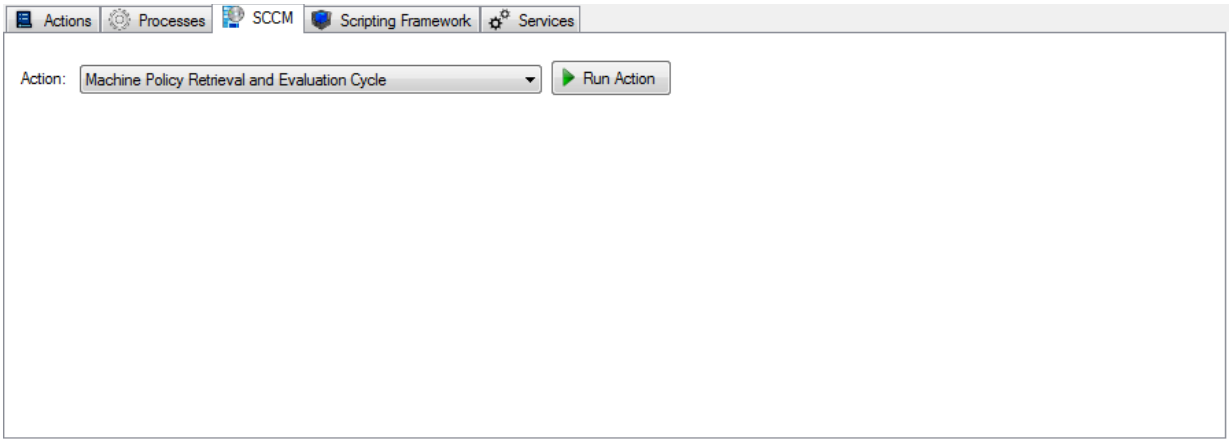
Die verfügbaren Aktionen sind über die Settings.xml Datei im Programmverzeichnis entsprechend erweiterbar.



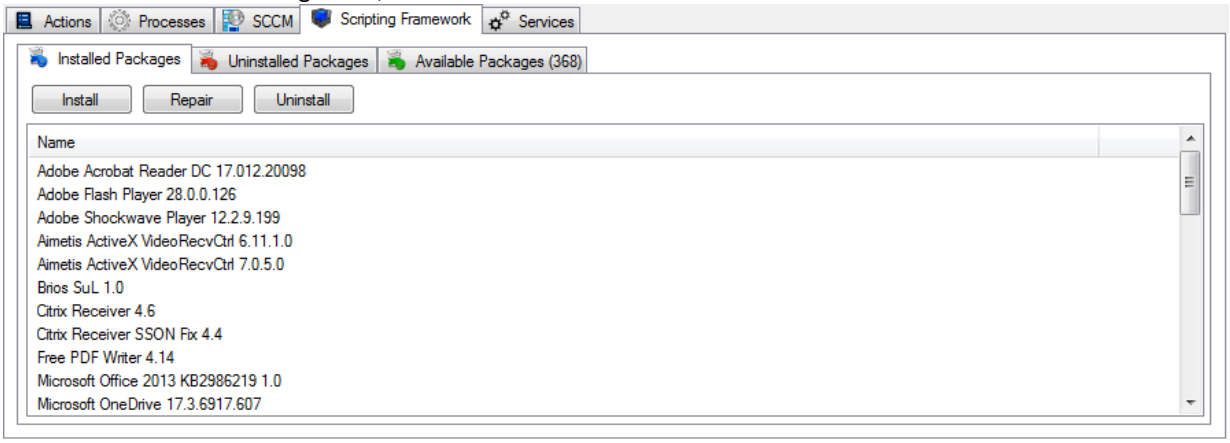
[Processes]
Unter Processes werden die aktuell laufenden Prozesse auf dem verbundenen Client angezeigt mit der Möglichkeit diese zu beenden.



[SCCM]
Ermöglicht das Starten der vorhandenen SCCM Client Actions auf dem verbundenen Client.



[Scripting Framework - Installed Packages]
Hier werden die Pakete aufgelistet, welche auf dem verbundenen Client installiert sind.



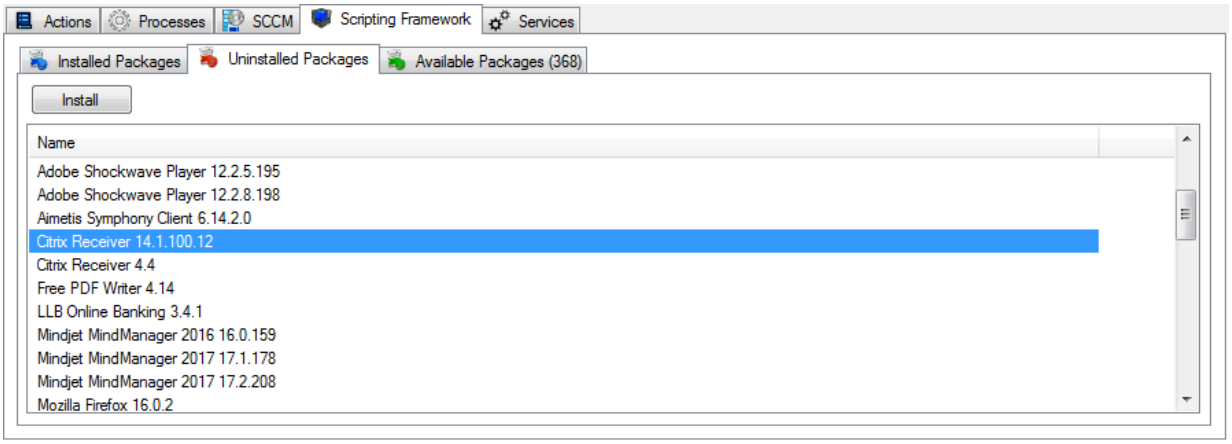
Nachdem Auswählen von einem Paket, stehen Ihnen folgende drei Optionen per Button zur Verfügung:

Install again
Das Paket wird nochmals neu installiert.

Repair
Das Paket wird zuerst deinstalliert und dann nochmals installiert.

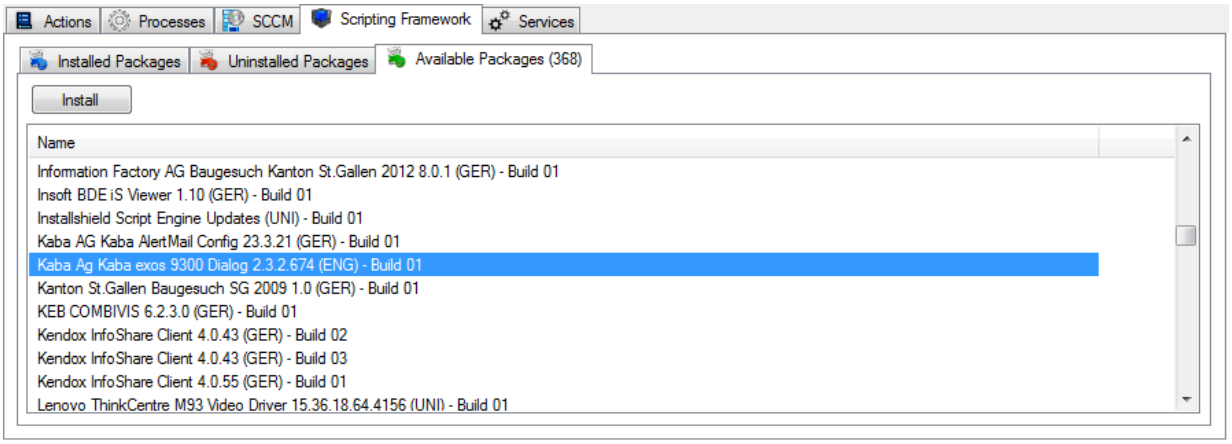
Uninstall
Das Paket wird deinstalliert.

[Scripting Framework - Uninstalled Packages]
Eine Auflistung der Pakete, welche auf dem Gerät deinstalliert wurden.

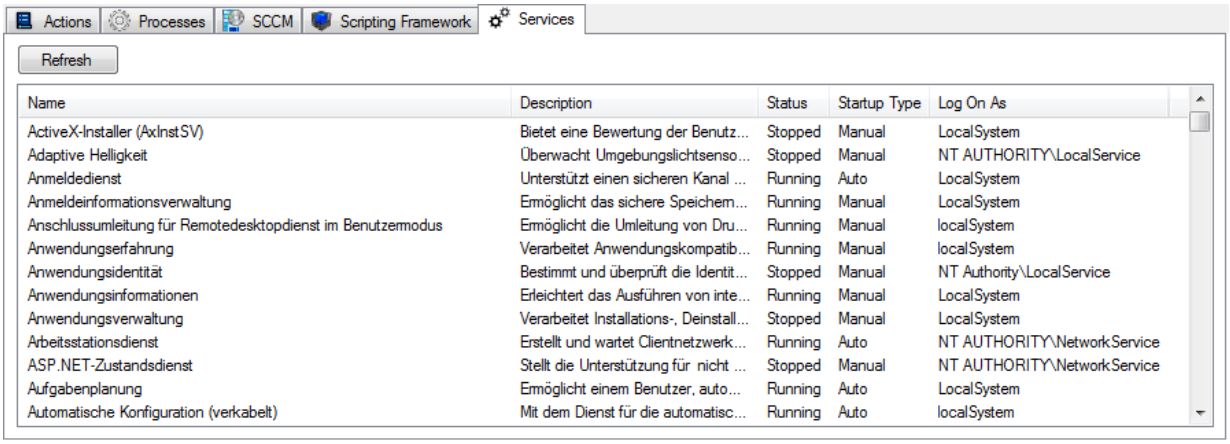


Um ein deinstalliertes Paket wieder zu installieren, wählen Sie es aus und klicken anschliessend auf den Button *Install*.

[Scripting Framework - Available Packages]
Hier sehen Sie alle verfügbaren Scripting Framework Pakete, welche Sie zuvor importiert haben. Wählen Sie das gewünschte Paket aus und klicken Sie auf den Button *Install* um das Paket zu installieren.



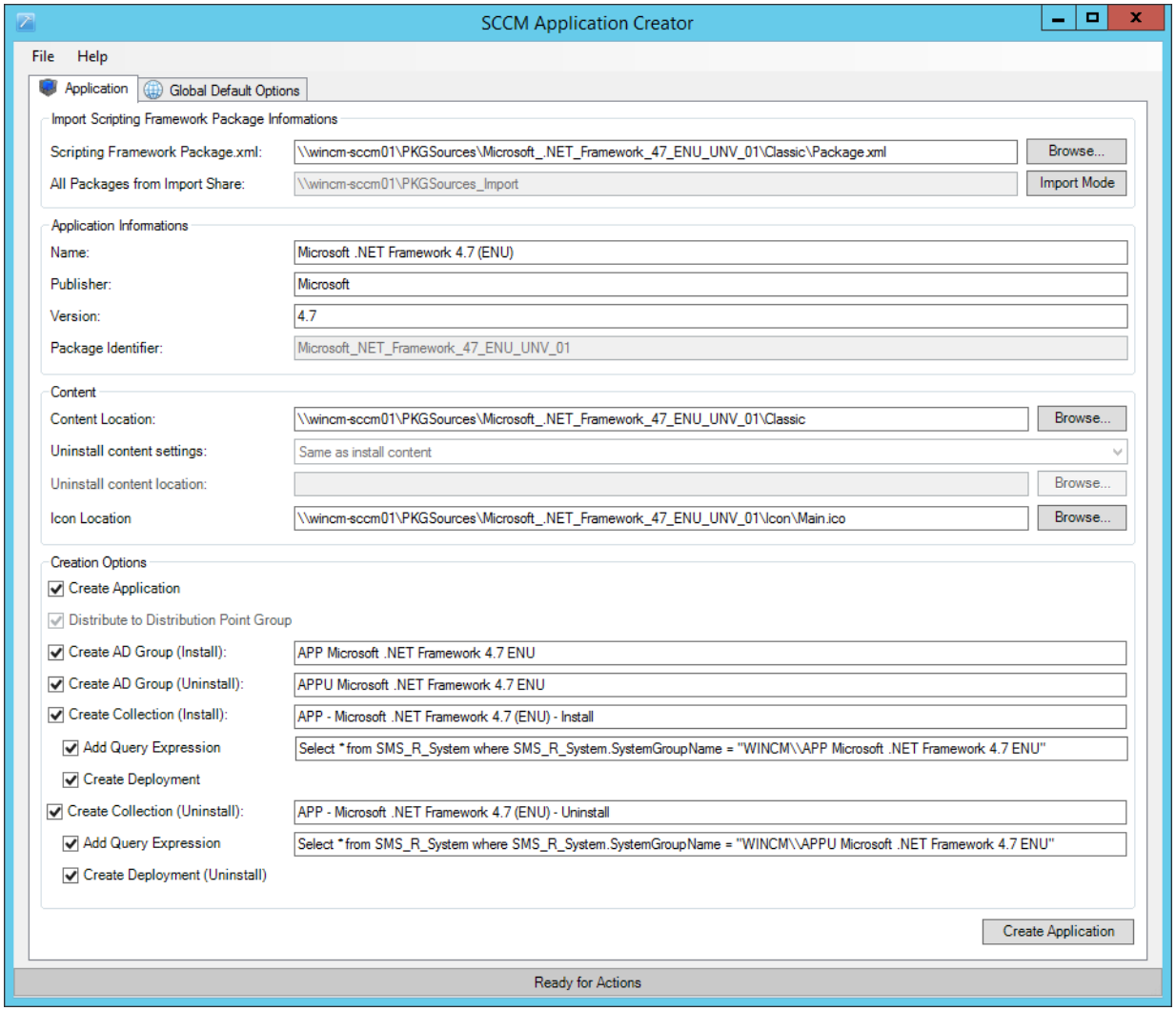
[Services]
Eine Auflistung aller Dienste, welche auf dem verbundenen Gerät vorhanden sind.



Name	Description	Status	Startup Type	Log On As
ActiveX-Installer (AxInstSV)	Bietet eine Bewertung der Benutz...	Stopped	Manual	LocalSystem
Adaptive Helligkeit	Überwacht Umgebungslichtenso...	Stopped	Manual	NT AUTHORITY\LocalService
Anmeldedienst	Unterstützt einen sicheren Kanal ...	Running	Auto	LocalSystem
Anmeldeinformationsverwaltung	Ermöglicht das sichere Speichem...	Running	Manual	LocalSystem
Anschlussumleitung für Remotedesktopdienst im Benutzermodus	Ermöglicht die Umleitung von Dru...	Running	Manual	localSystem
Anwendungserfahrung	Verarbeitet Anwendungskompatib...	Running	Manual	localSystem
Anwendungsidentität	Bestimmt und überprüft die Identit...	Stopped	Manual	NT Authority\LocalService
Anwendungsinformationen	Erleichtert das Ausführen von inte...	Running	Manual	LocalSystem
Anwendungsverwaltung	Verarbeitet Installations-, Deinstall...	Stopped	Manual	LocalSystem
Arbeitsstationsdienst	Erstellt und wartet Clientnetzwerk...	Running	Auto	NT AUTHORITY\NetworkService
ASP.NET-Zustandsdienst	Stellt die Unterstützung für nicht ...	Stopped	Manual	NT AUTHORITY\NetworkService
Aufgabenplanung	Ermöglicht einem Benutzer, auto...	Running	Auto	LocalSystem
Automatische Konfiguration (verkabelt)	Mit dem Dienst für die automatisc...	Running	Auto	localSystem

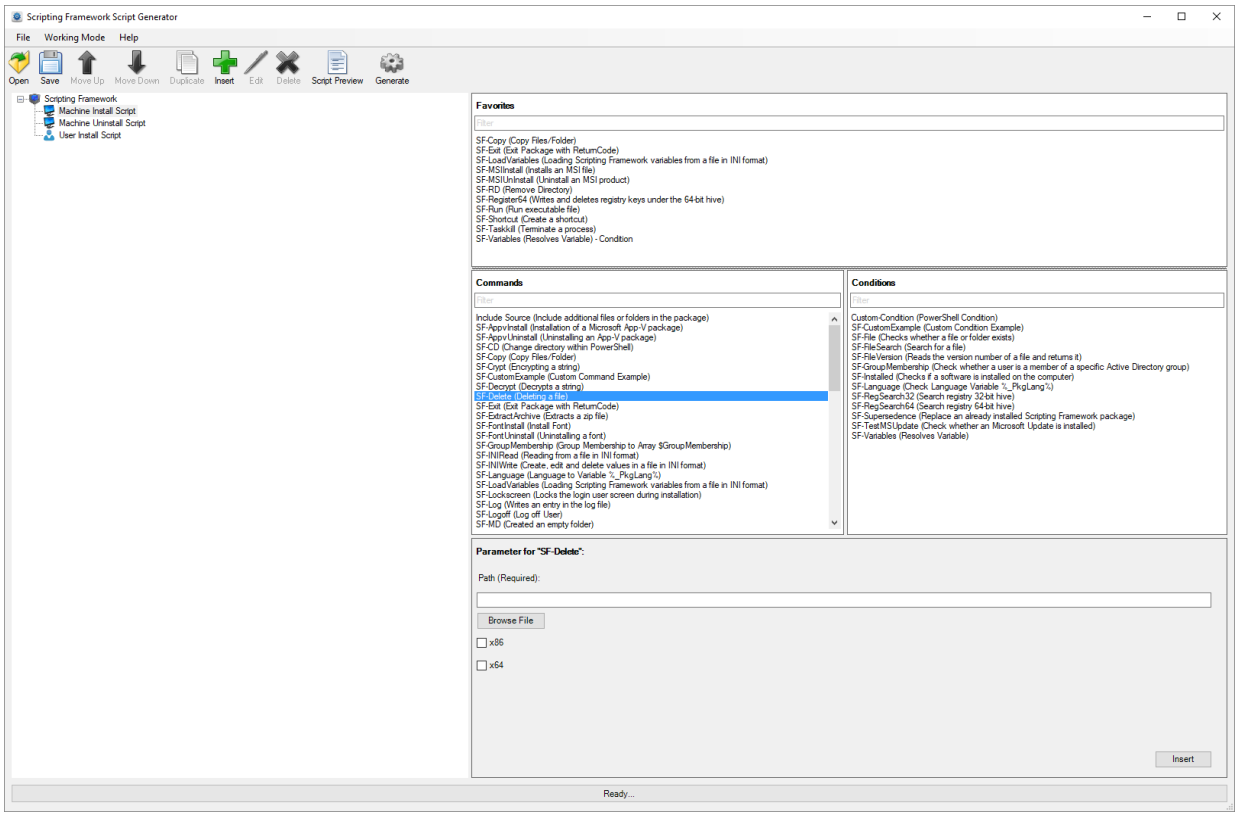
3.8 SCCM Application Creator

Schnell und effizient ein Scripting Framework Package im SCCM als Applikation inkl. Collections, AD Gruppen, etc. erstellen. Die Erstellung erfolgt anhand der von Ihnen definierten Einstellungen in den "Global Default Options".



3.9 Script Generator

Der Script Generator ermöglicht es, den kompletten Code mit den verfügbaren Funktionen von Scripting Framework einfach zu erstellen. Es ist möglich, zu einem späteren Zeitpunkt, jederzeit das erzeugte Paket mit dem Script Generator zu bearbeiten.



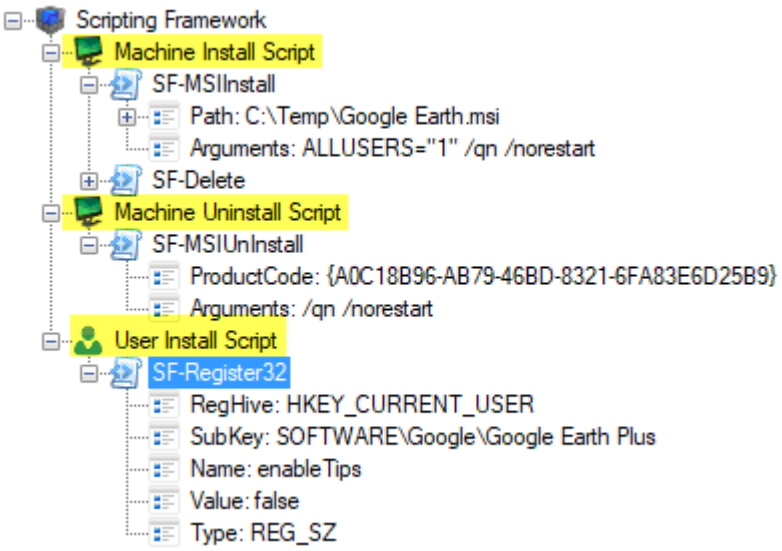
Auf der rechten Seite sehen Sie die verfügbaren *Scripting Framework Funktionen*. Diese werden mit einem einfachen Click angewählt. Im unteren Bereich rechts, werden die verfügbaren Parameter ausgewählt und ergänzt.

Durch klicken auf den *Insert*-Button wird die Funktion in den linken Bereich verschoben, welcher den eigentlichen Script darstellt. Dieser ist unterteilt in die drei Bereiche *[Machine Install Script], [Machine Uninstall Script]* und *[User Install Script]*.

