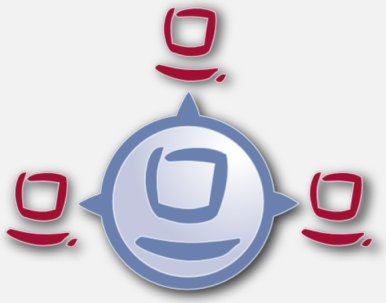

# opsi Setup Detector

uib gmbh
Bonifaziusplatz 1b
55118 Mainz
Tel.:+49 6131 275610
www.uib.de
info@uib.de

Stand: $Date$

# Inhaltsverzeichnis

# opsi Setup Detector (frei)

## Einführung

Die grundsätzlichen Schritte bei der Paketierung und Verteilung einer Software sind:

- Analyse des Setups und Erzeugen der Paket-Dateien auf der opsi Workbench
- Packen des opsi Paketes
- Installation des opsi Paketes auf dem opsi Server
- Installation der Software auf den Clients

Der *opsi Setup Detector* ist ein opsi Werkzeug zur Unterstützung der Software-Paketierung. Von seiner grafischen Benutzeroberfläche können alle Schritte von der Auswahl der zu paketierenden Setup-Datei, über das Erstellen der Installationsverzeichnisse auf der opsi Workbench, bis zur Paketierung und Installation auf dem opsi Server durchgeführt werden. Hierzu arbeitet der Setup Detector mit dem Community Projekt *opsi Package Builder* zusammen. Der *opsi Setup Detector* erkennt beim Öffnen einer Setup-Datei, ob es sich um einen bekanneten Installer-Typ handelt und ermittelt automatisch die verfügbaren Informationen aus der Setup-Datei. Mit der Zeit wird durch weitere Erfahrungen mit verschiedenen Installer-Typen und Rückmeldungen der Benutzer die Fähigkeit der automatischen Erkennung verschiedener Setup-Typen sicher noch weiter wachsen.

## Vorbedingungen für die Benutzung des opsi Setup Detectors

Der *opsi Setup Detector* läuft auf Windows Systemen ab Windows XP.

Er ist Teil der Standard-Repositories von opsi. Er kann mit dem folgenden Befehl installiert werden:

```
opsi-product-updater -i -p opsi-setup-detector
```

Falls die Pakete auch gepackt und auf dem Server installiert werden sollen, wird zusätzlich das Community Projekt opsi Package Builder benötigt, welches auf dem gleichen Windows-Rechner zu installieren ist. Weitere Informationen zum opsi Package Builder finden Sie im opsi Forum bei den opsi Community Projekten: https://forum.opsi.org/viewforum.php?f=22

Für die Erzeugung der Pakete ist ein Samba-Share mit Schreibberechtigung auf die opsi Workbench notwendig. Ausserdem ist zu beachten, dass manche Software-Pakete (z.B. manche 64 Bit Software) auch für die Integration ein entsprechendes Windows-System voraussetzen. Das heißt z.B. ein Setup für eine 64bit Windows7 Software kann eventuell nicht auf einem 32bit WindowsXP paketiert werden. Das hängt im Einzelfall ab vom Verhalten des jeweiligen Setups.

Hier nochmal die Voraussetzungen im Überblick:

- Windows-Rechner ab Windows XP
- opsi package Builder installieren und konfigurieren
- Samba-Share mit Schreibrechten zur opsi Workbench
- *opsi Setup Detector* installieren

## Inbetriebnahme des opsi Setup Detectors

Der *opsi Setup Detector* ist ein mit Lazarus entwickeltes Windows-Programm, das zusätzlich einige Hilfsdateien zur Analyse spezieller Setup-Formate sowie die Vorlagen-Dateien zur Erstellung eines neuen opsi Paketes benötigt. Der *opsi Setup Detector* mit seinen Zusatz-Dateien ist als opsi Paket verfügbar. Nach der Installation muss noch der Samba-Share auf die opsi Workbench als Basisverzeichnis eingetragen werden und schon kann mit *Datei öffnen* eine Setup-Datei geöffnet werden, die dann automatisch analysiert wird.

### Sprachunterstützung

Beim Start des Programmes wird automatisch ermittelt, unter welcher Sprache das Windows System läuft. Wenn für die Sprache *xx* eine passende Sprachdatei *languages\opsisetupdetector.xx.po* gefunden wird, wird diese verwendet. Von Haus aus unterstützt der *opsi Setup Detector* Deutsch und Englisch. Weitere Sprachen können recht einfach über eine entsprechend zu übersetzende Sprachdatei nachgerüstet werden.

