SamLogic

# User's Guide

# SamLogic Visual Installer 2020

# SamLogic Visual Installer 2020

## User's Guide

---

***by SamLogic***

*This user's guide describes how to use Visual Installer 2020. It contains also reference chapters that describe the dialog boxes and main tabs in the program.*

# User's Guide - SamLogic Visual Installer 2020



**Contact Information**

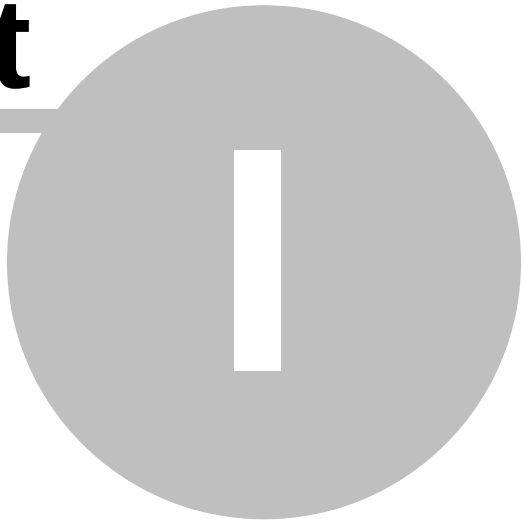
SamLogic
Box 102
135 23 TYRESO
SWEDEN

Tel: +46 8 531 83 900
Fax: +46 8 531 88 403

E-mail: info@samlogic.com
Internet: www.samlogic.net

# Contents

# Part



1 Contents

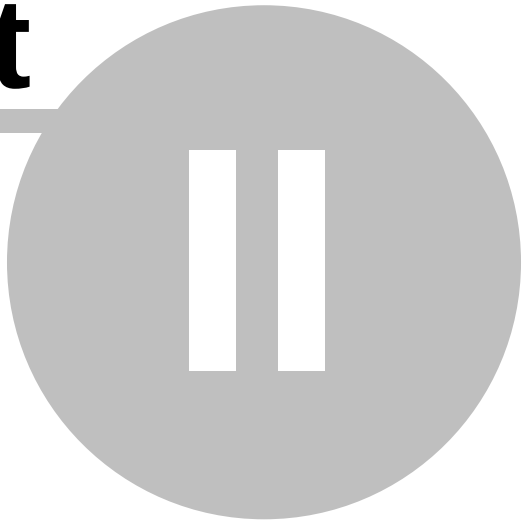
# SamLogic Visual Installer 2020

**SamLogic Visual Installer 2020** is a powerful setup tool that can be used to create setup programs for CD, DVD, USB flash drives and the Internet. No programming knowledge is required - you can create your installation program visually - but if you want to fine tune your setup programs, a script language is available.

There are many built-in and ready-to-use setup dialog boxes included in the Visual Installer installation software, and you can easily select those you want to use for your setup programs. You can add images and text to these dialog boxes in an easy way. Any language can be used. Stylish and impressive setup screens that are shown in the background during the installation can be created, but an installation without any setup screen can also be created.

**SamLogic Visual Installer 2020** supports all version of Windows from Windows 98 to Windows 10. Both 32 and 64 bit Windows are fully supported.

# Part



# Overview

People demand a lot from setup programs nowadays. Copying files to users' hard disks is the easy part. Besides that you must often update Windows Registry, add shortcuts to menus and desktops, register binary files, show graphics and text during the setup process, verify license keys, handle updates, install databases, handle different operating systems etc. With Visual Installer you can do all this, and best of all, no programming is needed!

Visual Installer can install files that are distributed on CDs, DVDs, USB flash drives and via the Internet. Self-extracting setup files can be created, so you only need to distribute one file.

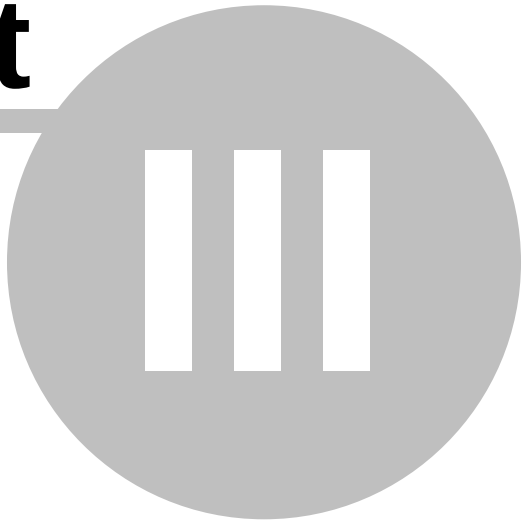
**Visual Installer 2020 - Features**

**The Tabs in Visual Installer**

For advanced users:

**Script commands in Visual Installer / Professional**

# Part



# Visual Installer 2020 - Features

Whether you need to create a simple or create an advanced setup program, you can do it quickly with **SamLogic Visual Installer 2020**. The tool is very flexible and versatile. Below are some of the features in Visual Installer 2020 listed:

## Some features:

- Can create installations for CDs, DVDs, USB flash drives and the Internet.
- Can create installations for all versions of Windows from Windows 98 to Windows 10.
- Supports both 32-bit and 64-bit Windows.
- Can create a self-extracting installer for the Internet (only one file needs to be distributed).
- Can create localized installations (you can specify your own text in any language).
- Supports silent installations (installations with no user interaction).
- Supports digital signing of created setup programs.
- Can uninstall an installed application.
- Files can be grouped into file components.
- Can add shortcuts (icons) to the Start menu, Program menu, Startup Menu and directly on the desktop.
- Can handle updates in an intelligent manner.
- Installations can be protected with license keys or passwords.
- Can import Visual Basic and Visual C# project files.
- Can handle DAO, ADO and RDO.
- Can handle AutoCAD profiles and install additional tools for Microsoft Excel and PowerPoint.
- Can prevent installations on non-supported operating systems.
- Can display images during an installation.
- Can play sounds and music during an installation (supports MP3, WAVE, and MIDI).
- Can add and delete keys and values to / from the Registry.
- Can add and remove values to / from INI files.
- Can register files, such as DLLs, ActiveX components and fonts.
- Can replace active/locked files (files that are used by a program that is currently running).
- Can replace write-protected files.
- Can launch applications or view documents after an installation.
- A large number of ready-to-use setup dialog boxes are included.
- Supports the Autorun feature in Windows (automatic start of programs on a CD or DVD).
- A large number of variables that can be used in different contexts are included.
- An effective ZIP-based compression is included.
- Installations can be simulated before they are created.

## Visual Installer 2020 / Professional includes:

- A script language that can be used for advanced installations.
- A project manager that keeps track of your created project files.
- Functionality to create multiple levels of menus with shortcuts in Windows.
- Functionality to install 64-bit applications.
- Functionality to send messages to Twitter.
- A special tool, SamLogic Selector, which can create system dependent program starts.

# Part

# Getting Started with Visual Installer

There are Getting Started pages available on our website that helps you getting started with Visual Installer. The Getting Started information is also available as a video. Click on the link below to read more (a web page will be opened in your web browser):

**Getting Started With Visual Installer (on the web)**

# Part V

5 The Tabs in Visual Installer

# The Tabs in Visual Installer

This chapter contains information about the tabs in the Visual Installer editor.

**Tab 1 - File list**
In this tab you add files to your setup projects and specifies destination folders for the files. You can also specify which files that should have shortcuts and which files that must be registered in the system. You can also set different installation options for the files.

**Tab 2 - Design**
In this tab you create a setup screen that will be shown during the setup. You can add text and graphics to the setup screen. It is also possible to add sound to the setup package via this tab.

**Tab 3 - Dialog boxes**
In this tab you choose which dialog boxes to show during the setup process. By pressing the "..." button to the right of a check box you can set some properties and specify texts for every dialog box.

**Tab 4 - Registry**
In this tab you specify keys and values that should be added to the Registry during the setup process.

**Tab 5 - INI files**
In this tab you specify sections and values that should be added to INI files during the setup process.

**Tab 6 - Components**
If you want to group your files in components, it can be done in this tab.

5.1 Tab 1 - File list

# Tab 1 - File list

In this tab you add files to your setup projects and specifies destination folders for the files. You can also specify which files that should have shortcuts and which files that must be registered in the system. You can also set different installation options for the files.

**Project name**
Here you enter the name of the project.

**Main folder**
Here you enter the main destination folder for your project. The folder path will be stored in the global variable %DESTDIR and can be used in many places in Visual Installer, for example in the file list.

To add files to the file list, choose the menu option **List - Add files**. if you want to add a folder tree, choose the menu option **List - Add tree**.

**More information about main folders**

**About copy settings (the 'Inst' column)**

**Variables**

5.2 Tab 2 - Design

# Tab 2 - Design

In this tab you create a setup screen that will be shown during the setup. You can add text and graphics to the setup screen. It is also possible to add sound to the setup package via this tab.

You can preview a setup screen by pressing the **Preview** button in the toolbar. You can return from the preview mode by pressing the **ESC** key on your keyboard.

5.3 Tab 3 - Dialog boxes

# Tab 3 - Dialog boxes

In this tab you choose which dialog boxes to show during the setup process. By pressing the "..." button to the right of a check box you can set some properties and specify texts for every dialog box.

5.4 Tab 4 - Registry

# Tab 4 - Registry

In this tab you specify keys and values that should be added to or deleted from the Registry during the setup process. You can add text strings, DWORD values, binary data and default values to the Registry. All Visual Installer variables can be used in this tab.

**Add**
Opens a dialog box where you can specify a key and value that should be added to the Registry. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here is an example of how a line can look like:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: AppDir="%DESTDIR"

**Delete**
Opens a dialog box where you can specify a key and value that should be deleted from the Registry during the setup process. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here are two examples of how a line can look like:

DELETE=HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\9.0
DELETE=HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\9.0 :: AppDir

**At uninstall**
Opens a dialog box where you can specify a key and value that should be deleted from the Registry during an uninstall process. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here are two examples of how a line can look like:

UNINSTALL=HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5
UNINSTALL=HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: AppDir

**Variables**
If you click on this button a list with variables will be shown. if you double click on a variable, it will be inserted in the cursor's position in the textarea of the tab.

**More information about the Registry**

Variables

## 5.5 Tab 5 - INI files

# Tab 5 - INI files

In this tab you specify sections and values that should be added to INI files during the setup process.

**New file**
Opens a dialog box where you can specify a filename of an INI file. You can use Visual Installer variables to create a file path. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here is an example of how a line can look like:

```
-- VISUINST.INI --
```

**Add section**
Opens a dialog box where you can specify a section to add to the INI file. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here is an example of how a line can look like:

```
[Directories]
```

**Add value**
Opens a dialog box where you can specify a value to add to the INI file. When you press the **OK** button in the dialog box, a line with information will be placed in the textarea of the tab. Here is an example of how a line can look like:

```
AppDir=%DESTDIR
```

**Variables**
If you click on this button a list with variables will be shown. if you double click on a variable, it will be inserted in the cursor's position in the textarea of the tab.

More information about INI files

Variables

## 5.6 Tab 6 - Components

# Tab 6 - Components

If you want to group your files in components, it can be done in this tab.

# Part VI

6 Dialog Boxes

# Dialog Boxes

There are more than 80 dialog boxes in Visual Installer where you can set various options for your setup packages. In this chapter we will describe these dialog boxes.

The easiest way to get help for a specific dialog box is to press the **F1** key when the dialog box is open. Then you will get help information for that specific dialog box immediately.

## 6.1 Dialog Box - Add empty folder

# Dialog Box - Add empty folder

Adds an empty folder to the file list.

## 6.2 Dialog Box - Add new INI file

# Dialog Box - Add new INI file

Here you specify a new INI file to add to the INI file editor.

## 6.3 Dialog Box - Add Registry key or value

# Dialog Box - Add Registry key or value

In this dialog box you specify a key or value that should be added to the Registry during the setup process.

When you specify a sub key, it should start with a backslash. Example: "**\SOFTWARE\SamLogic\Visual Installer\10.5**". You can use all of Visual Installer's variables with the values that you enter.

**Variables**

## 6.4 Dialog Box - Add section

# Dialog Box - Add section

Here you specify a name of a section to add to the INI file editor.

6.5 Dialog Box - Add value

## Dialog Box - Add value

In this dialog box you can specify a value that should be added to an INI file. You can use all of Visual Installer's variables with the values that you enter.

**Variables**

6.6 Dialog Box - Background

## Dialog Box - Background

Specifies a background for your setup screen.

6.7 Dialog Box - Background sound/music

## Dialog Box - Background sound/music

Via this dialog box you can choose background music or sound effects for your installation package. The music / sound effects will be played during the setup process.

6.8 Dialog Box - Block list

## Dialog Box - Block list

If you want to block some license keys from being accepted you can create a block list in this dialog box. If the user enters a license key that exists in the block list, an error message is shown. You can enter a list with license keys to block in the **Block list** text box. You must enter one license key per line. To add a line feed, press **Shift + Enter**.

You can specify which text to display in the error message in the **Information message** text box. The title of the information message box is read from the current language file.

The block list is useful if you have created and distributed prime number based license keys, and need to block some license keys for some reason.

6.9 Dialog Box - Change contents for variables

## Dialog Box - Change contents for variables

Via this dialog box you can change the contents of some of the variables in Visual Installer.

**Variables**

6.10 Dialog Box - Change source path for file

## Dialog Box - Change source path for file

Via this dialog box you can change the source file path for a file.

6.11 Dialog Box - Comment

## Dialog Box - Comment

In this dialog box you can enter a comment for the selected file in the file list. The comment vill be saved with the project file.

6.12 Dialog Box - Component - Advanced options

## Dialog Box - Component - Advanced options

This dialog box contains advanced options for file groups / components.

6.13 Dialog Box - Component - More options and description

## Dialog Box - Component - More options and description

In this dialog box you can enter a description for a component and set some options for the component.

6.14 Dialog Box - Component number

## Dialog Box - Component number

You can enter a new component number for the selected files in the file list here.

6.15 Dialog Box - Compression options

## Dialog Box - Compression options

In this dialog box you can set some compression options for your setup package.

6.16 Dialog Box - Copy options for file

# Dialog Box - Copy options for file

In this dialog box you can set copy options for one or more files that are selected in the file list. Below are the available options in this dialog box described:

**Check date and time**
Checks a file's date and time before it is replaced. The file will only be installed if it is newer.

**Check version**
Checks a file's version number before it is replaced. The file will only be installed if it have a higher version number.

**Never replace existing file**
If a file with the same filename already exists in a specified folder, the file will not be installed.

**Register file**
Check this option if a file must be registered during the installation. Some components and DLL files require this.

**Turn on write protection**
Write protection will be turned on for the file after it has been installed in the hard disk.

**Warn if file already exists**
An information message box will be shown if the file already exists on the destination drive. The end-user can decide if the file should be replaced or not.

**Make a backup if file already exists**
If you check this option, a backup will be made of the file if it already exists in the destination folder. The backup file will get the filename extension ".BAK" and it will be placed in the same folder as the original file.

**Replace also write protected file**
If this option is checked, a file will be replaced also when it is write protected.

**Handle the file if it is locked and in use**
Sometimes a file can be in use during the setup process, and then it can not be replaced at that moment. If you check this option, the file will be marked in the system to be replaced next time the computer restarts (during the boot process).

**Shared file**
If a file is shared with other programs, you can check this option. This will prevent the file to be uninstalled by mistake.

**Do not copy if same version**
If this option is checked a file will not be copied if the version number of the source and destination files are equal.

**Never uninstall this file**
Prevent uninstallation of the file. If your project is uninstalled, the file will remain on the user's hard disk.

**Why register a file?**

6.17 Dialog Box - Create setup package

# Dialog Box - Create setup package

From this dialog box you an create a setup package that will be distributed via a CD, DVD, USB flash drive or the Internet. Below is the contents of this dialog box explained:

**Type**
Here you specify the type of installation.

**Operating system**
Specifies which operating systems that should be supported by the setup program.

**Create**
Here you specify a creation folder.

**FTP Server**
If you want to upload your setup package to a web site directly after it has been created you can turn on that option here. This option is only available if you have chosen the **Internet** installation type. You must also have specified FTP settings in the **FTP settings** dialog box.

### 6.18 Dialog Box - Delete Registry key or value

## Dialog Box - Delete Registry key or value

In this dialog box you specify a key or value in the Registry that should be deleted during the installation of your program. Be sure to only specify such a key or value that no other programs are dependent of because otherwise they might stop working.

