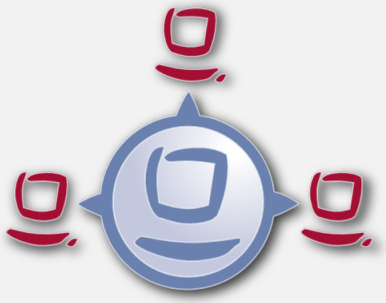

opsi Setup Detector

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

Stand: $Date: 2013-10-23 23:06:53 +0200 (mer 23 oct 2013) $

Contents

opsi Setup Detector (free)

Introduction

The basic steps of software packaging and distribution are:

- analyzing the setup and creating the files on the opsi workbench
- packing the opsi package
- installing the opsi package on the opsi server
- installing the software on the clients (roll out)

The opsi setup detector is a tool for supporting the packaging of software to prepare for software rollouts. All the steps can be done from the graphical user interface: selecting and analyzing the setup, creating the files on the opsi workbench, packing and installing the package on the opsi server. For packing and installing, the opsi setup detector invokes the community projekt *opsi Package Builder*. When opening a setup file, the opsi setup detector checks whether it is a well known Installer type and then automatically detects the available parameters from the setup file. Over the time, by feed back from opsi users and increasing experience with different installer types and setup files, the setup detector will improve its ability to detect different setup types.

Preconditions for using the opsi Setup Detector

The *opsi Setup Detector* runs on Windows XP or above.

It is part of the default repositories of opsi. It can be installed with the following command:

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

If also packets shall be packed and installed on the opsi server, the community projekt *opsi Package Builder* is required to be installed on the same Windows client. For further information about the *opsi Package Builder* see the Community Projects section of the opsi Forum: https://forum.opsi.org/viewforum.php?f=22

For creating the files and packets a writable Samba share to the opsi Workbench is required. Further on some software setups (especially for 64 bit software) require for analyzing and integration the matching Windows system. This means, that analyzing and integrating software for 64bit Windows7 might not run on a 32bit Windows XP Client. This must be considered in each individual case.

So this is the list of preconditions:

- Windows clients from Windows XP and above
- opsi package Builder installed and configured
- writable Samba share to the opsi Workbench
- opsi Setup Detector itself with all its helper files

Setting up the opsi Setup Detector

The *opsi Setup Detector* is a Windows program developed with Lazarus. It requires some additional helper files to analyze special Setup types and some templates to generate the files for a new opsi packet from. The *opsi Setup Detector* is available as an opsi package, which includes all its helper files (but not the community project *opsi Packet Builder*). After installing the *opsi Setup Detector*, just the Share to the opsi Workbench has to be configured as *Packet BaseDir* and then a setup file to analyze can be opened with *File - Open Setup File And Analyze*.

Language Support

When starting the *opsi Setup Detector*, the language of the Windows system is detected and, if for language *xx* a suitable language file *languages\opsisetupdetector.xx.po* is detected, it will be used. The *opsi Setup Detector* comes with German and English language files. Additional languages can be supported by generating a suitable language file.

Files of the opsi Setup Detector

The *opsi Setup Detector* package contains the following files:

- opsisetupdetector.exe - the *opsi Setup Detector* itself
- extractMSI.cmd - helper file for analyzing Setups with embedded MSI files
- MSIInfo.js - helper file to analyze MSI files
- innounp.exe - helper file to analyze Inno Setup files
- innounp.htm - help for the innounp.htm helper file
- favicon.ico - opsi Icon
- languages\ - directory for language specific files
- languages\opsisetupdetector.po - default language file (English)
- languages\opsisetupdetector.de.po - German language file
- languages\Help.en.html - English help file
- languages\Help.de.html - German help file

To create new opsi packages on the opsi workbench, several template files are required. These templates will be copied and patched according to the information from the *opsi Setup Detector*:

- files2opsi\OPSI\ - templates for the opsi packaging
- files2opsi\OPSI\control
- files2opsi\OPSI\postinst
- files2opsi\OPSI\preinst
- files2opsi\CLIENT_DATA\ - templates for the opsiscripts
- 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