### Dateien des opsi Setup Detectors

Folgende Dateien sind im *opsi Setup Detector* Paket enthalten und werden zur Ausführung benötigt:

- opsisetupdetector.exe - der *opsi Setup Detector*
- extractMSI.cmd - Hilfsdatei zur automatischen Analyse von MSI-Dateien
- MSIInfo.js - Hilfsdatei zur automatischen Analyse von MSI-Dateien
- innounp.exe - Hilfsdatei zur automatischen Analyse von Inno Setup Paketen
- innounp.htm - Hilfsdatei zur automatischen Analyse von Inno Setup Paketen
- favicon.ico - opsi Icon
- languages\ - Verzeichnis für die sprachspezifischen Komponenten
- languages\opsisetupdetector.po - default Sprachdatei (Englisch)
- languages\opsisetupdetector.de.po - Deutsche Sprachdatei
- languages\Help.en.html - Englische Hilfe
- languages\Help.de.html - Deutsche Hilfe

Zum Erzeugen der opsi Pakete werden verschiedene Vorlagen-Dateien benötigt. Diese werden dann beim Erzeugen eines neuen Paketes in die opsi Workbench kopiert und entsprechend den Einträgen des im *opsi Setup Detector* angezeigten Formulars gepatcht:

- files2opsi\OPSI\ - Vorlagen für die Paketierung-Informationen
- files2opsi\OPSI\control
- files2opsi\OPSI\postinst
- files2opsi\OPSI\preinst
- files2opsi\CLIENT_DATA\ - Vorlagen für die opsiscripte
- files2opsi\CLIENT_DATA\setup.opsiscript
- files2opsi\CLIENT_DATA\uninstall.opsiscript
- files2opsi\CLIENT_DATA\delsub.opsiscript
- files2opsi\CLIENT_DATA\check_advanced_exitcode.opsiscript
- files2opsi\CLIENT_DATA\check_inno_exitcode.opsiscript
- files2opsi\CLIENT_DATA\check_installshield_exitcode.opsiscript
- files2opsi\CLIENT_DATA\check_installshieldmsi_exitcode.opsiscript
- files2opsi\CLIENT_DATA\check_inno_exitcode.opsiscript
- files2opsi\CLIENT_DATA\check_msi_exitcode.opsiscript

## Das Menü des opsi Setup Detektors

Das Menü am oberen linken Rand enthält die Punkte

- Datei - (öffnet ein Untermenü, s.u.)
- Hilfe - interne Hilfe aufrufen. Um die enthaltenen Links zu verwenden, kann die Hilfe-Datei *%ProgramFiles-Dir%\opsiSetupDetector\languages\Help.de.html* alternativ auch direkt in einem Browser angezeigt werden.
- Info - Programm-Version anzeigen

Der Menüpunkt *Datei* öffnet ein Untermenü mit:

- *Datei öffnen und analysieren* - Auswählen des zu analysiernden Setup. Dieser Menüpunkt hat die gleiche Funktion wie der *Datei öffnen* Button auf dem *Analysiere* Tab rechts unten und öffnet einen Datei Auswahl Dialog, der alle Dateien anzeigt. Der Datei Auswahl Dialog auf den jeweiligen Installer Tabs zeigt nur Dateien an mit der Endung *.MSI*(*MSI* Tab) bzw. *.EXE* (alle anderen Installer Tabs).
- *Schreibe Logdatei* - Der Inhalt des Analyze Tab wird als Logdatei geschrieben. Der Name der erzeugten Datei entspricht *%TEMP%\opsitmp\opsiSetupDetector.log* und wird beim Erzeigen in einem Dialog angezeigt. Die Logdatei wird direkt beim Aufruf dieses Kommandos erzeugt und enthält nur die Informationen, die zum Zeitpunkt des Aufrufs im **Analysiere** Tab stehen.
- *Beenden* - beendet den opsi Setup Detektor

## Automatische Analyse eines Setup-Paketes

Nach dem Öffnen einer Setup-Datei mit *Datei öffnen* wird diese automatisch analysiert und, falls es sich um einen bekannten Installer-Typ handelt, automatisch auf den entsprechenden Tab gewechselt. Falls der Installer-Typ nicht erkannt wird, bleibt weiterhin das **Analysiere** Tab stehen. Auch wenn der Installer-Typ erkannt wurde, kann jederzeit auf das **Analysiere** Tab zurück gewechselt werden um zu sehen, was bei der automatischen Analyse gefunden wurde.