When you press **OK** in this dialog box, a line will be inserted in the text editor in the **Registry** tab that starts with "**DELETE=**".

### 6.19 Dialog Box - Destination folder

## Dialog Box - Destination folder

In this dialog box you choose a destination folder for the files that you have chosed.

You can include a variable in the destination folder, for example **%DESTDIR\Documents**. You can view a list with available variables by pressing the "..." button to the right of the **Folder** text box.

**Variables**

### 6.20 Dialog Box - Destination folder (Setup dialog box)

## Dialog Box - Destination folder

In this dialog box you can set some options for the setup dialog box that asks for a destination folder during the installation. If you want to preview the dialog box, press the **Preview** button.

### 6.21 Dialog Box - Destination folders

## Dialog Box - Destination folders

Shows a list with destination folders used by the files in the file list. These folders will be created in the end-users hard disk.

6.22 Dialog Box - Drive (setup)

## Dialog Box - Drive

In this dialog box you can set some options for the setup dialog box that asks for a drive during the installation. If you want to preview the dialog box, press the **Preview** button.

6.23 Dialog Box - Editor Options

## Dialog Box - Editor Options

Contains options for the Visual Installer editor.

6.24 Dialog Box - Execute script commands

## Dialog Box - Execute script commands

If you want to execute script command during the setup process you can enter script commands in this dialog box. You can choose between executing script commands before the installation or after the installation.

**Script commands - Overview**

**Script commands - Reference**

**Script examples**

6.25 Dialog Box - File information

## Dialog Box - File information

Shows information about a file.

6.26 Dialog Box - Folder tree

## Dialog Box - Folder tree

In this dialog box you can choose how the files will be added to the file list. Choose on of the options in the dialog box and press the **Add files** button to add them to the file list.

More options is available if you click on the **More options** button.

### 6.27 Dialog Box - FTP settings

## Dialog Box - FTP settings

If you want to upload your installation automatically to a web site you can specify the necessary settings below. When you specify a folder path at **Server folder**, the path must start with a slash (/). Example:

**/www/download**

### 6.28 Dialog Box - General dialog boxes (for Setup)

## Dialog Box - General dialog boxes

There are some general setup dialog boxes available in Visual Installer that can be used to show different kind of information. Here you can enter title and information text for the setup dialog box that you have chosen. If you want to preview the dialog box, press the **Preview** button. If you need to enter a linefeed in the dialog box text, press **Shift + Enter**.

It is possible to include a clickable e-mail address and web address in the general setup dialog boxes. You can read more in this section:

**Using an e-mail address and web address in a dialog box**

### 6.29 Dialog Box - Icon

## Dialog Box - Icon

Via the list in this dialog box you can choose an icon to the **Support Information** dialog box. The list contains all files in Visual Installer's file list.

### 6.30 Dialog Box - If .NET Framework is missing

## Dialog Box - If .NET Framework is missing

If the Microsoft .NET Framework version that your program requires is not installed in the end-user's computer, you can let the user download and install it directly from Visual Installer. In this dialog box you can specify a URL to a download page from where the user can download .NET Framework. Or, you can specify a file path to an installation of .NET Framework (for example on a CD/DVD or USB flash drive) if you are going to distribute .NET Framework with your software.

**Action**
Here you specify if the end-user should download .NET Framework from the Internet (for example from Microsoft's download page) or if it follows your CD/DVD or USB flash drive, and you want to start the installation from Visual Installer.

**Download page / File path**
Here you specify a URL to a download page or a file path to an installation package that installs Microsoft .NET Framework. An URL must always begin with "http://" or "https://", and a file path must always be entered without a drive letter (for example: "\NET4\dotNetFx40_Full_setup.exe").

**Language**
If you are going to use a download page on Microsoft's website, you can specify which language to use for the texts on the web page via this combobox.

**Standard**
Resets the URL in the **Download page** text box to the built-in standard URL. If you have entered your own URL, but change your mind and want to use the default URL provided by Visual Installer, you can click on this button.

### 6.31 Dialog Box - Import Visual Basic .NET project

## Dialog Box - Import Visual Basic .NET project

Visual Installer can read a Visual Basic .NET project file and import the files necessary to create a fully functioning setup package.

If your program is dependent of third-party DLL files you must add them manually to the Visual Installer project.

### 6.32 Dialog Box - Import Visual Basic 6.0 project

## Dialog Box - Import Visual Basic 6.0 project

Visual Installer can read a Visual Basic 6.0 project file and import the files necessary to create a fully functioning setup package.

If your program is dependent of third-party DLL files you must add them manually to the Visual Installer project.

### 6.33 Dialog Box - Import Visual C# project

## Dialog Box - Import Visual C# project

Visual Installer can read a Visual C# project file and import the files necessary to create a fully functioning setup package.

If your program is dependent of third-party DLL files you must add them manually to the Visual Installer project.

6.34 Dialog Box - Language

## Dialog Box - Language

Via this dialog box you choose a language file for your installation. A language file is a text file that contains general texts and messages that will be shown during the setup process. Click on the link below to get more information about language files.

**More information about language files**

6.35 Dialog Box - License information (RTF file) (Setup dialog box)

## Dialog Box - License information (RTF file)

Via this dialog box you can select a RTF file with a end-user license agreement (EULA) text that the user must accept before he/she continues the installation. The size of the RTF file can be unlimited.

6.36 Dialog Box - License information (Setup dialog box)

## Dialog Box - License information

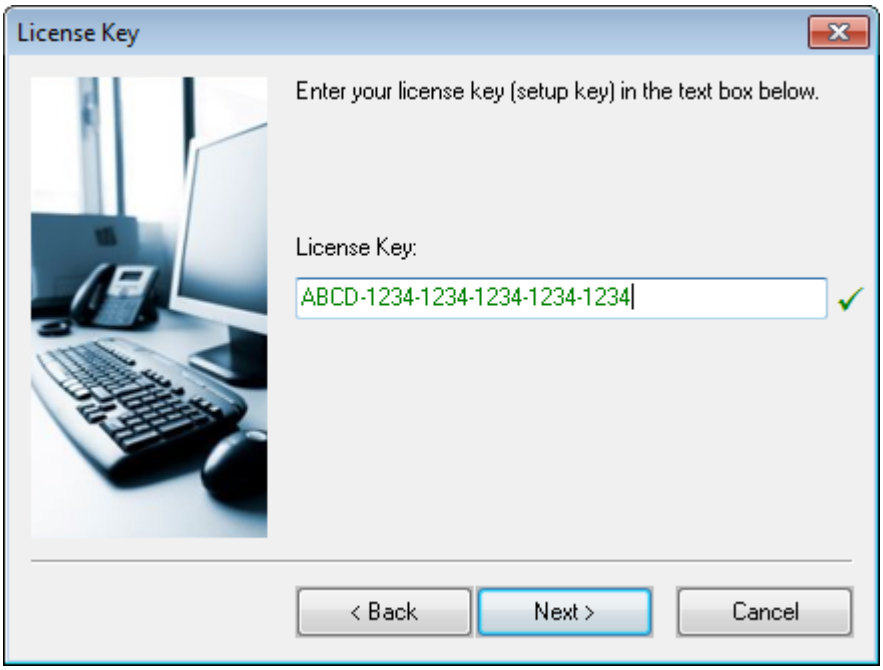
In this dialog box you can enter (or paste) an end-user license agreement (EULA) text that the user must accept before he/she continues the installation. You can insert up to 16 KB text in the **License text** text box.

6.37 Dialog Box - License key - More options

## Dialog Box - License key - More options

**Show all entered letters in the 'License Key' input box in uppercase**
If you turn on this option all letters that are entered in the **License Key** input box will be shown with uppercase letters.

**Show visually when correct license key has been entered**
If you turn on this option the text color in the input box where the license key is entered will be set to green when the user has entered correct license key. A green check mark will also be shown to the right of the input box. The picture below shows how it can look like:




**Show this license key as default when dialog box opens**
Here you can enter a default license key that is shown for the user in the **License Key** dialog box when the dialog box is opened for the first time.

**The license key must start with this prefix**
If the license key that the user enters must start with a specific prefix (for example with "**2001-**") you can enter the prefix here. This is especially useful when prime number based license keys are used because you can block all other prefixes from being accepted except the one that you enter in this text box.

**Block List**
If you press this button a new dialog box is shown, where you can block some license keys from being accepted. Read more here.

### 6.38 Dialog Box - License key (Setup dialog box)

## Dialog Box - License key

In this dialog box you can enter a license key that the user must enter before he/she can continue the setup. Below is the contents of the dialog box explained:

**Dialog text**
The text that is displayed in the dialog box

**License key**
Correct license key. The setup can not continue until the user has entered the correct license key. A license key can contain the letters **A-Z** and **a-z**, and the number characters **0-9**. Also the following special characters are allowed: "**+ - * / & . = # ¤ ?**". Below are some examples of valid license keys:

9682-1002-1639
SAM-VI-1028
ORANGE

If you want to specify more than one license key it is also possible. You must then enter a semicolon between the license keys. Example:

ORANGE;APPLE;BANANA

You can specify up to 10 license keys.

**Prime number based license keys & More options**
Opens dialog boxes with more settings. These two buttons are only available in the Professional version of Visual Installer.

**Preview**
Press this button to prevew the dialog box.

**More information about license keys**

6.39 Dialog Box - License keys - Prime number based

# Dialog Box - License keys - Prime number based

Via this dialog box you can create prime number based license keys. You can specify a fixed prefix (4 digits) that is placed in the beginning of the license key. Thereafter some random prime numbers generated by Visual Installer, that is encoded in the license key. In the end of the license key, a checksum is placed. The checksum assures that nobody enter characters just randomly after the prefix, when Visual Installer asks for a license key.

Below is the contents of the dialog box explained:

**Activate use of prime number based license keys**
If you check this option, Visual Installer will handle prime number based license keys.

**Accept also static license keys**
If you check this option, you can also use static license keys simultaneously.

**Insert hyphen automatically**
If you check this option, hyphens will be inserted automatically when a user enters a prime number based license key.

**Prefix / ID**
Here you specify a prefix that will be placed in the beginning of the license key. The prefix must consists of exactly 4 digits. Example: "1101".

**Create keys**
Press this button to create keys.

**Clear**
Clears the settings in the dialog box.

**Open license key file**
Opens a text file with license keys.

**Save all**
Saves all license keys (both new ones and old ones) to a text file.

**Save new license keys**
Saves only new license keys to a text file.

**More information about license keys**

6.40 Dialog Box - Logotype

## Dialog Box - Logotype

In this dialog box you can specify a logotype that should be visible in the setup screen.

6.41 Dialog Box - Main Folder Variable Suggestions

## Dialog Box - Main Folder Variable Suggestions

In this dialog box you can choose a variable for the main folder. You can combine this variable with a project name or similar to create a main folder (main setup directory) for the installation.

**More information about main folders**

**Variables**

6.42 Dialog Box - Microsoft .NET Framework - Check if installed

## Dialog Box - Microsoft .NET Framework - Check if installed

You can make Visual Installer check that a specific version of Microsoft .NET Framework is installed in the user's computer before the actual installation starts. In this dialog box you can specify a minimum version number for .NET Framework that your software requires, and tell Visual Installer how to react if the installed version of .NET Framework is too old.

6.43 Dialog Box - Microsoft Data Access - Details

## Dialog Box - Microsoft Data Access - Details

In this dialog box you can set some options for installation of Microsoft Data Access components.

6.44 Dialog Box - New language file

## Dialog Box - New language file

In this dialog box you specify the filename for a new language file. A language file should always have the filename extension "**.LNG**".

6.45 Dialog Box - New project

## Dialog Box - New project

In this dialog box you choose project type and sets some initial project settings. All settings can be changed later.

6.46 Dialog Box - Operating system

## Dialog Box - Operating system

In this dialog box you specify a minimum Windows version for a file. If the file can be installed in all Windows version the option **All Windows versions** should be chosen. The **All Windows versions** option is also the default option if no other option has been selected.

6.47 Dialog Box - Options for shortcut

## Dialog Box - Options for shortcut

Contains additional options for shortcuts.

6.48 Dialog Box - Overview - Objects

## Dialog Box - Overview - Objects

Shows an overview of all objects in a setup screen.

6.49 Dialog Box - Password (Setup dialog box)

## Dialog Box - Password

In this dialog box you can specify a password that the user must enter to be able to make an installation. A password can contain the letters **A-Z** and **a-z**, and the number characters **0-9**. Also the following special characters are allowed: "**+ - * / & . = # ¤ ?**".

The password check is case sensitive, so the passwords "**orange**" and "**ORANGE**" are considered as two different passwords.

6.50 Dialog Box - Percent gauge

## Dialog Box - Percent gauge

In this dialog box you choose the percent gauge's positon on the setup screen.

The percent gauge will not be shown when you preview the setup screen because the percent gauge is a dialog box, not a visual control. But the percent gauge will be shown when you do a test installation via **File - Test**.

### 6.51 Dialog Box - Pictures

## Dialog Box - Pictures

Via this dialog box you can add pictures to the setup screen. After that a picture has been placed on the setup screen, you can drag the picture to a position that you prefer.

### 6.52 Dialog Box - Port number (FTP)

## Dialog Box - Port number (FTP)

In this dialog box you can change the port number for the FTP communication. The standard number for the port is 21 and normally you do not need to change this number.

### 6.53 Dialog Box - Preview Registry keys and values

## Dialog Box - Preview Registry keys and values

Shows how the Registry keys that you have specified in the **Registry** tab will affect the Windows Registry.

### 6.54 Dialog Box - Print file list

## Dialog Box - Print file list

Prints the file list.

### 6.55 Dialog Box - Program group / menu

## Dialog Box - Program group / menu

In the **Name of program group / menu** text box you can enter a name for a program group / program menu where all shortcuts for the files in your setup project will be placed. At **Users** you can specify if the program group / program menu should be created to only the current user or to all users in a computer.

The text that you enter in the **Name of program group / menu** text box will be stored in the %PROGRAMGROUP variable in Visual Installer.

6.56 Dialog Box - Project manager

## Dialog Box - Project manager

Here you can organize your Visual Installer project files that you have created.

6.57 Dialog Box - Project manager (Information)

## Dialog Box - Project manager (Information)

Shows detailed information about a project file.

6.58 Dialog Box - Register files

## Dialog Box - Register files

If you want to register files in different order than in the file list you can list them here in the desired order. Line 1 will be registered first, then line 2 etc. You must enter full paths to the files. The paths can contain Visual Installer variables. Below are some examples of valid paths:

%SYSDIR\ALPHA.DLL
%SYSDIR\GAMMA.DLL

If you want to get all files in the file list that are marked to be registered, you can press the **Get all register marked files from the file list** button. If you want to get all DLL files in the file list, press the **Get all DLL files from file list** button.

6.59 Dialog Box - Register font

## Dialog Box - Register font

If a font file needs to be registered in the system after installation you can specify it here. Below is the contents of this dialog box explained:

**Font name**
The name of the font, for example: "Bremen Bold BT (TrueType)".

**Register this font after the installation**
If this option is checked, the font will be registered in the system.

**Never replace an existing font file**
If this option is checked, an existing font file will never be replaced. It is recommended to have this option turned on if you distribute common used font files, because the computer must be restarted if the font file is in use. If you know that you are distributing an updated font file, or a font file that you have created at your own, it is recommended to have this option turned off.

6.60 Dialog Box - Register .NET assembly

# Dialog Box - Register .NET assembly

If you want to register a .NET assembly instead of a traditional DLL you can specify it here. Visual Installer will run the assembly registration tool **RegAsm.exe** to register the .NET assembly.

**Register this .NET assembly during the installation**
If this option is selected, the .NET assembly will be registered during the setup.

**.NET Version**
Specifies which .NET version the .NET assembly is compiled to (or will be compiled to). It is important to specify the correct version because different versions of the **RegAsm.exe** tool will be run depending on the version of .NET.

**Bitness**
There exists both a 32 bit version and a 64 bit version of the assembly registration tool **RegAsm.exe**. The bitness specified in the **32/64 bit** tab in the **Setup options** dialog box determines which version to use for the registration, if the **Same as the setup project** option is selected in this dialog box. If the **Both 32 bit and 64 bit** option is selected, both the 32 bit version and a 64 bit version of **RegAsm.exe** will be run.