The Menu of opsi Setup Detektor

The menu on the upper left contains the following options:

- File - (opens a sub menu, see below)
- Help - show help in an *opsi Setup detector* window. The help file *%ProgramFiles-Dir%\opsiSetupDetector\languages\Help.de.html* also can be displayed with a browser.
- About - Show program version

The menu option *File* opens a sub menu with:

- *Open Setup File and Analyze* - select the Setup to be analyzed. This option has the same effect as the *Open Setup File* button on the *Analyze* Tab down at the right side and opens a file selection dialog, which shows all file types. The file selection dialog on the other setup tabs only shows files with the extension *.MSI*(*MSI* Tab) or *.EXE* (on all the other Tabs).
- *Create Logfile* - The current content of the Analyze Tab will be written as logfile. The name of the created logfile is *%TEMP%\opsitmp\opsiSetupDetector.log* and will be shown when generating a logfile. The logfile will be generated when executing this command and only contains information, that is currently shown in the *Analyze* tab.
- Exit - quits the *opsi Setup Detektor*

Automated Analysis of a Setup file

When opening a setup file with *Open Setup File and Analyze*, it will be analyzed automatically and, if it is a well known Installer type, automatically switched to the suitable tab. If the installer type is not detected, the **Analyze** tab keeps the focus. Even when the installer type was detected, you can switch back to the **Analyze** tab any time to check the info details sampled by the analysis.

When analyzing a MSI file, the extension *.MSI* already indicates a MSI setup file. When analyzing an *.EXE* file the extension is not very specific and could be anything. So the EXE file is scanned for special markers to indicate the installer type. If the marker of a well known Installer type is found, the focus switches to the corresponding installer type tab. This might give some clue to what kind of installer it is. If the Setup type cannot be detected, the file will be scanned for a special set of words like (e.g. "Installer" or "Setup") and the corresponding lines will be written to the **Analyze** tab. This also might show some cryptic text passages and error codes, which are contained within the setup file like:

```
o@dejDs@s@t@t@du@u@<v@w@w@lx@Hy@x@x{@X|@|@@@@L@x@@Inno Setup Setup Data
The Setup has detected a serious error and quits
```

This just means, that these strings are found within the setup file and match the search pattern, so they are written to the **Analyze** Tab . In this case the search pattern is "Setup". In case of an unknown setup type this might give some clue about the installer type. All output on the **Analyze** Tab, which is preceded by:

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

und closed by:

```
default grep finished.
```

derives from the automated analysis. If the setup type is detected, it is written below "default grep finished". In case of a well known installer type, the focus switches to the corresponding Tab.

According to the installer type some more information is to be found in the **Analyze** tab. This will be discussed in each of the installer chapters.

The content of the **Analyze** tab is cleared when starting a new analysis and so only contains information from the current setup analysis.

Setup-EXE with embedded MSI

Before going into the different installer types, just a few general things about setup files with embedded MSI: at the moment the *opsi Setup Detector* automatically detects embedded MSI that are wrapped by by Advanced Installer and Installshield based EXE files. The *opsi Setup Detector* extracts and analyses the MSI and writes the detected information to the **Advanced+MSI** Tab or **InstallShield+MSI** Tab and additionally to the **MSI** Tab. An EXE file with embedded MSI can contain several different MSI packages, e.g. for different Windows platforms 32bit/64bit. Usually a setup like this detects the system to use the matching MSI. In this case, when analyzing a setup, only the MSI will be detected for the system where the *opsi Setup Detector* is running on. Such EXE files often have command line options to get more information about the package and install options. Also there might be several MST-files, which will be used in case of detectable conditions. To make a long story short: you never really know, what a setup might do under different conditions. So in most cases you just need the MSI code to patch the deinstallation script. So you just have to make sure, that you have captured all of the codes to be handled in the deinstallation script. For instance a MSI for 32bit and 64bit usually has got a different code. To get all of the codes, you can run the *opsi Setup Detector* on different platforms to grab the MSI codes. When the *opsi Setup Detector* detects an embedded MSI, it will be analyzed and the information is shown in the MSI tab. It is possible to create the opsi packet based on the **MSI** tab. But usually you would take the EXE file for the packet, for it is easier to adapt it to updates and also the EXE file might do some additional jobs.