3.9.1 Scriptfenster

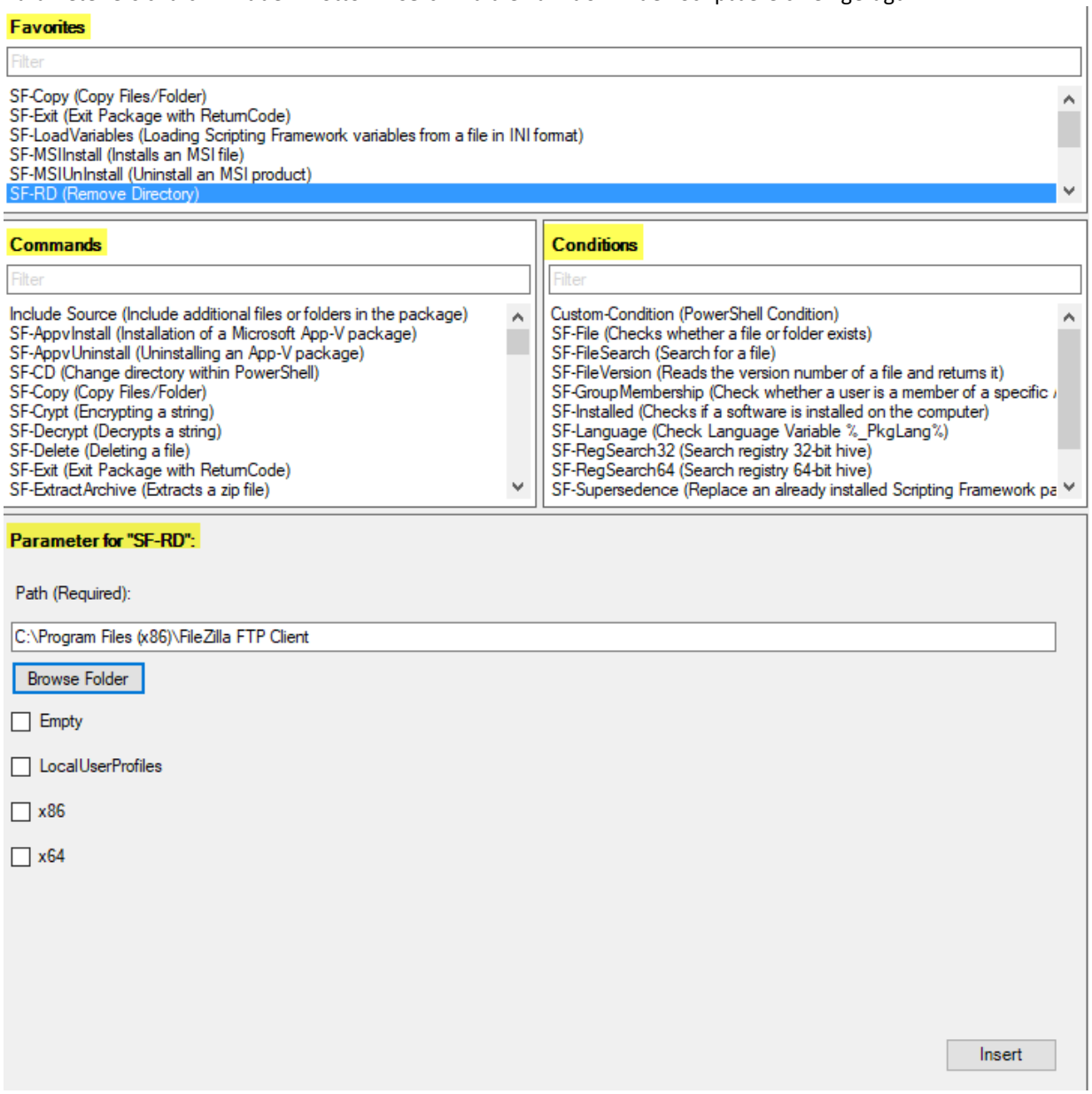
Das Scriptfenster hat drei Bereiche:

- **Machine Install Script** -> entspricht Install.ps1
- **Machine Uninstall Script** -> entspricht Uninstall.ps1
- **User Install Script** -> entspricht InstallUser.ps1

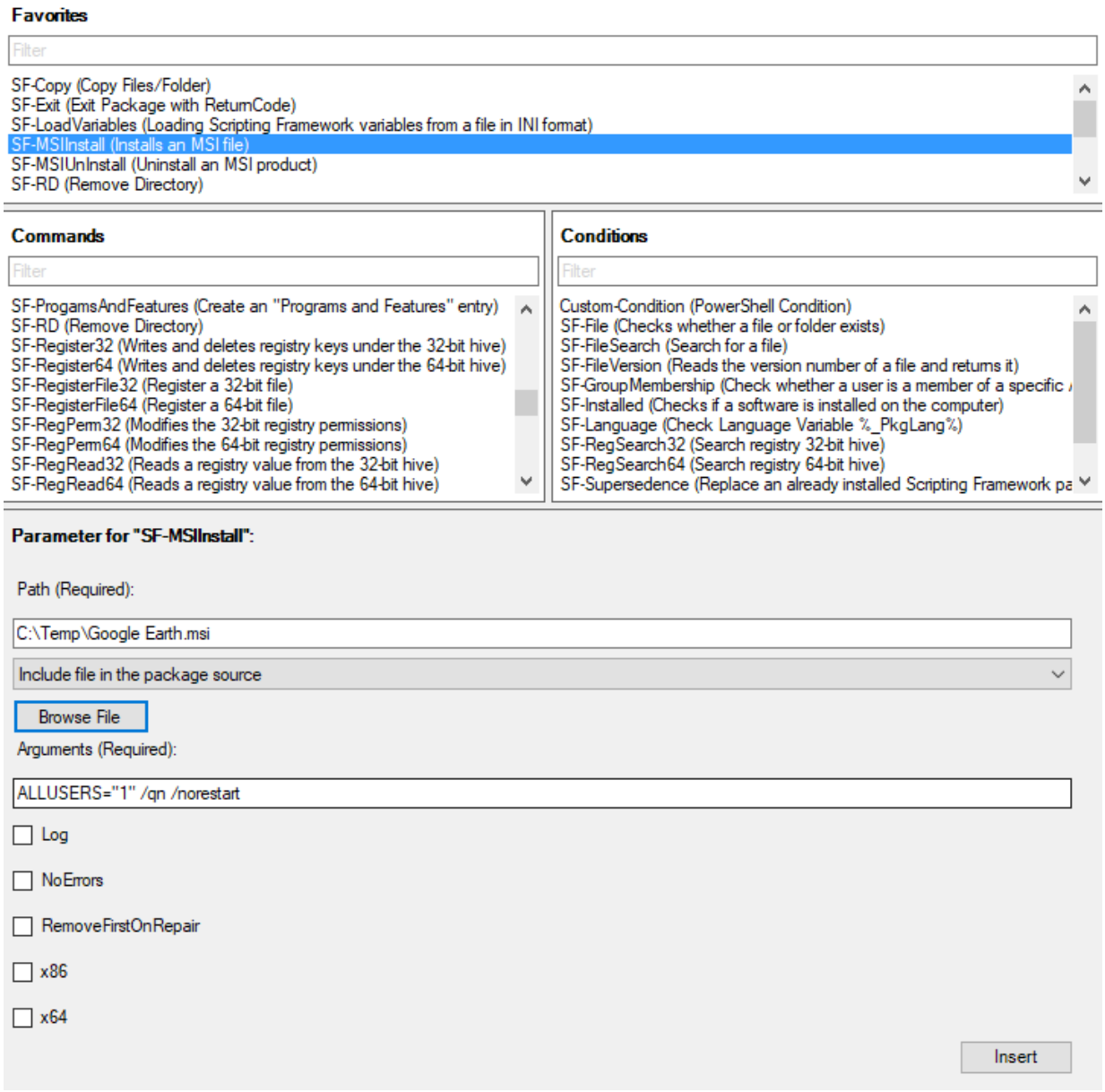


3.9.2 Der Funktionsbereich

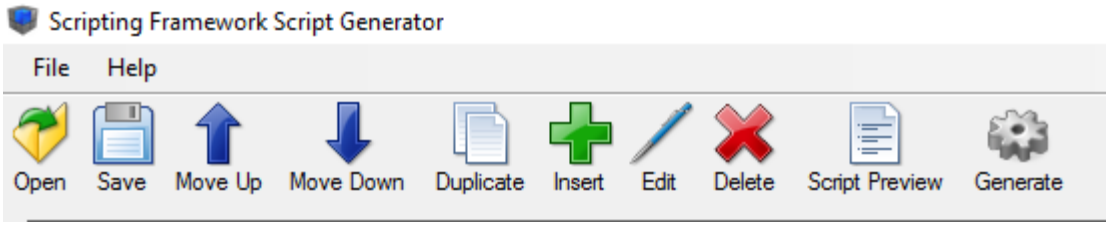
Hier stehen alle Funktionen von Scripting Framework, sowie der möglichen Conditions zur Verfügung.
Bei Auswahl einer Funktion oder Condition werden im unteren Bereich die dafür notwendigen und möglichen Parameter ersichtlich. Mit dem Botton **Insert** wird die Funktion in den Scriptbereich eingefügt.



Beispiel „SF-MSIInstall“

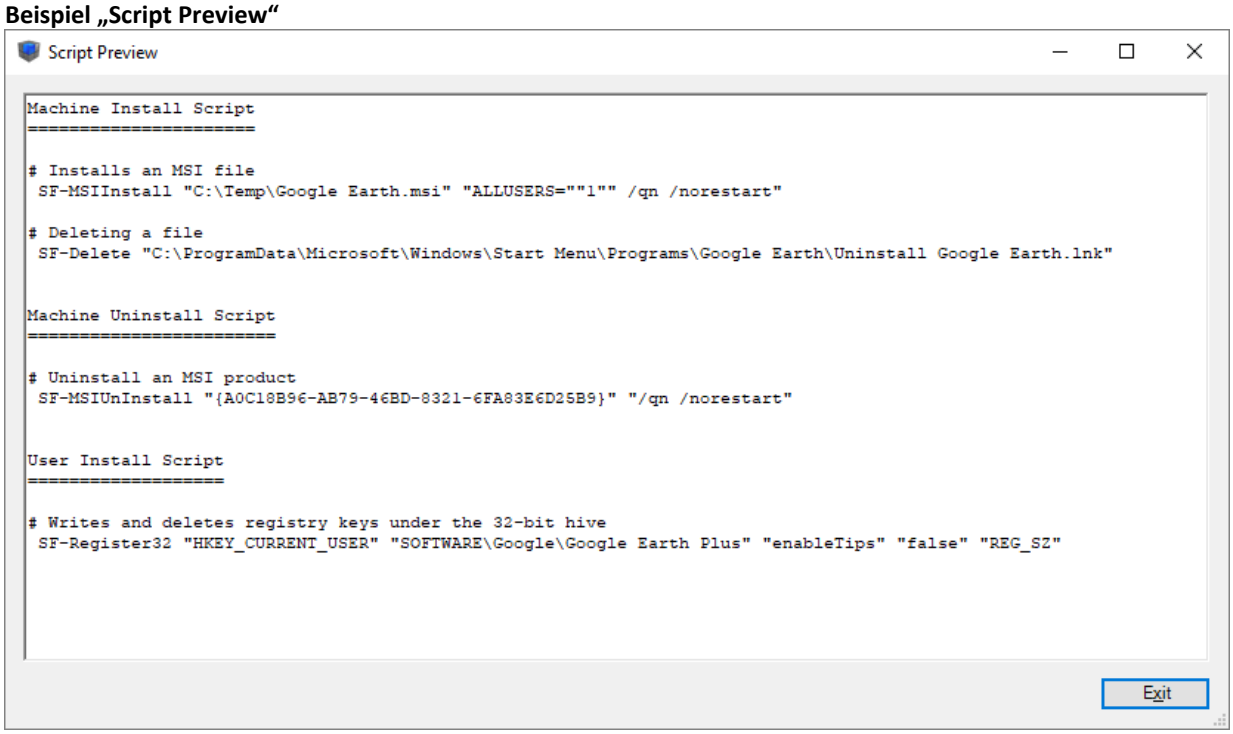


3.9.3 Der Menübalken

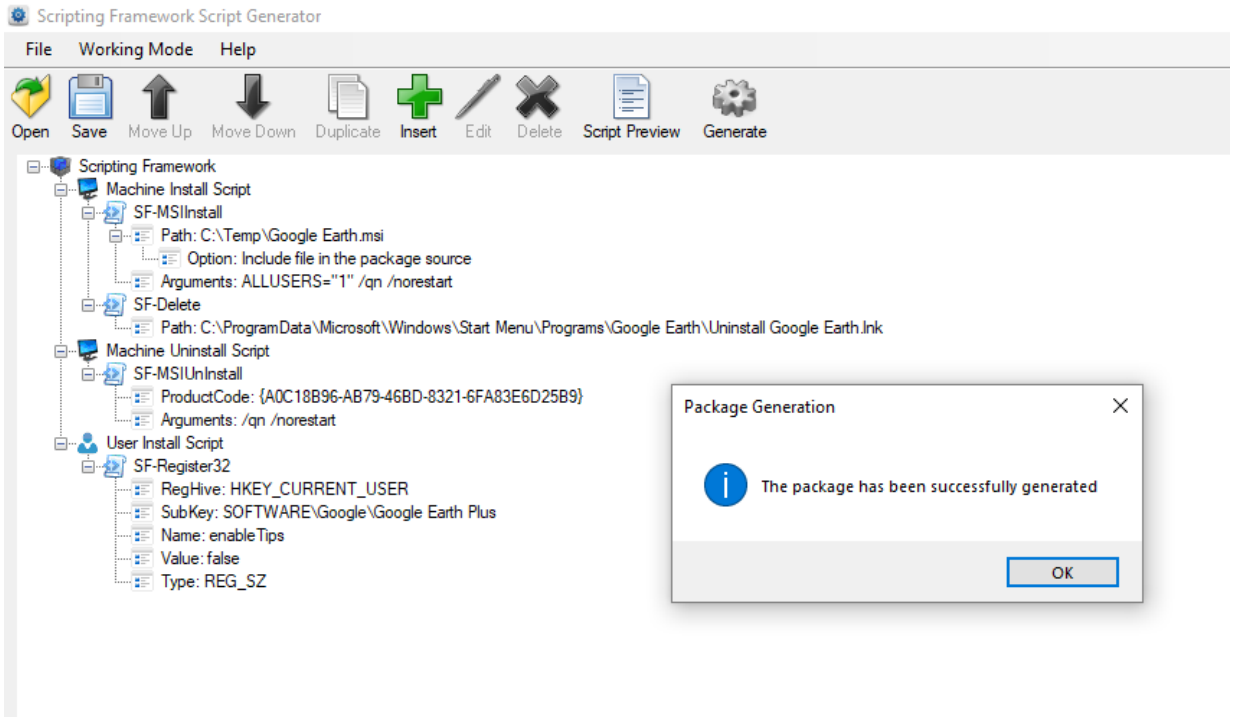


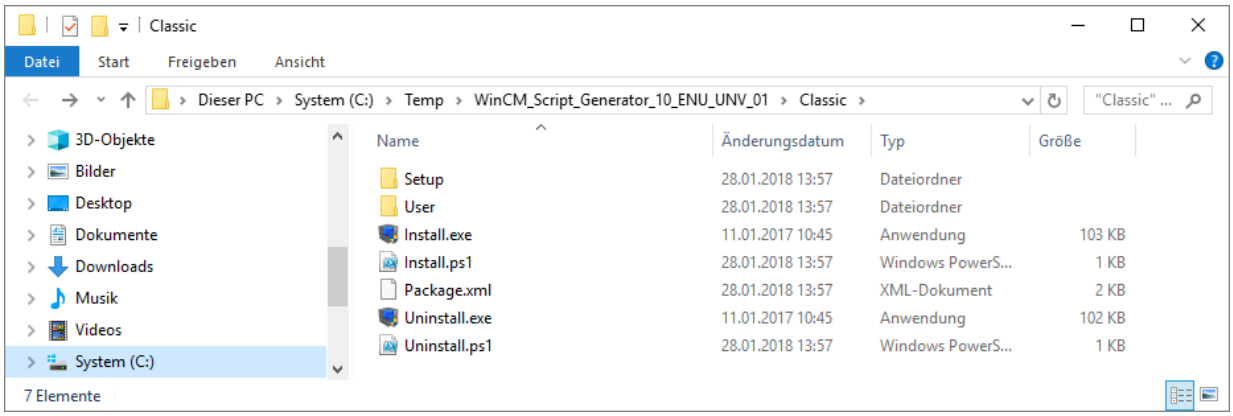
Open – Öffnet ein bestehendes Paket
Save – Speichert das Projekt
Move Up – Verschieben der ausgewählten Funktion im Scriptbereich nach oben
Move Down - Verschieben der ausgewählten Funktion im Scriptbereich nach unten
Duplicate – Kopieren einer Funktion / Condition
Insert – Einfügen der Ausgewählten Funktion aus dem Rechten Fenster in den Scriptbereich
Edit – Editieren der ausgewählten Funktion im Scriptbereich
Delete – Löschen einer Funktion / Condition
Script Preview – Zeigt den Script als Scripting Framework Code an
Generate – Erzeugt das Software Paket

Beispiel „Script Preview“



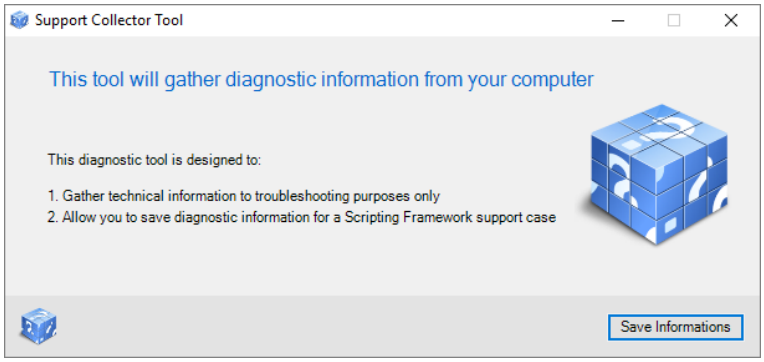
Beispiel „Generate"





3.10 Support Collector Tool

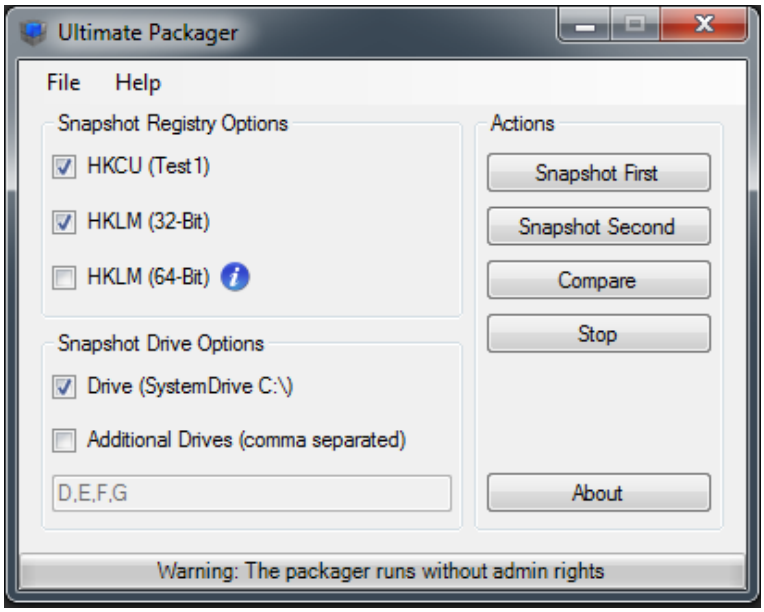
Das Support Collector Tool sammelt alle wichtigen Informationen und Logs, welche für einen Support Case relevant sind.



Um die Informationen zu sammeln klicken Sie auf „Save Informations“. Sobald der Vorgang abgeschlossen ist, erhalten Sie eine Meldung und die Datei wird auf dem Desktop gespeichert:

3.11 Ultimate Package

Mit dem *Ultimate Packager* erstellen Sie schnell und einfach Software-Pakete auf Basis der Scripting Framework Funktionen. Das Tool verwendet das Snapshot-Verfahren. Vor und nach der Installation einer Applikation wird ein Snapshot aufgezeichnet. Aus den Änderungen wird ein funktionsfähiges Paket erstellt.



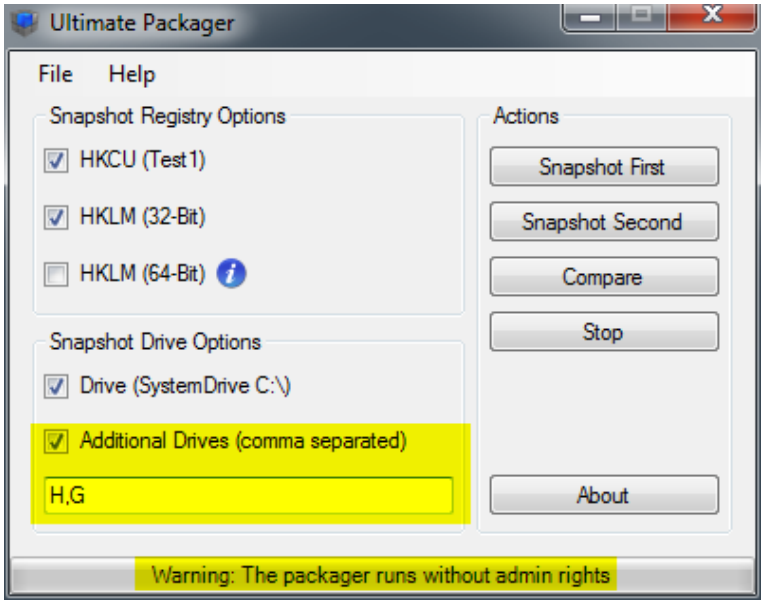
Verwendung

1. *Snapshot First* ausführen
   Es sind verschiede Einstellungen verfügbar. Wählen Sie, welche Bereiche der Registry berücksichtigt werden, sowie mit oder ohne Files inkl. zusätzlichen Laufwerken (nicht nur Drive C).
2. Installation oder Einstellungen auf dem Client manuell durchführen.
3. *Snapshot Second* ausführen
4. *Compare*
   Nun werden die Veränderungen, welche durch eine Installation oder Einstellungen, in einem Scripting Framework Paket aufgezeichnet.

Im Ordner *C:\Scripting_Framework\Suite\Ultimate Packager\Output\Scripting Framework* wird das Paket erstellt und kann für die weitere Verwendung kopiert werden.

Wichtiger Hinweis
Sie können den Ultimate Packager ohne die Suite starten, indem Sie das File *„C:\Scripting_Framework\Suite\Ultimate Packager\UltimatePackager.exe“* ausführen.

Starten Sie den *Ultimate Packager (Suite)* im „Normalfall“ mit lokalen Administratoren Rechten.
Wenn Sie aber für den Snapshot Netzlaufwerke berücksichtigen, wie zum Beispiel das Userhome, dann starten Sie bitte den *Ultimate Packager (Suite)* mit „normalen“ User Rechten.



4 Hilfe und Beispiele aufrufen

4.1 Funktionen innerhalb des Scripts

Beispiel um alle verfügbaren Befehle für den Engineer anzuzeigen:
Get-Command SF-*

```
PS C:\WINDOWS\system32> get-command SF-*

CommandType     Name
-----------     ----
Function        SF-AppvInstall
Function        SF-AppvUninstall
Function        SF-CD
Function        SF-Copy
Function        SF-Crypt
Function        SF-Decrypt
Function        SF-Delete
Function        SF-Exit
Function        SF-File
Function        SF-FontInstall
Function        SF-FontUninstall
Function        SF-GroupMembership
Function        SF-INIRead
Function        SF-INIWrite
Function        SF-Installed
Function        SF-Language
Function        SF-LoadVariables
Function        SF-LockScreen
Function        SF-Log
Function        SF-MD
Function        SF-MSIInstall
Function        SF-MSIPatchInstall
Function        SF-MSIRepair
Function        SF-MSIUninstall
Function        SF-MSIUninstallByDisplayName
Function        SF-NTFSPerm
Function        SF-Path
Function        SF-PinnedApplication
Function        SF-RD
Function        SF-Register32
Function        SF-Register64
Function        SF-RegisterFile32
Function        SF-RegisterFile64
Function        SF-RegPerm32
Function        SF-RegPerm64
Function        SF-RegRead32
Function        SF-RegRead64
Function        SF-RegSearch32
Function        SF-RegSearch64
Function        SF-RemoveVariables
Function        SF-Rename
Function        SF-Replace
Function        SF-Run
Function        SF-Service
Function        SF-ServiceInstall
Function        SF-Set
Function        SF-Shortcut
Function        SF-ShowInstallNotification
Function        SF-SystemReboot
Function        SF-Taskkill
Function        SF-Textfile
Function        SF-UnRegisterFile32
Function        SF-UnRegisterFile64
Function        SF-Variables
Function        SF-Wait
Function        SF-WUSAInstall
```

Jede Funktion verfügt über eine Hilfe und entsprechende Beispiele.