Bei der Analyse einer MSI-Datei ist bereits durch die Endung MSI bekannt, dass es sich um ein MSI-Paket handelt. Einer EXE-Datei sieht man es aber nicht an der Endung an, mit welchem Installer-Typ sie erstellt wurde. Deshalb wird die EXE-Datei vom *opsi Setup Detector* automatisch nach bestimmten Kennungen durchsucht. Falls die Kennung eines bekannten Installer-Typs gefunden wurde, wird direkt auf den Tab dieses Installers gewechselt. Falls kein bekannter Setup-Typ erkannt wurde, wird die Datei nach bestimmten allgemeinen Stichworten (z.B. "Installer" oder "Setup") durchsucht und die entsprechenden Zeilen werden auf dem **Analysiere** Tab ausgegeben. Darin können auch kryptische Passagen und bestimmte Fehlermeldungen enthalten sein, die das Setup-Programm in bestimmten Fällen, z.B. bei aufgetretenen Fehlern während der Installation, ausgeben möchte, wie z.B.:

```
o@dejDs@s@t@t@du@u@<v@w@w@lx@Hy@x@x{@X|@|@@@@L@x@@Inno Setup Setup Data
Das Setup hat einen schwerwiegenden Fehler erkannt und wird abgebrochen
```

Dies besagt aber nur, dass diese Zeichen innerhalb des Setups gefunden wurden und im **Analysiere** Tab angezeigt werden, da sie einem der verwendeten Suchmuster entsprechen. Oben kommt z.B. das Suchmuster "Setup" vor. Bei einem nicht automatisch erkannten Setup-Typ lässt sich in diesen Angaben eventuell ein Hinweis finden, um welchen Installer-Typ es sich handeln könnte. Alle Angaben auf dem **Analysiere** Tab, die eingeleitet werden von:

```
Analyzing: F:\<setup.exe>
```

und abgeschlossen werden mit:

```
default grep finished.
```

entstammen dieser automatischen Analyse und können im Normalfall unbeachtet bleiben. Falls der Setup-Typ erkannt wurde, steht dieser unterhalb von "default grep finished". Dass ein bekannter Setup Typ erkannt wurde merkt man auch daran, dass automatisch auf den entsprechenden Tab gewechselt wurde.

Je nach dem gefundenen Installer-Typ sind im **Analysiere** Tab noch weitere Informationen zu finden zur weiteren Analyse des Setups. Diese werden in den jeweiligen Kapiteln der entsprechenden Installer-Typen behandelt.

Der Inhalt des **Analysiere** Tabs wird vor einer neuen Analyse gelöscht und enthält dementsprechend nur Informationen zu dem aktuellen Setup.

## Setup-EXE mit eingebettetem MSI

Bevor auf die einzelnen Installer-Typen eingegangen wird, noch ein paar allgemeine Bemerkungen zu Setup-Dateien mit eingebettetem MSI: derzeit werden Advanced Installer oder InstallShield basierte EXE-Dateien mit eingebettetem MSI erkannt. Das MSI Paket wird automatisch entpackt und die analysierten Eigenschaften im **Advanced+MSI** Tab bzw. **InstallShield+MSI** Tab sowie zusätzlich im **MSI** Tab angezeigt. Eine EXE-Datei mit eingebettetem MSI kann auch mehrere unterschiedliche MSI-Pakete enthalten, z.B. für unterschiedliche Windows-Versionen oder 32bit/64bit. Das Setup entscheidet dann meist je nach den System-Gegebenheiten automatisch, welches MSI-Paket ausgepackt und verwendet wird. In diesem Fall erhält man bei Analyse des Paketes mit dem *opsi Setup Detector* nur das für das aktuelle System, auf dem der *opsi Setup Detector* läuft, passende MSI-Paket. Solche EXE-Dateien haben oft eine Kommandozeilen-Hilfe, die weitere Auskunft über die enthaltenen Pakete und die Aufruf-Möglichkeiten gibt. Auch kann eine Setup-EXE eventuell je nach System oder sonstigen Eigenschaften verschiedene MST-Dateien verwenden. Kurz und gut, eine Setup.EXE kann sich je nach den Umständen unterschiedlich verhalten. In den meisten Fällen braucht man lediglich den MSI Code des eingebetteten MSI für das opsi Deinstallations-Script. Hierbei ist zu beachten, dass ein MSI-Paket für 32bit und 64bit im Normalfall einen unterschiedlichen MSI Codes enthält. Das opsi Deinstallations-Script muss dann dafür sorgen, dass es alle diese MSI Codes kennt und bei der Deinstallation handhabt. Bei solchen Paketen sollte man den *opsi Setup Detector* auf unterschiedlichen Plattformen laufen lassen, um alle MSI-Kennungen zu erfassen. Wenn der *opsi Setup Detector* ein eingebettetes MSI findet, analysiert er dieses und zeigt die Inhalte zusätzlich im **MSI** Tab an. Es kann dann auch das **MSI** Tab als Basis für das opsi Paket genommen werden, aber im Normalfall sollte man besser die EXE-Datei nehmen, da auf diese Weise schneller eine neue Version eingebunden werden kann und die EXE-Datei eventuell auch zusätzliche Dinge tut.