During the analysis the MSI and other components of the EXE file will be extracted to the directory `%TEMP%\opsitmp`. Before doing so, the directory will be cleaned, so all files to be found there are from the current setup.

Supported Installer Types

Some of the well known installer types have pretty good standards and are accessible for automated analysis. From a MSI packet or an Inno Setup based Setup a lot of useful information can be detected automatically, whereas other Installer types, like NSIS, don't make any information accessible. So in many cases the automatic analysis gives good results, but in other cases it doesn't work at all or requires to work it over. With increasing experience from user results the detection skill of the *opsi Setup Detector* will surely improve.

MSI packages are the most important and widespread Installer type, so the **MSI** tab is the first one to follow the *Analyze tab, and then the others in alphabetical order.

Installer Type MSI

MSI files are the installer format of the Microsoft Windows Installer. Setups of type MSI can be detected by the file extension .MSI. The internal format of MSI packages is a well known standard, so the *opsi Setup Detector* can read several information from the packet to show on the **Analyze** tab and the **MSI** Tab, where the focus will switch to. When analysing a MSI, the detected information shown in the **Analyze** Tab looks like this:

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

Microsoft (R) Windows Script Host, Version 5.8
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

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
```

The detected information is extracted from the MSI packet and automatically fills in the **MSI** tab form. It is used to patch the opsiscript installation/deinstallation scripts:

- **MSI file:** Name of the analysed MSI file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from the MSI. This name can be changed, but most likely the detected name will fit.
- **Product Version:** the version of the software as detected from the MSI. Usually the detected value can be taken, but it should contain only numbers and points.
- **MSI Product Code:** the MSI product code as detected from the MSI. It will be used to patch the deinstall opsiscript. Caveat: a MSI product for 32bit and 64bit usually has a different product code.
- **Required Space:** this value is not read from the MSI packet, but is assumed as being six times the size of the MSI file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **MSI File Size:** the size of the MSI file.
- **use MST File:** if there is a MST file (Transform file) to be found in the same directory (where the MSI file is), its name will be filled in this edit field and will be checked to use for installation. By using the *Select* button on the right, another MST file can be selected. By removing the selection mark on the left, the MST file will not be used.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

In case of analysing an EXE file with embedded MSI, the MSI file will be extracted automatically and the detected information will be shown in the **MSI** tab in addition to the corresponding tab (**Advanced+MSI** or **InstallShied+MSI**).

Installer Type Advanced+MSI

The Advanced Installer is a tool for creating MSI packets that also can be embedded in EXE setup files. When the *opsi Setup Detector* detects an EXE file as an MSI embedded with Advanced Installer, it extracts the MSI file from the EXE file and analyses the information. The result is shown in the **Advanced+MSI** tab and in addition in the **MSI** tab. For further information on the **MSI** tab see chapter Section 1.7.1.

When extracting and analysing an embedded MSI from an Advanced Installer setup the **Analyze** tab will look like this:

```
xxxxxxxxxxxxxxxxxxxxxxxxx
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
```

The MSI will be extracted to the directory =%TEMP%\opsitmp= and then analysed. With the result the **Advanced+MSI** tab will be filled in:

- **Setup file:** Name of the analysed setup file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from the MSI. This name can be changed, but most likely the detected name will fit.
- **Product Version:** the version of the software as detected from the MSI. Usually the detected value can be taken, but it should contain only numbers and points.
- **MSI Product Code:** the MSI product code as detected from the MSI. It will be used to patch the deinstall opsiscript. Caveat: a MSI product for 32bit and 64bit usually has a different product code and an Advanced Installer packet might contain different MSI files. In this case it depends on the system the analysis is running on, which MSI is extracted and analysed. Usually a 64bit MSI cannot be extracted on a 32bit system. So for packing a 64bit software, a 64 system should be used.
- **Required Space:** this value is not read from the MSI packet, but is assumed as being six times the size of the EXE file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **Setup File Size:** the size of the EXE file.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

Signature Advanced+MSI: for detection of an Advanced+MSI setup, the EXE file is scanned for the marker:

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

Installer Type Inno Setup

Setups created with Inno Setup follow a pretty well known standard and contain a special file named *install_script.iss*, which can be extracted from the EXE file. The information from *install_script.iss* is filled into the edit fields of the **Inno Setup** tab and the focus switches to this tab. Going back to the **Analyze** Tab it looks like this:

```
.........................................
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
```

The information from *install_script.iss* is taken to fill in the edit fields of the **Inno Setup** tabs to patch the opsiscripts:

- **Setup file:** Name of the analysed setup file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from *install_script.iss*. This name can be changed, but most likely the detected name will fit.
- **Product Version:** the version of the software as detected from *install_script.iss*. Usually the detected value can be taken, but it should contain only numbers and points.
- **Install Directory:** this is the name of the directory, where the software is supposed to be installed on the client. This is taken to patch the deinstall script. The installation dir is read from *install_script.iss* and usually can be taken as is. But sometimes this entry might contain some Inno Setup specific notations, which are usually enclosed in curly braces. Some of them the *opsi Setup Detector* will translate to opsiscript syntax, like There could be a number of other entries in curly brackets, which opsiscript does not understand. In this case manual translation to opsiscript syntax is required. The entry *UninstallDisplayName* from *install_script.iss* might give some clue to this. A list and description of Inno Setup constants can be found in the Inno Setup help, e.g. http://www.jrsoftware.org/ishelp/index.php
- **Required Space:** this value is not read from *install_script.iss*, but is assumed as being six times the size of the EXE file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **Setup File Size:** the size of the EXE file.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

Signature Inno Setup: for detecting a setup as Inno Setup, the EXE file is scanned for the marker:

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

Installer Type InstallShield

EXE files created with Installshield are not accessible for automated analysis. It just can be detected, that it is a setup of type Installshield. So all the entries to be found in the **InstallShield** tab are derived from the name of the setup file. So if the filename is simply *Setup.exe*, the edit fields will be filled with *Setup*. In addition it depends on the Installshield version, what command line parameters are accepted by the setup and the deinstallation program. So integrating an Installshield setup is mainly manual work. The first test could be to check out available command

line parameters of the setup, e.g. with *setup -?*. More and more Installshield packets contain embedded MSI files. So pure Installshield packets without MSI will become less important.

When analysing an Installshield setup, the **Analyze** tab looks like this:

```
.....................................
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
```

The edit fields of the **InstallShield** tab are:

- **Setup file:** Name of the analysed setup file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from the setup filename. Most likely it has to be adapted.
- **Product Version:** the version of the software as detected from from the setup filename. It should contain only numbers and points and most likely has to be adapted .
- **Install Directory:** this is the name of the directory, where the software is supposed to be installed on the client. This is taken to patch the deinstall script and must be set manually
- **Required Space:** this value is assumed as being six times the size of the EXE file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **Setup File Size:** the size of the EXE file.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

Signature InstallShield: for detecting an Installshield setup file, the EXE is scanned for the marker:

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

Finding the Installshield signature without detecting the marker for embedded MSI, the setup is assumed to be of type **InstallShield**. If also the marker for an embedded MSI is found, the setup is assumed to be of type **InstallShield+MSI** (see below).

Installer Type InstallShield+MSI

Setups created with InstallShield often contain embedded MSI files. To extract the MSI file from the EXE file, a batch is invoked to start the setup and wait for a MSI file to be extracted. The batch will wait for about 10 seconds for a MSI file to appear and then kill the setup tasks. The extracted MSI will be analysed and the gathered information will be shown in the **InstallShield+MSI** tab and the **MSI** tab. It depends on the EXE file and the system conditions,

whether it extracts an MSI file or not. Eventually it does not extract any MSI for the operating system is not suitable or because ther is a GUI question waiting to be answered. If the automated extraction fails, the MSI might be unpacked manually and analysed as MSI. These entries can be transfered to the **InstallShield+MSI** tab.

In case of a successful InstallShield+MSI analysis the **Analyze** tab looks like this:

```
.....................................
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
```

If the automated extraction of the MSI succeeds, the detected valueswill be written to the **InstallShield+MSI** tab:

- **Setup file:** Name of the analysed setup file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from the setup filename. Most likely it has to be adapted.
- **Product Name:** this is the name of the software as detected from the MSI. This name can be changed, but most likely the detected name will fit.
- **Product Version:** the version of the software as detected from the MSI. Usually the detected value can be taken, but it should contain only numbers and points.

- **MSI Product Code:** the MSI product code as detected from the MSI. It will be used to patch the deinstall opsiscript. Caveat: a MSI product for 32bit and 64bit usually has a different product code and an Advanced Installer packet might contain different MSI files. In this case it depends on the system the analysis is running on, which MSI is extracted and analysed. Usually a 64bit MSI cannot be extracted on a 32bit system. So for packing a 64bit software, a 64 system should be used.
- **Required Space:** this value is not read from the MSI packet, but is assumed as being six times the size of the EXE file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **Setup File Size:** the size of the EXE file.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

Signature InstallShield+MSI: for the automated detection of an Installshield setup with embedded MSI the EXE file is scanned for the Installshield marker and the marker for an embedded MSI:

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

If both of them are found, the EXE file is supposed to be InstallShield+MSI.

Installer Type NSIS

NSIS setups are known to be small and fast. But it is not possible to extract further information from the packet, besides from detecting it as a NSIS setup. All further information to be shown is derived from the name of the setup. So the information must be sampled manually (for instance by installing the setup).

The entries in the **Analyze** tab might look like this:

```
.................................
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
```

The edit fields of the **NSIS** tab are:

- **Setup file:** Name of the analysed setup file
- **opsi Product ID:** this is the name of the new opsi packet to be created. It is derived from the product name below, by substituting spaces and other invalid characters with -. The suggested opsi Product ID can be changed.
- **Product Name:** this is the name of the software as detected from the setup filename. Most likely it has to be adapted.

- **Product Version:** the version of the software as detected from from the setup filename. It should contain only numbers and points and most likely has to be adapted.
- **Install Directory:** this is the name of the directory, where the software is supposed to be installed on the client. This is taken to patch the deinstall script and must be set manually.
- **Required Space:** this value is assumed as being six times the size of the EXE file. Before installting the software on the client, the setup.opsiscript checks, whether there is enough disc space.
- **Setup File Size:** the size of the EXE file.
- **Description:** The product name is set to this edit space and can be supplemented with additional information to be shown in the product description of the opsi product.
- **License required:** When this selection mark is checked, the opsiscript will be patched with *license=true* to indicate that a license is required.

Signatur NSIS: for detecting a NSIS Setup, the EXE file will be scanned for the marker:

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

Creating a new opsi Packet

After analyzing the setup and the edit fields of the corresponding tab being filled, clicking the button *Create opsi Packet* (on the lower right) generates the files for a new opsi product.

Attention:

- the *Packet BaseDir* must be a writable share to the opsi Workbench.
- for safety reasons a new opsi packet only can be created if it doesn't exist. To rewrite the product, first delete the product directory from the opsi workbench.

Left of the *Create opsi Packet* button are three available options to specify the operation of the button:

- **create opsi packet:** the product directory and the corresponding files will be generated and patched on the opsi workbench. The packet will not be packed nor installed.
- **create opsi packet and start interactive PacketBuilder:** as above the files will be created and patched and then the *opsi Packet Builder* will be started. The product data will be passed to the Packet Builder and the packet can be packed and installed in interactive mode. After completion the *opsi Packet Builder* should be closed to be startable again for the next data transfer.
- **create and auto -build -install -quiet:** as above the directory and the files will be generated and patched and for further actions the *opsi Packet Builder* will be started. The opsi product data will be passed to the packet Builder and the packet will be built and installed, if the corresponding options are checked. If e.g. the packet just should be packed but not installed, the checkmark at **install** can be removed. Additionally the checkmark **quiet** can be set to execute the *opsi Packet Builders* without showing its user interface. After completion the *opsi Packet Builder* should be closed again.

For installation, configuration and instructions for the Community Project *opsi Packet Builder* see https://forum.opsi.org/viewforum.php?f=22