**Parameters**
Three common used options that can be passed to **RegAsm.exe** are **/codebase**, **/registered** and **/tlb**. Here you can specify if you need to pass these paramerers to the assembly registration tool.

6.61 Dialog Box - Registration (Setup dialog box)

# Dialog Box - Registration

Sets some properties for the **Registration** dialog box. The **Registration** dialog box is a dialog box that asks for name, company name and other information during the installation.

6.62 Dialog Box - Remove current picture

# Dialog Box - Remove current picture and use this picture instead

Via this dialog box you can remove your own picture that you specified before. When you remove your own picture, you can choose if the standard picture of the installation or a classic uninstallation picture will be used instead.

6.63 Dialog Box - Remove Registry key or value at uninstall

# Dialog Box - Remove Registry key or value at uninstall

In this dialog box you specify a key or value in the Registry that should be deleted when your program is uninstalled. Be sure to only specify such a key or value that no other programs are dependent of because otherwise they might stop working.

When you press **OK** in this dialog box, a line will be inserted in the text editor in the **Registry** tab that starts with "**UNINSTALL=**".

6.64 Dialog Box - Replace

## Dialog Box - Replace

Searches for a text in the file list and replaces it with a text that you have specified.

6.65 Dialog Box - RTF box

## Dialog Box - RTF box

It is possible to place a text box with formatted RTF text on the setup screen. Via this dialog box you can create such a text box. After that the text box has been placed on the setup screen, you can drag the box to a position that you prefer.

6.66 Dialog Box - Run program after installation

## Dialog Box - Run program after installation

You can run one or two programs after the installation if you specify file paths to the programs in this dialog box. You can also pass parameters to the programs. Below is the contents of the dialog box explained:

**Filename**
A file path to the program to run. If the program that you want to run is distributed with your setup package you can use the variable %DESTDIR to create a file path to the program, e.g: "**%DESTDIR\Apps\MyApp.exe**".

You can also specify a file path to an MSI file here, e.g: "**%DESTDIR\Setup\MySetup.msi**".

**Parameters**
If you want to pass parameters to the program, you can enter them here.

**Do not run if the program is already running**
If this option is checked, the program will not be started if it is already running.

**Show check box in the 'Installation completed' dialog box**
If this option is selected, a check box will be shown in the **Installation completed** setup dialog box. The user can choose whether he/she want to run the program(s) or not by checking or unchecking this check box.

**Check box text**
Specifies a text for the check box in the **Installation completed** setup dialog box. See above for more information.

**Variables**

6.67 Dialog Box - Run as administrator

## Dialog Box - Run as administrator

If you select the **Run program as administrator** option in this dialog box, the program will be run with administrator privileges.

6.68 Dialog Box - Select component

## Dialog Box - Select component

In this dialog box you can choose which component (file group) to use for the selected files. If you don't want to place the files in a component, you can select the "**No component**" option.

6.69 Dialog Box - Select folder

## Dialog Box - Select folder

You can select a folder via this dialog box.

6.70 Dialog Box - Select variable or previous destination folder

## Dialog Box - Select variable or previous destination folder

Via this dialog box you can select a variable or a previous used destination folder.

6.71 Dialog Box - Send Message To Twitter

## Dialog Box - Send Message To Twitter

You can post a message to your Twitter account directly from Visual Installer. Enter your message in the text box and press the **Send Message** button to send the message to Twitter.

6.72 Dialog Box - Settings

## Dialog Box - Settings

The option setup dialog box that is shown for the end-user during an installation can either contain check boxes or radio buttons. In this dialog box you can choose which to use.

### 6.73 Dialog Box - Setup options - .NET

## Dialog Box - Setup options

If your software requires a specific version of .NET Framework you can specify the version number here. You can also choose what Visual Installer should do if the required version of .NET Framework is missing.

### 6.74 Dialog Box - Setup options - 32/64 bit

## Dialog Box - Setup options

In this tab you specify if the program that you will install is a 32 or 64 bit program. If you install other types of files, e.g. documents, or do not know the bitness of your program you should select 32 bit.

32 bit is the default option; most programs are still 32 bit. A 32 bit program can be run in a 64 bit Windows without problem, but it is important that the files are installed in 32 bit folders (and not in 64 bit folders). Visual Installer will do this for you if you tell Visual Installer that your program is a 32 bit program.

More information about 32 bit and 64 bit folders in Windows is available in this article in our web site.

### 6.75 Dialog Box - Setup options - Code signing

## Dialog Box - Setup options

If you want to code sign your installation package (add a digital certificate to your installation package) you can set the necessary options in this tab. Only a self-extracting installation package can be code signed.

To enable the code signing function you must first check the **Code sign self-extracting installation package** option in this tab. Then you need to inform Visual Installer if your digital certificate is located on a USB token (USB stick) or on a file. If your digital certificate is stored on a USB token you must select the **My digital certificate is located on a USB token** option. If your digital certificate is not located on a USB token, but was downloaded from a website, keep this option unselected.

The actual code signing is made by an external code signing tool, and at **Code signer program** you must specify a file path to such a tool. Normally this file path is specified automatically, but if not, you can enter the file path manually. We recommend you to use the latest version of **SIGNTOOL**.

If your digital certificate is not located on a USB token but was downloaded from a website, you must specify a file path to a PFX file, or to a SPC and PVK file. You can do this via the **PFX file** sub tab or the **SPC/PVK files** sub tab in the **Code Signing** tab. If your digital certificate is located on a USB token, you don't need to specify these file paths.

A hash algorithm (SHA-1 or SHA-2) must also be specified. You can specify the hash algorithm in the **Options** sub tab in the **Code Signing** tab. More information about hash algorithms is available on this page.

You must also choose a time stamp server. This is made via the **Time stamp** combo box.

There is always a password connected to a digital certificate (for security reasons), and you can specify such a password in this dialog box tab if your digital certificate is *not* located on a USB stick. If your digital certificate is located on a USB stick, the software that handles the USB token and the digital certificate will ask for the password.

### Tip

We recommend you to read this tip page on our web site to learn how to code sign setup packages when using Visual Installer. It shows step by step how to code sign a setup package:

**Tip: How to code sign a setup package**

### 6.76 Dialog Box - Setup options - General

## Dialog Box - Setup options

In this tab you can set some general options for your setup package; for example add uninstall support or specify a name of the setup program.

### 6.77 Dialog Box - Setup options - Internet

## Dialog Box - Setup options

If your setup package will be distributed via Internet you can set some options here. You can for example specify the filename of the self-extracting setup package and tell Visual Installer if the setup package should be automatically uploaded to a web server after creation.

### 6.78 Dialog Box - Setup options - Operating systems

## Dialog Box - Setup options

In this tab you can specify in which operating systems the setup program is allowed to run in.

### 6.79 Dialog Box - Setup options - Various

## Dialog Box - Setup options

Contains various options.

**Variables**

### 6.80 Dialog Box - Setup window

## Dialog Box - Setup window

Sets properties for the setup window (background window). If you want to hide the setup window completely you can choose the **No setup window** option.

You can also specify your own application title for Visual Installer's setup program via the **Title** text box in this dialog box. The application title is shown in Window's task bar, and at the top of the setup window (in the caption area) if you also have selected the **Show title** option.

### 6.81 Dialog Box - Setup window - Absolute size

## Dialog Box - Absolute size

Sets width and height of the setup window.

### 6.82 Dialog Box - Setup window - Part of screen

## Dialog Box - Setup window

By dragging the slider in this dialog box you can adjust the percentage of the screen that the setup window should cover.

### 6.83 Dialog Box - Setup window - Settings

## Dialog Box - Setup window - Settings

Here you can choose type of setup window (background window) that will be shown in the background during the installation.

You can set more options via the **Design** tab in the editor.

### 6.84 Dialog Box - Shortcut

## Dialog Box - Shortcut

Via this dialog box you specify if a file should have a shortcut or not. A shortcut can be added to Windows menu system or to the desktop.

In the **Description** text box you specify a description text for the shortcut. In the **Parameters** text box you can specify command line parameters to a program file. In the **Start in** text box you can set a working folder for the file. The **Add shortcut to created program group** option must checked if you want to create a shortcut during the setup process.

**Variables**

6.85 Dialog Box - Shortcut (.ICO file)

# Dialog Box - Shortcut (.ICO file)

In this dialog box you can specify a full command line for the icon file. You can include parameters in the command line, for example like this:

"%DESTDIR\textpad.exe" readme.txt -a

If you want to use parameters you must set the file path between quotes.

**Variables**

6.86 Dialog Box - Show document after installation

# Dialog Box - Show document after installation

Specifies a document that should be opened after the setup. Below is the contents of the dialog box explained:

**Document**
In this text box you specify a document file that should be opened after the setup. You can use the %DESTDIR variable to create a file path to the document file, for example like: "**%DESTDIR\Docs\Readme.doc**".

**Show check box in the 'Installation completed' dialog box**
If this option is selected, a check box will be shown in the **Installation completed** setup dialog box. The user can choose whether he/she want to open the document or not by checking or unchecking this check box.

**Check box text**
Specifies a text for the check box in the **Installation completed** setup dialog box. See above for more information.

**Use internal RTF viewer to show the RTF document**
If your document is a RTF document you can check this option if you want the RTF document to be shown in a built-in dialog box in Visual Installer setup program. Otherwise, the RTF document will be opened via an associated program; usually Wordpad or Word.

**Window title**
Specifies a title for the dialog box that shows the RTF document.

6.87 Dialog Box - Sort file list

# Dialog Box - Sort file list

Specifies which columns in the file list to sort after. You can also click on a column to sort the file list after that column.

6.88 Dialog Box - Specify number of license keys to create

## Dialog Box - Specify number of license keys to create

Here you can enter the number of license keys to create.

6.89 Dialog Box - Submenu

## Dialog Box - Submenu

If you want to create a submenu to the program group / program menu, you can specify the name of it in the **Name of submenu** text box in this dialog box. If you want to remove the submenu from the project you can delete the text below and press **OK**.

6.90 Dialog Box - Subtitle

## Dialog Box - Subtitle

If you want a subtitle to appear below the main title you can specify it in this dialog box. The subtitle may contain variables; these will then be expanded in the setup screen during the installation.

6.91 Dialog Box - Support Information

## Dialog Box - Support Information

It is possible to include some extra information in **Add / Remove Program** in Windows that gives information about your software and your company. You can for example include information about version number, company name, support URL etc. In this dialog box you can specify such information.

Variables can be used in this dialog box. If you already have specified for example version number and company name in the **Version Information** dialog box, you can reuse the information in an easy way by using variables, for example the %COMPANY and %PRODUCTVERSION variables.

If you want to specify your own icon to **Add / Remove Program** you can specify a file path to an icon at **Icon** . If the icon is included in the file list you can press the "..." to choose an icon from the list. If possible you should always use variables in the file path and not hard code any paths. Icons can be extracted from EXE, DLL and ICO files.

6.92 Dialog Box - Switchable pictures

## Dialog Box - Switchable pictures

In this dialog box you can specify pictures that the setup programs should switch between during the setup process. If you have a large installation, that will take some time, you can show the user pictures when he/she is waiting for the installation to complete.

6.93 Dialog Box - Test

## Dialog Box - Test

From this dialog box you can start a simulated installation. Dialog boxes, graphics, gauges, setup screens etc will be shown in the same way as in a real installation, but no files are actually copied. The Registry will also remain untouched.

6.94 Dialog Box - Text box

## Dialog Box - Text box

Via this dialog box you can create a framed box with information text that will be placed on the setup screen. After that the text box has been placed on the setup screen, you can drag the box to a position that you prefer.

6.95 Dialog Box - Title

## Dialog Box - Title

In this dialog box you specify a title that should be shown in the setup screen. If you want to specify a subtitle, you can press the **Subtitle** button.

6.96 Dialog Box - Underline

## Dialog Box - Underline

In this dialog box you can create an underline to a title that is placed on the setup screen.

6.97 Dialog Box - Uninstall

## Dialog Box - Uninstall

In this dialog box you can specify your own picture for the uninstallation dialog box. If no picture is specified, a standard picture in the uninstallation program is used.

Here you can also see current texts for the uninstallation dialog box. You can change the texts via the **Special - Language** menu item.

6.98 Dialog Box - Updates

## Dialog Box - Updates

In this dialog box you can specify how Visual Installer will handle updates. Below is the contents of this

dialog box explained:

**Handle updates**
If you check this option, Visual Installer will automatically know if an installation should be regarded as an update or as a first-time installation. You can distribute one setup package, and the setup program will behave different if your product are installed in a user's computers for the first time, or if the product is installed again. If a product is installed again, the dialog box that asks for a destination folder will not be shown. Visual Installer will already know what the destination folder is.

**Reference file**
In this text box you specify a filename of a reference file. This reference file is used to tell Visual Installer if the product already has been installed or not. If the reference file is found, Visual Installer assumes that the product already has been installed once and that the current installation is an update.

The reference file must always be placed in the main folder of the installation (%DESTDIR) and must never be placed in a sub folder. The reference file should always be a program file (a file with the filename extension ".EXE").

**Advanced**
If you press this button a dialog box with advanced settings will be shown.

### 6.99 Dialog Box - Updates - Advanced

## Dialog Box - Updates - Advanced

Update information is normally located in the following key in the Windows Registry:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths

If you want to use another Registry key instead, you can specify such a key in this dialog box.

### 6.100 Dialog Box - User Options

## Dialog Box - User Options

You can include your own options that are shown for the end-user via a setup dialog box that are shown during the installation process. The end-user can select and unselect these options before he/she continue the installation. Visual Installer's scripting language and **Registry** tab contain condition statements that can co-operate with these options, so the end-user can control which script lines to execute and which registry keys and values to add, during the installation.

**Text**
In the **Dialog title** text box you can enter a title for the setup dialog box and in the **Dialog text** text box you can enter an information text for the setup dialog box. The information text will be displayed in the upper part of the setup dialog box.

**Options**
Here you can choose so many options that you need for your installation and enter a descriptive text for every option. These texts will be shown to the right of every option check box in the setup dialog box. If you click on a button marked with "..." you can choose if the option should be selected or unselected from the beginning.

**Your own dialog picture**
If you want to have another picture for the option setup dialog box than the standard picture, you can select it here.

**Preview**
You can click on this button to preview the setup dialog box.

**How this setup dialog box co-operates with the scripting language and the Registry tab**
The options that you specify in this dialog box can co-operate with condition statements in Visual Installer's scripting language and **Registry** tab. This means that you can let the user decide if some special operations must be done in the computer during the installation process or not. If you click on the links below you can read more about how the options in this dialog box co-operate with Visual Installer's scripting language and **Registry** tab:

**Script commands - Conditions - Check user option**

6.101 Dialog Box - User Options - Settings

# Dialog Box - User Options - Settings

If you select the **This option should be selected as default in the setup dialog box** option, the corresponding user option will be selected (checked) from the beginning in the option setup dialog box that is shown for the end-user.

6.102 Dialog Box - Variables

# Dialog Box - Variables

Shows a list of all variables that are included in Visual Installer, and their current values (if applicable). You can change some of the variable's values by clicking the **Change values** button.

**Variables**

6.103 Dialog Box - Version information

# Dialog Box - Version information

In this dialog box you can enter version information for your Visual Installer project.

The contents of the **Company**, **Product name** and **Version number** text boxes are stored in the %COMPANY, %PRODUCTNAME and %PRODUCTVERSION variables.

**Save version information also in the self-extracting setup package**
If this option is checked, the version information will also be saved in the self-extracting setup package. All fields except the **Comments** field will be used

# Part VII

7 Registry

# Registry

In this chapter you find some information about how Visual Installer handles the Windows Registry.



7.1 What is Registry?

# What is Registry?

The Registry in Windows (often referred to as the Windows Registry) is database that contain information about hardware, software, operating system, document types etc. When you install a program to Windows, it is very likely that the program will use the Registry in some way. With Visual Installer you can add information to the Registry during the installation.

If you want to view the Registry in an easy way you can start the program REGEDIT.EXE that is included in Windows. Via this program you can also change data in the Registry.

7.2 How Visual Installer handles Registry

# How Visual Installer handles Registry

There are very powerful functions included in Visual Installer to handle the Registry in Windows. You can for example add text strings, numbers and binary data to the Registry via Visual Installer. You can also delete keys and values from the Registry. There are also special commands available that handles Microsoft Excel, Microsoft PowerPoint and AutoCAD. You can also add conditions, so information is only added to the Registry in certain circumstances.