Bei der Analyse wird das MSI und andere Komponenten der EXE-Datei in das Verzeichnis `%TEMP%\opsitmp` entpackt. Zuvor wird eventueller anderer Inhalt gelöscht, so dass im Verzeichnis `%TEMP%\opsitmp` immer nur Dateien des aktuell zu analysierenden Setups zu finden sind.

## Unterstützte Installer-Typen

Die einzelnen Installer-Typen sind sehr unterschiedlich bezüglich der Standardisierung ihrer Struktur und automatisierten Zugänglichkeit. Bei einem MSI-Paket oder einem auf Inno-Setup basierenden Setup kann eine Menge nützlicher Informationen automatisch detektiert werden, während andere Installer-Typen, wie z.B. NSIS, keine internen Informationen des Setups zugreifbar machen. In vielen Fällen bringt die automatisierte Analyse gute Ergebnisse, aber es kann auch immer wieder Fälle geben, bei denen die automatisierte Vorgehensweise nicht funktioniert oder zumindest händige Nacharbeit erfordert. Mit zunehmenden Erfahrungen beim Einsatz wird der *opsi Setup Detector* seine automatisierten Fähigkeiten mit der Zeit erweitern.

Da MSI-Pakete der wichtigste und am meisten verwendete Installer Typ sind, folgt nach dem **Analysiere** Tab der **MSI** Tab, danach alle weiteren Installer Typ Tabs in alphabetischer Reihenfolge.

### Installer-Typ MSI

MSI-Dateien sind das Installations-Datenformat des Microsoft Windows Installers. Setups vom Installer-Typ MSI sind bereits an der Endung MSI als solche zu erkennen. Das Paketformat ist standardisiert, so dass der *opsi Setup Detector* automatisch weitere Informationen aus dem Paket auslesen und im **MSI** Tab, auf den automatisch gewechselt wird, anzeigen kann. Die im **Analysiere** Tab angezeigten Informationen sehen bei einem MSI-Paket z.B. so aus:

```
Analyzing MSI: F:\Setups\MSI\7-zip\7-Zip.msi

Microsoft (R) Windows Script Host, Version 5.8
Copyright (C) Microsoft Corporation 1996-2001. Alle Rechte vorbehalten.
```

```
MSI file: F:\uib\setupdetector\Setups\MSI\7-zip\7-Zip.msi
Manufacturer: Igor Pavlov
ProductName: 7-Zip 4.57
ProductVersion: 4.57.00.0
ProductCode: {23170F69-40C1-2701-0457-000001000000}
UpgradeCode: {23170F69-40C1-2701-0000-000004000000}
MSI file size is: 0,9 MB
Estimated required space is: 5,2 MB

get_MSI_info finished
```

Diese Informationen werden aus dem MSI-Paket gelesen und automatisch in die einzelnen Felder des **MSI** Tabs übernommen. Diese Angaben werden benötigt zum Patchen der opsi Installations-Scripte:

- **MSI Datei:** Name der geöffneten MSI Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein MSI-Paket für 32bit und 64bit hat normalerweise einen jeweils unterschiedlichen Produkt Code.
- **erforderlich:** dieser vorgeschlagene Wert stammt nicht aus dem MSI-Paket, sondern ist geschätzt als sechsmal die Größe der MSI-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Größe MSI:** die ermittelte Größe des MSI-Paketes.
- **MST Datei:** wird im gleichen Verzeichnis (also dort, wo die MSI-Datei liegt) eine MST-Datei (Transform-Datei) gefunden, wird diese eingetragen und beim Aufruf des Setup verwendet. Gegebenenfalls kann mit dem *Auswahl*-Button rechts neben dem Feld eine andere MST-Datei gewählt werden oder durch Entfernen des MST Datei-Häkchens vor dem Feld die Verwendung der MST-Datei ausgeschaltet werden.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *license=true* gesetzt.

Im Falle einer EXE-Datei mit eingebettetem MSI wird das MSI Paket automatisch extrahiert und die gefundenen Informationen zusätzlich zum Tab der EXE-Date (**Advanced+MSI** oder **InstallShied+MSI**)ebenfalls hier im **MSI** Tab angezeigt.

#### Installer-Typ Advanced+MSI

Der Advanced Installer ist ein Werkzeug zur Erstellung von MSI-Paketen, die auch in EXE-Dateien eingebettet sein können. Erkennt der opsi Setup Detektor eine EXE-Datei als ein mit Advanced Installer eingebettetes MSI-Paket, wird die MSI-Datei automatisch aus der EXE-Datei extrahiert und analysiert und die gefundenen Informationen im Tab **Advanced+MSI** angezeigt. Die entsprechenden Informationen werden zusätzlich auch im **MSI** Tab eingetragen. Nähere Angaben zum **MSI** Tab siehe Kapitel Abschnitt 1.7.1.