Beispiel um die Hilfe aufzurufen:
Get-Help SF-Exit

```
PS C:\WINDOWS\system32> Get-Help SF-Exit

NAME
    SF-Exit

ÜBERSICHT
    Exit the PowerShell script


SYNTAX
    SF-Exit [[-Message] <Object>] [<CommonParameters>]


BESCHREIBUNG
    Write a log entry and closes the PowerShell Script with the ExitCode 1603


VERWANDTE LINKS


HINWEISE
    Zum Aufrufen der Beispiele geben Sie Folgendes ein: "get-help SF-Exit -examples".
    Weitere Informationen erhalten Sie mit folgendem Befehl: "get-help SF-Exit -detailed".
    Technische Informationen erhalten Sie mit folgendem Befehl: "get-help SF-Exit -full".
    Geben Sie zum Abrufen der Onlinehilfe Folgendes ein: "get-help SF-Exit -online"
```

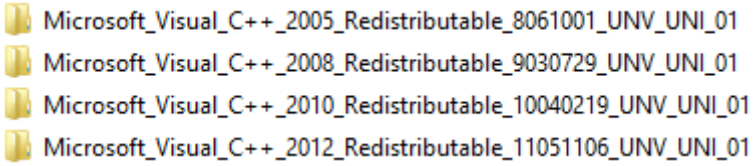
Beispiel um die Beispiele anzuzeigen:
Get-Help SF-Exit -Examples

```
PS C:\WINDOWS\system32> Get-Help SF-Exit -Examples

NAME
    SF-Exit

ÜBERSICHT
    Exit the PowerShell script

    -------------------------- BEISPIEL 1 --------------------------

    PS C:\>SF-Exit "My Example Message"


    -----------
    Description
    Exit the Script with the log entry "My Example Message"
```

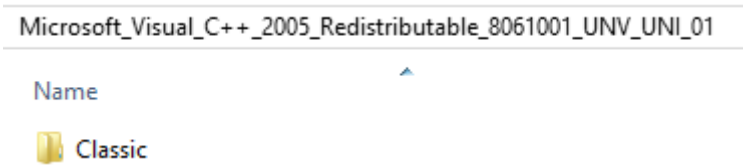
5 Scripting Framework Software Paket (Definition)

5.1 Paketname (Main Folder)

Jedes Paket wird in einem eigenen Ordner abgelegt:



Darunter empfehlen wir nochmals eine Unterteilung, diese ist jedoch keine Voraussetzung. Wir unterscheiden dabei jeweils zwischen Classic und AppV:



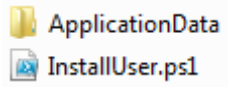
Classic entspricht einem klassischen Paket (MSI, Silent Installation, Snapshot) und AppV einem Microsoft AppV Paket.

5.2 Ordner- und Dateistruktur

Die Hauptstruktur eines Paketes unter „Classic“ oder „AppV“ sieht wie folgt aus:

Config Setup User Install.exe Install.ps1 Package.xml Uninstall.exe Uninstall.ps1	**Config**	Konfigurationsdatei welche zu dem Paket gehört (.cfg)
	Setup	Source der Setup Dateien (MSI, Silent Setups, etc.)
	User	User spezifische Dateien für die Benutzereinstellungen.
	Install.exe	Die Datei wird für die Installation des Paketes aufgerufen.
	Install.ps1	In dieser Datei befindet sich die Logik des Pakets (Installation).
	Package.xml	Definition des Paketes (Hersteller, Produktname, Version, etc.)
	Uninstall.exe	Die Datei wird für die Deinstallation des Paketes aufgerufen.
	Uninstall.ps1	In dieser Datei befindet sich die Logik des Pakets (Deinstallation).

5.2.1 User Struktur - Unter dem Ordner „User“

	**ApplicationData**	Als Beispiel könnten mittels der InstallUser.ps1 der Ordner ApplicationData nach %_ApplicationData% kopiert werden.
	InstallUser.ps1 **oder** **Disabled_InstallUser.ps1**	Die Datei beinhaltet die Logik für Benutzer spezifische Vorgänge (z.B. das Schreiben von einem Registry Key, das Kopieren von Dateien, etc.).

Sofern die Datei „User\InstallUser.ps1“ existiert, wird während der Installation automatisch ein Eintrag für die Benutzereinstellungen in der Registry gesetzt und der benutzerspezifische Inhalt wird nach „C:\Windows_ScriptingFramework\Cache“ ausgelagert. Scripting Framework regelt den kompletten Ablauf automatisch, es sind keine weiteren Aktionen durch den Paketierer notwendig. Um den Benutzerteil zu deaktivieren muss die Datei entsprechend umbenannt werden („Disabled_InstallUser.ps1“). Scripting Framework verlangt zwingend eine Datei, löschen Sie diese deshalb nicht!

5.3 Details der einzelnen Dateien

5.3.1 Install.exe

Kommentar:
Startet Scripting Framework und verarbeitet die Install.ps1. Anschliessend wird die EXE mit dem entsprechenden Return Code beendet.

5.3.2 Install.ps1

Kommentar:
Beinhaltet das Installationsscript. Ein Beispiel von einer MSI Installation:

Inhalt:

```
# ================================================================================================
# Installation
# ================================================================================================

# Terminate Process
 SF-Taskkill "AcroRd32.exe"

# Language
  SF-Language "1031,1033,1036,1040"

# Installation Reader
 SF-MSIInstall "%_PkgSource%\Setup\AcroRead.msi"
"TRANSFORMS='%_PkgSource%\Setup\%_PkgLang%.mst' ALLUSERS='1' DISABLEDESKTOPSHORTCUT='1'
EULA_ACCEPT='YES' SUPPRESS_APP_LAUNCH='YES' /qn /norestart"

# Installation Language Packs
 SF-MSIInstall "%_PkgSource%\Setup\FontPack11000_XtdAlSF-Lang.msi" "ALLUSERS='1' /qn
/norestart"

# Installation Patches
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11001_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11002.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11003_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11004_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11005.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11006_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11007_MUI.msp" "ALLUSERS='1' /qn
/norestart"
```

```
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11008.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11009_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 SF-MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11010_MUI.msp" "ALLUSERS='1' /qn
/norestart"

# Delete Registry
 SF-Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe
Reader Speed Launcher" "{nil}" "REG_SZ"
 SF-Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe ARM"
"{nil}" "REG_SZ"
 SF-Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
"AdobeAAMUpdater-1.0" "{nil}" "REG_SZ"

# Disable and Stop Service
 SF-Service "AdobeARMservice" -Stop
 SF-Service "AdobeARMservice" -Disabled

# Modify Registry
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\AdobeViewer" "EULA" "1"
"REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\AdobeViewer" "Launched"
"1" "REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableAcrobatHS" "0" "REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableShareFile" "0" "REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableRTCAuth" "0" "REG_DWORD"

# Delete Updater
 SF-Delete "%_ProgramFiles32%\Adobe\Reader 11.0\Reader\plug_ins\Updater.api"
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\FeatureLockDown"
"bUpdater" "0" "REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\FeatureLockDown"
"bCreatePDFOnline" "0" "REG_DWORD"
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe ARM"
"{NIL}" "REG_SZ"
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Adobe ARM\1.0\ARM" "iCheck" "0" "REG_DWORD"

# Disable Product Improvement Program
 SF-Register32 "Hkey_Local_Machine" "SOFTWARE\Policies\Adobe\Acrobat
Reader\11.0\FeatureLockDown" "bUsageMeasurement" "0" "REG_DWORD"

# ================================================================================================
# END
# ================================================================================================
```

5.3.3 Package.xml

Kommentar:
Im XML werden die wichtigsten Informationen über das Paket hinterlegt. Die Angaben werden für diverse Tasks im Scripting Framework verwendet. Zum Beispiel für MSI Log Dateien, Config Dateien, Package Identifier, etc.

Zu definierende Werte im XML:

Name:	Wert:
BuildNumber	Entspricht der Build Version des Paketes. Diese wird für die Unterscheidung benötigt, sofern das Paket bereits in der identischen Version existiert. Dies ist in der Regel bei einer Erweiterung des Paketes oder einer anderen Änderung der Fall.
Company	Angabe ob das Paket nur für eine bestimmte Firma oder einen Standort gedacht ist. Der Wert ist frei definierbar. Für ein Paket welches unabhängig eingesetzt werden kann, empfehlen wir die Abkürzung UNV für „universal“. Sollte ein Paket spezielle Settings für einen Standort/Firma beinhalten, verwenden wir in der Regel eine dreistellige Abkürzung. Für Zürich z.B. ZHR.
Manufacturer	Hersteller der Software
ProductName	Produktname der Software
Version	genaue Version der Software
VersionShort	Hauptversionsnummer
Language	Sprache der Software. Wir empfehlen die Namenskonvention von Microsoft. Beispiele: Deutsch = DEU Französisch= FRA Italienisch = ITA Englisch = ENU Mehrsprachig = MUI Universal = UNI → Dabei handelt es sich zum Beispiel um ein Konfigurationspaket, welches Einstellungen setzt und keine eigentliche Sprache besitzt.

Aufbau der XML Datei (Inhalt):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 # ================================================================================================
 # (c) Copyright by Windows Client Management - www.wincm.ch
 # ================================================================================================
 #
 # Package created :   01.01.2015 by : Windows Client Management AG
 # Package changed : 00.00.0000 by :
 #
 # ================================================================================================

 # ================================================================================================
 # Prerequisites:
 # - None
 # ================================================================================================
-->

<!-- Package Definition -->
<Package>
        <BuildNumber>01</BuildNumber>
        <Company>UNV</Company>
        <Manufacturer>Adobe</Manufacturer>
        <ProductName>Reader</ProductName>
        <Version>11.0.10</Version>
        <VersionShort>11</VersionShort>
        <Language>MUI</Language>
</Package>
```

5.3.4 Uninstall.exe

Kommentar:
Startet Scripting Framework und verarbeitet die Uninstall.ps1. Anschliessend wird die EXE mit dem entsprechenden Return Code beendet.

5.3.5 Uninstall.ps1

Kommentar:
Beinhaltet das Uninstallationsscript. Ein Beispiel von einer MSI Deinstallation:

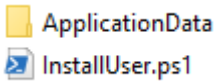
Inhalt:
```
# ================================================================================================
# Uninstallation
# ================================================================================================

# Terminate Process
 SF-Taskkill "AcroRd32.exe"

# MSI Uninstall Adober Reader
 SF-MSIUninstall "{AC76BA86-7AD7-FFFF-7B44-AB0000000001}" "/qn /norestart"

# MSI Uninstall Software Reader FontPack
 SF-MSIUninstall "{AC76BA86-7AD7-2530-0000-A00000000004}" "/qn /norestart"


# ================================================================================================
# END
# ================================================================================================
```

5.3.6 InstallUser.ps1

Kommentar:
Beinhaltet die Benutzerkonfiguration. Beispiel:

Inhalt:
```
# ================================================================================================
# Install User
# ================================================================================================

# Copy Files
 SF-Copy "%_PkgCache%\User\ApplicationData" "%_ApplicationData%"

# Registry write
 SF-Register32 "HKey_Current_User" "Software\Adobe\Acrobat Reader\11.0\AdobeViewer" "EULA" "1"
"REG_DWORD"
 SF-Register32 "HKey_Current_User" "Software\Adobe\Acrobat Reader\11.0\AVGeneral"
"bBrowserDisplayInReadMode" "0" "REG_DWORD"

# ================================================================================================
# END
# ================================================================================================
```

Dateistruktur (Classic\User):



6 Scripting Mode

Alle Funktionen von Scripting Framework können durch den „Scripting Mode“ ausserhalb einer Pakethülle verwendet werden. Der Anwendungsbereich dafür sind generelle Scripts, Launcher oder Konfigurationen innerhalb eines App-V Paketes.

Um einen Script im „Scripting Mode“ auszuführen, wird eine beliebige .ps1 Datei angelegt. Diese darf **nicht** den Namen Install.ps1, Uninstall.ps1 oder InstallUser.ps1 verwenden:

```
# ================================================================================
# Scripting Framework Script
# ================================================================================

# Example Parameter
SF-Log "$Parameter1"

# Example (Start Notepad)
SF-Run "%_Windows%\notepad.exe" "$Parameter1" -Wait -NoErrors -Show

# ================================================================================
# END
# ================================================================================
```

Die .ps1 Datei lässt sich mit folgendem CMD Befehl ausführen:
"%SystemRoot%_ScriptingFramework\Modul\ScriptingFramework.exe" -File "C:\Temp\ScriptingModeExample.ps1" -Parameter1 "%WINDIR%\WindowsUpdate.log"

In diesem Beispiel wird der *Parameter1* mit dem Wert *C:\Windows\WindowsUpdate.log* an das Script übergeben, welcher anschliessend für den Run von Notepad verwendet wird. Es können beliebig viele Parameter dem Script übergeben werden, der Parametername entspricht jeweils dem Namen der PowerShell Variabel.

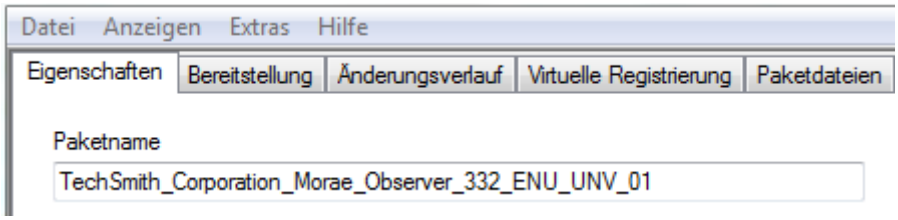
6.1 Einsatzbereich - Launcher

Um Konfigurationen vor dem Start einer Applikation repetitiv zu schreiben, ist der Einsatz von sogenannten Launchern möglich. Dazu wird der originale Shortcut der Applikation auf den Launcher umgebogen und der Launcher Script setzt zuvor die gewünschten Einstellungen und startet anschliessend die eigentliche Applikation. Auch in diesem Szenario ist es möglich dynamische Variablen mittels *SF-LoadVariables* aus einer CFG Datei zu laden und anhand von diesen Aktionen durchzuführen. Aufgrund der fehlenden XML Datei, ist innerhalb eines Launchers der *PkgIdentifier* nicht bekannt. Der *PkgIdentifier* wird auf den Wert *Scripting_CurrentProccessID* gesetzt. Jedoch kann der Identifier per Command Line Parameter *-PkgIdentifier* an den Scripting Mode übergeben werden, wodurch ebenfalls die Variable *%_PkgCFGName%* automatisch korrekt gesetzt wird. Laden Sie sich hier ein Beispiel herunter.

Innerhalb des Launchers können wie gewohnt alle Funktionen verwendet werden, für administrative Tasks muss der ausführende Benutzer über die entsprechenden Berechtigungen verfügen.

6.2 Einsatzbereich - App-V

Bei der Anwendung des Scripting Modes innerhalb eines App-V Paketes, entspricht die Variabel %_PkgCFGName% dem App-V Paketname:



Der Befehl *SF-LoadVariables* lässt sich dadurch in einem App-V Paket analog wie in einem Classic Paket anwenden. Innerhalb eines Scripts für App-V, stehen wie gewohnt alle Scripting Framework Funktionen zur Verfügung.

Beispiel XML für einen Aufruf aus einem App-V Paket:

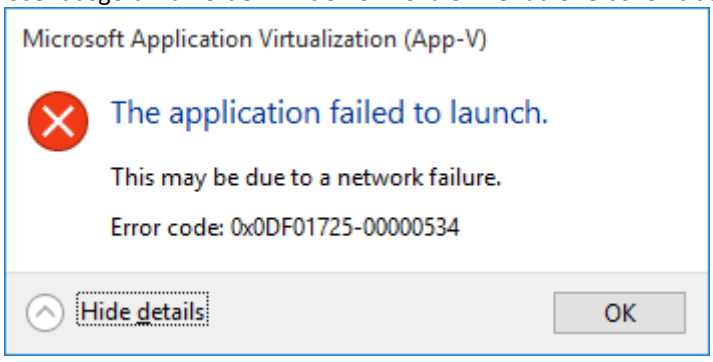
```
<!-- User Scripts -->
<UserScripts>
 <StartProcess RunInVirtualEnvironment="true">
  <Path>"C:\Windows\_ScriptingFramework\Modul\ScriptingFramework.exe"</Path>
  <Arguments>-File "[{AppVPackageRoot}]\..\Scripts\Launcher.ps1"</Arguments>
  <Wait RollbackOnError="true" />
  <ApplicationId>[{AppVPackageRoot}]\MoraeObserver.exe</ApplicationId>
 </StartProcess>
</UserScripts>
```

Um innerhalb eines virtuellen App-V Environments das Root anzusprechen, steht die Scripting Framework Variable *%_AppVPackageRoot%* zur Verfügung. Ebenso ist die Verwendung der PowerShell Variabel *$AppVPackageRoot* möglich.

Wichtig: Der App-V Clients muss die Ausführung von Scripts erlauben. Die Konfiguration kann einmalig per PowerShell oder GPO gesetzt werden. PowerShell Command:

```
Set-AppvClientConfiguration -EnablePackageScripts $true
```

Bis und mit dem App-V Client 5.1.86.0 können App-V Pakete mit integrierten Scripts nur unter einem Domain User ausgeführt werden. Mit einem lokalen Benutzer erscheint beim Aufruf der Applikation eine Fehlermeldung:

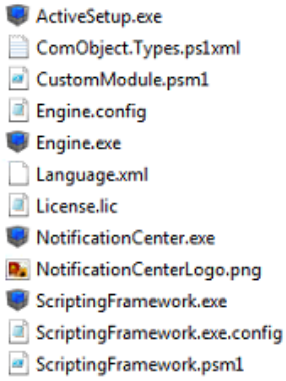


Wir hoffen, dass Microsoft das Problem mit einem Hotfix beheben wird.

7 Aufbau des Toolkits und weitere Informationen

7.1 Dateien

Folgende Dateien werden vom Scripting Framework benötigt und werden nach „C:\Windows_ScriptingFramework“ installiert:



Inhalt Ordner „Modul_x86“ und „Modul_x64“

ActiveSetup.exe
Verarbeitet die ausstehenden Scripting Framework Benutzereinstellungen.

ComObject.Types.ps1xml
Die Datei ist eine XML-basierte Textdatei, mit der den Objekten von Windows PowerShell, Eigenschaften und Methoden hinzufügt werden. Diese wird intern benötigt.

CustomModule.psm1
Beinhaltet die selber erstellten Scripting Framework Funktionen. Details sind im Custom Module Handbuch zu finden.

Engine.exe | Engine.config
Beinhaltet die Scripting Framework Engine.

Language.xml
Übersetzungstexte für das Notification Center.

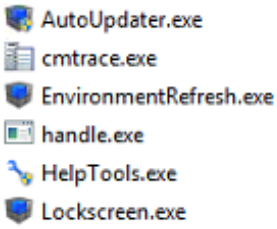
License.lic
Darin befinden sich die Lizenzangeben, welche Sie von uns erhalten.

NotificationCenter.exe
Zeigt dem Benutzer die entsprechenden Benachrichtigungen an, sofern eine entsprechende Funktion verwendet wird.

NotificationCenterLogo.png
Das Notification Center Logo, dies kann mit einem eigenen Bild ersetzt werden (480x50)

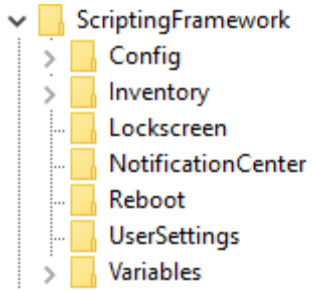
ScriptingFramework.exe | ScriptingFramework.exe.config
Hauptdatei, welche den Aufruf der Engine handelt.

ScriptingFramework.psm1
Reduziertes PowerShell Script welcher für die Entwicklung (Engineering) von Paketen gebraucht wird.

	**Inhalt Ordner „Tools“** **AutoUpdate.exe** Startet die Autoupdate Funktionalität. **cmtrace.exe** Zum Anzeigen der Log Datei. **EnvironmentRefresh.exe** Aktualisiert die User Environment Variablen. Das Tool wird durch Scripting Framework automatisch angesprochen. **Handle.exe** Tool von Sysinternals (Microsoft) um offene Handles zu beenden. **HelpTools.exe** Hilfe Tool von Scripting Framework, womit die Konfiguration für die Installation erstellt wird. **Lockscreen.exe** Um den Bildschirm des Benutzers zu sperren, dies wird für die Funktion SF-Lockscreen verwendet.

7.2 Registry

Unter „HKLM\Software*Wow6432Node*\ScriptingFramework“ befindet sich folgende Struktur:

	Config	Konfigurationsparameter von Scripting Framework.
	Inventory	Nach der erfolgreichen Installation von einem Paket, werden die Informationen unter Inventory abgelegt.
	NotificationCenter	Interne Einträge für das Notifcation Center.
	Reboot	Zwischenspeicherung des Reboot Flags.
	UserSettings	Informationen über die Pakete mit Benutzereinstellungen.
	Variables	Alle relevanten Scripting Framework Variablen, welche innerhalb eines Scripts verwendet werden können.

7.2.1 Config

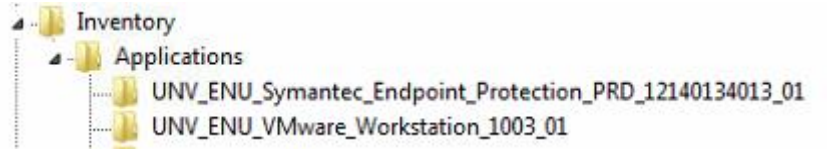
Folgende Konfigurationswerte existieren:

Name:	Wert:	Beschreibung
DisableAutoupdate	0 oder 1	Deaktiviert die automatische Aktualisierung von Scripting Framework.
DisableImmediateUserInstallation	0 oder 1	Deaktiviert die automatische Abarbeitung der User Einstellungen (InstallUser.ps1) nach einer Paket Installation.
DisableIsRegistryPOLGood	0 or 1	Per Default prüft SF auf eine korrupte Registry.pol Datei und korrigiert diese, da dies zu einem Absturz von SF führt. Mit dem Wert 1 lässt sich die Prüfung und Korrektur deaktivieren.
DisableNotificationCenter	0 oder 1	Mit dem Wert 1, lässt sich das Notification Center komplett deaktivieren. Der Parameter -Notification wird jeweils ignoriert.
DisableRebootExitCode	0 oder 1	Sofern ein Scripting Framework Paket ein Reboot anfordert, wird anstelle des Return Codes 3010 der Code 0 zurückgegeben. Dies wird in der Regel nur im Citrix Umfeld aktiviert.

DisableShowUserSettings	0 oder 1	Kein sichtbares Fenster während der Installation von Benutzereinstellungen.
ForceMachineLanguage	1031 1033 1036 1040	Fixiert die Sprache für die Funktion SF-Language. Dies wird in der Regel nur auf einem sprachspezifischen Citrix benötigt.
LogPathMachine	${env:WINDIR}_ScriptingFramework\Logs	Path of the Scripting Framework Log files (Machine Actions)
LogPathUser	${env:LOCALAPPDATA}_ScriptingFramework\Logs	Path of the Scripting Framework Log files (User Actions)
NotificationTimeoutShowInstallNotification	0 oder x	Der Notification Timeout (SF-ShowInstallNotification) in Sekunden, sofern keine weitere Verschiebung der Installation mehr möglich ist. Nach diesem Timeout (wird dem Benutzer als Countdown angezeigt), wird die Installation fortgesetzt. Mit dem Wert 0 wird der automatische Timeout deaktiviert, was jedoch nicht die Empfehlung ist.
NotificationTimeoutTaskkill	0 oder x	Notification Timeout (SF-Taskkill) in Sekunden bis die Tasks automatisch geschlossen werden. Mit dem Wert 0 wird der automatische Timeout deaktiviert, was jedoch nicht die Empfehlung ist.
NotificationTopMost	0 oder 1	Soll das Notification Center permanent im Vordergrund angezeigt werden. 0=Aus 1=Ein
Password	**********	Passwort des Service Accounts für den Share Zugriff
PkgIdentifier	Manufacturer|ProductName|Version|Language|Company|BuildNumber	Mit diesem Wert wird der PkgIdentifier bestimmt
Username	Domain.local\User	Username des Service Accounts für den Share Zugriff

7.2.2 Inventory

Nach der Installation von einem Paket, sind dessen Informationen im Application Inventory abgelegt:



Zu jedem Scripting Framework Paket sind folgende Details zu finden:

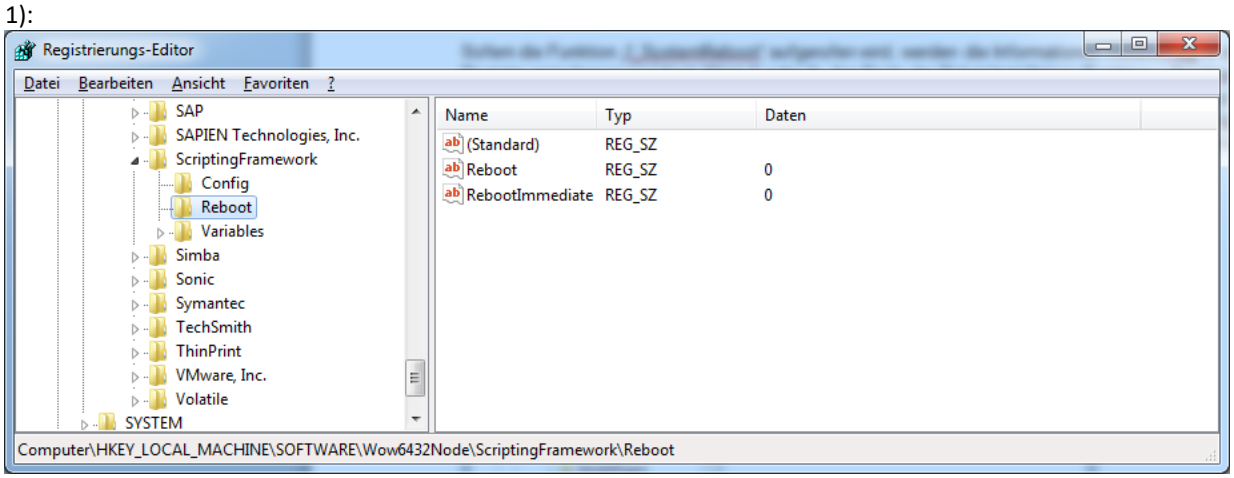
Name	Typ	Daten
(Standard)	REG_SZ	(Wert nicht festgelegt)
BuildNumber	REG_SZ	01
Company	REG_SZ	UNV
DisplayName	REG_SZ	Symantec Endpoint Protection PRD 12.1.4013.4013
Identifier	REG_SZ	UNV_ENU_Symantec_Endpoint_Protection_PRD_12140134013_01
InstallDate	REG_SZ	201411210131
InstallState	REG_SZ	Installed
Language	REG_SZ	ENU
Manufacture	REG_SZ	Symantec
ProductName	REG_SZ	Endpoint Protection PRD
Version	REG_SZ	12.1.4013.4013
VersionShort	REG_SZ	12

Nach der Deinstallation des Paketes ändert sich der InstallState auf „Uninstalled“, zusätzlich wird das UninstallDate eingetragen:

Name	Typ	Daten
(Standard)	REG_SZ	(Wert nicht festgelegt)
BuildNumber	REG_SZ	01
Company	REG_SZ	UNV
DisplayName	REG_SZ	Symantec Endpoint Protection PRD 12.1.4013.4013
Identifier	REG_SZ	UNV_ENU_Symantec_Endpoint_Protection_PRD_12140134013_01
InstallDate	REG_SZ	201411210131
InstallState	REG_SZ	Uninstalled
Language	REG_SZ	ENU
Manufacture	REG_SZ	Symantec
ProductName	REG_SZ	Endpoint Protection PRD
UninstallDate	REG_SZ	201501180216
Version	REG_SZ	12.1.4013.4013
VersionShort	REG_SZ	12

7.2.3 Reboot

Bei einem Aufruf der Funktion SF-SystemReboot, oder sofern eine andere Scripting Framework Funktion einen Neustart registriert, wird die Neustart Information mittels Flag in der Registry zwischengespeichert (Wert 0 oder 1):



Am Ende der Paket Installation erfolgt durch Scripting Framework eine Überprüfung, ob ein Reboot ausstehend ist. Falls ein Neustart registriert wurde, beendet sich Scripting Framework mit dem Return Code 3010. Der Neustart wird nicht durch Scripting Framework selbst durchgeführt! Durch den Return Code registriert z.B. Microsoft SCCM automatisch einen Reboot und dieser wird anschliessend durch das System Center ausgeführt. Die Reboot Flags werden in der Registrierung zwischengespeichert, damit auch ein anderes Verteilungssystem die Werte gegebenenfalls auslesen kann. In der Regel werden die Werte nicht benötigt, die meisten Verteilungssystem sind in der Lage den Return Code auszuwerten.

Achtung: Bei jedem starten des PowerShell Script erfolgt eine Rücksetzung der Flags.

7.2.4 Variables

Die Default Variablen welche in einem Scripting Framework Script zur Verfügung stehen, werden jeweils dynamisch beim Start des Frameworks geschrieben. Der Schritt wird nur durchgeführt, sofern die Boot Time des Clients (Computer Variablen) oder die LogonTime (User Variablen) nicht übereinstimmt. Diese Abfrage findet aufgrund von Performanceoptimierungen statt.

Zusätzlich zu den Default Variablen besteht die Möglichkeit, innerhalb des Scripts weitere Variablen mit der Funktion *SF-LoadVariables* aus einer CFG Datei dynamisch zu laden. Einzelne Variablen lassen sich mit dem Befehl *SF-Set* setzen oder auch wieder entfernen. Die Variablen stehen sofort zur Verfügung und sind zur Anwendung bereit.

Innerhalb von Scripting Framework werden die Variablen mit ***%VariableName%*** angesprochen. **Beispiel:**

```
# Run
SF-Run "%_Windows%\notepad.exe" "" -Show
```

Der Befehl startet Notepad, die Variable *%_Windows%* wird durch Scripting Framework aufgelöst → C:\Windows\Notepad.exe

7.2.4.1 Maschine - Default Variablen

Name:	Wert:
_CommonAdminTools	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools
_CommonApplicationData	C:\ProgramData
_CommonDesktop	C:\Users\Public\Desktop
_CommonDocuments	C:\Users\Public\Documents
_CommonFiles	**32-Bit System:** C:\Program Files\Common Files **64-Bit System:** C:\Program Files (x86)\Common Files
_CommonFiles32	**32-Bit System:** C:\Program Files\Common Files **64-Bit System:** C:\Program Files (x86)\Common Files
_CommonFiles64	**32-Bit und 64 Bit System:** C:\Program Files\Common Files
_CommonMusic	C:\Users\Public\Music
_CommonPictures	C:\Users\Public\Pictures
_CommonPrograms	C:\ProgramData\Microsoft\Windows\Start Menu\Programs
_CommonStartMenu	C:\ProgramData\Microsoft\Windows\Start Menu
_CommonStartup	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
_CommonTemplates	C:\ProgramData\Microsoft\Windows\Templates
_CommonVideos	C:\Users\Public\Videos
_ComputerChassisName	Chassis Type Name. Beispiel: Tablet

_ComputerChassisType	Chassis Type (Win32_Systemenclosure)
_ComputerDomain	Computer Domain oder Workgroup
_ComputerFQDN	COMPUTERNAME.domain.ch
_ComputerHardwareType	Physical oder Virtual
_ComputerHardwareTypeDetail	Hyper-V, Virtual PC, Xen, VMWare, None
_ComputerModel	Modell Typ des Computers. Beispiel: HP Compaq Elite 8300 SFF
_ComputerModelVersion	Modell Version des Computers
_ComputerName	Name des Computers
_ComputerOU	Domain OU des Computers
_ComputerType	WORKSTATION, NOTEBOOK, SERVER oder CITRIX
_Fonts	C:\Windows\Fonts
_GroupAdministrators	Administratoren
_GroupEveryone	Jeder
_GroupGuests	Gäste
_GroupPowerUsers	Hauptbenutzer
_GroupRemoteDesktopUsers	Remotedesktopuser
_GroupUsers	Benutzer
_OfficePlatform	Office Platform (x86 oder x64)
_OSLang	1031
_OSType	x64
_OSVersion	10.0.19042
_OSVersionCurrent	10.0 (Major.Minor)
_ProgramFiles	**32-Bit System:** C:\Program Files **64-Bit System:** C:\Program Files (x86)
_ProgramFiles32	**32-Bit System:** C:\Program Files **64-Bit System:** C:\Program Files (x86)
_ProgramFiles64	**32-Bit und 64Bit System:** C:\Program Files
_Public	C:\Users\Public
_sProgramFiles	C:\PROGRA~2
_sProgramFiles32	C:\PROGRA~2
_sProgramFiles64	C:\PROGRA~1
_SystemDrive	C:
_Temp	C:\Windows\TEMP
_UserAdministrator	Administrator
_UserSystem	NT AUTORITÄT\SYSTEM
_Windows	C:\Windows
_WindowsSystem	**32-Bit System:**

	C:\Windows\system32 **64-Bit System:** C:\Windows\SysWOW64
_WindowsSystem32	**32-Bit System:** C:\Windows\system32 **64-Bit System:** C:\Windows\SysWOW64
_WindowsSystem64	**32-Bit und 64-Bit System:** C:\Windows\system32

7.2.4.2 Maschine - Laufzeit Variablen

Name:	Wert:
_AppVPackageRoot	Innerhalb eines App-V Packages steht diese Variable zur Verfügung, welche in das App-V Root zeigt
_PkgBuildNumber	01
_PkgCache	C:\Windows_ScriptingFramework\Cache\PkgIdentifier (siehe InstallUser.ps1)
_PkgCFGName	PkgIdentifier (aus XML)
_PkgDisplayName	Hersteller ProduktName Version (aus XML)
_PkgIdentifier	PkgIdentifier (aus XML)
_PkgIdentifierShort	PkgIdentifier ohne Version und BuildNumber (aus XML)
_PkgLang	Language Code (wird mit der Funktion SF-Language generiert)
_PkgLanguage	Sprache (aus XML)
_PkgManufacturer	Hersteller (aus XML)
_PkgProductName	ProduktName (aus XML)
_PkgSource	Pfad von welchem das Paket gestartet wurde
_PkgVersion	Detaillierte Version (z.B. 14.0.0.179)
_PkgVersionShort	Kurze Version (aus XML)

7.2.4.3 Benutzer - Default Variablen

Diese Variablen stehen innerhalb der InstallUser.ps1 oder dem Scripting Mode zur Verfügung:

Name:	Wert:
_ApplicationData	C:\Users*Username*\AppData\Roaming
_Cookies	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Cookies
_Desktop	C:\Users*Username*\Desktop
_Favorites	C:\Users*Username*\Desktop
_History	C:\Users*Username*\AppData\Local\Microsoft\Windows\History
_HomeDrive	H:
_HomePath	\Users*Username*
_HomeShare	
_InternetCache	C:\Users*Username*\AppData\Local\Microsoft\Windows\Temporary Internet Files

_LastLogonTime	12/17/2014 07:48:51
_LocalApplicationData	C:\Users*Username*\AppData\Local
_LocalLowApplicationData	C:\Users*Username*\AppData\LocalLow
_MyDocuments	C:\Users*Username*\Documents
_MyMusic	C:\Users*Username*\Music
_MyPictures	C:\Users*Username*\Pictures
_MyVideos	C:\Users*Username*\Videos
_Personal	C:\Users*Username*\Documents
_Programs	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Start Menu\Programs
_Recent	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Recent
_SendTo	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\SendTo
_StartMenu	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Start Menu
_Startup	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
_Temp	C:\Users*Username*\AppData\Local\Temp
_Templates	C:\Users*Username*\AppData\Roaming\Microsoft\Windows\Templates
_User	*Username*
_UserProfile	C:\Users*Username*

7.2.4.4 Benutzer - Default Variablen (Active Directory)

Achtung: Diese Informationen können nur ausgelesen werden, sofern das AD vorhanden oder erreichbar ist.

Name:	**Wert:**
_City	Bischofszell
_Company	Windows Client Management AG
_Country	CH
_Department	Informatik
_Description	Domain Admin ohne GPO und Roaming Profile
_DisplayName	Test1
_EMail	test1@wincm.ch
_Fax	071 451 14 41
_FirstName	Test1
_Homepage	www.wincm.ch
_Initials	TT
_LastName	Test
_Office	Büro 101
_Pager	10 72
_PostalCode	9220
_Province	TG
_Street	Industriestrasse 6

_Telephone	n/A
_TelephoneHome	
_TelephoneIP	
_TelephoneMobile	
_Title	Microsoft Certified Enterprise Administrator

7.2.4.5 Benutzer - Laufzeit Variablen

Name:	Wert:
_AppVPackageRoot	Innerhalb eines App-V Packages steht diese Variable zur Verfügung, welche in das App-V Root zeigt
_PkgBuildNumber	01
_PkgCache	C:\Windows_ScriptingFramework\Cache\PkgIdentifier (siehe InstallUser.ps1)
_PkgCFGName	PkgIdentifier (aus XML)
_PkgDisplayName	Hersteller ProduktName Version (aus XML)
_PkgIdentifier	PkgIdentifier (aus XML)
_PkgIdentifierShort	PkgIdentifier ohne Version und BuildNumber (aus XML)
_PkgLang	Language Code (wird mit der Funktion SF-Language generiert)
_PkgLanguage	Sprache (aus XML)
_PkgManufacturer	Hersteller (aus XML)
_PkgProductName	ProduktName (aus XML)
_PkgSource	Pfad von welchem das Paket gestartet wurde
_PkgVersion	Detaillierte Version (z.B. 14.0.0.179)
_PkgVersionShort	Kurze Version (aus XML)

7.3 Command Line Options (Install.exe und Uninstall.exe)

Die Install.exe und Uninstall.exe unterstützt folgende optionale Parameter:

Name:	Beschreibung:
-AdditionalLog	Um ein eigenes Log pro Paket zu erstellen, kann dieser Parameter verwendet werden. **Beispiel:** Install.exe -AdditionalLog "C:\Logs\Example.log"
-DisableCustomFinalize	Deaktiviert die Custom Finalize Phase.
-DisableCustomInitialize	Deaktiviert die Custom Initialize Phase.
-DisableLogging	Die Log-Ausgabe wird deaktiviert. Der Parameter kann mit dem Switch -AdditionalLog kombiniert werden: **Beispiel:** Install.exe -AdditionalLog "C:\Logs\Example.log" -DisableLogging
-NoError	Bei einem Error (1603) wird trotzdem der Return Code 0 zurückgegeben.
-NoReboot	Anstelle eines 3010 Return Codes wird 0 zurückgegeben.
-PerformanceMode	Scripting Framework startet mit der bestmöglichen Performance, um das Script schnellstmöglich auszuführen. Dafür werden einige Überprüfungen und Schritte weggelassen. Der Modus sollte nur im Scripting Mode verwendet werden.
-Silent	Keine Anzeige des Installationsprozesses.

7.4 Logs und Fehlercodes

Jede Funktion von Scripting Framework schreibt detaillierte Angaben in das Log, wodurch jeder einzelne Schritt nachvollziehbar ist. Es werden Standard Return Codes zurückgegeben, welche unabhängig von der Softwaredeployment Lösung ausgewertet werden können.

7.4.1 Log Dateien

Im Computer Log „C:\Windows_ScriptingFramework\Logs“ und im User Log (%LOCALAPPDATA%_ScriptingFramework\Logs) wird zwischen Informationen, Warnungen und Fehlern unterschieden. Um über einen längeren Zeitraum im Log die Vorgänge nachzuvollziehen zu können, werden die Log Dateien nach 1 MB entsprechend archiviert, ohne die Performance des Scripts zu beeinflussen.

Log Filename:	Beschreibung:
._ScriptingFramework\Logs\ ScriptingFramework.log	Diese Datei beinhaltet die aktuellen Einträge.
._ScriptingFramework\Logs\ScriptingFramework_1.log	Archivnummer 1
._ScriptingFramework\Logs\ScriptingFramework_2.log	Archivnummer 2
._ScriptingFramework\Logs\ScriptingFramework_3.log	Archivnummer 3
._ScriptingFramework\Logs\ScriptingFramework_4.log	Archivnummer 4
._ScriptingFramework\Logs\ScriptingFramework_5.log	Archivnummer 5 (älteste Einträge)

Beispiel ScriptingFramework.log:

17:01:2015-21:25:34 INFO: ***** © Copyright 2013 Windows Client Management AG - www.wincm.ch **********

17:01:2015-21:25:35 INFO: ***** ScriptingFramework.exe **********

17:01:2015-21:25:35 INFO: Command Line: "C:\WINDOWS_ScriptingFramework\Modul\ScriptingFramework.exe" "\\srvdeployment01\PkgSources\ UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\\Install.ps1" "0000"

```
17:01:2015-21:25:35   INFO:      Install Script: \\srvdeployment01\PkgSources\
UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\Install.ps1
17:01:2015-21:25:35   INFO:      LanguageCode: 0000
17:01:2015-21:25:35   INFO: ***** START Initialization Version 1.6.3.0 **********************************************************
17:01:2015-21:25:35   INFO: ===== Get Machine Variables ================================================================
17:01:2015-21:25:37   INFO:      Script is not running in InstallUser mode, user actions are not performed
17:01:2015-21:25:37   INFO: ===== ResetRebootFlag ======================================================================
17:01:2015-21:25:37   INFO: ***** END Initialization
*****************************************************************************************************************
17:01:2015-21:25:37   INFO: ***** START Installation
*****************************************************************************************************************
17:01:2015-21:25:37   INFO: ===== GetPackageDefinition ==================================================================
17:01:2015-21:25:37   INFO:      File:        \\srvdeployment01\PkgSources\ UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic
\Package.xml
17:01:2015-21:25:37   INFO:      DisplayName:   Adobe Photoshop Elements 11 MUI
17:01:2015-21:25:37   INFO:      Identifier:    UNV_MUI_Adobe_Photoshop_Elements_11_01
17:01:2015-21:25:37   INFO:      Manufacturer:  Adobe
17:01:2015-21:25:37   INFO:      ProductName:   Photoshop Elements
17:01:2015-21:25:37   INFO:      Version:      11
17:01:2015-21:25:37   INFO:      VersionShort:  11
17:01:2015-21:25:37   INFO:      Language:     MUI
17:01:2015-21:25:37   INFO:      BuildNumber:   01
17:01:2015-21:25:37   INFO:      Company:      UNV
17:01:2015-21:25:37   INFO: ===== Taskkill ============================================================================
17:01:2015-21:25:37   INFO:      ProcessOrFolder: C:\Program Files (x86)\Adobe
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Adobe
17:01:2015-21:25:37   INFO: ===== Taskkill ============================================================================
17:01:2015-21:25:37   INFO:      ProcessOrFolder: C:\Program Files (x86)\Adobe Media Player
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Adobe Media Player
17:01:2015-21:25:37   INFO: ===== Taskkill ============================================================================
17:01:2015-21:25:37   INFO:      ProcessOrFolder: C:\Program Files (x86)\Common Files\Adobe
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Common Files\Adobe
17:01:2015-21:25:37   INFO: ===== LoadVariables ========================================================================
17:01:2015-21:25:38   INFO:      Folder Level 1:
\\pkgconfig\Config_Eng\Pkg_Config_Files\MGB\UNV_MUI_Adobe_Photoshop_Elements_11_01.cfg
17:01:2015-21:25:38   INFO:      Folder Level 2:
\\pkgconfig\Config_Eng\Pkg_Config_Files\UNV_MUI_Adobe_Photoshop_Elements_11_01.cfg
17:01:2015-21:25:38   INFO:      INISection: CommonClientSettings
17:01:2015-21:25:38   INFO:      License = 1057-4754-0084-0281-4585-8195
17:01:2015-21:25:38   INFO: ===== Language ============================================================================
17:01:2015-21:25:38   INFO:      1031: German
17:01:2015-21:25:38   INFO:      1033: English
17:01:2015-21:25:38   INFO:      1036: French
17:01:2015-21:25:38   INFO:      1040: Italian
17:01:2015-21:25:38   INFO:      ValidLanguages: 1031,1033,1036
17:01:2015-21:25:38   INFO:      Set _PkgLang (Detected OS Language): 1031
17:01:2015-21:25:38   INFO: ===== Installed ===========================================================================
17:01:2015-21:25:38   INFO:      ProductCode: {1D181764-DCD0-41B8-AA7B-0A599F027A72}
17:01:2015-21:25:38   INFO:      Result: False
17:01:2015-21:25:38   INFO: ===== Installed ===========================================================================
17:01:2015-21:25:38   INFO:      ProductCode: {D4D065E1-3ABF-41D0-B385-FC6F027F4D00}
```