### Supported root keys

Visual Installer can handle the following root keys (main keys) in the Registry:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
HKEY_DYN_DATA

### How to add a key and value to the Registry

You can specify a key and value that must be added to the Registry in the **Registry** tab in Visual Installer, by

pressing the **Add** button in this tab and entering the necessary information in the dialog box that is opened. You can also enter the information directly in the text editor in the **Registry** tab. Below we show how a line with Registry information can look like in the text editor in the **Registry** tab:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: AppDir="%DESTDIR"

### Special characters and special commands

A line with Registry information in the Registry tab uses some special characters and special commands. Below you find a list with them:

| | |
|---|---|
| \ | : Is used to separate subkeys in a line with Registry information. |
| :: | : Informs Visual Installer that now follows a value name and a value. |
| "Value" | : Quotes around a value indicates that it's a text string. No quotes indicates that it's a number. |
| =! | : An exclamation mark after the equal sign informs that binary data will be specified. |
| (Standard)= | : Informs that the value is a default value. |
| DELETE= | : Informs that this Registry key or value name should be deleted from Registry during installation. |
| UNINSTALL= | : Informs that this Registry key or value name should be deleted from Registry during uninstall. |
| // | : Informs that this line is a comment. |

### Syntax

The syntax of a line with Registry information look like:

**Key :: Name = Value**

where **Key** contains the Registry key, **Name** contains the value name and **Value** contains the value. The value can be a text string, a number or binary data. If you add a string to the Registry you must set the text within quotes. Example:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: ProgramName="Visual Installer"

All of Visual Installer's variables can be used in the value. Below we show an example of how to store the contents of the %DESTDIR variable in the Registry:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: AppDir="%DESTDIR"

### Default values

You can also add default values to Registry keys via Visual Installer. A default value is the value that is read if no value name is specified. The syntax of a line that adds a default value to the Registry looks like:

**Key :: (Standard) = Value**

Below we shows an example of a line that adds a default value:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: (Standard)="Visual Installer"

### Comments

You can also add comments. Comments are not stored in the Registry during the installation; they will only help you to better understand and remember what you are adding to the Registry. A comment always starts with two slashes ("//"). Example:

```
// Destination Folder
HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: AppDir="%DESTDIR"

// Source Folder
HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: SrcDir="%SRCDIR"
```

**Registry - binary data**

**Registry - example**

7.3 Registry - binary data

# Registry - binary data

It is possible to add binary data to the Registry via Visual Installer. Binary data are specifed as a series of hexadecimal numbers with spaces between the numbers. Every number must contain exactly two digits (e.g. **A8**). You must also add an exclamation mark after the equal sign if you want to add binary data to Registry. Example:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: BinaryData=!A0 32 FF B7 60

You can add up to 256 hexadecimal numbers to a value name in this way.

7.4 Uninstallation of Registry keys and values

# Uninstallation of Registry keys and values

It is possible to have keys and values removed from the Registry during an uninstallation process. If a line in the text editor in the **Registry** tab starts with "**UNINSTALL=**", this key or value will be deleted when the program is uninstalled.

When you use "**UNINSTALL=**" to uninstall keys and values from Registry, you should add the lines in the same order as when you added the keys and values to Registry. Visual Installer will start removing keys and values backwards, so the last line with uninstall information (that starts with "**UNINSTALL=**") will be removed first. If you need to remove a tree with keys during an uninstallation, it is especially important to have this in mind because Visual Installer can not remove a key if the key still have subkeys beneath it.

7.5 Special variables, commands and keys

# Special variables, commands and keys

We have added some special variables, commands and keys that will simplify some common tasks that are made via the Registry tab in Visual Installer.

**Installation of an Excel add-in**

**Installation of a PowerPoint add-in**

**AutoCAD profiles**

**Launch a program automatically when Windows restarts**

**Compatibility mode**

**Conditional execution of a line**

7.5.1 Installation of an Excel add-in

# Installation of an Excel add-in

It is possible to install an Excel add-in by using Visual Installer. The add-in will also be activated during the installation, so the add-in is active when Microsoft Excel starts the next time. To install an add-in you must first add the file to Visual Installer's file list, and then use a special command with the name **XLADDIN** in the **Registry** tab. Example:

XLADDIN=%DESTDIR\MyAddin.xla

In the example above, the add-in with the filename **MyAddin.xla** will be added to Microsoft Excel during the setup.

Uninstallation of add-ins is also supported, so when a user uninstalls your project, the add-in will be automatically removed from Microsoft Excel.

**More details about XLADDIN**

7.5.1.1 More details about XLADDIN

# More details about XLADDIN

When you use the **XLADDIN** command, Visual Installer will add necessary information in the Registry to inform Microsoft Excel where to find your Excel add-in. Information will be added to the following Registry keys:

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\8.0\Excel\Microsoft Excel
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\9.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\10.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\11.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\12.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\14.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\15.0\Excel\Options
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Office\16.0\Excel\Options

Microsoft Excel version 8 to 16 (Excel 97 to Excel 2016) are supported.

7.5.2 Installation of a PowerPoint add-in

# Installation of a PowerPoint add-in

With Visual Installer you can install a PowerPoint add-in in an easy way. The add-in will also be activated during the installation, so the add-in is active when Microsoft PowerPoint starts the next time. To install an add-in you must first add the file to Visual Installer's file list, and then use a special command with the name **PPADDIN** in the **Registry** tab. The syntax is:

PPADDIN : Name = File path

"Name" must be replaced with the name of the add-in, and "File path" must be replaced with the file path to the PPA file. Example:

PPADDIN : MyAddIn = %DESTDIR\AddIns\MyAddIn.ppa

Uninstallation is also supported, so when a user uninstalls your project, the add-in will be automatically removed from Microsoft PowerPoint.

7.5.3 AutoCAD profiles

# AutoCAD profiles

With Visual Installer you can create and register AutoCAD profiles in an easy way. There are lots of special variables in Visual Installer that can be used to handle this. They can be used in the **Registry** tab in the Visual Installer editor.

**AutoCAD profiles - AutoCAD**

**AutoCAD profiles - AutoCAD LT**

7.5.3.1 AutoCAD

# AutoCAD profiles - AutoCAD

With Visual Installer you can create and register AutoCAD profiles in an easy way. There are lots of special variables in Visual Installer that can be used to handle this. They can be used in the **Registry** tab in the Visual Installer editor. Below we show a list of all supported AutoCAD specific special variables:

## AutoCAD

### Version number

**%ACR14CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R14.0 :: CurVer
Contains current version number for installed AutoCAD release 14. This variable is used to get access to profiles.

**%ACR15CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R15.0 :: CurVer
Contains current version number for installed AutoCAD release 15. This variable is used to get access to profiles.

**%ACR16CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R16.0 :: CurVer
Contains current version number for installed AutoCAD release 16. This variable is used to get access to profiles.

**%ACR17CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.0 :: CurVer
Contains current version number for installed AutoCAD release 17. This variable is used to get access to profiles.

**%ACR17_1CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.1 :: CurVer
Contains current version number for installed AutoCAD release 17.1. This variable is used to get access to profiles.

**%ACR17_2CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.2 :: CurVer
Contains current version number for installed AutoCAD release 17.2. This variable is used to get access to profiles.

**%ACR18_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R18.0 :: CurVer

Contains current version number for installed AutoCAD release 18. This variable is used to get access to profiles.

**%ACR18_1CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R18.1 :: CurVer
Contains current version number for installed AutoCAD release 18.1. This variable is used to get access to profiles.

**%ACR18_2CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R18.2 :: CurVer
Contains current version number for installed AutoCAD release 18.2. This variable is used to get access to profiles.

**%ACR19_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R19.0 :: CurVer
Contains current version number for installed AutoCAD release 19. This variable is used to get access to profiles.

**%ACR19_1CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R19.1 :: CurVer
Contains current version number for installed AutoCAD release 19.1. This variable is used to get access to profiles.

**%ACR20_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R20.0 :: CurVer
Contains current version number for installed AutoCAD release 20.0. This variable is used to get access to profiles.

**%ACR20_1CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R20.1 :: CurVer
Contains current version number for installed AutoCAD release 20.1. This variable is used to get access to profiles.

**%ACR21_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R21.0 :: CurVer
Contains current version number for installed AutoCAD release 21.0. This variable is used to get access to profiles.

**%ACR22_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R22.0 :: CurVer
Contains current version number for installed AutoCAD release 22.0. This variable is used to get access to profiles.

**%ACR23_0CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R23.0 :: CurVer
Contains current version number for installed AutoCAD release 23.0. This variable is used to get access to profiles.

## Installation folder

**%AC14LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R14.0\%ACR14CV :: AcadLocation
Contains the main folder where AutoCAD release 14 is installed.

**%AC15LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R15.0\%ACR15CV :: AcadLocation
Contains the main folder where AutoCAD release 15 is installed.

**%AC16LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R16.0\%ACR16CV :: AcadLocation
Contains the main folder where AutoCAD release 16 is installed.

**%AC17LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.0\%ACR17CV ::

AcadLocation
Contains the main folder where AutoCAD release 17 is installed.

**%AC17_1LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.1\%ACR17_1CV :: AcadLocation
Contains the main folder where AutoCAD release 17.1 is installed.

**%AC17_2LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R17.2\%ACR17_2CV :: AcadLocation
Contains the main folder where AutoCAD release 17.2 is installed.

**%AC18_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R18.0\%ACR18_0CV :: AcadLocation
Contains the main folder where AutoCAD release 18 is installed.

**%AC18_1LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R18.1\%ACR18_1CV :: AcadLocation
Contains the main folder where AutoCAD release 18.1 is installed.

**%AC18_2LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R18.2\%ACR18_2CV :: AcadLocation
Contains the main folder where AutoCAD release 18.2 is installed.

**%AC19_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R19.0\%ACR19_0CV :: AcadLocation
Contains the main folder where AutoCAD release 19 is installed.

**%AC19_1LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R19.1\%ACR19_1CV :: AcadLocation
Contains the main folder where AutoCAD release 19.1 is installed.

**%AC20_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R20.0\%ACR20_0CV :: AcadLocation
Contains the main folder where AutoCAD release 20.0 is installed.

**%AC20_1LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R20.1\%ACR20_1CV :: AcadLocation
Contains the main folder where AutoCAD release 20.1 is installed.

**%AC21_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R21.0\%ACR21_0CV :: AcadLocation
Contains the main folder where AutoCAD release 21.0 is installed.

**%AC22_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R22.0\%ACR22_0CV :: AcadLocation
Contains the main folder where AutoCAD release 22.0 is installed.

**%AC23_0LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R23.0\%ACR23_0CV :: AcadLocation
Contains the main folder where AutoCAD release 23.0 is installed.

The variables that end with the letters "CV", for example %ACR14CV, are special because they can exists *in* the key. This is necessary because you can not locate the profile until you know the version number of the installed AutoCAD. Below we show how the variable %ACR14CV can be used:

HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R14.0\%ACR14CV\Profiles\MyProfile :: Data="MyData"

7.5.3.2 AutoCAD LT

# AutoCAD profiles - AutoCAD - LT

With Visual Installer you can create and register AutoCAD profiles in an easy way. There are lots of special variables in Visual Installer that can be used to handle this. They can be used in the **Registry** tab in the Visual Installer editor. Below we show a list of all supported AutoCAD LT specific special variables:

## AutoCAD LT

### Version number

**%ACLT97CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R4.0 :: CurVer
Contains current version number for installed AutoCAD LT 97. This variable is used to get access to profiles.

**%ACLT98CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R5.0 :: CurVer
Contains current version number for installed AutoCAD LT 98. This variable is used to get access to profiles.

**%ACLT2000CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R2000 :: CurVer
Contains current version number for installed AutoCAD LT 2000. This variable is used to get access to profiles.

**%ACLT2000iCV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R7 :: CurVer
Contains current version number for installed AutoCAD LT 2000i. This variable is used to get access to profiles.

**%ACLT2002CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R8 :: CurVer
Contains current version number for installed AutoCAD LT 2002. This variable is used to get access to profiles.

**%ACLT2004CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R9 :: CurVer
Contains current version number for installed AutoCAD LT 2004. This variable is used to get access to profiles.

**%ACLT2005CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R10 :: CurVer
Contains current version number for installed AutoCAD LT 2005. This variable is used to get access to profiles.

**%ACLT2006CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R11 :: CurVer
Contains current version number for installed AutoCAD LT 2006. This variable is used to get access to profiles.

**%ACLT2007CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R12 :: CurVer
Contains current version number for installed AutoCAD LT 2007. This variable is used to get access to profiles.

**%ACLT2008CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R13 :: CurVer
Contains current version number for installed AutoCAD LT 2008. This variable is used to get access to

profiles.

**%ACLT2009CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R14 :: CurVer
Contains current version number for installed AutoCAD LT 2009. This variable is used to get access to profiles.

**%ACLT2010CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R15 :: CurVer
Contains current version number for installed AutoCAD LT 2010. This variable is used to get access to profiles.

**%ACLT2011CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R16 :: CurVer
Contains current version number for installed AutoCAD LT 2011. This variable is used to get access to profiles.

**%ACLT2012CV**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R17 :: CurVer
Contains current version number for installed AutoCAD LT 2012. This variable is used to get access to profiles.

**%ACLT2013CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R18 :: CurVer
Contains current version number for installed AutoCAD LT 2013. This variable is used to get access to profiles.

**%ACLT2014CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R20 :: CurVer
Contains current version number for installed AutoCAD LT 2014. This variable is used to get access to profiles.

**%ACLT2015CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R21 :: CurVer
Contains current version number for installed AutoCAD LT 2015. This variable is used to get access to profiles.

**%ACLT2016CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R22 :: CurVer
Contains current version number for installed AutoCAD LT 2016. This variable is used to get access to profiles.

**%ACLT2017CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R23 :: CurVer
Contains current version number for installed AutoCAD LT 2017. This variable is used to get access to profiles.

**%ACLT2018CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R24 :: CurVer
Contains current version number for installed AutoCAD LT 2018. This variable is used to get access to profiles.

**%ACLT2019CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R25 :: CurVer
Contains current version number for installed AutoCAD LT 2019. This variable is used to get access to profiles.

**%ACLT2020CV**
Reads the value at HKEY_CURRENT_USER\Software\Autodesk\AutoCAD LT\R26 :: CurVer
Contains current version number for installed AutoCAD LT 2020. This variable is used to get access to profiles.

## Installation folder

**%ACLT97LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R4.0\%ACLT97CV :: AcltLocation

Contains the main folder where AutoCAD LT 97 is installed.

**%ACLT98LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R5.0\%ACLT98CV :: AcltLocation
Contains the main folder where AutoCAD LT 98 is installed.

**%ACLT2000LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R2000\%ACLT2000CV :: Location
Contains the main folder where AutoCAD LT 2000 is installed.

**%ACLT2000iLOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R7\%ACLT2000iCV :: Location
Contains the main folder where AutoCAD LT 2000i is installed.

**%ACLT2002LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R8\%ACLT2002CV :: Location
Contains the main folder where AutoCAD LT 2002 is installed.

**%ACLT2004LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R9\%ACLT2004CV :: Location
Contains the main folder where AutoCAD LT 2004 is installed.

**%ACLT2005LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R10\%ACLT2005CV :: Location
Contains the main folder where AutoCAD LT 2005 is installed.

**%ACLT2006LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R11\%ACLT2006CV :: Location
Contains the main folder where AutoCAD LT 2006 is installed.

**%ACLT2007LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R12\%ACLT2007CV :: Location
Contains the main folder where AutoCAD LT 2007 is installed.

**%ACLT2008LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R13\%ACLT2008CV :: Location
Contains the main folder where AutoCAD LT 2008 is installed.

**%ACLT2009LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R14\%ACLT2009CV :: Location
Contains the main folder where AutoCAD LT 2009 is installed.

**%ACLT2010LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R15\%ACLT2010CV :: Location
Contains the main folder where AutoCAD LT 2010 is installed.

**%ACLT2011LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R16\%ACLT2011CV :: Location
Contains the main folder where AutoCAD LT 2011 is installed.

**%ACLT2012LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R17\%ACLT2012CV :: Location
Contains the main folder where AutoCAD LT 2012 is installed.