Beim automatischen Auspacken einer eingebetteten MSI-Datei wird im **Analysiere** Tab Folgendes angezeigt:

```
xxxxxxxxxxxxxxxxxxxxxxxx
Analyzing AdvancedMSI Setup:
F:\Setups\AdvancedMSI\Phase5\phase562install.exe
Analyzing MSI from Setup
F:\Setups\AdvancedMSI\Phase5\phase562install.exe
cmd.exe /C "F:\Setups\AdvancedMSI\Phase5\phase562install.exe" /extract:e:\Temp\opsitmp\
!! PLEASE WAIT !!
!! PLEASE WAIT !! Extracting and analyzing MSI ...
!! PLEASE WAIT !!
e:\Temp\opsitmp\*.msi
Analyzing MSI: e:\Temp\opsitmp\phase5.msi
```

Das MSI wird in das Verzeichnis =%TEMP%\opsitmp= extrahiert und analysiert. Mit dem Ergebnis werden die Felder des **Advanced+MSI** Tab gefüllt:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein Advanced-Installer-Paket kann unterschiedliche MSI-Pakete mit unterschiedlichem MSI Produkt Code für z.B. 32bit und 64bit enthalten und es hängt vom System des Paketierungs-PCs ab, welches MSI-Paket ausgepackt und analysiert wird. Meist kann ein 64bit-Paket nicht auf einem 32bit-Rechner entpackt werden. Zur Paketierung einer 64bit-Software wird somit auch ein 64bit-Paketierung-Rechner gebraucht.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und kann gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *$LicenseRequired$=true* gesetzt.

**Signatur Advanced+MSI:** bei der automatischen Erkennung einer Advanced+MSI-Datei wird in der EXE-Datei gesucht nach dem Marker:

```
name="microsoft.windows.advancedinstallersetup
```

#### Installer-Typ Inno Setup

Mit Inno Setup erstellte EXE-Dateien sind weitgehend standardisiert und enthalten eine spezifische Datei mit Namen *install_script.iss*, die automatisch aus der EXE-Datei extrahiert und analsiert werden kann. Mit den ermittelten Informationen werden die Felder des **Inno Setup Tab** gefüllt und dieser angezeigt. Wenn man zurück in den **Analysiere** Tab klickt, sieht man ungefähr folgende Angaben:

```
.....................................
Analyzing: F:\Setups\Inno\openssl\Win32OpenSSL_Light-1_0_0i.exe
stringsgrep started (verbose:false, skipzero:false)
found string "<description>Inno Setup</description>"
detected Inno Setup
stringsgrep completed (verbose:false, skipzero:false)
.....................................
Analyzing Inno-Setup:
extract install_script.iss from F:\Setups\Inno\openssl\Win32OpenSSL_Light-1_0_0i.exe to
C:\ProgramData\opsi setup detector\INNO\Win32OpenSSL_Light-1_0_0i\install_script.iss
"E:\Program Files (x86)\opsiSetupDetector\innounp.exe" -x -a -y -d"C:\ProgramData\opsi setu
; Version detected: 5402
#66 install_script.iss
C:\ProgramData\opsi setup detector\INNO\Win32OpenSSL_Light-1_0_0i\install_script.iss
........
[Setup]
AppName=OpenSSL Light (32-bit)
AppVerName=OpenSSL 1.0.0i Light (32-bit)
AppPublisher=OpenSSL Win32 Installer Team
AppPublisherURL=http://www.openssl.org
AppSupportURL=http://www.slproweb.com
AppUpdatesURL=http://www.slproweb.com/products/Win32OpenSSL.html
DefaultDirName={sd}\OpenSSL-Win32
DefaultGroupName=OpenSSL
OutputBaseFilename=Win32OpenSSL_Light-1_0_0i
Compression=bzip2
........
Setup file size is: 1,9 MB
Estimated required space is: 11,1 MB
........
get_inno_info finished
Inno Setup detected
```