```
17:01:2015-21:25:38   INFO:      Result: False
17:01:2015-21:25:38   INFO: ===== Run ================================================================================
17:01:2015-21:25:38   INFO:      Target: \\srvdeployment01\PkgSources\
UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\Setup\Setup.exe
17:01:2015-21:25:38   INFO:      Parameters: /UL1031 /V"SERIALNUMBER=1057-4754-0084-0281-4585-8195"
17:01:2015-21:25:38   INFO:      Timeout: 9999 (Seconds)
17:01:2015-21:27:08   INFO:      Reporting task Exitcode: (0)
17:01:2015-21:27:08   INFO: ===== Delete =============================================================================
17:01:2015-21:27:08   INFO:      File: C:\Users\Public\Desktop\Adobe Photoshop Elements 11.lnk
17:01:2015-21:27:08   INFO: ===== NTFSPerm ===========================================================================
17:01:2015-21:27:08   INFO:      FileOrFolder: C:\ProgramData\Adobe\Elements Organizer
17:01:2015-21:27:08   INFO:      User: Users
17:01:2015-21:27:08   INFO:      Permission: Modify
17:01:2015-21:27:08   INFO:      Command: icacls.exe "C:\ProgramData\Adobe\Elements Organizer" /grant "Users":(OI)(CI)M /Q /T /C
17:01:2015-21:27:08   INFO: ===== NTFSPerm ===========================================================================
17:01:2015-21:27:08   INFO:      FileOrFolder: C:\ProgramData\Adobe\Photoshop Elements
17:01:2015-21:27:08   INFO:      User: Users
17:01:2015-21:27:08   INFO:      Permission: Modify
17:01:2015-21:27:08   INFO:      Command: icacls.exe "C:\ProgramData\Adobe\Photoshop Elements" /grant "Users":(OI)(CI)M /Q /T /C
17:01:2015-21:27:10   INFO: ===== INIWrite ===========================================================================
17:01:2015-21:27:10   INFO:      INIFile: C:\ProgramData\Adobe\Elements Organizer\11.0\Elements.ini
17:01:2015-21:27:10   INFO:      Section: PSEInformation
17:01:2015-21:27:10   INFO:      Name: Country
17:01:2015-21:27:10   INFO:      Value: 223
17:01:2015-21:27:10   INFO: ***** Finalize
********************************************************************************************************************
17:01:2015-21:27:10   INFO:      PackageName: Adobe Photoshop Elements 11 MUI
17:01:2015-21:27:10   INFO:      User settings disabled: Disabled_InstallUser.ps1
17:01:2015-21:27:10   INFO:      System Reboot: No reboot is pending. Exit with Code: 0
17:01:2015-21:27:10   INFO: ***** END Installation
********************************************************************************************************************
```

7.4.2 Rückgabewerte

Scripting Framework beendet sich jeweils mit folgende Return Codes, welche MSI konform sind:

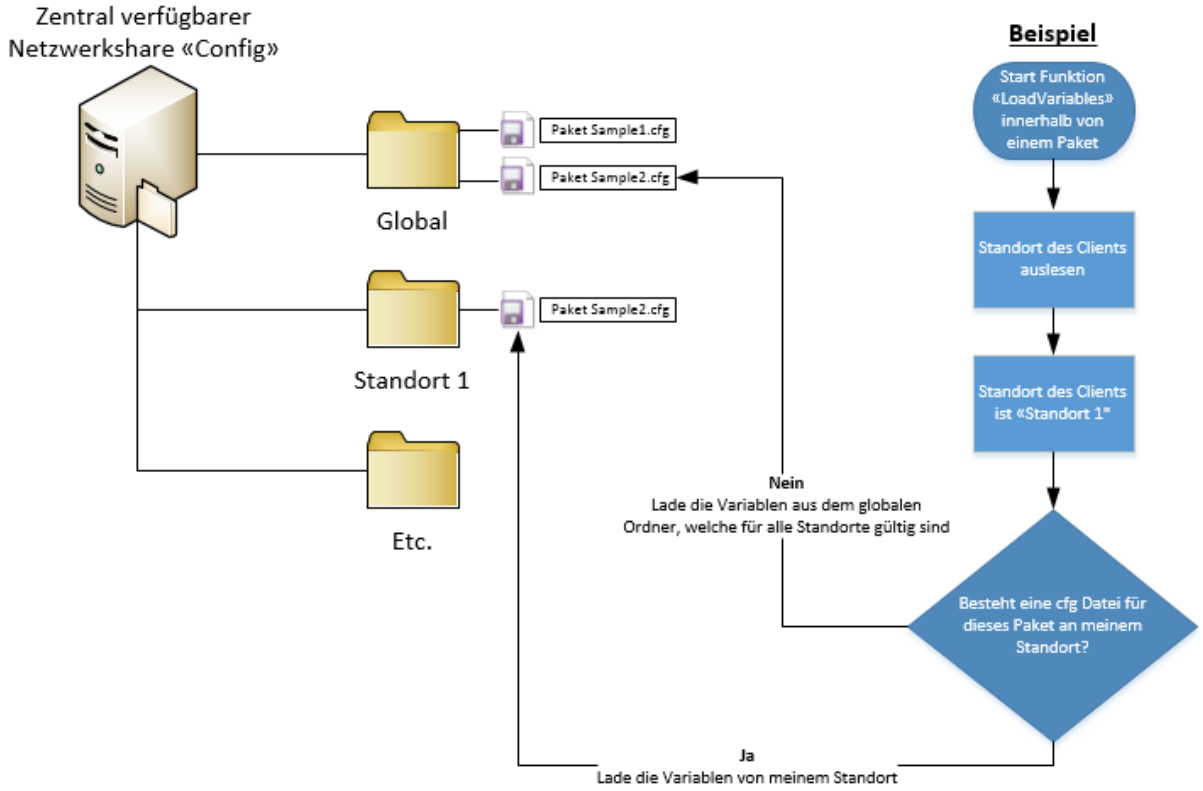
ErrorCode:	Beschreibung:
0	Die Aktion wurde erfolgreich abgeschlossen.
1603	Schwerwiegender Fehler bei der Installation.
3010	Zum Fertigstellen der Installation ist ein Neustart erforderlich.
1641	Sofortiger Neustart

Der ErrorCode ist jeweils auch im Scripting Framework Log ersichtlich.

7.5 Laden von Paketvariablen für dynamische Pakete

Beim Laden von Variablen mit der Funktion SF-LoadVariables, wird automatisch überprüft, ob eine standortspezifische .cfg Datei zur Verfügung steht. Der an die Funktion übergebene Pfad c_ConfigPath, wird automatisch mit dem entsprechenden Standort erweitert. Für die Abfrage des Standortes wird intern die Variabel c_configcompany verwendet.

Ablaufschema:



Sollten keine standortspezifischen Einstellungen (Level 1) vorhanden sein, werden die Variablen aus dem „global“ gültigen Ordner (Level 2) geladen, dies entspricht dem Wert %c_ConfigPath%. Bei den globalen Einstellungen handelt es sich um Konfiguration, welche standortunabhängig für alle gültig sind. Sollte auch unter dem Level 2 keine Konfiguration gefunden werden, wird ein Error ausgegeben. Beispiel:

Scripting Framework Befehl:

```
SF-LoadVariables "%c_ConfigPath%\%_PkgCFGName%.cfg" "CommonClientSettings"
```

Registry Werte (HKEY_LOCAL_MACHINE\SOFTWARE*Wow6432Node*\ScriptingFramework\Variables):
c_configpath = \\WINCM-SRV01.wincm.local\Config
c_configcompany = WinCM

Log Ausgabe:

```
29:04:2015-14:22:20   INFO: ===== LoadVariables ==========================================================================
29:04:2015-14:22:21   INFO:      Folder Level 1: \\WINCM-SRV01.wincm.local\Config\WinCM\UNV_UNI_WinCM_Sample_10_01.cfg
29:04:2015-14:22:21   INFO:      Folder Level 2: \\WINCM-SRV01.wincm.local\Config\UNV_UNI_WinCM_Sample_10_01.cfg
```

7.6 Abhandlung Benutzereinstellungen

Die InstallUser.ps1 wird für das Setzen von Benutzereinstellungen verwendet. Es stehen alle Funktionen von Scripting Framework zur Verfügung und können verwendet werden, sofern die Aktion keine Administrationsrechte benötigt.

Scripting Framework erkennt während der Installation von einem Paket automatisch ob die Benutzerkonfiguration aktiviert (InstallUser.ps1) oder deaktiviert (Disabled_InstallUser.ps1) wurde. Sofern diese aktiviert ist, wird die gesamte Konfiguration des Benutzers im Ordner „C:\Windows_ScriptingFramework\Cache*PackageIdentifier*“ zwischengespeichert und die Informationen entsprechend in die Registrierung geschrieben. Der Name des eindeutigen PackageIdentifiers wird aus den Angaben im Package.xml generiert, wobei alle Punkte „.“ entfernt werden. Die Deinstallation von einem Paket entfernt automatisch den Cache und die entsprechenden Registrierungseinträge.

Sofern die Konfiguration *DisableImmediateUserInstallation=1* auf dem Computer nicht aktiv ist, werden die Benutzereinstellungen nach der Installation automatisch durch Scripting Framework auf allen auf dem Gerät angemeldeten Benutzern angewendet, es wird kein Benutzer Log Off dazu benötigt! Sollte ein Benutzer nicht angemeldet sein, oder die sofortige Installation wurde über die Konfiguration deaktiviert, so wird die Benutzerkonfiguration erst bei der nächsten Anmeldung ausgeführt. Siehe auch unter sofortige Benutzerinstallation.

7.6.1 Technische Beschreibung

Bei der Anmeldung eines Benutzers vergleicht die Scripting Framework ActiveSetup.exe die zwei Registry Keys und deren Sub Keys miteinander:

- HKEY_LOCAL_MACHINE\SOFTWARE*WOW6432Node*\ScriptingFramework\UserSettings
- HKEY_CURRENT_USER\SOFTWARE\ScriptingFramework\UserSettings

Unterhalb dieses Pfades befindet sich für jedes Paket mit Benutzereinstellungen ein eigener Schlüssel. Jeder dieser Unterschlüssel beinhaltet folgende Werte:

Registry Wert:	**Beschreibung:**
(default)	Der Default Wert kann einen beliebigen Text enthalten, wie zum Beispiel der Name des Programms oder eine Beschreibung.
StubPath	Die Befehlszeile, die ausgeführt werden soll, wenn sich der Benutzer anmeldet: "C:\Windows_ScriptingFramework\Modul\ScriptingFramework.exe" -File "C:\Windows_ScriptingFramework\Cache*PkgIdentifier*\User\InstallUser.ps1"
Version	Die Version wird ebenfalls abgeglichen, ist die Version unter HKLM grösser als die Version unter HKCU so wird der StubPath erneut ausgeführt. Beispiel: 1,0,0,1 Achtung: Die Versionsnummer wird durch ein Komma getrennt und nicht durch Punkte.

Wichtiger Hinweis: Seit der Scripting Framework Version 2.5.0.0 wird nicht mehr der ActiveSetup von Microsoft verwendet, sondern ein eigener Mechanismus, welcher analog MS ActiveSetup funktioniert. Der Grund dafür ist, dass bei einem Windows 10 Upgrade alle benutzerdefinierten ActiveSetup Keys automatisch aus der Registry entfernt werden. Für die einmalige Migration auf den internen Benutzereinstellungen Mechanismus von Scripting Framework bieten wir ein kostenlosen Script an, kontaktieren Sie hierfür unseren Support.

7.6.2 Ablauf

1. Ein Benutzer meldet sich an.
2. Die ActiveSetup.exe überprüft die UserSettings unter HKLM und vergleicht Sie mit den vorhandenen Registry Einträgen im Benutzerprofil HKCU.
3. Fehlt unter HKCU ein Eintrag oder ist die Version in HKLM grösser als die Version unter HKCU, so wird das Kommando im Registry Wert StubPath (aus HKLM) ausgeführt und der Eintrag in die Registry (HKCU) geschrieben.

7.6.3 Published Application (Citrix)

Bei einem Aufruf von einer Published Application wird der Prozess „Explorer.exe“ nicht gestartet, wodurch auch die ActiveSetup.exe nicht automatisch aufgerufen wird. Dadurch werden keine Benutzereinstellungen von Scripting Framework verarbeitet. Um dieses Problem zu lösen kann die ActiveSetup.exe wie im MS Artikel beschrieben in den Login Prozess eingebunden werden:
http://support.microsoft.com/kb/195461/en-us

8 Bereitstellung und Konfiguration

8.1 Zentraler Share einrichten

Richten Sie als erstes einen Share auf einem Server ein, worauf später die CFG Dateien der dynamischen Pakete abgelegt werden können (SF-LoadVariables). Alle Geräte welche Scripting Framework einsetzen, sollten diesen Share erreichen können. Tragen Sie folglich den Share in dem von uns zur Verfügung gestellten Konfigurationstool ein.

8.2 Service Account im AD anlegen

Für den Zugriff auf Netzlaufwerke (Shares) benötigt Scripting Framework einen entsprechenden Benutzer. Wir empfehlen zu diesem Zweck einen separaten Service Account zu erstellen. Der Benutzer benötigt lediglich „Read“ Berechtigungen auf diesem Share, ansonsten sind keine weiteren Rechte notwendig. Den Benutzer tragen Sie anschliessen auch im Konfigurationstool ein.

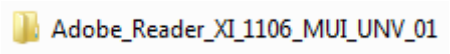
8.3 Definieren des zukünftigen Paketnamens (Identifier)

Jedes Paket verfügt über einen eindeutigen Identifier, welcher aus dem XML des jeweiligen Paketes generiert wird. Im Konfigurationstool können Sie die den Namen des Identifiers definieren:
PkgIdentifier = Manufacturer|ProductName|Version|Language|Company|BuildNumber

Beispiel XML:

```
<Package>
        <BuildNumber>01</BuildNumber>
        <Company>UNV</Company>
        <Manufacturer>Adobe</Manufacturer>
        <ProductName>Reader XI</ProductName>
        <Version>11.0.6</Version>
        <VersionShort>11</VersionShort>
        <Language>MUI</Language>
</Package>
```

Daraus resultiert der Paketname:

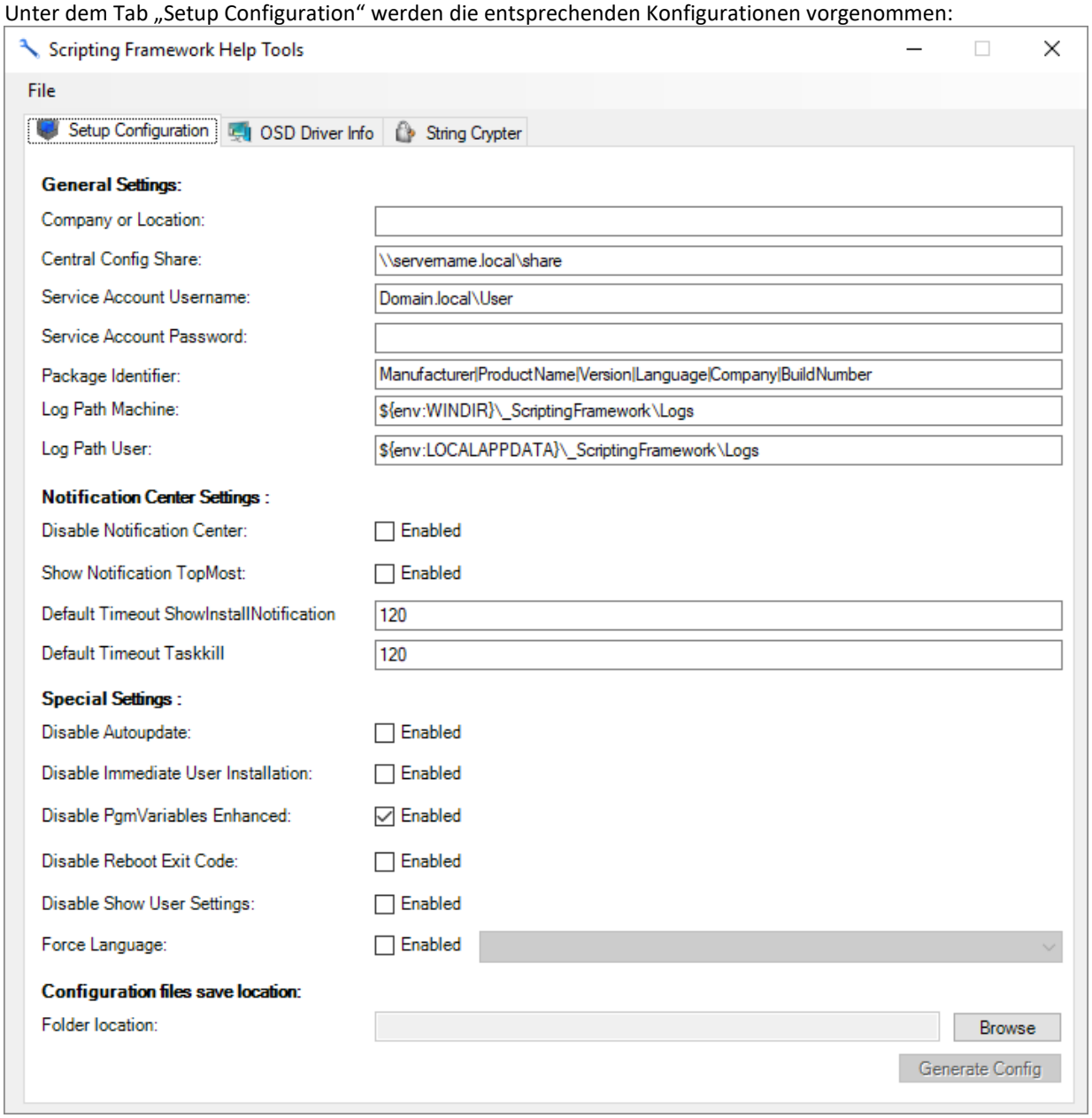


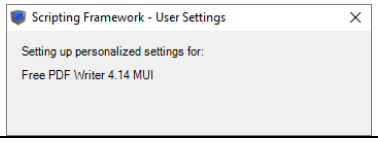
Achtung: Sobald Pakete im produktiven Einsatz sind, kann der Identifier nicht mehr angepasst werden! Weitere Details über den Aufbau der XML Datei.

8.4 Erstkonfiguration durchführen

Kopieren Sie das Installationspaket „WinCM_Scripting_Framework_1xxx_UNI_UNV_01“ nach „C:\Temp“. Starten Sie jetzt die Help Tools, diese befinden sich im soeben kopierten Installationspaket unter dem Pfad „…\Classic\Setup\Tools\HelpTools.exe“.

Unter dem Tab „Setup Configuration“ werden die entsprechenden Konfigurationen vorgenommen:



Option	Beschreibung
General Settings	
Company or Location	Geben Sie die Firma oder den Standort ein. Dieser Wert wird für die Funktion SF-LoadVariables verwendet. Für Details lesen Sie das Kapitel Laden von Paketvariablen für dynamische Pakete. Der Wert gilt in der Regel nur initial und wird später z.B. dynamisch per SCCM Tasksequenz oder GPO mit dem korrekten Standort abgefüllt.
Central Config Share	Tragen Sie den soeben erstellten Share (UNC Pfad) ein.
Service Account Username	Tragen Sie den erstellten Benutzer ein, welcher auf dem Central Config Share über „Read“ Berechtigungen verfügt.
Service Account Password	Tragen Sie das Passwort des Benutzers ein.
Package Identifier	Definieren Sie den gewünschten Paket Identifier. Wir empfehlen diesen so zu belassen.
Log Path Machine	Pfad der Scripting Framework Logdateien (Machine Actions)
Log Path User	Pfad der Scripting Framework Logdateien (User Actions)
Notification Center Settings	
Disable Notification Center	Deaktiviert das Scripting Framework Notification Center. Dem Benutzer werden dadurch keine Mitteilungen angezeigt, bei der Verwendung von Funktionen mit Notification Möglichkeit (z.B. SF-Taskkill).
Show Notification TopMost	Die Mitteilung des Notification Centers werden permanent im Vordergrund angezeigt.
Default Timeout ShowInstallNotification	Der Notification Timeout (SF-ShowInstallNotification) in Sekunden, sofern keine weitere Verschiebung der Installation mehr möglich ist. Nach diesem Timeout (wird dem Benutzer als Countdown angezeigt), wird die Installation fortgesetzt. Mit dem Wert 0 wird der automatische Timeout deaktiviert, was jedoch nicht die Empfehlung ist.
Default Timeout Taskkill	Timeout in Sekunden bis die Tasks (SF-Taskkill mit Switch -Notification) automatisch geschlossen werden. Mit dem Wert 0 wird der automatische Timeout deaktiviert, was jedoch nicht die Empfehlung ist.
Special Settings	
Disable Autoupdate	Deaktiviert die Scripting Framework Autoupdate Funktionalität, welche über die Datei AutoUpdater.exe durchgeführt wird.
Disable Immediate User Installation	Die User Einstellungen (InstallUser.ps1) werden automatisch nach der Paket Installation auf allen auf dem Gerät angemeldeten Benutzern angewendet. Mit dieser Option lässt sich diese Funktion deaktivieren.
Disable PgmVariables Enhanced	Deaktiviert die Erstellung der erweiterten Pgm Variablen bei einer Paket Installation.
Disable Reboot Exit Code	Sofern Scripting Framework einen Reboot entdeckt, wird anstelle des Return Codes 3010 ein Error Code 0 zurückgegeben. Dies kann z.B. auf einem Citrix Server verwendet werden.
Disable Show User Settings	Beim Anwenden der Benutzereinstellungen wird das Fenster nicht angezeigt: 
Force Language	Die Funktion SF-Language welche in Paketen verwendet werden kann, liest dynamisch jeweils die Sprache aus und gibt diese dann zurück. Sofern es gewünscht ist die Sprache der Funktion SF-Language zu fixieren, kann hier der Language Code eingetragen werden. In Regel wird das nur auf sprachspezifischen Citrix Servern verwendet.

Browse Folder	Tragen Sie hier folgenden Pfad ein: C:\Temp\WinCM_Scripting_Framework_*1xxx*_UNI_UNV_01\Classic\Config

Sofern alle Angaben gemacht wurden, klicken Sie auf Generate Config. Überprüfen Sie nun, ob in dem angegebenen Config Ordner die Datei Config.ps1 korrekt angelegt wurde. Das Paket für die Engine Installation ist damit fertig erstellt und kann verwendet werden. Vergessen Sie nicht die License.lic Datei nach „Classic\Setup\Modul_x64“ und „Classic\Setup\Modul_x86“ zu kopieren.

8.5 Installation

Vergessen Sie nicht vor der Installation von Scripting Framework die Erstkonfiguration durchzuführen. Die Installationen der Prerequisites basieren auf einem Command Line Script, da ohne die entsprechende PowerShell Version Scripting Framework nicht funktioniert.

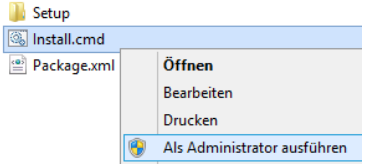
Um die Pakete nach der Installation direkt aus der PowerShell ISE Konsole ausführen zu können, installieren Sie als Paketierer zusätzlich die Engineer Erweiterung, wie im Handbuch beschrieben.

Alle benötigten Setup Dateien sowie Beispiel Pakete, können Sie von unserer Homepage beziehen:
http://www.scriptingframework.ch

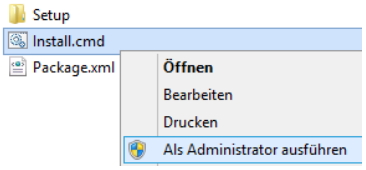
8.5.1 Manuelle Installation

Falls die Prerequisites bereits installiert sind, können Sie diese Installationen auslassen. Ansonsten verwenden Sie die Setuproutinen von uns, entpacken Sie dazu die heruntergeladenen Sourcen nach C:\Temp.

Installation .Net Framework:
Ausführen der Datei „C:\Temp\Microsoft_.Net_Framework_*4xx*_UNI_UNV_01\Classic\Install.cmd“ als Administrator:

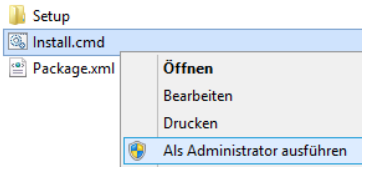


Installation PowerShell:
Ausführen der Datei „C:\Temp\Microsoft_Powershell_5*x*_UNI_UNV_01\Classic\Install.cmd“ als Administrator:



Achtung: Damit die neue PowerShell Version verwendet werden kann, ist ein Neustart nötig.

Installation Scripting Framework
Ausführen der Datei „C:\Temp\ WinCM_Scripting_Framework_2*xxx*_UNI_UNV_01\Classic\Install.cmd“ als Administrator:



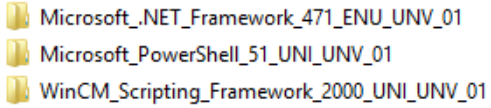
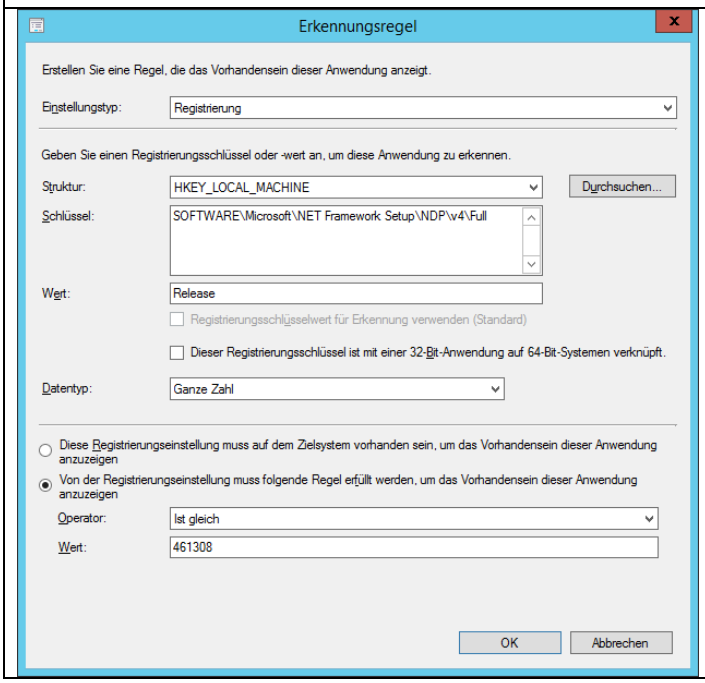
Achtung: Achten Sie darauf, dass Sie vorher die „Erstkonfiguration“ durchgeführt haben und die korrekten REG Dateien im Config Ordner abgelegt sind.

8.5.2 Mittels Softwareverteilung (Generell)

Um Scripting Framework über ihre Softwareverteilung zu installieren, erstellen Sie eine entsprechende Pakethülle und starten Sie darin die von uns gelieferte Install.cmd. Es sind keine Silent Parameter, etc. notwendig. Stellen Sie sicher, dass die Prerequisites auch installiert werden.

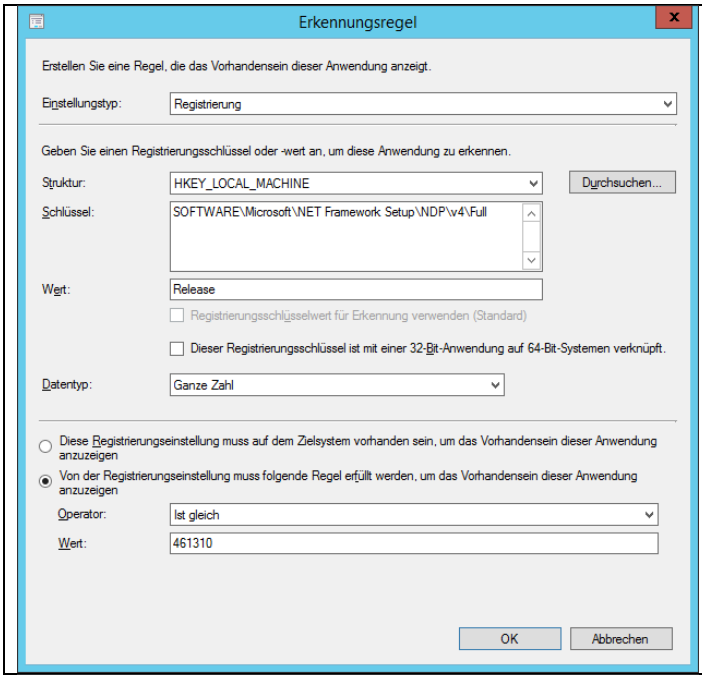
8.5.3 Mittels Softwareverteilung (Microsoft SCCM)

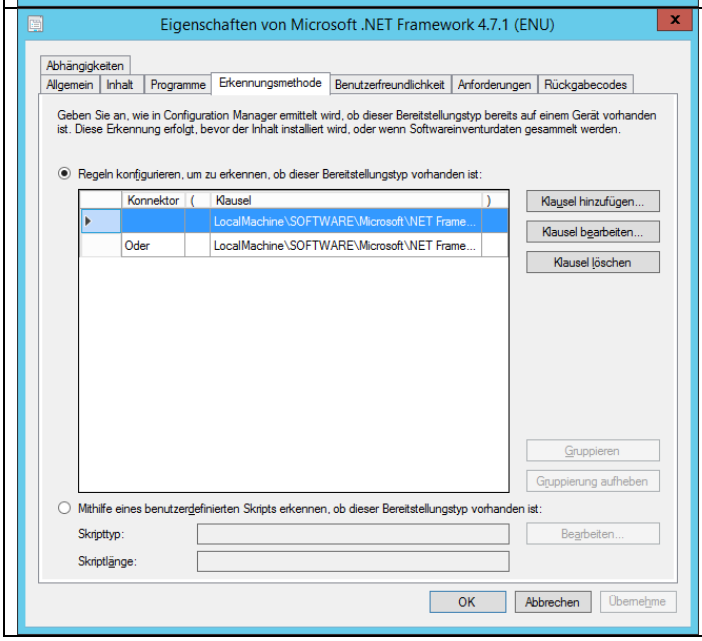
Für die Microsoft System Center Implementierung, werden die drei Pakete als Anwendung erfasst. Im Handbuch ist nur die Erfassung der Scripting Framework Anwendung Schritt für Schritt dokumentiert. Für die Prerequisites sind jede glich die Erkennungsregel (Detection Method) angegeben.

<table>
<tr><td> 
</td><td>Kopieren Sie die drei heruntergeladenen Pakete in die SCCM Application Source.</td></tr>
<tr><td colspan="2">Microsoft .NET Framework</td></tr>
<tr><td> 
</td><td>Legen Sie im SCCM als erstes die Microsoft .NET Framework Anwendung an.

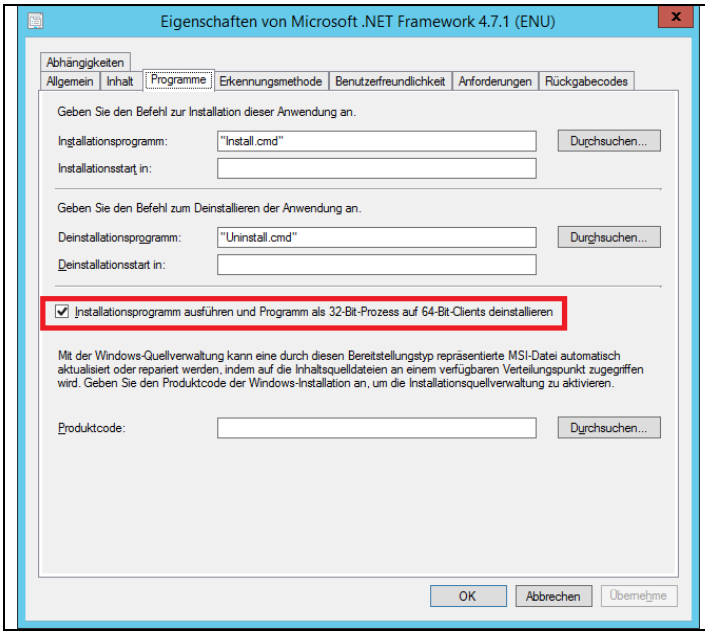
Die Erkennungsregel entnehmen Sie den folgenden drei Screenshots.

Für Windows 10 Fall Creators Update: 461308
Alle anderen OS Versionen: 461310</td></tr>
</table>



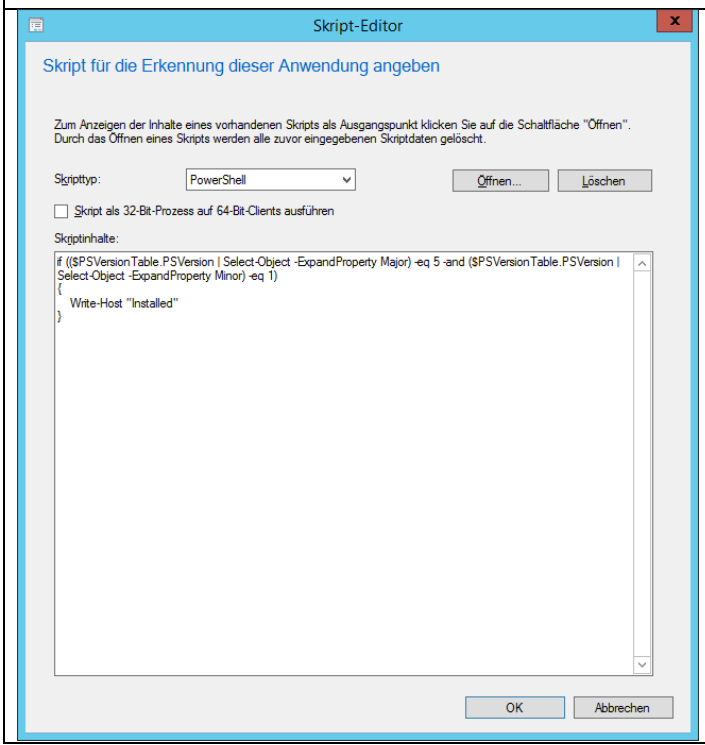


In der Spalte Konnektor wählen Sie „Oder“ aus.



Wichtig: Für die .NET Framework Installation wird zwingend die Option "Installationsprogramm ausführen und Programm als 32-Bit-Prozess auf 64-Bit-Clients deinstallieren" benötigt. Ansonsten tritt während der Installation ein Fehler auf.

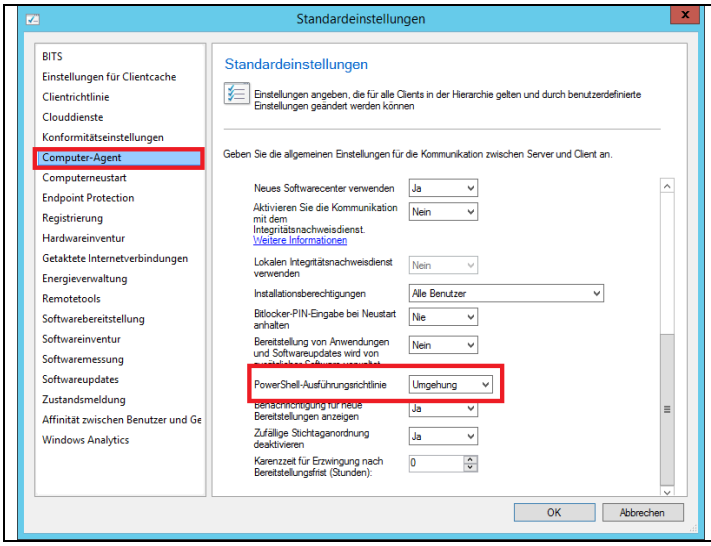
Microsoft PowerShell



Erfassen Sie als nächstes die PowerShell Anwendung mit der Abhängigkeit zu dem soeben erstellten .NET Framework Paket.

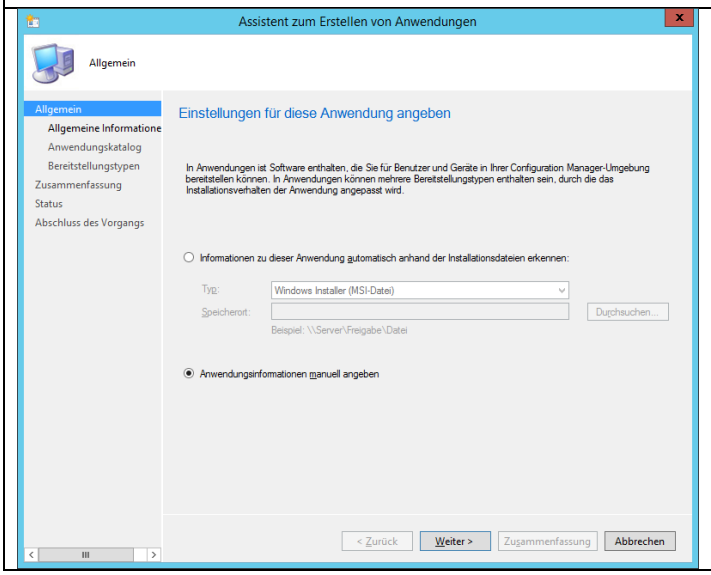
Als Erkennungsregel wird folgender PowerShell Script verwendet:

```
if (($PSVersionTable.PSVersion | Select-Object -ExpandProperty Major) -eq 5 -and ($PSVersionTable.PSVersion | Select-Object -ExpandProperty Minor) -eq 1)
{
   Write-Host "Installed"
}
```

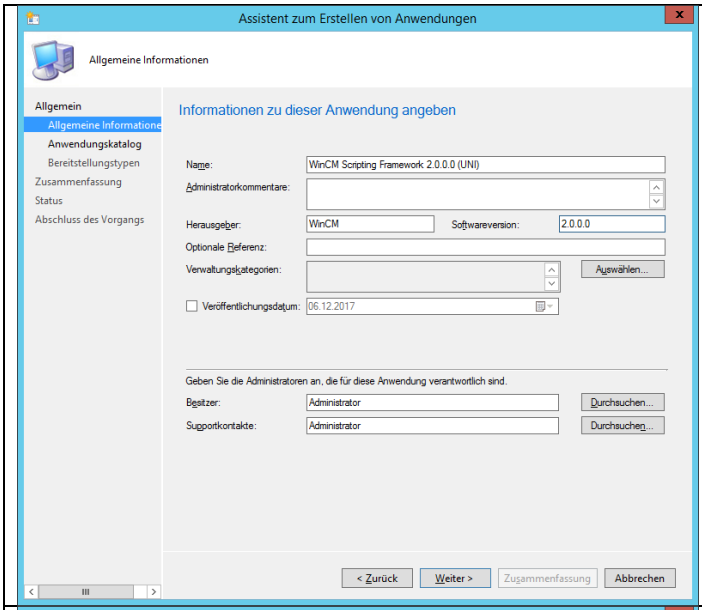
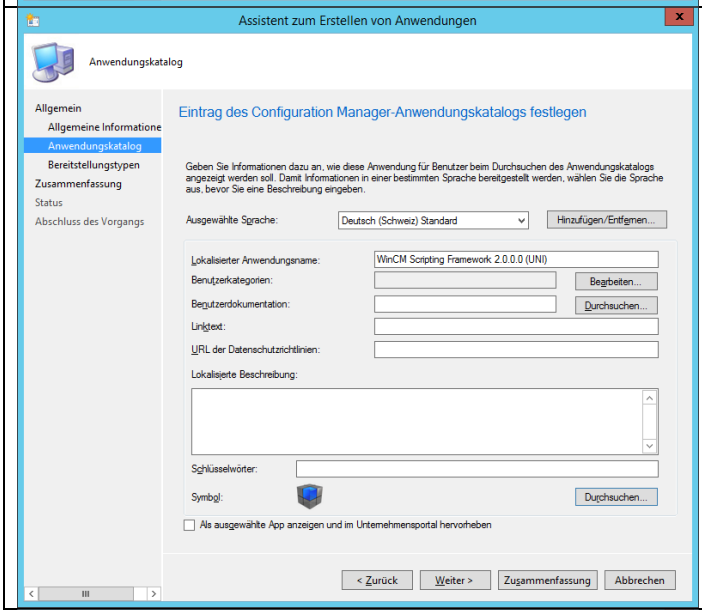
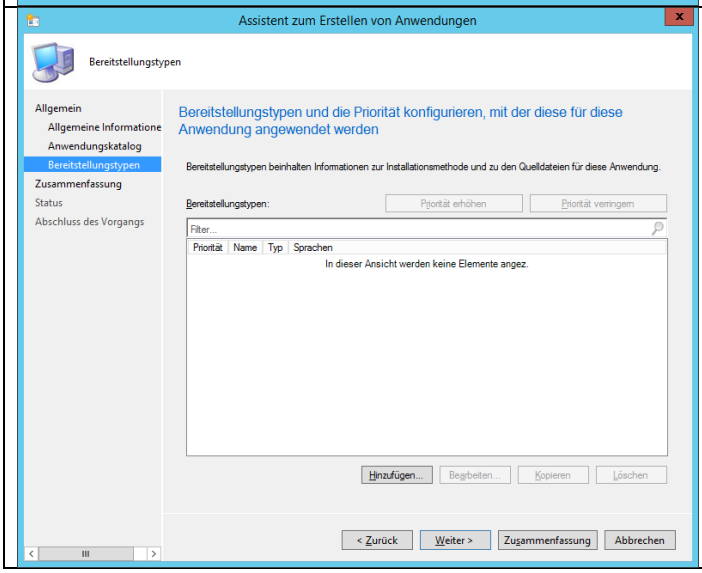


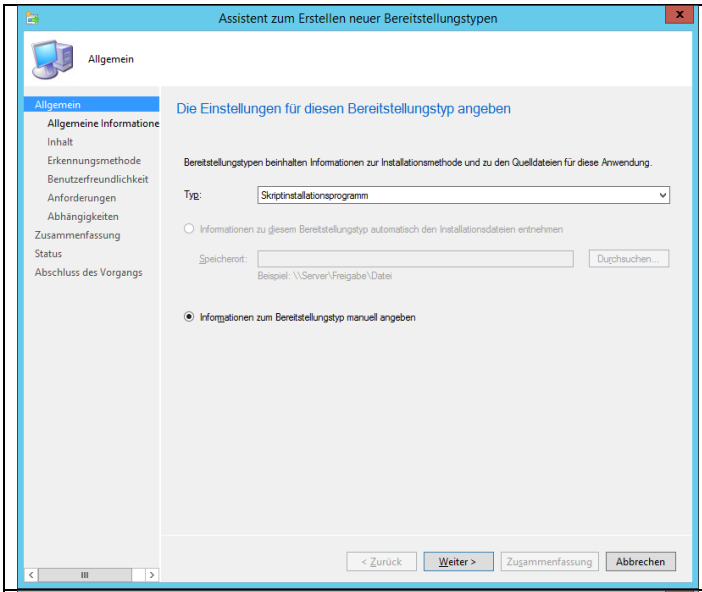
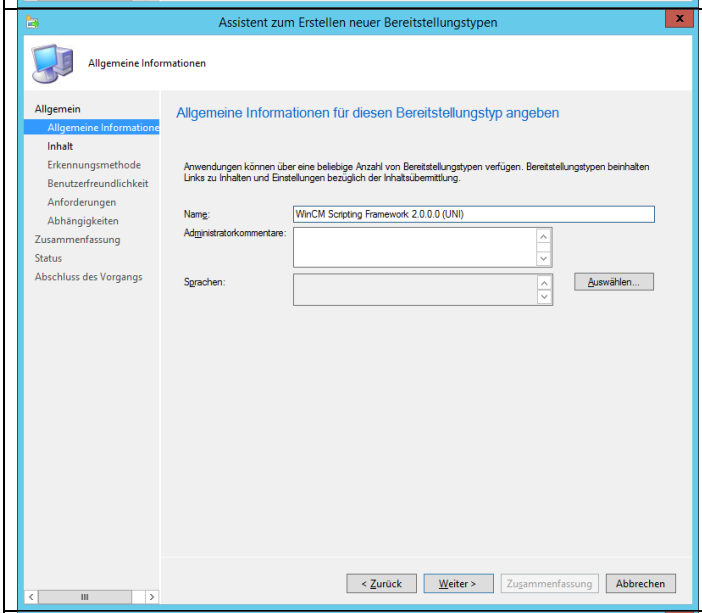
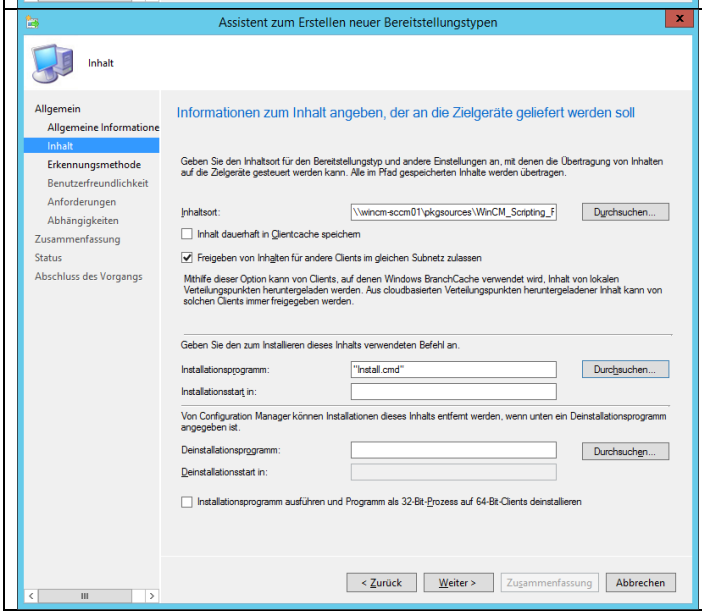
Wichtig: Für die PowerShell Script Erkennungsregel, muss die PowerShell Execution Policy auf Umgehung gesetzt werden. Siehe Screenshot.

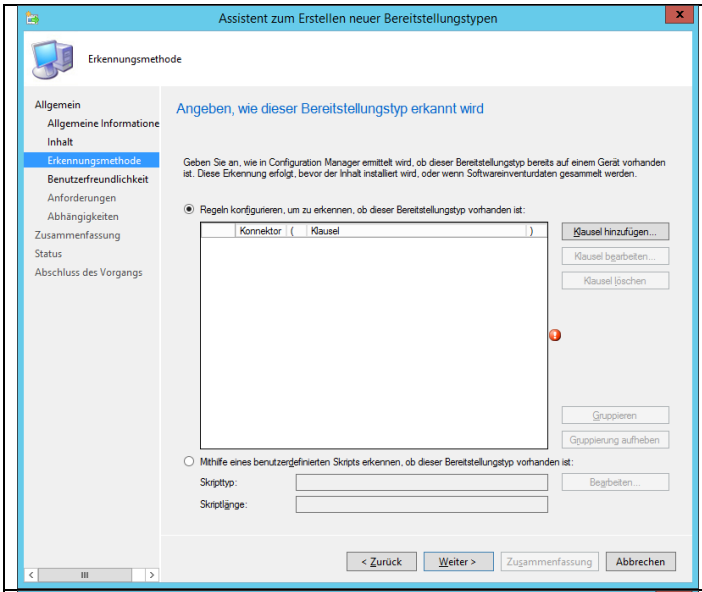
WinCM Scripting Framework (Schritt für Schritt)



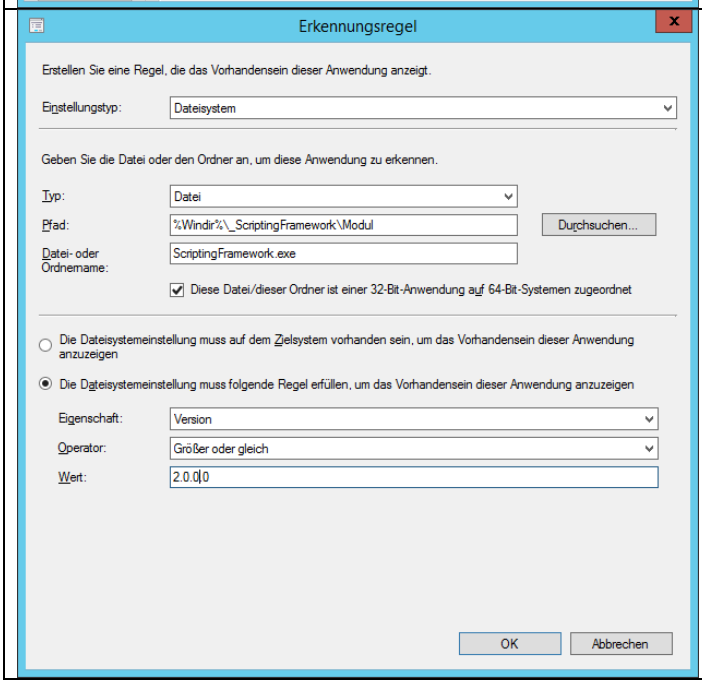
Erstellen Sie eine neue Anwendung.

	Die entsprechenden Informationen eingeben
	Geben Sie das Symbol an. Dieses ist im Paket im Ordner Icon zu finden. Weiter
	Hinzufügen

	**Typ:** Skriptinstallationsprogramm
	Name des Deployments eingeben. Weiter
	Inhaltsort: \\servername.local\share\PKGSources\WinCM_Scripting_Framework_2xxx_UNI_UNV_01\Classic Installationsprogramm: Install.cmd Ein Remove ist nicht vorgesehen, deshalb existiert keine Uninstall.

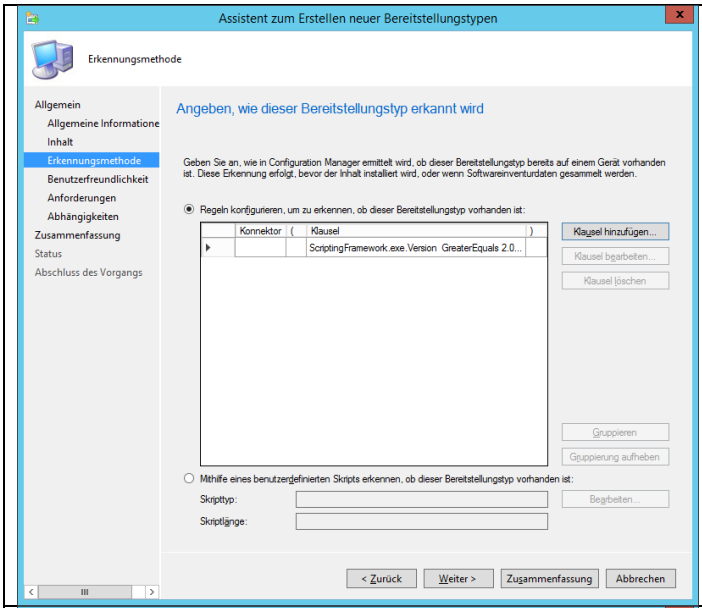
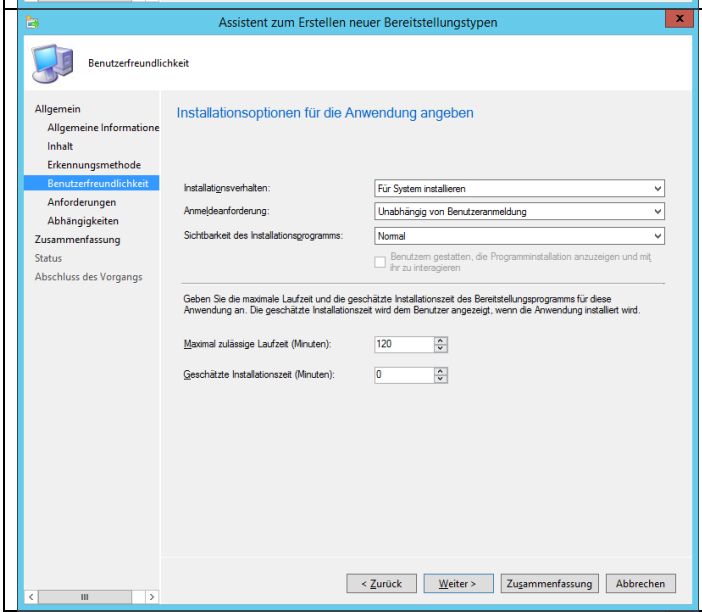
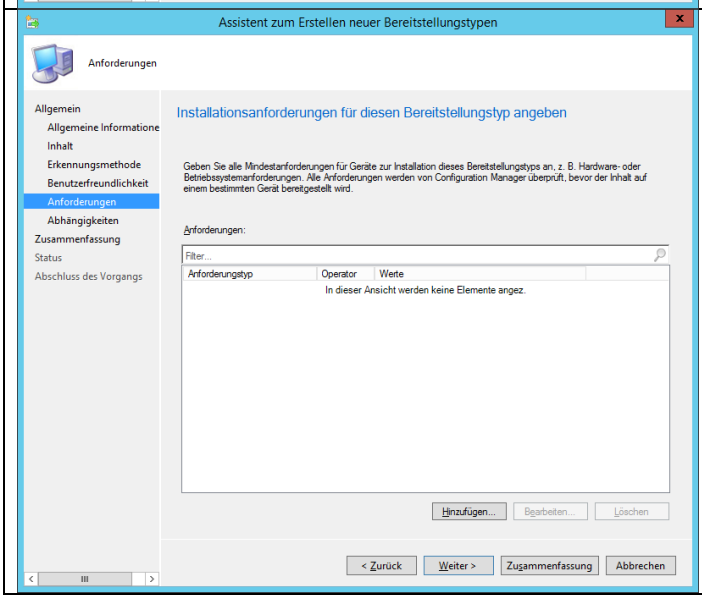


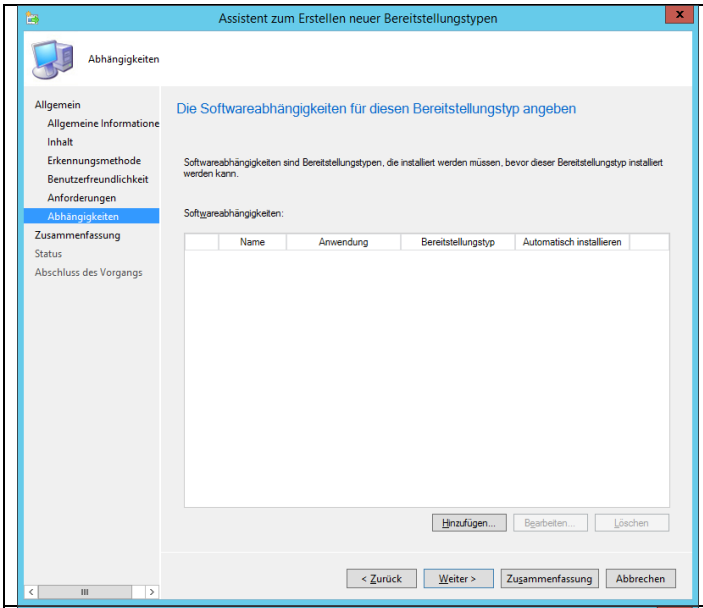
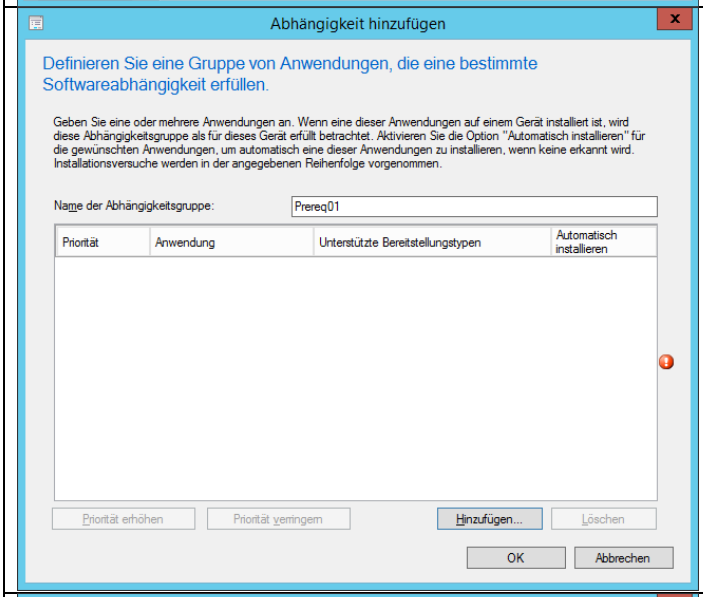
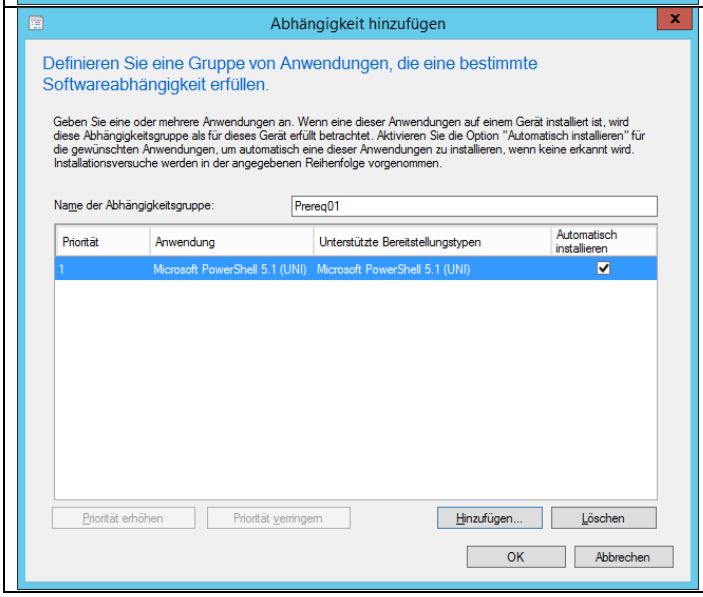
Hinzufügen

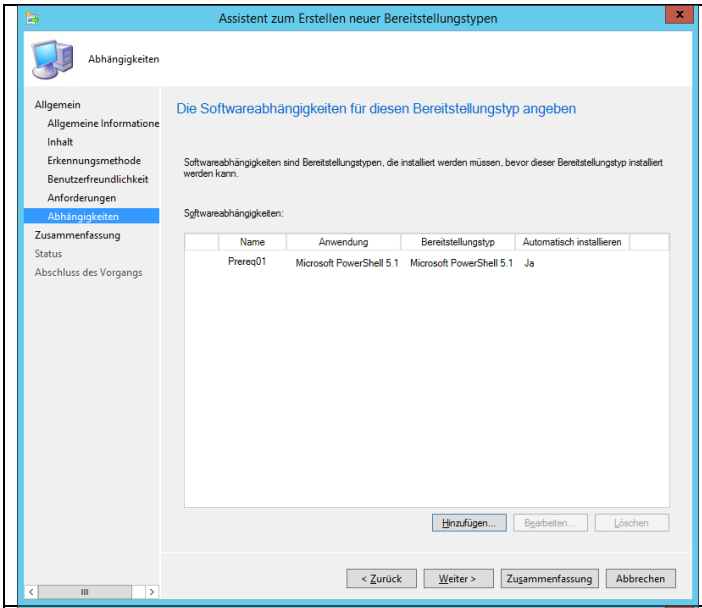
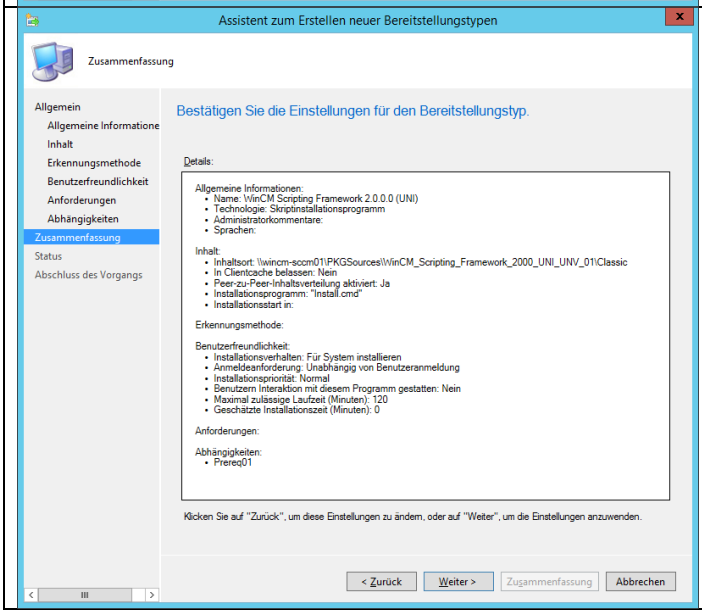


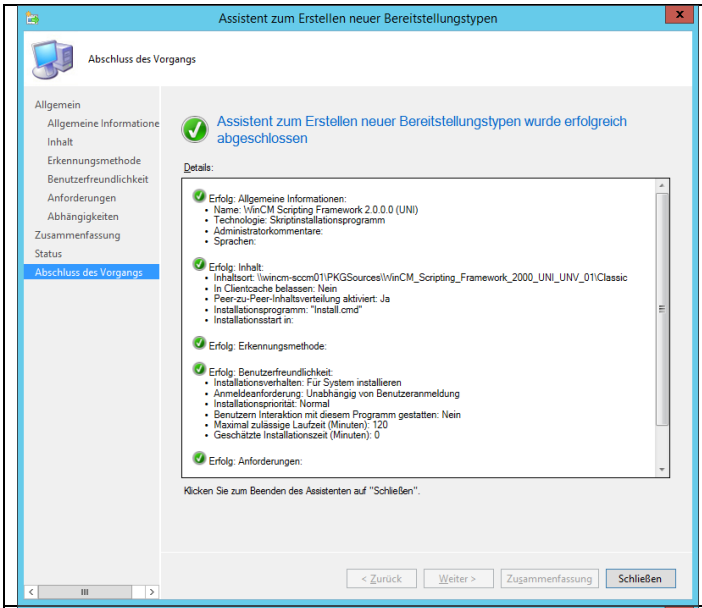
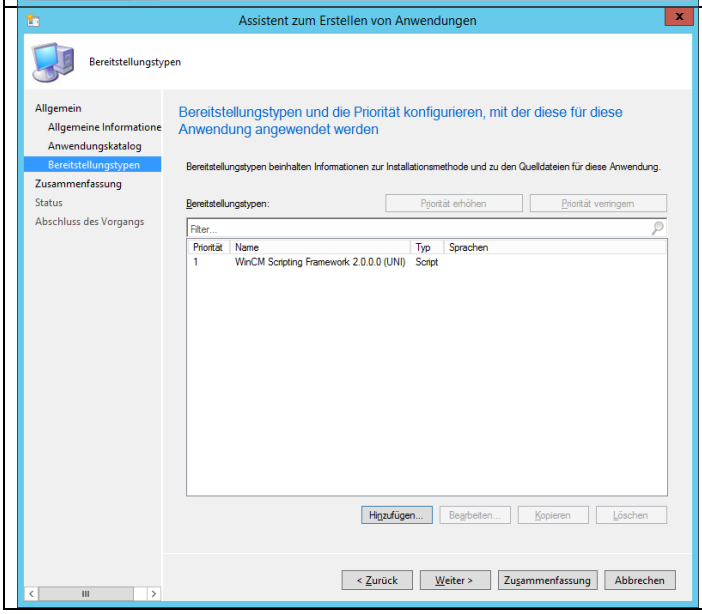
Regel gemäss Screenshot erfassen.

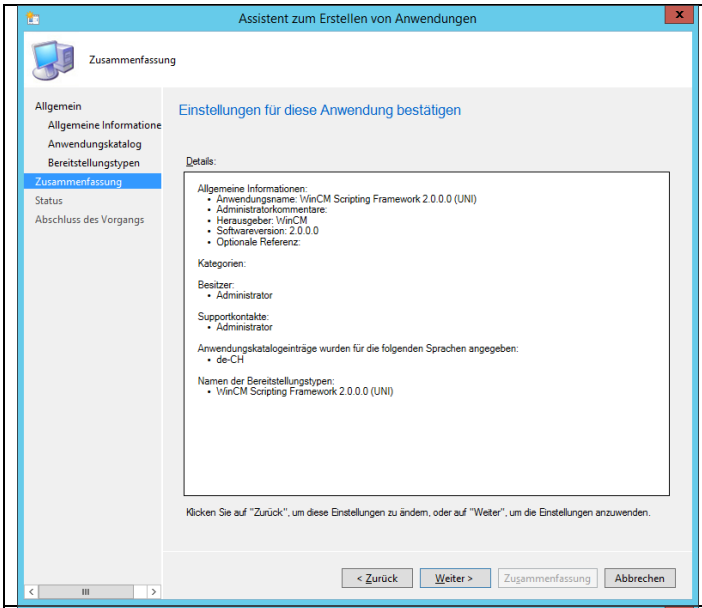
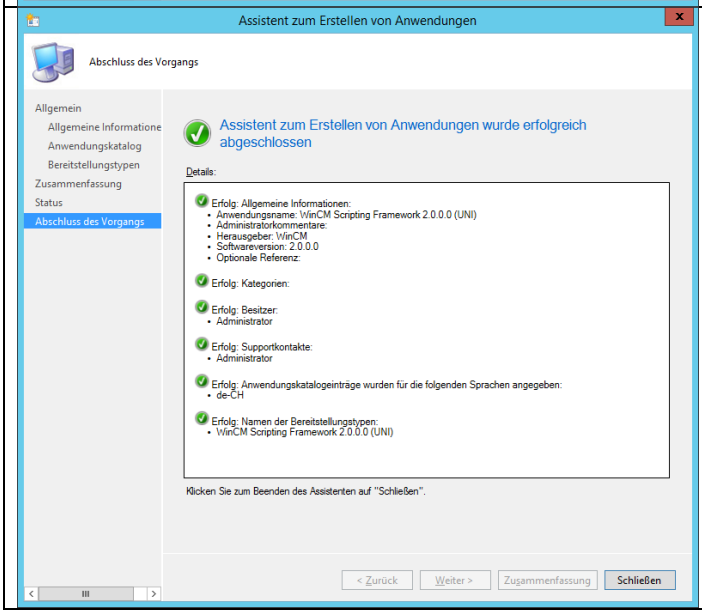
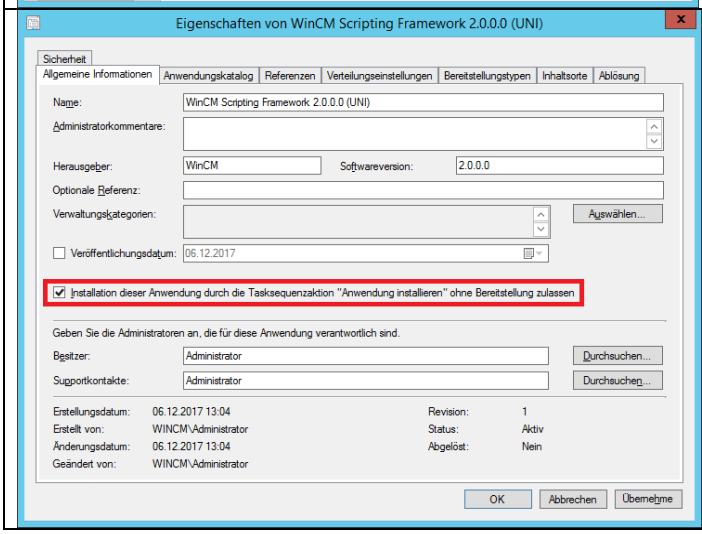
Passen Sie den Wert der Version der verwendeten Scripting Framework Version an.

	Weiter
	**Installationsverhalten:** Für System installieren **Anmeldeanforderung:** Unabhängig von Benutzeranmeldung Weiter
	Weiter

	Hinzufügen…
	Fügen Sie als Abhängigkeit die zuvor erfasste PowerShell Anwendung hinzu.
	OK

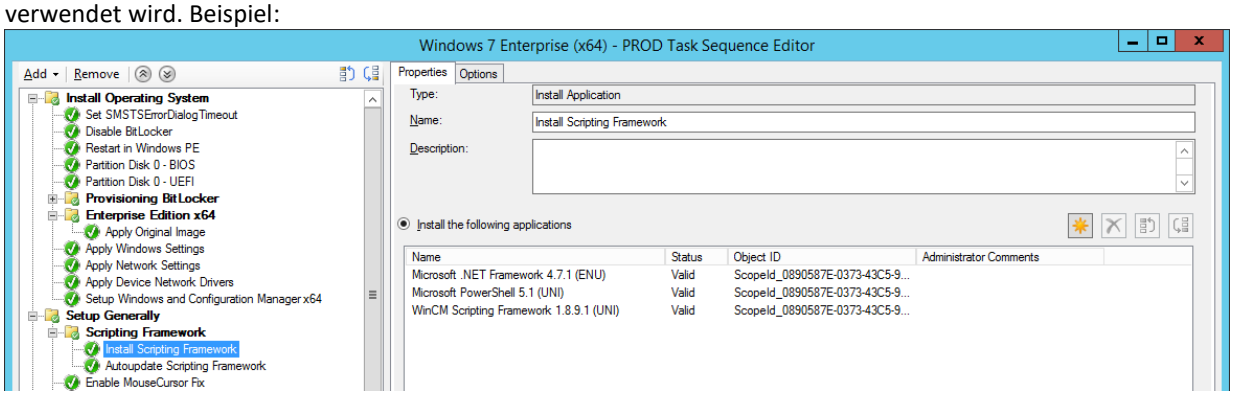
	Weiter
	Weiter

	Schliessen
	Weiter

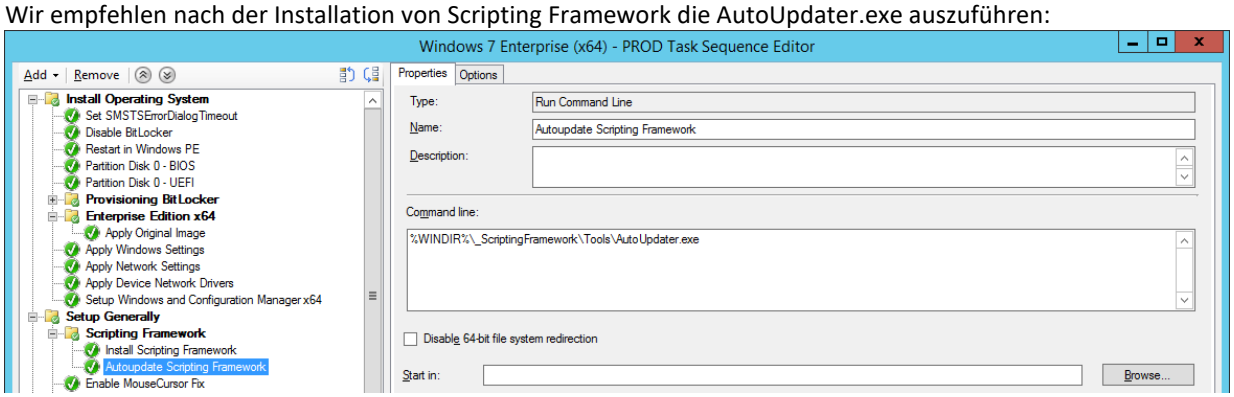
	Weiter
	Schliessen
	Öffnen Sie die Eigenschaften der Anwendungen und aktivieren Sie die im Screenshot markierte Option. Verteilen Sie anschliessend die Anwendung auf die Distribution Points. Somit ist die Anwendung für die Bereitstellung bereit.

8.5.3.1 Tasksequenz Integration

Scripting Framework muss so früh wie möglich installiert werden, bevor das erste Scripting Framework Paket verwendet wird. Beispiel:



Wir empfehlen nach der Installation von Scripting Framework die AutoUpdater.exe auszuführen:

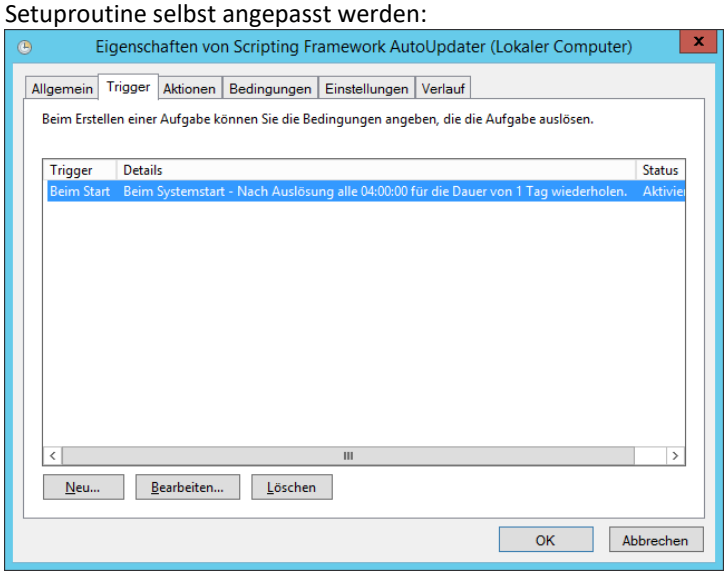


%WINDIR%_ScriptingFramework\Tools\AutoUpdater.exe

Dies ist zum Beispiel bei der Verwendung von einem Capture Image sehr nützlich, auf welchem die Detection Methode bereits erfüllt ist. Durch diesen Schritt findet immer eine Aktualisierung der Scripting Framework Version inkl. der Lizenz statt.

8.6 Automatische Aktualisierung von Scripting Framework (AutoUpdate)

Das Setup von Scripting Framework (Install.cmd) erstellt bei der Installation einen Schedule Task mit dem Namen „Scripting Framework AutoUpdater“. Dieser ermöglicht es Scripting Framework Dateien automatisch über den Config Share zu aktualisieren inkl. der Lizenz. Auf dem Task ist ein Trigger gesetzt, dieser kann falls nötig in der Setuproutine selbst angepasst werden:



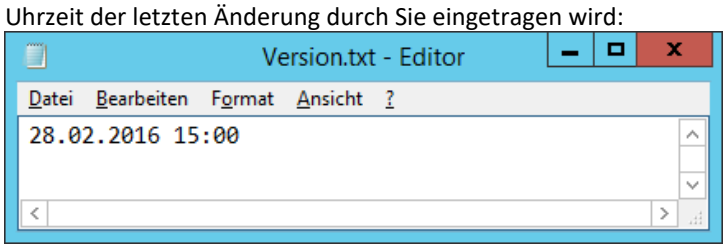
Die Autoupdate Funktion benötigt kaum Ressourcen und ist in der Regel innert Sekunden fertig.

8.6.1 Funktionsweise

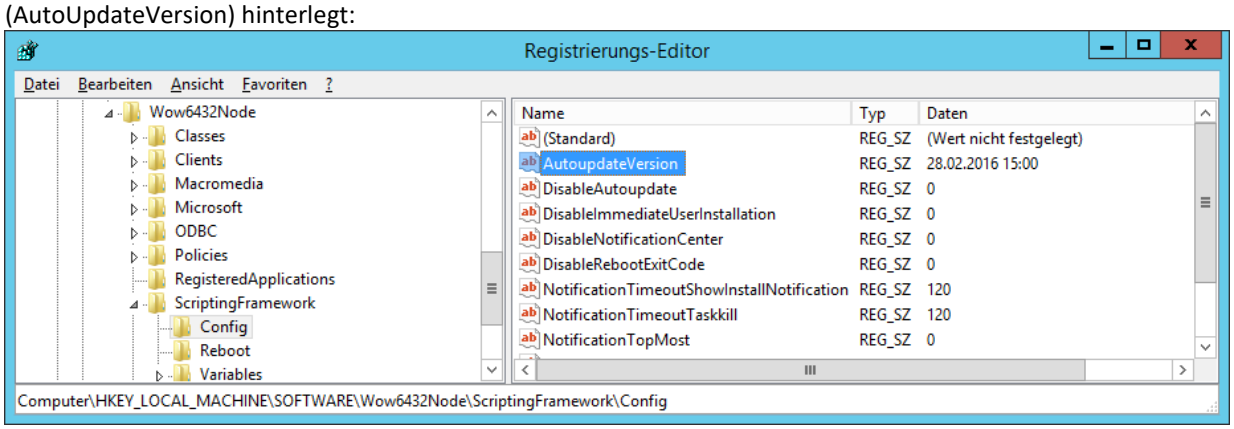
Nach dem Aufruf der AutoUpdater.exe wird als erster Schritt überprüft, ob in der Registry (Config) die Funktionalität über den Wert DisableAutoupdate=1 deaktiviert wurde. Bei einer Deaktivierung, beendet sich der AutoUpdater, ohne eine weitere Aktion durchzuführen.

Als nächstes wird geprüft, ob die benötigten Angaben in der Registry aus der Erstkonfiguration korrekt sind. Dazu gehören die Werte Username, Password und c_configpath. Anhand von diesen wird die Authentifizierung auf dem Config Share durchgeführt, auf welchem die Scripting Framework Update Dateien abgelegt werden. Falls der "Scripting Framework Autoupdate" Ordner auf dem Config Share nicht existiert, oder die definierte Konfiguration nicht korrekt ist, beendet sich der AutoUpdater automatisch.

Innerhalb des Ordners auf dem Share befindet sich die Textdatei „Version.txt“, in welcher das Datum und die Uhrzeit der letzten Änderung durch Sie eingetragen wird:



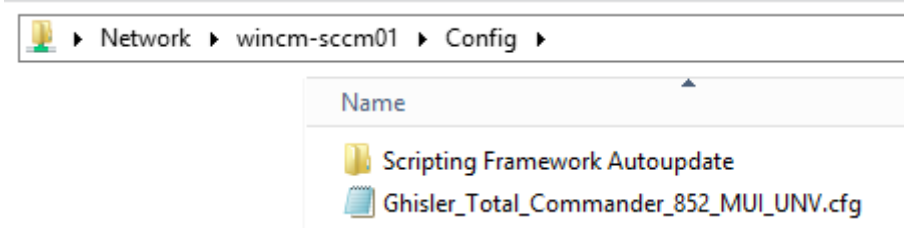
Nach einem erfolgreichen Autoupdate wird das Datum aus der Textdatei auf dem Client in der Registry (AutoUpdateVersion) hinterlegt:



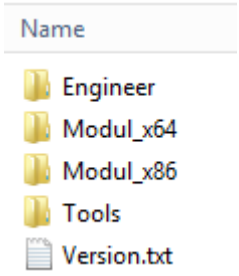
Nur sofern diese zwei Angaben nicht übereinstimmen, wird jeweils das automatische Update durchgeführt. Die Aktualisierung der Lizenzdatei findet unabhängig davon statt. Diese wird aktualisiert, wenn das Dateidatum der Lizenz auf dem Share nicht mit der lokalen Lizenzdatei übereinstimmt.

8.6.2 Benötigte Share Struktur

Auf dem Config Share wird der Ordner „Scripting Framework Autoupdate" angelegt. Die gesamte Struktur können Sie auch auf unserer Homepage herunterladen:



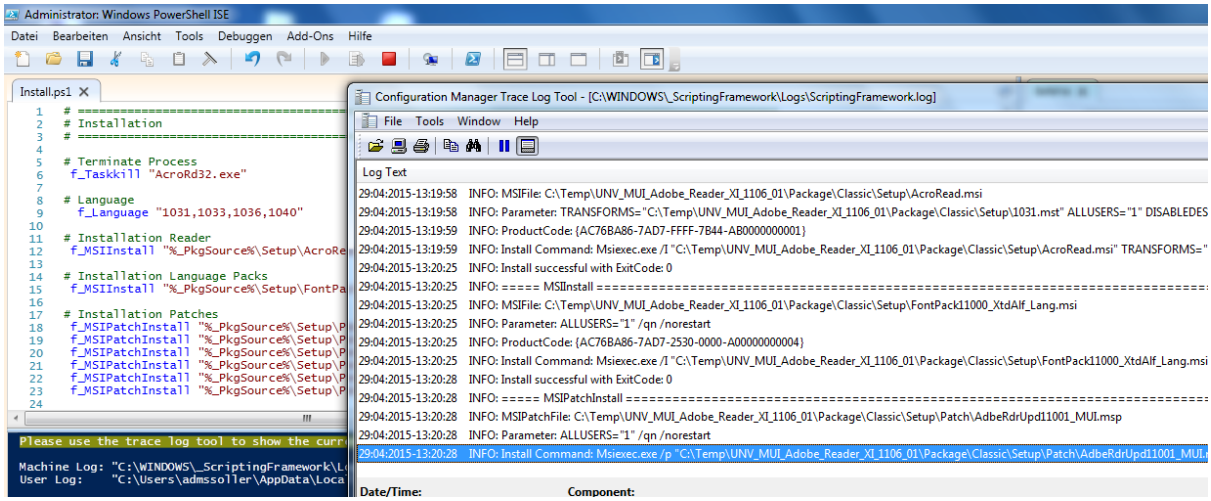
Die Struktur des Scripting Framework Autoupdate Ordners:



Ordner	Inhalt und Beschreibung
Engineer	Der Ordner beinhaltet das Engineer Modul "ScriptingFramework.psm1". Dieses wird auf dem Client nur aktualisiert, sofern es installiert wurde.
Modul_x64	Alle x64 Modul Dateien inkl. der persönlichen Lizenz.
Modul_x86	Alle x86 Modul Dateien inkl. der persönlichen Lizenz.
Tools	Alle Tools Dateien von Scripting Framework.
Version.txt	Das Datum und die Uhrzeit der letzten Änderung, werden in dieser Datei eingetragen (siehe Funktionsweise).

8.7 Installation Engineer Erweiterung (für die Entwicklung von Paketen)

Um Pakete schnell und einfach zu entwickeln und um die Pakete direkt aus der PowerShell ISE Console zu starten, existiert eine zweite Setuproutine welche die entsprechenden Erweiterungen installiert. Führen Sie dazu die Install.cmd des „Scripting Framework Engineer“ als Administrator aus.

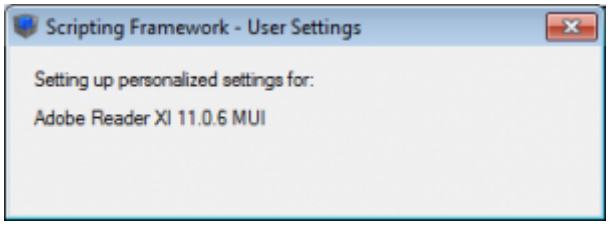


Nach der Installation können Sie die Pakete direkt aus der ISE Console starten. Wir empfehlen die Engineer Erweiterung nur auf Clients zu installieren, auf welchen Scripting Framework Pakete erstellt werden.

9 Wissenswertes und weitere Einsatzbereiche

9.1 Sofortige Benutzerinstallation

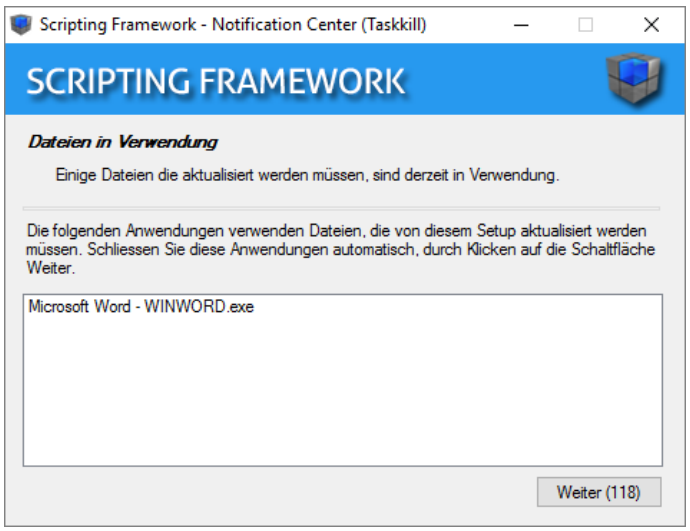
Die Benutzerkonfiguration (InstallUser.ps1) wird nach einer Installation automatisch auf allen auf dem Gerät angemeldeten Benutzern gestartet, sofern das Feature nicht über die Konfiguration deaktiviert wurde. Falls das Feature deaktiviert wurde, besteht die Möglichkeit den Task manuell über die EXE Datei zu starten: C:\Windows_ScriptingFramework\Modul\ActiveSetup.exe



Beim Logon kann Einfluss auf die Reihenfolge vom Anwenden der User Einstellungen genommen werden. Dazu kann im Install.ps1 der Wert Priority für die Benutzer Einstellungen geschrieben werden: SF-Register32 "HKEY_LOCAL_MACHINE" "SOFTWARE\ScriptingFramework\UserSettings\%_PkgIdentifier%" "Priority" "1" "REG_SZ". Die Priorität 1 ist dabei die höchste.

9.2 Notification Center (User Notifications)

Mit dem Scripting Framework Notification Center erhalten Sie die Möglichkeit, dass eine Interaktion mit dem Benutzer während einer Installation stattfinden (funktioniert auch wenn die Installation unter dem System Account abläuft). Beispiel einer Interaktion:

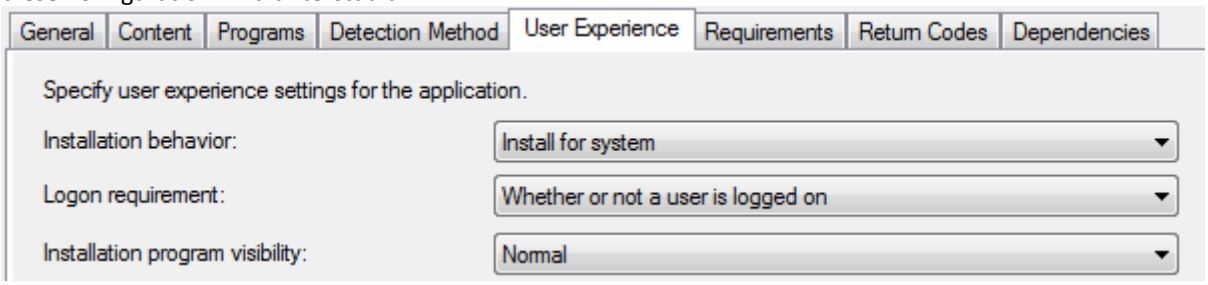


Das Interface unterstützt aktuell die Sprachen Deutsch, Englisch, Italienisch und Französisch und ist per Language.xml erweiterbar. Bei der Ausgabe von einer Meldung, wird das Notification Center automatisch in der jeweiligen Systemsprache des Benutzers angezeigt.

Sofern die Sprache nicht im XML hinterlegt ist, wird das Interface per Default in Englisch angezeigt. Das Titel Logo kann mit einem eigenen Bild ersetzt werden (C:\Windows_ScriptingFramework\Modul\NotificationCenterLogo.png).

Default Logo (480x50):

Im Zusammenhang mit SCCM funktioniert die Notification unabhängig der "User Experience" Einstellungen, auch diese Konfiguration wird unterstützt:



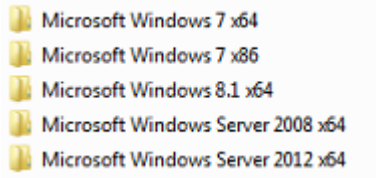
Folgende Befehle unterstützen aktuell eine Notification:

- SF-ShowInstallNotification
- SF-Taskkill

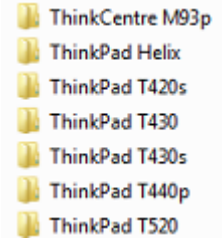
9.3 Installation von Treibern

Wir haben ein fertiges Scripting Framework Paket, für die einfache Integration von Treibern. Auf Anfrage stellen wir es Ihnen gerne zur Verfügung.

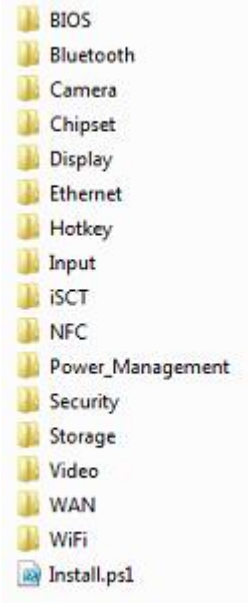
Mit dem Paket lassen sich PnP Treiber installieren, sowie auch Setuproutinen ausführen. Wir verwenden es auch für BIOS Updates und das Schreiben von BIOS Settings. Das Paket löst automatisch den Treiber Pfad des jeweiligen Hardwaremodells auf. Dafür wird ein Share mit folgender Struktur verwendet:



Struktur unter „Microsoft Windows 8.1 x64“:



Struktur unter Thinkpad Helix:



In der Install.ps1 können wie gewohnt alle Befehle von Scripting Framework verwendet werden. Beispiel:

```
# === Intel Bluetooth Driver ==========================================
f_Run "%_SystemDrive%\Drivers\Bluetooth\Setup.exe" "/qn" -Wait -NoErrors

# === Broadcom NFC Driver =============================================
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\NFC\GFWN04WW\*.inf""" -Wait -NoErrors

# === Video Intel Driver ==============================================
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\Video\GFDAS2WW\DisplayAudio\*.inf""" -Wait -NoErrors
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\Video\GFDAS2WW\Graphics\*.inf""" -Wait -NoErrors

# === Hotkey ==========================================================
f_Run "%_SystemDrive%\Drivers\Hotkey\SETUP.EXE" "/S" -Wait -NoErrors

# === WAN Driver ======================================================
f_Run "%_SystemDrive%\Drivers\WAN\setup.exe" "-S" -Wait -NoErrors

# === Cleaning ========================================================
f_RD  "%_SystemDrive%\Intel"
```

9.4 Lizenzpflichtige Fonts per User Session laden (Citrix)

Mit der Funktion *SF-FontInstall* haben Sie mit dem Parameter *-CurrentSessionLoad* die Möglichkeit Schriftarten nur für die aktuelle Session zu laden. Nach einer Abmeldung des Benutzers stehen die Schriften nicht mehr zur Verfügung. Zusammen mit dem „Scripting Mode“ setzen wir diese Funktion erfolgreich ein.