**%ACLT2013LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R18\%ACLT2013CV :: Location
Contains the main folder where AutoCAD LT 2013 is installed.

**%ACLT2014LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R20\%ACLT2014CV :: Location
Contains the main folder where AutoCAD LT 2014 is installed.

**%ACLT2015LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R21\%ACLT2015CV :: Location
Contains the main folder where AutoCAD LT 2015 is installed.

**%ACLT2016LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R22\%ACLT2016CV :: Location
Contains the main folder where AutoCAD LT 2016 is installed.

**%ACLT2017LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R23\%ACLT2017CV :: Location
Contains the main folder where AutoCAD LT 2017 is installed.

**%ACLT2018LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R24\%ACLT2018CV :: Location
Contains the main folder where AutoCAD LT 2018 is installed.

**%ACLT2019LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R25\%ACLT2019CV :: Location
Contains the main folder where AutoCAD LT 2019 is installed.

**%ACLT2020LOC**
Reads the value at HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD LT\R26\%ACLT2020CV :: Location
Contains the main folder where AutoCAD LT 2020 is installed.

The variables that end with the letters "CV", for example %ACLT2004CV, are special because they can exists in the key. This is necessary because you can not locate the profile until you know the version number of the installed AutoCAD. Below we show how the variable %ACLT2004CV can be used:

HKEY_LOCAL_MACHINE\Software\Autodesk\AutoCAD\R9\%ACLT2004CV\Profiles\MyProfile :: Data="MyData"

### 7.5.4 Launch a program automatically when Windows restarts

## Launch a program automatically when Windows restarts

Sometimes it is necessary to run a program automatically when Windows restarts. In Visual Installer there are two special keys that you can use in the **Registry** tab to achieve this:

**KEY_STARTUP_RUN** : launch a program every time Windows starts
**KEY_STARTUP_RUNONCE** : launch a program only the next time Windows starts

The syntax is:

<Specialkey> : Name = File path

where "Name" is the name of the application and "File path" is a full file path to the application. The path can contain variabes. Examples:

KEY_STARTUP_RUN : MyApplication = %DESTDIR\MyApp.exe

KEY_STARTUP_RUNONCE : MyProg = %DESTDIR\MyProg.exe

### 7.5.5 Compatibility mode

## Compatibility mode

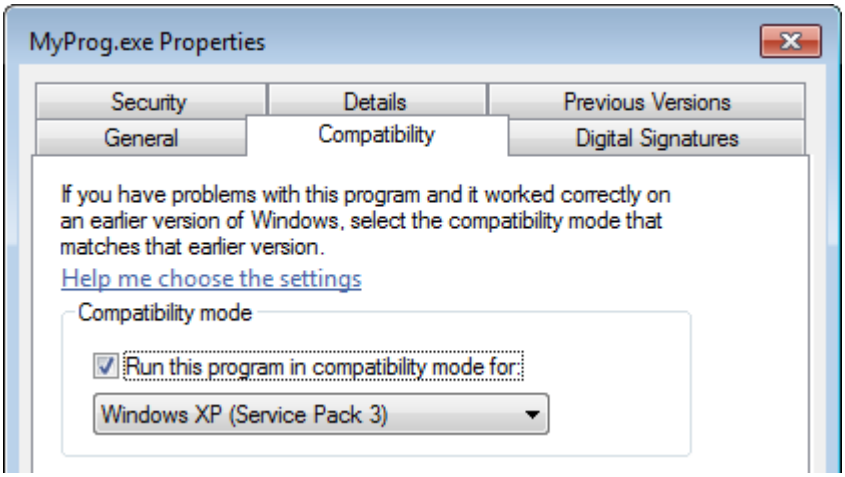
Old legacy programs sometimes get problems when they are run in new Windows versions. To reduce the risk for problems you can run the program in compatibility mode. A common problem is that applications that works well for Windows XP and older, may get problems when run in Windows Vista or newer. To handle this situation we have therefore added a special command with the name **XPCOMPAT** that sets the program in compatibility mode for Windows XP SP3 (the latest version of Windows XP). The command is used in this way:

XPCOMPAT = File path

Example:

XPCOMPAT = %DESTDIR\MyProg.exe

In the example above, the program MyProg.exe will be run in compatibility mode for Windows XP SP3. This is same as setting the following setting in Windows:




### 7.5.6 Conditional execution of a line

## Conditional execution of a line

Sometimes it is necessary to determine during the installation if a key or value should be added to the Registry or not. In Visual Installer this is possible by using conditional execution of lines in the **Registry** tab. A specific line will only be executed if a condition is true. The following conditions are supported:

### Operating system test

**IF_WIN10**

Execute the line only if the program is run in Windows 10 (or newer). Example:

IF_WIN10 HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: ThisIsWin10="TRUE"

**IF_WIN8**
Execute the line only if the program is run in Windows 8 (or newer). Example:

IF_WIN8 HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: ThisIsWin8="TRUE"

**IF_WIN7**
Execute the line only if the program is run in Windows 7 (or newer). Example:

IF_WIN7 HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: ThisIsWin7="TRUE"

**IF_VISTA**
Execute the line only if the program is run in Windows Vista (or newer). Example:

IF_VISTA HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: ThisIsVista="TRUE"

**IF_XP**
Execute the line only if the program is run in Windows XP (or newer). Example:

IF_XP HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: ThisIsXP="TRUE"

## License key test

**IF_PRIMEKEY**
Execute the line only if the license key that was entered by the user is a prime number based license key. Example:

IF_PRIMEKEY HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: KeyType="PrimeNumber"

**IF_DEMOKEY**
Execute the line only if the license key that was entered by the user is "**DEMO**". Example:

IF_DEMOKEY HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Data :: KeyType="Demo"

## User option test

**IF_OPTION(*n*)** or **IF_OPTION(*n*)=ON**
Where *n* is a number between 1 and 9. Executes the line if option number *n* was selected by the user in the User Options dialog box. Examples:

Example 1:
IF_OPTION(2) HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Options :: Option2="TRUE"

Example 2:
IF_OPTION(3)=ON HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Options :: Option3="TRUE"

**IF_OPTION(*n*)=OFF**
Where *n* is a number between 1 and 9. Executes the line if option number *n* was not selected by the user in the User Options dialog box. Example:

IF_OPTION(3)=OFF HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\TEST\Options :: Option3="FALSE"

7.6 Registry - example

# Registry - example

Below we show how script lines in the **Registry** tab can be used to register an icon and document type for a specific filename extension:

HKEY_CLASSES_ROOT\.vip :: (Standard)="VisualInstaller"

```
HKEY_CLASSES_ROOT\VisualInstaller :: (Standard)="Visual Installer Project"
HKEY_CLASSES_ROOT\VisualInstaller\DefaultIcon :: (Standard)="%DESTDIR\VI.EXE"
HKEY_CLASSES_ROOT\VisualInstaller\shell\open\command :: (Standard)="%DESTDIR\VI.EXE %1"
```

The example above is an authentic example that shows how Visual Installer registers the **.VIP** filename extension in the system during the installation of Visual Installer.

# Part VIII

8 INI files

# INI files

In this chapter you will find some information about how Visual Installer handles INI files.

## 8.1 More information about INI files

# More information about INI files

An INI file is a text file that contains data and settings for programs etc. An INI file has usually the filename extension ".INI". Visual Installer can add information to INi files via the **INI files** tab in the Visual Installer editor. Below is an example with lines with data that have been specified via Visual Installer:

```
-- VISUINST.INI --
[Directories]
AppDir=%DESTDIR
SrcDir=%SRCDIR

[InstallData]
InstallDate=%DATE
InstallTime=%TIME

[Coordinates]
XPos=200
YPos=100
```

Some variables are been used in the example above. During the setup process the variables will be replaced with actual values, so in the computer a INI file like this will be created on the user´s hard disk:

```
[Directories]
AppDir=C:\Program Files\SamLogic\Visual Installer
SrcDir=A:\

[InstallData]
InstallDate=2014-12-08
InstallTime=18:45:16

[Coordinates]
XPos=200
YPos=100
```

## 8.2 How to delete values from INI files

# How to delete values from INI files

You can also delete values from INI files by using Visual Installer. You can delete a value by giving it the value "%%". When Visual Installer encounters a "%%" value, the value will be removed from the INI file. Example:

```
[Directories]
AppDir=%DESTDIR
SrcDir=%%
```

# Part

9 Variables

# Variables

Variables are used as storage places for information that can be used in different parts of a setup program. There exist variables that store folder and file paths, date and time, version information, license keys etc. Click on the links below to get more information about the variables in Visual Installer.

**Variables (reference)**

**More information about variables**

9.1 Variables (reference)

# Variables (reference)

Variables are used as storage places for information that can be used in different parts of a setup program. There exist variables that store folder and file paths, date and time, version information, license keys etc. Below you find a complete list of all variables that Visual Installer handles:

## Folders - Setup

**%DESTDIR**
Contains the main folder path for the setup project. The path is read from the **Main folder** text box in the **File list** tab.

**%COMPDIR**
Contains destination folders for file components (file groups) if such are used. The component's number is stored after the variable name, e.g. **%COMPDIR1**, **%COMPDIR2** etc.

**%SRCDIR**
Contains the source folder path for the setup package. More information.

## Folders - System

**%PROGRAMFILES**
Contains the path to the **Program Files** folder in the system. This path is used for storage of program files and the path is usually **C:\Program Files** or **C:\Program Files (x86)**. More information.

**%COMMONFILES**
Contains the folder where common files usually are copied to.

**%SYSDIR**
Contains the system directory in Windows (usually **C:\Windows\System32**). More information.

**%WINDIR**
Contains the Windows directory (usually **C:\Windows**).

**%FONTDIR**
Contains the folder where font files are installed to (usually **C:\Windows\Fonts**).

**%MYDOCUMENTS**
Contains the folder path to the **My Documents** folder.

**%MYPICTURES**
Contains the folder path to the **My Pictures** folder.

**%DESKTOPDIR**
Contains the folder path to the desktop. More information.

**%APPDATADIR**
Contains the path to the folder in the system that is used to store settings for specific users (the **Application Data** or **AppData\Roaming** folder). More information.

**%LOCALAPPDATADIR**
Contains the path to the folder in the system that is used to store local settings. More information.

**%TEMPLATESDIR**
Contains the path to the folder in Windows that is used for storage of templates.

**%DAODIR**
Contains the path to the folder that is used for storage of Microsoft DAO database object files.

### Folders - System - Shared data

**%SHAREDDOCUMENTS**
Contains the path to the folder in the system that is used for storage of shared documents (the **All Users\Document** or **Users\Public\Documents** folder). More information.

**%PUBLICDIR**
Contains the path to the folder in the system that is used for storage of shared files (the **All Users\Document** or **Users\Public** folder). More information.

**%APPDATADIR_ALLUSERS**
Contains the path to the folder in the system that is used to store shared settings for all users in a computer (the **All Users\Application Data** or **ProgramData** folder). More information.

### Folders - Microsoft Office

**%WORDSTARTUPDIR**
Contains the Microsoft Word startup folder. If no such a folder exists, this variable is set to %DESTDIR.

**%EXCELSTARTUPDIR**
Contains the Microsoft Excel startup folder. If no such a folder exists, this variable is set to %DESTDIR.

**%OFC_TEMPLATESDIR**
Contains the folder that is used for document templates in Microsoft Office. If no such a folder can be found, this variable is set to %MYDOCUMENTS. More information.

**%OFC_SHAREDTEMPLATESDIR**
Contains the folder that is used for shared document templates in Microsoft Office. If no such a folder can be found, this variable is set to %MYDOCUMENTS. More information.

### Drives

**%DESTDRIVE**
Contains the destination drive for the setup.

**%SRCDRIVE**
Contains the source drive for the setup. More information.

### License keys

**%ENTEREDLKEY**
Contains the license key (installation key) that the user entered during the setup.

**%ENTEREDLKEYPREFIX**
Contains the prefix (the first 4 characters) of an prime number based license key that was entered during the setup. If another type of license key was used, this variable is empty.

### Date and time

**%DATE**
Contains the current date.

**%TIME**
Contains the current time.

**%JULIANDAY**
Contains the current date as a Julian day number. For example, the Julian day number for September 1th 2011 is 2455806.

## Version information

**%COMPANY**
Contains the name of your company (specified in the **Versions information** dialog box).

**%PRODUCTNAME**
Contains the name of your product (specified in the **Versions information** dialog box).

**%PRODUCTVERSION**
Contains the version number of your product (specified in the **Versions information** dialog box).

**%XPRODUCTVERSION**
Contains a version number that is read automatically from a specified file with version data.

## User registration

**%USERCOMPANY**
Contains the company name that the user entered in the **Registration** setup dialog box.

**%USERNAME**
Contains the name that the user entered in the **Registration** setup dialog box.

**%USERADDRESS**
Contains the address that the user entered in the **Registration** setup dialog box.

**%USERREGION**
Contains the region or state that the user entered in the **Registration** setup dialog box.

**%USERZIPCODE**
Contains the zip code that the user entered in the **Registration** setup dialog box.

**%USERPHONE**
Contains the phone number that the user entered in the **Registration** setup dialog box.

**%USERFAX**
Contains the fax number that the user entered in the **Registration** setup dialog box.

**%USEREMAIL**
Contains the e-mail address that the user entered in the **Registration** setup dialog box.

## Miscellaneous

**%PROJECTNAME**
Contains the name of the project. This variable gets it value from the **Project name** text box in the **File list** tab.

**%PROGRAMGROUP**
Contains the name of the program group / program menu that will be created during the installation. This name is specified in the **Program group / menu** dialog box (can be opened via the menu item **List - Program group / menu**.

**%REG1 .. %REG5**
General variables that read their values from the Windows Registry. More information.

**More information about variables**

9.1.1 More information about %PROGRAMFILES

# More information about %PROGRAMFILES

The %PROGRAMFILES variable contains the folder path to the Program Files folder in Windows. It is usually **C:\Program Files** or **C:\Program Files (x86)** but the path can sometimes be something else. You should never "hard code" the path; instead, always use the %PROGRAMFILES variable. Visual Installer will obtain the correct folder path by asking the system.

**The %PROGRAMFILES32 and %PROGRAMFILES64 variables**
If you want to create an installation project that will install both 32 and 64 bit files, there are two special variables that you can use to install files to a Program Files folder of a specific bitness. The name of the variables are %PROGRAMFILES32 and %PROGRAMFILES64, and they can be used if you first select the **Both 32 bit and 64 bit files are included in the project** option in the **32/64 bit** tab in the **Setup options** dialog box. These two variables should only be used if the installation project have a mix of 32 bit and 64 bit files; otherwise you should always use the %PROGRAMFILES variable as a destination.

9.1.2 More information about %SYSDIR

# More information about %SYSDIR

The %SYSDIR variable contains the folder path to the system directory in Windows. It is usually **C:\Windows\System32** but the path can sometimes be something else. You should never "hard code" the path; instead, always use the %SYSDIR variable. Visual Installer will obtain the correct folder path by asking the system.

**The %SYSDIR32 and %SYSDIR64 variables**
If you want to create an installation project that will install both 32 and 64 bit files, there are two special variables that you can use to install files to a system directory of a specific bitness. The name of the variables are %SYSDIR32 and %SYSDIR64, and they can be used if you first select the **Both 32 bit and 64 bit files are included in the project** option in the **32/64 bit** tab in the **Setup options** dialog box. These two variables should only be used if the installation project have a mix of 32 bit and 64 bit files; otherwise you should always use the %SYSDIR variable as a destination.

9.1.3 More information about %SRCDIR and %SRCDRIVE

# More information about %SRCDIR and %SRCDRIVE

The %SRCDIR and %SRCDRIVE variables contain the source folder and source drive for the setup program. A drive path is stored in the %SRCDRIVE variable with a backslash at the end, e.g. "**E:**". The %SRCDIR variable contains a path that do not ends with a backslash, e.g. "**E:\MySetup**".

### Self-extracting setup packages
If you create a self-extracting setup package, the %SRCDIR and %SRCDRIVE variables may not always contain the folder to the self-extracting EXE file. These variables contain always the path to the setup program (SETUP.EXE etc.) and this program file can be placed in another drive or folder after extraction. If you don't create a self-extracting setup package, the %SRCDIR and %SRCDRIVE variables will always contain the folder path to the EXE file that the user starts.