Die Informationen aus *install_script.iss* werden automatisch in die einzelnen Felder des **Inno Setup** Tabs übernommen und dann verwendet zum Patchen der opsi Installations-Scripte:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus *install_script.iss* ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl in vielen Fällen beibehalten. Sie darf allerdings nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes. Das Installations-Verzeichnis wird aus *install_script.iss* ausgelesen und kann in vielen Fällen direkt verwendet werden. Manchmal kann die Angabe des Installations-Verzeichnisses auch Inno Setup-spezifische Konstanten und Variablen enthalten, die in geschweiften Klammern angegeben sind. Einige der Inno Setup Variablen werden automatisch in opsi Syntax übersetzt, z.B. Es sind aber auch andere Einträge möglich. Hier ist dann händige Nacharbeit angesagt, so dass in diesem Feld nur Bezeichner stehen, die vom opsi Installer (Winst) verstanden werden. In obigem Fall wäre das z.B.: %Systemdrive%\lazarus Eventuell kann hierzu auch der *install_script.iss* Eintrag *UninstallDisplayName* ausgewertet werden. Eine Beschreibung der Inno Setup Variablen ist in der Inno Setup Hilfe zu finden.

- **erforderlich:** dieser vorgeschlagene Wert stammt nicht aus der *install_script.iss*, sondern ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *$LicenseRequired$=true* gesetzt.

**Signatur Inno Setup:** bei der automatischen Erkennung einer Inno Setup-Datei wird in der EXE-Datei gesucht nach dem Marker:

```
<description>inno setup</description>
```

#### Installer-Typ InstallShield

Mit Installshield erstellte EXE-Dateien sind für eine automatisierte Analyse nicht zugänglich. Es kann lediglich automatisch erkannt werden, dass es sich um ein Installshield Setup handelt. Daher werden die vorgeschlagenen Werte aus dem Namen der Setup-Datei erstellt. Wenn diese Datei lediglich *Setup.exe* heißt, wird in den Feldern auch nur *Setup* eingetragen. Hinzu kommt noch, dass es von der jeweiligen Version des verwendeten Installshield abhängt, welche Aufruf-Parameter die Setup-Datei sowie das Deinstallations-Programm akzeptieren. Bei einem Installshield-Paket ist also weitgehend Handarbeit angesagt. Der erste Test könnte darin bestehen, die Aufrufparameter des Setup zu ermitteln, z.B. durch *Setup.exe -?*. Da mit InstallShield erstellte Pakete inzwischen kaum mehr reines InstallShield, sondern zunehmend interne MSI-Pakete enthalten, kommen bei der Paketierung reine InstallShield-Pakete kaum mehr vor, was den Paketierer in Zukunft von einem chronischen Sorgenkind befreit.

Wenn man zurück in den **Analysiere** Tab klickt, sieht man ungefähr folgende Angaben:

```
.....................................
Analyzing: F:\uib\setupdetector\Setups\InstallShield\viclient\VMware-viclient.exe
stringsgrep started (verbose:false, skipzero:false)
found string "InstallShield"
detected InstallShield Setup
stringsgrep completed (verbose:false, skipzero:false)
.....................................
Analyzing InstallShield-Setup:
Setup file size is: 32,1 MB
Estimated required space is: 192,6 MB
........
get_InstallShield_info finished
InstallShield Setup detected
```

Die einzelnen Felder des **InstallShield** Tab sind:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der Name der zu installierenden Software. Dieser muss evtl. händig korrigiert werden
- **Produkt Version:** die aus dem Name der Setup-Datei ermittelte Versionsnummer muss wahrscheinlich händig korrigiert werden. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.

- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes und muss händig eingetragen werden.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *$LicenseRequired$=true* gesetzt.

**Signatur InstallShield:** bei der automatischen Erkennung einer InstallShield-Datei ohne eingebettetes MSI wird in der EXE-Datei gesucht nach dem Marker:

```
<description>InstallShield.Setup</description>
```

ohne dass der Marker für ein eingebettetes MSI gefunden wird. Falls auch der Marker für ein eingebettetes MSI gefunden wird, handelt es sich um ein **InstallShield+MSI** (s.u.).

#### Installer-Typ InstallShield+MSI

Mit InstallShield erstellte Software-Pakete enthalten meist intern eine (oder mehrere?) MSI-Dateien. Um die MSI-Datei zu extrahieren, wird ein Batch gestartet, der das Setup startet und zehn Sekunden wartet, welche MSI-Datei beim Starten ausgepackt wird. Dann wird der Setup-Task abgeschossen. Das extrahierte MSI-Paket wird dann automatisch analysiert und die Informationen in den **InstallShield+MSI** Tab sowie den **MSI** Tab geschrieben. Hier hängt es eventuell auch von der jeweiligen EXE-Datei ab, ob und wenn ja welches MSI-Paket sie auspackt. Eventuell packt sie keines aus, weil das Betriebssystem unpassend ist oder weil erst noch an der GUI eine Frage beantwortet werden muss, bevor ein MSI ausgepackt wird. Falls die automatische Extrahierung nicht klappt, sollte das MSI händig ausgepackt werden und als MSI analysiert werden. Diese Angaben können dann in den **InstallShield+MSI** Tab über tragen werden.

Im **Analysiere** Tab sind typischerweise solche Einträge zu finden:

```
.....................................
Analyzing: F:\Setups\Installshield-MSI\javavm\jre-6u22-windows-x64.exe
stringsgrep started (verbose:false, skipzero:false)
found strings "Installer,MSI,Database" and "InstallShield"
detected InstallShield+MSI Setup (InstallShield with embedded MSI)
found strings "Installer,MSI,Database" and "InstallShield"
detected InstallShield+MSI Setup (InstallShield with embedded MSI)
stringsgrep completed (verbose:false, skipzero:false)
.....................................
Analyzing InstallShield+MSI Setup: F:\Setups\Installshield-MSI\javavm\jre-6u22-windows-x64
Analyzing MSI from InstallShield Setup F:\Setups\Installshield-MSI\javavm\jre-6u22-windows
cmd.exe /C E:\Program Files (x86)\opsiSetupDetector\extractMSI.cmd "F:Setups\Installshield
!! PLEASE WAIT !!
!! PLEASE WAIT !! Extracting and analyzing MSI ...
!! PLEASE WAIT !!
e:\Temp\opsitmp\*.msi
Analyzing MSI: e:\Temp\opsitmp\phase5.msi
cmd.exe /C cscript.exe "E:\Program Files (x86)\opsiSetupDetector\msiinfo.js" "e:\Temp\opsit
........
Microsoft (R) Windows Script Host, Version 5.8
Copyright (C) Microsoft Corporation 1996-2001. Alle Rechte vorbehalten.
```

```
MSI file: e:\Temp\opsitmp\phase5.msi
Manufacturer: Systemberatung Schommer
ProductName: Phase 5 HTML-Editor
ProductVersion: 5.6.2.3
ProductCode: {20B1B020-DEAE-48D1-9960-D4C3185D758B}
UpgradeCode: {C63B6E47-6A28-44B6-A2C9-2BF084491FAD}
MSI file size is: 0,3 MB
Estimated required space is: 1,7 MB
........
get_MSI_info finished
Setup file size is: 15,4 MB
Estimated required space is: 92,7 MB
........
get_InstallShield_info finished
InstallShield+MSI Setup detected
```

Falls die automatische Extrahierung des MSI geklappt hat, werden die gefundenen Werte automatisch in den **InstallShield+MSI** Tab eingetragen:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der aus dem MSI Paket ausgelesene Name der zu installierenden Software. Diesen Namen möchte man wahrscheinlich im Normalfall beibehalten.
- **Produkt Version:** die ausgelesene Version der zu installierenden Software. Auch diese Angabe möchte man wohl im Normalfall beibehalten. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **MSI Produkt Code:** der aus dem Paket ausgelesene Produkt-Code, welcher für das Deinstallations-Script benötigt wird. Vorsicht: ein MSI-Paket kann unterschiedliche MSI Produkt Codes für z.B. 32bit und 64bit enthalten und es hängt vom System des Paketierungs-PCs ab, welches MSI-Paket ausgepackt und analysiert wird. Meist kann ein 64bit-Paket nicht auf einem 32bit-Rechner entpackt werden. Zur Paketierung einer 64bit-Software wird somit auch ein 64bit-Paketierung-Rechner gebraucht.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und kann gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *$LicenseRequired$=true* gesetzt.

**Signatur InstallShield+MSI:** bei der automatischen Erkennung einer InstallShield-Datei mit eingebettetem MSI wird in der EXE-Datei gesucht nach dem Marker für InstallShield und dem Marker für ein eingebettetes MSI:

```
<description>InstallShield.Setup</description>
...
installer,msi,database
```

### Installer-Typ NSIS

NSIS Pakete zeichnen sich insbesondere dadurch aus, dass sie kompakt und schnell sind. Allerdings ist es dadurch nicht möglich, dem Paket automatisiert nähere Angaben zu entlocken. Ausser der Erkennung, dass es sich um ein NSIS-Paket handelt, können alle weiteren Informationen lediglich aus dem Namen der Setup-Datei vorgeschlagen werden. Hier ist also händige Nacharbeit angesagt.