9.1.4 More information about %SHAREDDOCUMENTS

## More information about %SHAREDDOCUMENTS

The %SHAREDDOCUMENTS variable contains the folder path to the folder in the system that is used for storage of shared documents. The path is different in different versions of Windows. The variable will usually contain one of the following folder paths:

Windows XP/2000 : **C:\Documents and Settings\All Users\Document**

Windows Vista/7/8/10 : **C:\Users\Public\Documents**

9.1.5 More information about %PUBLICDIR

## More information about %PUBLICDIR

The %PUBLICDIR variable contains the folder path to the folder in the system that is used for storage of shared files. The path is different in different versions of Windows. The variable will usually contain one of the following folder paths:

Windows XP/2000 : **C:\Documents and Settings\All Users\Dokument**

Windows Vista/7/8/10 : **C:\Users\Public**

9.1.6 More information about %APPDATADIR and related variables

## More information about %APPDATADIR and related variables

The %APPDATADIR, %APPDATADIR_ALLUSERS and %LOCALAPPDATADIR variables are used to return paths to folders in the system that are used to store settings for applications etc. Below are the three variables described in detail:

### %APPDATADIR

Contains the folder path to the folder in the system that is used to store settings for specific users. The variable will usually contain one of the following folder paths:

Windows XP/2000 **C:\Documents and Settings\<User>\Application Data**

Windows Vista/7/8/10 **C:\Users\<User>\AppData\Roaming**

### %APPDATADIR_ALLUSERS

Contains the folder path to the folder in the system that is used to store shared settings for all users in a computer. The variable will usually contain one of the following folder paths:

Windows XP/2000 **C:\Documents and Settings\All Users\Application Data**

Windows Vista/7/8/10 **C:\ProgramData**

### %LOCALAPPDATADIR

Contains the folder path to the folder in the system that is used to store local settings. The variable will usually contain one of the following folder paths:

| | |
|---|---|
| Windows XP/2000 | **C:\Documents and Settings\<User>\Lokala inställningar\Application Data** |
| Windows Vista/7/8/10 | **C:\Users\<User>\AppData\Local** |

### 9.1.7 More information about %DESKTOPDIR

## More information about %DESKTOPDIR

The %DESKTOPDIR variable contains the folder path to the desktop in Windows. The settings in the **Program group / menu** dialog box (that is opened via the menu item **List - Program group / menu**) decides if the path to the desktop for all users or the path to the desktop for the current user is stored in this variable.

### 9.1.8 More information about %OFC_TEMPLATESDIR and %OFC_SHAREDTEMPLATESDIR

## More information about %OFC_TEMPLATESDIR and %OFC_SHAREDTEMPLATESDIR

To determine which contents to store in the %OFC_TEMPLATESDIR and %OFC_SHAREDTEMPLATESDIR variables, Visual Installer will first examine which version of Microsoft Word that is installed. Thereafter, Visual Installer will obtain information from Word to determine what paths to store in these two variables. Visual Installer assumes that if Microsoft Word is installed, Microsoft Office is also very likely installed.

If you want to install files that are specific creted for Microsoft Excel or Microsoft PowerPoint, it can be better to use the following special variables instead:

%OFC_TEMPLATESDIR_XL
Check if Excel is installed, and get templates folder based on information from Excel.

%OFC_SHAREDTEMPLATESDIR_XL
Check if Excel is installed, and get shared templates folder based on information from Excel.

%OFC_TEMPLATESDIR_PP
Check if PowerPoint is installed, and get templates folder based on information from PowerPoint.

%OFC_SHAREDTEMPLATESDIR_PP
Check if PowerPoint is installed, and get shared templates folder based on information from PowerPoint.

Sometimes Excel or PowerPoint can be installed in a computer without an Office installation. So using the variables above will give higher security.

### 9.1.9 %REG1 .. %REG5 - examples

## %REG1 .. %REG5 - examples

The %REG1, %REG2, %REG3, %REG4 and %REG5 variables get their values from the Registry. By specifying a key and value name, the variables will know where in Registry to obtain their values. When you inform Visual Installer where in Registry to get a value, you must use the same syntax that in the Registy tab in the Visual Installer editor. Example:

HKEY_LOCAL_MACHINE\SOFTWARE\SamLogic\Visual Installer\10.5 :: Path

If you set %REG1 to the key and value name above, %REG1 will read it contents from this location in Registry during the setup.

9.2 More information about variables

# More information about variables

Visual Installer can handle variables in many parts of the editor. You can for example use variables in all main tabs of the editor and in most dialog boxes that handles file or folder paths. You can also use variables with the script language.

A variable must always contain uppercase letters. Examples:

%DESTDIR : CORRECT !
%PROGRAMFILES : CORRECT !

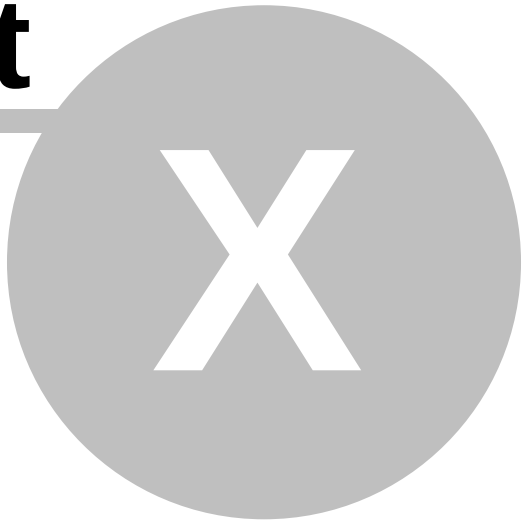
A variable can not contain lowercase letters. So the following examples are wrong:

%destdir : WRONG !!!
%ProgramFiles : WRONG !!!

You can combine variables with text, for example with folder names. Examples:

%DESTDIR\MyApp
%DESTDIR\MyApp\Examples
%PROGRAMFILES\SamLogic\Visual Installer
%MYDOCUMENTS\SamLogic\Word-Docs

# Part



10 Script commands

# Script commands - Overview

The script commands in Visual Installer are used to handle operations that the usual functions in Visual Installer can not handle. There exist script commands to copy files, create folders, add shortcuts, run programs, install external MSI setup packages etc. All Visual Installer variables can be used with script commands.

All script commands use one or more parameters. The parameters must be specified without quotes around the values. If pure text strings are specified, for example information messages, usage of quotes is optional A comma is used to separate two parameters from each other. All commands must be entered using uppercase letters.

Below is an example of a script command that copies a file from one location to another:

```
COPY %DESTDIR\App.exe, %DESTDIR\Bin\App.exe
```

The file with the name "**App.exe**" is copied from the "**%DESTDIR**" folder to the "**%DESTDIR\Bin**" folder.

**Script commands - Reference**

**Script examples**

**Variables**

10.1 Script commands - Reference

# Script commands - Reference

Below you find a list with all available script commands in Visual Installer. Click on a command to get more information.

## Files (operations)

| | |
|---|---|
| **COPY** | : Copies a file. |
| **MCOPY** | : Copies many files at once. |
| **MMOVE** | : Moves many files at once. |
| **DELETE** | : Deletes a file. |
| **XDELETE** | : Deletes a file, also if it's write protected. |
| **MDELETE** | : Deletes many files at once. |
| **RENAME** | : Change filename of a file. |
| **XRENAME** | : Change filename of a file (with extended error handling). |
| **WRITEPROTECT** | : Write-protects a file. |
| **MWRITEPROTECT** | : Write-protects many files at once. |

## Files (tests)

| | |
|---|---|
| **XFILEACTIVE** | : Checks if a file is in use (is active). |
| **XFILEEXISTS** | : Checks if a file exists. |
| **XNOTFILEEXISTS** | : Checks if a file not exists. |

## Folders (operations)

| | |
|---|---|
| **CREATEDIR** | : Creates a folder. |
| **REMOVEDIR** | : Removes a folder. |
| **XREMOVEDIR** | : Removes a folder, also if it's contain files. |
| **DELTREE** | : Deletes a folder tree. |

## Folders (tests)

**XDIREXISTS** : Checks if a folder exists.
**XNOTDIREXISTS** : Checks if a folder not exists.

## Executable files

**RUN** : Runs a program.
**XRUN** : Runs a program. Different settings can be used.

## MSI installations

**MSIEXEC** : Runs an MSI installation.

## Registration

**REGISTER** : Registers a file.
**REGISTER_ASSEMBLY** : Registers a .NET assembly.

## Permissions

**XCREATEDIR** : Creates a folder and adds specific access rights to the folder.
**SETPERM** : Changes access rights for a folder.

## Registry

**REG_KEYEXISTS** : Checks if a key exists in the Registry.
**REG_KEYNOTEXISTS** : Checks if a key not exists in the Registry.

## Environment variables

**SETENV** : Sets an environment variable to a value.

## Command interpreter

**CMD** : Starts Windows' command interpreter and executes a command.

## Menus, shortcuts and icons

**CREATEMENU** : Creates a menu (program group).
**ADDICON_MENU** : Adds a shortcut to a menu.
**ADDICON_MENU_PROGRAM** : Adds a shortcut to the Program menu.
**ADDICON_MENU_START** : Adds a shortcut to the Start menu.
**ADDICON_MENU_STARTUP** : Adds a shortcut to the Startup menu.
**ADDICON_DESKTOP** : Adds a shortcut to the desktop.
**DELETEICON** : Deletes a shortcut from a menu.
**DELETEICON_DESKTOP** : Deletes a shortcut from the desktop.
**RENAMEICON** : Changes the description text for a shortcut in a menu.
**RENAMEICON_DESKTOP** : Changes the description text for a shortcut in the desktop.

## Uninstallation

**UNINSTALL_DELETE** : Logs a file for deletion at uninstallation.
**UNINSTALL_DELETE_AT_REEBOT** : Logs a file for deletion at reboot during an uninstallation.
**UNINSTALL_DELETEICON_STARTUP** : Logs a startup shortcut for deletion at uninstallation.
**UNINSTALL_REMOVEDIR** : Logs a folder for deletion at uninstallation.
**UNINSTALL_REMOVEDIR_AT_REBOOT** : Logs a folder for deletion at reboot during an uninstallation.
**UNINSTALL_RESTORE** : Logs a file to restore at uninstallation.
**UNINSTALL_MSIEXEC** : Logs an MSI installation/uninstallation to be run at uninstallation.
**UNINSTALL_XRUN** : Logs a program file to be run at uninstallation.
**UNINSTALL_RENAME_LOGFILE** : Changes the filename of the uninstallation log file.

## Conditions

**IF / ELSE / END IF** : Conditions. IF, ELSE and END IF.

## Error handling

| | |
|---|---|
| **SHOWERROR** | : Error handling; display information on errors. |
| **HANDLEERROR** | : Error handling; how should errors be handled? |
| **USELOG** | : Activates or deactivates the script log. |

### Miscellaneous

| | |
|---|---|
| **SLEEP** | : Pause. |
| **EXIT** | : Stops the installation and closes the setup program. |
| **MESSAGE** | : Displays a message. |
| **MSGBOX** | : Shows a message box. |
| *//* | : Comment. |

**Script examples**

10.2 Script commands - Files

# Script commands - Files

## COPY %Source, %Destination

Copies a file from one location to another.

**%Source** : Source path for file.
**%Destination** : Destination path for file.

## MCOPY %Source, %DestinationDir

Copies many files at once. The command can handle wildcards.

**%Source** : Source path. Wildcards can be used, ex: "C:\Data*.*" or "%DESTDIR\Data*.txt".
**%Destination** : Destination folder for files.

## MMOVE %Source, %DestinationDir

Moves many files at once. The command can handle wildcards.

**%Source** : Source path. Wildcards can be used, ex: "C:\Data*.*" or "%DESTDIR\Data*.txt".
**%Destination** : Destination folder for files.

## DELETE %Filename

Deletes a file.

**%Filename** : Path to file to delete.

## MDELETE %Filename

Deletes many files at once. The command can handle wildcards.

**%Filename** : Files to delete. Wildcards can be used, ex: "C:\Data*.*" or "%DESTDIR\Data*.txt".

## XDELETE %Filename

Deletes a file, also if it's write protected.

**%Filename** : Path to file to delete.

## RENAME %OldName, %NewName

Change filename of a file.

**%OldName** : File path to file to rename.
**%NewName** : The new filename (including the file path)

### WRITEPROTECT %Filename, %Mode
Write-protects a file.

**%Filename** : The file path to the file that you want to write-protect.
**%Mode** : 1 = turn on write-protection, 0 = turn off write-protection.

### MWRITEPROTECT %Filename, %Mode
Write-protects many files at once.

**%Filename** : Files to protect/unprotect. Wildcards can be used, ex: "C:\Data*.*" or "%DESTDIR\Data*.txt".
**%Mode** : 1 = turn on write-protection, 0 = turn off write-protection.

**Script commands - Files (2)**

**Script examples**

10.3 Script commands - Files (2)

## Script commands - Files (2)

### XFILEACTIVE %Filename, %Title, %Text, %Type
Checks if a file is in use (is active). If the specified file is in use, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%Filename** : File path to the file to test.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### XFILEEXISTS %Filename, %Title, %Text, %Type
Checks if a file exists. If the specified file exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%Filename** : File path to the file to test.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### XNOTFILEEXISTS %Filename, %Title, %Text, %Type
Checks if a file not exists. If the specified file does not exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%Filename** : File path to the file to test.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### XRENAME %OldName, %NewName, %Title, %Text, %Type
Renames a file during the setup. If any error occurs when the file is renamed, a message box is shown.

**%OldName** : File path to file to rename.
**%NewName** : The new filename (including the file path)
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### The %Type parameter

The **%Type** parameter that is mentioned above can have one of the following two values:

**1 :** A message box with a **OK** button is shown. When the user presses OK, the installation is cancelled.
**2 :** A message box with a **Yes** and **No** button is shown. The user can decide if he/she want to continue the installation or want stop it. If the user presses No, the installation is cancelled.

### Using commas with the %Title and %Text parameters

In Visual Installer's script language the comma character is used to separate parameters from each other, so you can not include a text that contains commas. However, sometimes it is necessary to shows commas in a text to make it more readable, so we have included a method to have commas in texts. If you include the semicolon character in a text, the character will be shown as a comma in the message box.

**Script commands - Folders**

**Script examples**

10.4 Script commands - Folders

## Script commands - Folders

### CREATEDIR %Dir

Creates a folder.

**%Dir** : Specifies name of the folder to create (including the folder path).

### REMOVEDIR %Dir

Removes a folder.

**%Dir** : Specifies name of the folder to remove (including the folder path).

### XREMOVEDIR %Dir

Removes a folder, also if it's contain files. All files in the folder will be deleted.

**%Dir** : Specifies name of the folder to remove (including the folder path).

### DELTREE %Dir, %Filter, %Mode

Deletes a folder tree (including files). The first folder level will not be removed, but all sub folder will be removed it the option for folder deletion has been specified.

**%Dir** : Specifies the first level of a folder tree to delete.
**%Filter** : Filter. Can contain wildcards. Optional parameter.
**%Mode** : Mode. Optional parameter.

The **%Mode** parameter can have one of the following 3 values:

**1 :** Delete only files in the folder tree.
**2 :** Delete only folders in the folder tree.

**3 :** Delete both files and folders in the folder tree. This is the default value.

**Script commands - Folders (2)**

**Script examples**

10.5 Script commands - Folders (2)

# Script commands - Folders (2)

## XDIREXISTS %Dir, %Title, %Text, %Type

Checks if a folder exists. If the specified folder exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%Dir** : Path to the folder to test.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

## XNOTDIREXISTS %Dir, %Title, %Text, %Type

Checks if a folder not exists. If the specified folder does not exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%Dir** : Path to the folder to test.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### The %Type parameter

The **%Type** parameter that is mentioned above can have one of the following two values:

**1 :** A message box with a **OK** button is shown. When the user presses OK, the installation is cancelled.
**2 :** A message box with a **Yes** and **No** button is shown. The user can decide if he/she want to continue the installation or want stop it. If the user presses No, the installation is cancelled.

### Using commas with the %Title and %Text parameters

In Visual Installer's script language the comma character is used to separate parameters from each other, so you can not include a text that contains commas. However, sometimes it is necessary to shows commas in a text to make it more readable, so we have included a method to have commas in texts. If you include the semicolon character in a text, the character will be shown as a comma in the message box.