Die Einträge im **Analysiere** Tab sehen typischerweise so aus:

```
.....................................
Analyzing: F:\Setups\NSIS\7z920\7z920.exe
stringsgrep started (verbose:false, skipzero:false)
found string "Nullsoft.NSIS.exehead"
detected NSIS Setup
stringsgrep completed (verbose:false, skipzero:false)
.....................................
Analyzing NSIS-Setup:
Setup file size is: 1,1 MB
Estimated required space is: 6,4 MB
........
get_nsis_info finished
NSIS (Nullsoft Install System) detected
```

Die Bedeutung der einzelnen Felder des **NSIS** Tab:

- **Setup-Datei:** Name der analysierten Setup-Datei
- **opsi Product ID:** dies ist der Name des zu erzeugenden opsi Paketes und wird aus dem weiter unten stehenden Produkt Namen erzeugt, wobei Leerzeichen und andere ungültigen Zeichen durch ein - ersetzt werden. Die vorgeschlagene opsi Product ID kann natürlich geändert werden.
- **Produkt Name:** der Name der zu installierenden Software. Dieser muss evtl. händig korrigiert werden
- **Produkt Version:** die aus dem name der Setup-Datei ermittelte Versionsnummer muss wahrscheinlich händig korrigiert werden. Sie darf nur Ziffern und Punkte enthalten, da sie für die Versionierung des opsi Paketes verwendet wird.
- **Install. Verzeichnis:** dies ist der Name des Verzeichnisses, in das die Software auf dem Client installiert wird. Dieses wird gebraucht zum Patchen des Deinstallations-Skriptes und muss händig eingetragen werden.
- **erforderlich:** dieser vorgeschlagene Wert ist geschätzt als sechsmal die Größe der Setup-Datei und sollte gegebenenfalls angepasst werden. Bei der Installation auf dem Client prüft das setup.opsiscript, ob entsprechend dieser Angabe genügend Plattenplatz vorhanden ist.
- **Dateigröße:** die ermittelte Größe der Setup-Datei.
- **Beschreibung:** In diesem Feld wird als Vorgabe der Produkt Name vorgegeben und sollte mit weiteren Hinweisen ergänzt werden, die dann als Produktbeschreibung des opsi Paketes gesetzt werden.
- **Lizenz pflichtig:** Wenn dieses Häkchen gesetzt wird, wird beim Patchen des opsiscripts *$LicenseRequired$=true* gesetzt.

**Signatur NSIS:** bei der automatischen Erkennung einer NSIS Setup-Datei wird in der EXE-Datei gesucht nach einem der Marker:

```
Nullsoft.NSIS.exehead
...
Nullsoft Install System
```

## Erzeugen eines neuen opsi Paketes

Wenn das Setup analysiert und die Felder des entsprechenden Tab gefüllt sind, kann mit dem Button *Erzeuge opsi Paket* (rechts unten) das neue Produkt erzeugt werden.

**Achtung:**

- als *Basisverzeichnis* muss ein beschreibbarer Share auf die opsi Workbench eingetragen sein
- aus Sicherheitsgründen kann ein opsi Paket nur dann neu erzeugt werden, wenn es noch nicht vorhanden ist. Falls ein bestehendes Paket überschrieben werden soll, muss zuerst das Verzeichnis von der opsi Workbench gelöscht werden.

Links neben dem Button *Erzeuge opsi Paket* befinden sich drei mögliche Auswahl Optionen, die sich auf die Funktion des Buttons beziehen:

- **opsi Paket erzeugen:** falls noch nicht vorhanden, wird das Verzeichnis für das neue opsi Paket erzeugt und die entsprechenden Dateien erstellt und gepatcht. Es wird kein Paket gebaut.
- **opsi Paket erzeugen und opsi Packet Builder starten:** wie oben werden das Verzeichnis und die Dateien erzeugt und dann zum Packen und installieren der *opsi Packet Builder* gestartet. Die Paket Daten werden beim Aufruf in diesen übernommen und das Paket kann gebaut und installiert werden. Danach sollte der *opsi Packet Builder* wieder geschlossen werden, damit er für den nächsten Aufruf neu gestartet werden kann.
- **erzeugen und auto -build -install -quiet:** wie oben werden das Verzeichnis und die Dateien erzeugt und dann zum Packen und installieren der *opsi Packet Builder* gestartet. Die Paket Daten werden beim Aufruf in diesen übernommen und das Paket wird automatisch gebaut und installiert, falls die entsprechenden Häkchen gesetzt sind. Wenn z.B. das Paket nur gepackt, aber nicht installiert werden soll, kann das Häkchen bei **install** entfernt werden. Zusätzlich kann das Häkchen bei **quiet** gesetzt werden, dann werden die angewählten Funktionen automatisch ausgeführt, ohne dass die Benutzeroberfläche des *opsi Packet Builders* angezeigt wird. Ansonsten sollte der *opsi Packet Builder* dann wieder geschlossen werden, damit er für den nächsten Aufruf neu gestartet werden kann.

Zu Installation, Konfiguration und Bedienung des Community Projektes *opsi Packet Builder* siehe https://forum.opsi.org/viewforum.php?f=22