**Script commands - Binary / Executable files**

**Script examples**

# Script commands - Executable files

## RUN %Filename

Runs a program. Can also be used to start an MSI setup.

**%Filename** : Specifies a file path to the program file to run.

## XRUN %Filename, %Flag, %Param, %OS, %Bitness

Runs a program. Different options are available.

**%Filename** : Specifies a file path to the program file to run.
**%Flag** : Specifies how to start the program. Must be a value between 1 and 4 (see below).
**%Param** : Command line parameters, if needed. Optional parameter.
**%OS** : Required operating system. Optional parameter.
**%Bitness** : Required bitness for the operating system. Optional parameter.

### More information about the %Flag, the %OS and the %Bitness parameter:

#### %Flag

The **%Flag** parameter must have one of the following values:

**1** : Starts a program and returns immediately. Same as **RUN**.
**2** : Starts a program and returns when the program has been initialized.
**3** : Starts an invisible program and returns when the program is closed.
**4** : Starts a visible program and returns when the program is closed.

#### %OS

The **%OS** parameter is an optional parameter that specifies in which operating system (Windows version) the program can be run on. The **%OS** parameter, if used, must have one of the following values:

**0** : In any Windows.
**7** : Only in Windows 7.
**8** : Only in Windows 8.
**10** : Only in Windows 10.

The use of this parameter is optional. If you omit this parameter the program will be run in any Windows version. If you specify an operating system, for example sets **%OS** to **7** (= Windows 7), the program will only be run in this operating and never run in other operating systems.

#### %Bitness

The **%Bitness** parameter is an optional parameter that specifies which bitness the operating system (Windows version) must have to let the program start. The **%Bitness** parameter, if used, must have one of the following values:

**0** : In any bitness.
**32** : Run only in a 32 bit Windows.
**64** : Run only in a 64 bit Windows.

The use of this parameter is optional. If you omit this parameter the program will be run in a Windows of any bitness (both 32 bit and 64 bit Windows). If you specify a bitness, for example sets **%Bitness** to **64**, the program will only be run in a Windows with this bitness. If the bitness is something else, the program is not run.

### Examples of usage

The example below starts **MyApp.exe** and returns immediately.

```
XRUN %DESTDIR\MyApp.exe, 1
```

The example below starts **CleanDataFolders.exe** and waits until the program has completed its operations. The program is run silently and invisible.

```
XRUN %DESTDIR\CleanDataFolders.exe, 3
```

The example below starts **MyWin7App.exe** if the operating system is Windows 7. If the operating system is anything else, for example Windows 10, the program is never run.

```
XRUN %DESTDIR\MyWin7App.exe, 1, "", 7
```

The example below starts **MyWin7App.exe** if the operating system is a 64 bit Windows 7. If the operating system is anything else the program is never run.

```
XRUN %DESTDIR\MyWin7App.exe, 1, "", 7, 64
```

**Script commands - MSI installations**

**Script examples**

10.7 Script commands - MSI installations

# Script commands - MSI installations

## MSIEXEC %Filename, %Param, %Wait

Runs an MSI installation. With this command you can start an MSI installation (Windows Installer installation) from Visual Installer. You can choose whether Visual Installer should pause its own installation until the MSI installation is finished or if Visual Installer should continue its own installation immediately after that the MSI installation has been started. If you choose the later option, the MSI installation will run parallel with Visual Installer's installation.

The **MSIEXEC** command takes the following three parameters:

**%Filename** : Complete path to the MSI file, for example **%DESTDIR\MySetup.msi**.
**%Param** : Command line parameters to send to Windows Installer. Optional. (see below).
**%Wait** : Wait until the MSI installation is completed. Optional. (see below).

## More information about the %Param parameter

With the **%Param** parameter you can send command line parameters to Windows Installer. A complete list with possible parameters is available on this Microsoft page:

http://technet.microsoft.com/en-us/library/bb490936.aspx

The **MSIEXEC** script command calls the **MSIEXEC.exe** program, so all parameters that the **MSIEXEC.exe** program can handle can also be sent via the **%Param** parameter. You should read the Microsoft page to get the full list, but some useful parameters are the following:

**/qn** : Runs a silent installation (no setup dialog boxes or messages are shown during the setup).
**/qb** : Shows a reduced user interface, only shows how the installation proceeds.
**/qf** : Shows a full user interface, with complete setup dialog boxes and error messages etc.

The **MSIEXEC** script command automatically adds the **/i** command line parameter when it calls the **MSIEXEC.exe** program, so you never need to specify this parameter.

## More information about the %Wait parameter

Sometimes it is necessary to pause Visual Installer's setup process until the MSI installation is completed. If you want Visual Installer to wait until the MSI installation is ready, you can set the third parameter of the **MSIEXEC** command to **WAIT** or **1**. If you set this parameter to 0 or does not specify it at all, Visual Installer

will continue its own installation immediately after it has started the MSI installation. The MSI installation will run parallel with Visual Installer's installation.

### Examples of usage

The example below installs **MYSetup.msi**. The installation runs parallel with Visual Installer's own installation. The installation is run silently (no user interface is shown).

```
MSIEXEC %DESTDIR\MySetup.msi, /qn
```

The example below installs **MYSetup.msi**. Visual Installer pauses its own installation until the installation of **MYSetup.msi** is completed. A full user interface, with setup dialog boxes etc., is shown for the end-user.

```
MSIEXEC %DESTDIR\MySetup.msi, /qf, WAIT
```

The example below first installs **MYSetup1.msi** and thereafter, when the installation of **MYSetup1.msi** is completed, it installs **MySetup2.msi**. When the installation of **MySetup2.msi** is completed, Visual Installer continues its own installation. All installations are run silently.

```
MSIEXEC %DESTDIR\MySetup1.msi, /qn, WAIT
MSIEXEC %DESTDIR\MySetup2.msi, /qn, WAIT
```

More examples are available on this tip page on our website:

http://www.samlogic.net/visual-installer/tips/tips-pages/run-msi-installation-from-script

**Script commands - Registration**

**Script examples**

10.8 Script commands - Registration

# Script commands - Registration

## REGISTER %Filename

Registers a file. The file can be a DLL file, an OCX file (ActiveX component) or an EXE file. The command takes the following parameter:

**%Filename** : Specifies a file path to the file to register.

## REGISTER_ASSEMBLY %Filename, %NetVersion, %Options, %3264bit

Registers a .NET assembly. If you need to register a .NET assembly you must use this command instead of the **REGISTER** command.

When you use the **REGISTER_ASSEMBLY** command, the Assembly Registration Tool (**RegAsm.exe**) is run from Visual Installer. The **REGISTER_ASSEMBLY** command takes the following parameters:

**%Filename** : Specifies a file path to the file to register.
**%NetVersion** : Specifies the .NET version. Can be one of these values: **2**, **3**, **3.5**, **4**, **4.5**.or **4.6**. Optional (if omitted **2** is used).
**%Options** : Options to pass to the Assembly Registration Tool (**RegAsm.exe**). Optional.
**%3264bit** : If this parameter is set to **1**, both a 32 bit and 64 bit registration is made. Optional.

The **%Options** parameter can contain the same options as the Assembly Registration Tool (**RegAsm.exe**). Open this web page at Microsoft's website for details. The **%NetVersion** parameter specifies the version of .NET Framework that the assembly has been (or will be) compiled to. If the **%3264bit** parameter is set to 1, both a 32 bit and 64 bit registration is made. Otherwise is same bitness used as the installation project is using.

### Examples of usage

The example below shows how to use the **REGISTER** command to register an ActiveX component (an OCX file):

```
REGISTER %SYSDIR\SLClock.ocx
```

The example below shows how to use the **REGISTER_ASSEMBLY** command to register a .NET assembly:

```
REGISTER_ASSEMBLY %DESTDIR\SLCalendar.dll, 2, /codebase
```

Another example that shows how to use the **REGISTER_ASSEMBLY** command to register a .NET assembly. The example below registers a .NET assembly that is compiled for .NET Framework 4.0 and that can be used in both 32 bit and 64 bit applications:

```
REGISTER_ASSEMBLY %DESTDIR\SLCalendarV4.dll, 4, /codebase /register, 1
```

**Script commands - Permissions**

**Script examples**

10.9 Script commands - Permissions

# Script commands - Permissions

## XCREATEDIR %Dir, %Permission

Creates a folder and adds specific permissions (access rights) to the folder.

**%Dir** : Specifies name of the folder to create (including the folder path).
**%Permission** : Specifies which permissions to add to the folder.

The **%Permission** parameter must contain one of the following three values:

**PERM_ALL** : Full access rights (read, write and execute).
**PERM_READ** : Permission to read/open.
**PERM_WRITE** : Permission to write (and read/open).

The example below shows how to create a folder that will have full access rights (read, write and execute):

```
XCREATEDIR %APPDATADIR_ALLUSERS\MyFolder, PERM_ALL
```

The **XCREATEDIR** command will not change permissions for a folder if it already exists. If you want to make sure that the specified folder will have specific access rights, regardless of previous access rights, you can combine the **CREATEDIR** command with the **SETPERM** command instead.

## SETPERM %Dir, %Permission

Changes permissions (access rights) for a folder that already exists.

**%Dir** : Folder to change access rights for.
**%Permission** : Specifies which permissions to add to the folder.

The **%Permission** parameter must contain one of the following three values:

**PERM_ALL** : Full access rights (read, write and execute).
**PERM_READ** : Permission to read/open.
**PERM_WRITE** : Permission to write (and read/open).

The example below shows how to change permissions for a folder to full access rights (read, write and execute):

```
SETPERM %APPDATADIR_ALLUSERS\MyFolder, PERM_ALL
```

**Script commands - Registry**

**Script examples**

10.10 Script commands - Registry

# Script commands - Registry

## REG_KEYEXISTS %HKEY, %SubKey, %Title, %Text, %Type

Checks if a key exists in the Registry. If the specified key exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%HKEY** : Root key. HKEY_CLASSES_ROOT, HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE.
**%SubKey** : Sub key.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

The example below shows how this command can be used:

```
REG_KEYEXISTS  HKEY_LOCAL_MACHINE, SOFTWARE\SamLogic\Visual Installer\10.5, Information, This program is already installed, 2
```

## REG_KEYNOTEXISTS %HKEY, %SubKey, %Title, %Text, %Type

Checks if a key does not exists in the Registry. If the specified key does not exists, a message box is shown and the user has the option to cancel the installation if he/she wants.

**%HKEY** : Root key. HKEY_CLASSES_ROOT, HKEY_CURRENT_USER or HKEY_LOCAL_MACHINE.
**%SubKey** : Sub key.
**%Title** : Title of the message box.
**%Text** : Information text for the message box.
**%Type** : Type of message box (can be 1 or 2). More information is available below.

### The %Type parameter

The **%Type** parameter that is mentioned above can have one of the following two values:

**1 :** A message box with a **OK** button is shown. When the user presses OK, the installation is cancelled.
**2 :** A message box with a **Yes** and **No** button is shown. The user can decide if he/she want to continue the installation or want stop it. If the user presses No, the installation is cancelled.

### Using commas with the %Title and %Text parameters

In Visual Installer's script language the comma character is used to separate parameters from each other, so you can not include a text that contains commas. However, sometimes it is necessary to shows commas in a text to make it more readable, so we have included a method to have commas in texts. If you include the semicolon character in a text, the character will be shown as a comma in the message box.

**Script commands - Environment variables**

**Script examples**

10.11 Script commands - Environment variables

# Script commands - Environment variables

## SETENV %Variable, %Value

Sets an environment variable in the system during the installation. The command take two parameters:

**%Variable** : Specifies the name of the environment variable that you want to set.
**%Value** : Specifies the value to which you want to set the environment variable.

The example below shows how to set an environment variable with the name **MyVariable** to the value **MyValue**:

```
SETENV  MyVariable, MyValue
```

You can use Visual Installer's variables (for example **%DESTDIR**) when you set values. The example below shows how to set an environment variable with the name **MyAppsFolder** to the path **%DESTDIR\MyApps**:

```
SETENV  MyAppsFolder, %DESTDIR\MyApps
```

**Script commands - Command interpreter**

**Script examples**

10.12 Script commands - Command interpreter

# Script commands - Command interpreter

## CMD %Command, %NoCloseWin

Starts the command interpreter in Windows and executes a command.

**%Command** : Command in Windows to execute, for example: "DIR *.txt".
**%NoCloseWin** : Specifies if the command interpreter window should be closed or not after use. Optional parameter.

The **%NoCloseWin** parameter can have one of the following values:

**0 :** Close the command interpreter window when the command has been executed. This is the default value.
**1 :** Do not close the command interpreter window when the command has been executed.

The **CMD** command can also be used to execute BAT files. This is useful if many Windows commands need to be executed at once. Below we show how to execute a BAT file using the **CMD** command:

```
CMD %DESTDIR\MyCmds.bat
```

**Script commands - Menus, shortcuts and icons**

**Script examples**

# Script commands - Menus, shortcuts and icons

## CREATEMENU %MenuName

Creates a menu (program group) in Windows. This menu will be used as a storage place for shortcuts that belongs to a product.

**%MenuName** : The name of the menu to create.

You can also create sub menus with this command. You separate different menu levels from each other by using backslashes ( \ ). Example: "Menu\SubMenu ".

## ADDICON_MENU %MenuName, %Text, %Filename, %Param, %WorkDir, %IconPath

Adds a shortcut to a menu that was created with the **CREATEMENU** command.

**%MenuName** : Specifies the name of the menu where the shortcut should be added to.
**%Text** : Specifies a description text for the shortcut.
**%Filename** : Filepath to a file that will be opened when a user clicks the shortcut.
**%Param** : Command line parameters. Optional.
**%WorkDir** : Working directory. Optional.
**%IconPath** : File path to a separate icon file, if needed. Optional.

## ADDICON_MENU_PROGRAM %Text, %Filename, %Param, %WorkDir, %IconPath

Adds a shortcut to the Program menu.

**%Text** : Specifies a description text for the shortcut.
**%Filename** : Filepath to a file that will be opened when a user clicks the shortcut.
**%Param** : Command line parameters. Optional.
**%WorkDir** : Working directory. Optional.
**%IconPath** : File path to a separate icon file, if needed. Optional.

## ADDICON_MENU_START %Text, %Filename, %Param, %WorkDir, %IconPath

Adds a shortcut to the Start menu.

**%Text** : Specifies a description text for the shortcut.
**%Filename** : Filepath to a file that will be opened when a user clicks the shortcut.
**%Param** : Command line parameters. Optional.
**%WorkDir** : Working directory. Optional.
**%IconPath** : File path to a separate icon file, if needed. Optional.

## ADDICON_MENU_STARTUP %Text, %Filename, %Param, %WorkDir, %IconPath

Adds a shortcut to the Startup menu.

**%Text** : Specifies a description text for the shortcut.
**%Filename** : Filepath to a file that will be opened when a user clicks the shortcut.
**%Param** : Command line parameters. Optional.
**%WorkDir** : Working directory. Optional.
**%IconPath** : File path to a separate icon file, if needed. Optional.

## ADDICON_DESKTOP %Text, %Filename, %Param, %WorkDir, %IconPath

Adds a shortcut to the desktop.

**%Text** : Specifies a description text for the shortcut.
**%Filename** : Filepath to a file that will be opened when a user clicks the shortcut.
**%Param** : Command line parameters. Optional.
**%WorkDir** : Working directory. Optional.
**%IconPath** : File path to a separate icon file, if needed. Optional.

### DELETEICON %MenuName, %Text
Deletes a shortcut from a menu.

**%MenuName** : Menu name.
**%Text** : Description text of the shortcut to delete.

### DELETEICON_DESKTOP %Text
Deletes a shortcut from the desktop.

**%Text** : Description text of the shortcut to delete.

### RENAMEICON %MenuName, %Old Text, %New Text
Renames a shortcut in a menu. The description text of the shortcut is changed to a new text.

**%MenuName** : Menu name.
**%Old Text** : Old description text.
**%New Text** : New description text.

### RENAMEICON_DESKTOP %Old Text, %New Text
Renames a shortcut on the desktop. The description text of the shortcut is changed to a new text.

**%Old Text** : Old description text.
**%New Text** : New description text.

**Script commands - Uninstallation**

**Script examples**

10.14 Script commands - Uninstallation

## Script commands - Uninstallation

*Files and folders*

### UNINSTALL_DELETE %Filename
Logs a file for deletion at uninstallation.

**%Filename** : File path to the file to delete at uninstallation.

### UNINSTALL_DELETE_AT_REBOOT %Filename
Logs a file for deletion at reboot during an uninstall process.

**%Filename** : File path to the file to delete at uninstallation.

### UNINSTALL_REMOVEDIR %Dir

Logs a folder for deletion at uninstallation.

**%Dir** : Folder path to the folder to delete at uninstallation.

## UNINSTALL_REMOVEDIR_AT_REBOOT %Dir

Logs a folder for deletion at reboot during an uninstall process.

**%Dir** : Folder path to the folder to delete at uninstallation.

## UNINSTALL_RESTORE %Filename, %Register

Logs a file to restore at uninstallation. During the installation, if this command is executed, a backup of the file is saved in the same folder as the original file. The backup will have the filename extension .BAK. During an uninstallation process, this file is restored to it's original filename. The **UNINSTALL_RESTORE** command should always be executed in the **Before installation** tab in the script window.

**%Filename** : File path to file to handle.
**%Register** : Flag if the file must be registered when it is restored. 1 = yes, 0 = no.

### *Icons*

## UNINSTALL_DELETEICON_STARTUP %Text

Logs a startup shortcut for deletion at uninstallation.

**%Text** : Description text of the shortcut to delete.

### *External installations and programs*

## UNINSTALL_MSIEXEC %Filename, %Param, %Wait

Logs an MSI installation/uninstallation to be run at uninstallation.

**%Filename** : Complete path to the MSI file, for example **%DESTDIR\MySetup.msi**.
**%Param** : Command line parameters to send to Windows Installer. Optional.
**%Wait** : Wait until the MSI installation is completed. Optional.

The **UNINSTALL_MSIEXEC** commands works as MSIEXEC but is run when the user does an uninstallation. This command can be be used to run both an installation and uninstallation. If you want to do an uninstallation you must add the **/x** parameter to **%Param**. For example:

```
UNINSTALL_MSIEXEC %DESTDIR\MySetup.msi, /qn /x
```

## UNINSTALL_XRUN %Filename, %Flag, %Param, %OS

Logs a program file to be run at uninstallation.

**%Filename** : Specifies a file path to the program file to run.
**%Flag** : Specifies how to start the program. Must be a value between 1 and 4. See XRUN for more information.
**%Param** : Command line parameters, if needed. Optional parameter.
**%OS** : Required operating system. Optional parameter. See XRUN for more information.

### *Other*

## UNINSTALL_RENAME_LOGFILE %Filename

The filename of the log file that contains uninstall information has normally the filename "**Vinstall.log**", but via this command you can give the log file another name. This can be useful if you install more than one product to the same destination folder, because your user can then uninstall the products separately if he/she wants.

**%Filename** : New filename for the log file.

**Script commands - Conditions**

**Script examples**

10.15 Script commands - Conditions (IF / ELSE / END IF)

# Script commands - Conditions (IF / ELSE / END IF)

It is possible to do conditional execution of script commands in Visual Installer. If the condition is true, the script lines that follow the condition will be executed. If the condition is false, the following script lines will be ignored. A condition starts with an IF and ends with an END IF. Below is an example of a condition:

```
IF OS=WIN10
   RUN %DESTDIR\MyWin10App.exe
END IF
```

In the example above, the script line "**RUN %DESTDIR\MyWin10App.exe**" will only be executed if the operating system where the setup program is running on is Windows 10. In other versions of Windows the line will be ignored.

## Available conditions (IF ...)

Click on a link below to get more information about a specific condition:

### Operating system

**Check operating system ( IF OS... )**
**Check bitness ( IF OSBIT... )**
**Check .NET version ( IF NET... )**

### Microsoft Office

**Check if Office installed ( IF OFFICEINSTALLED )**
**Check bitness of Office ( IF OFFICEBIT... )**

### Products

**Check if a product is installed ( IF INSTALLED... )**

### File components

**Check for component ( IF COMP... )**

### User options

**Check user option ( IF OPTION... )**

## How to end a condition (END IF)

To end script lines that are part of a condition, place END IF in a single line. Like this:

```
IF OS=WIN10
   RUN %DESTDIR\MyWin10App.exe
END IF
```

## 'Else' is also supported (ELSE)

Visual Installer's scripting language also supports *else*. If the IF condition gives false, script lines that are between ELSE and END IF can be executed instead. Below is an example of how to use ELSE:

```
IF OS=WIN10
   RUN %DESTDIR\MyWin10App.exe
ELSE
   RUN %DESTDIR\MyGeneralApp.exe
END IF
```

Script commands - Error handling

Script examples

10.15.1 Script commands - Conditions - Check operating system

# Script commands - Conditions - Check operating system

The conditions below check which version of Windows that the setup program is running on. Windows XP, Windows Vista, Windows 7, Windows 8 and Windows 10 are supported.

**Windows 10**

## IF OS=WIN10
Gives true if the operating system that the setup program is running on is Windows 10.

## IF OS>=WIN10
Gives true if the operating system that the setup program is running on is Windows 10 or newer.

## IF OS<WIN10
Gives true if the operating system that the setup program is running on is older than Windows 10.

**Windows 8**

## IF OS=WIN8
Gives true if the operating system that the setup program is running on is Windows 8.

## IF OS>=WIN8
Gives true if the operating system that the setup program is running on is Windows 8 or newer.

## IF OS<WIN8
Gives true if the operating system that the setup program is running on is older than Windows 8.

**Windows 7**

## IF OS=WIN7
Gives true if the operating system that the setup program is running on is Windows 7.

## IF OS>=WIN7
Gives true if the operating system that the setup program is running on is Windows 7 or newer.

## IF OS<WIN7
Gives true if the operating system that the setup program is running on is older than Windows 7.

**Windows Vista**

## IF OS=WINVISTA
Gives true if the operating system that the setup program is running on is Windows Vista.

## IF OS>=WINVISTA
Gives true if the operating system that the setup program is running on is Windows Vista or newer.

### IF OS<WINVISTA
Gives true if the operating system that the setup program is running on is older than Windows Vista.

**Windows XP**

### IF OS=WINXP
Gives true if the operating system that the setup program is running on is Windows XP.

### IF OS>=WINXP
Gives true if the operating system that the setup program is running on is Windows XP or newer.

### IF OS<WINXP
Gives true if the operating system that the setup program is running on is older than Windows XP.

**Example**

```
IF OS=WINVISTA
   RUN %DESTDIR\MyVistaApp.exe
END IF
```

10.15.2 Script commands - Conditions - Check bitness

## Script commands - Conditions - Check bitness

The conditions below check if the operating system is a 32 bit or a 64 bit system.

### IF OSBIT=32
Gives true if the operating system is a 32 bit system.

### IF OSBIT=64
Gives true if the operating system is a 64 bit system.

**Example**

```
IF OSBIT=64
   RUN %DESTDIR\App64.exe
END IF
```

10.15.3 Script commands - Conditions - Check .NET version

## Script commands - Conditions - Check .NET version

The conditions below check which versions of Microsoft .NET Framework that are installed in a computer. All versions from 1.1 to 4.8 are supported.

Note that version 4 and 4.5 of .NET does not include version 2.0, 3.0 or 3.5 of .NET. So a user may have for example version 4.5 installed, but not version 3.0.

**Test if installed**

### IF NET=1.1

Gives true if .NET version 1.1 is installed.

### IF NET=2.0
Gives true if .NET version 2.0 is installed.

### IF NET=3.0
Gives true if .NET version 3.0 is installed.

### IF NET=3.5
Gives true if .NET version 3.5 is installed.

### IF NET=4.0
Gives true if .NET version 4.0 is installed.

### IF NET=4.5
Gives true if .NET version 4.5 is installed.

### IF NET=4.5.1
Gives true if .NET version 4.5.1 is installed.

### IF NET=4.5.2
Gives true if .NET version 4.5.2 is installed.

### IF NET=4.6
Gives true if .NET version 4.6 is installed.

### IF NET=4.6.1
Gives true if .NET version 4.6.1 is installed.

### IF NET=4.6.2
Gives true if .NET version 4.6.2 is installed.

### IF NET=4.7
Gives true if .NET version 4.7 is installed.

### IF NET=4.7.1
Gives true if .NET version 4.7.1 is installed.

### IF NET=4.7.2
Gives true if .NET version 4.7.2 is installed.

### IF NET=4.8
Gives true if .NET version 4.8 is installed.

**Test if not installed**

### IF NET!=1.1
Gives true if .NET version 1.1 is not installed.

### IF NET!=2.0
Gives true if .NET version 2.0 is not installed.

### IF NET!=3.0
Gives true if .NET version 3.0 is not installed.

### IF NET!=3.5
Gives true if .NET version 3.5 is not installed.

### IF NET!=4.0
Gives true if .NET version 4.0 is not installed.

### IF NET!=4.5
Gives true if .NET version 4.5 is not installed.

### IF NET!=4.5.1
Gives true if .NET version 4.5.1 is not installed.

### IF NET!=4.5.2
Gives true if .NET version 4.5.2 is not installed.

### IF NET!=4.6
Gives true if .NET version 4.6 is not installed.

### IF NET!=4.6.1
Gives true if .NET version 4.6.1 is not installed.

### IF NET!=4.6.2
Gives true if .NET version 4.6.2 is not installed.

### IF NET!=4.7
Gives true if .NET version 4.7 is not installed.

### IF NET!=4.7.1
Gives true if .NET version 4.7.1 is not installed.

### IF NET!=4.7.2
Gives true if .NET version 4.7.2 is not installed.

### IF NET!=4.8
Gives true if .NET version 4.8 is not installed.

#### Example

```
IF NET!=4.0
   MSGBOX Information, You must install .NET 4.0 before you can run this app. Press OK to continue., 2
   RUN \DOTNET\dotNetFx40_Full_setup.exe
END IF
```

10.15.4 Script commands - Conditions - Check if Office installed

## Script commands - Conditions - Check if Office installed

The conditions below check if Microsoft Office is installed in the system.

### IF OFFICEINSTALLED
Gives true if Microsoft Office is installed in the system.

### IF NOT OFFICEINSTALLED
Gives true if Microsoft Office is not installed in the system.

#### Example

```
IF OFFICEINSTALLED
   RUN %DESTDIR\InstallOfficeTools.exe
END IF
```

### 10.15.5 Script commands - Conditions - Check bitness of Office

## Script commands - Conditions - Check bitness of Office

The conditions below check if the installed Microsoft Office is 32 bit or 64 bit.

### IF OFFICEBIT=32
Gives true if the installed Microsoft Office is a 32 bit.

### IF OFFICEBIT=64
Gives true if the installed Microsoft Office is a 64 bit.

#### Example

```
IF OFFICEBIT=64
   RUN %DESTDIR\InstallExcel64Tools.exe
END IF
```

### 10.15.6 Script commands - Conditions - Check if a product is installed

## Script commands - Conditions - Check if a product is installed

The conditions below check if a product with a specified product name or product code has been installed or not.

### IF INSTALLED(name)
Gives true if a product with the name *name* has been installed in the system. *name* can be a product name or a product code. If *name* is a product code, you must surround it with braces (example: "{4815BD99-96A4-49FE-A885-DCF06E9E4E78}").

### IF NOT INSTALLED(name)
As above, but gives true if a product with the name *name* has not been installed.

#### Examples
This condition is useful if you need to check if a product has already been installed or not, and only run an installation if the product has not been installed. Below we show two examples of how to use the NOT INSTALLED condition:

```
// Example 1 - check for product name
IF NOT INSTALLED("My Application")
   MSIEXEC %DESTDIR\MySetup.msi, /qn
END IF

// Example 2 - check for product code
IF NOT INSTALLED("{4815BD99-96A4-49FE-A885-DCF06E9E4E78}")
   MSIEXEC %DESTDIR\MySetup.msi, /qn
END IF
```

10.15.7 Script commands - Conditions - Check for component

# Script commands - Conditions - Check for component

The conditions below check if the user has selected to install a specific file component (file group). A file component is a group of files that are installed together, and are handled via the **Components** tab in Visual Installer's editor.

## IF COMP(n)=ON
Gives true if component number *n* (where *n* = 1..9) has been selected for installation.

## IF COMP(n)=OFF
Gives true if component number *n* (where *n* = 1..9) has not been selected for installation.

### Example

```
IF COMP(2)=ON
   COPY %DESTDIR\App1.exe, %DESTDIR\Binary\App1.exe
   COPY %DESTDIR\App2.exe, %DESTDIR\Binary\App2.exe
END IF
```

10.15.8 Script commands - Conditions - Check user option

# Script commands - Conditions - Check user option

The conditions below check which options that are selected (checked) in the setup dialog box with user options.

## IF OPTION(n)=ON
Gives true if option number *n* (where *n* = 1..9) is selected in the setup dialog box with user options.

## IF OPTION(n)=OFF
Gives true if option number *n* (where *n* = 1..9) is not selected in the setup dialog box with user options.

### Example

```
// Copies the two files App1.exe and App2.exe if option 2 in the setup dialog box is selected
IF OPTION(2)=ON
   COPY %DESTDIR\App1.exe, %DESTDIR\Binary\App1.exe
   COPY %DESTDIR\App2.exe, %DESTDIR\Binary\App2.exe
END IF
```

10.16 Script commands - Error handling

# Script commands - Error handling

## SHOWERROR %Mode
Specifies if an error information dialog box should be shown at errors in the script. As default this option is

turned off, but via this command you can turn the option on. The following values can be used for the parameter **%Mode**:

**%Mode** : **1** = show error information dialog box, **0** = don't show error information dialog box.

### HANDLEERROR %Mode

Specifies how to handle errors in the script. The following values can be used for the parameter **%Mode**:

**0 :** Ignore errors. This is the default setting.
**1 :** Cancel script processing at errors.
**2 :** Cancel the installation if errors occurs in the script.
**3 :** Warn the user if an error occurred. The user has also the possibility to cancel the installation.

This command does not handle syntax errors etc. Only errors that are reported by Windows are handled.

### USELOG %Mode, %Dir

If you want to obtain detailed information about the execution of the script lines, you can activate a log that logs all activity when the script lines are executed. Via the log you can for example see what parameter values are passed to the script commands and see the return values of the script commands. If you use variables in the script lines, they will be expanded to their real values.

**%Mode** : **1** = Start logging, **0** = Stop logging.
**%Dir** : Specifys a folder path to the log file. Optional parameter.

The **%Dir** parameter is optional. If you omit this parameter, the log file is stored in the main destination folder (in the **%DESTDIR** folder).

The filename of the log file is **Vinstall-Script--Before.log** if the script lines are executed in the **Before installation** tab in the **Execute script commands** window and the filename is **Vinstall-Script--After.log** if the script lines are executed in the **After installation** tab in the **Execute script commands** window. The filenames can not be changed, but you can change the folder where the log files are stored (via the **%Dir** parameter).

**Script commands - Miscellaneous**

**Script examples**

10.17 Script commands - Miscellaneous

## Script commands - Miscellaneous

### SLEEP %milliseconds

Pauses the installation for a specified amount of time (milliseconds).

**%milliseconds** : How many milliseconds to wait. Max value is 30000 (= 30 sconds).

### EXIT

Stops the installation and closes the setup program.

### EXIT_SCRIPT

Stops the execution of scripts but does not stop the installation.

### MESSAGE %text

Displays an information message on the progress bar window.

**%text** : Text to display.

### MSGBOX %Title, %Text, %Symbol

Shows a message box with a title, text message and a **OK** button, and pauses the installation. The installation will continue when the user presses the OK button.

**%Title** : Title for the message box.
**%Text** : Text for the message box.
**%Symbol** : Icon (symbol) to show in the message box:
**1**=information icon, **2**=exclamation icon, **3**=error icon

### //

If you start a line with two slashes ("**//**") the line will be regarded as a comment line. The text to the right of the two slashes will not be processed.

**Script examples**

10.18 Script examples

## Script examples

Below you will some examples of usage with Visual Installer's script commands.

### Example 1 - Copy a file

Copies the file **App.exe** from one location to another.

```
COPY %DESTDIR\App.exe, %DESTDIR\Binary\App.exe
```

### Example 2 - Copy many files at once

Copies all files in the **%DESTDIR\Data** folder to **%DESTDIR\NewPlace**.

```
MCOPY %DESTDIR\Data\*.*, %DESTDIR\NewPlace
```

### Example 3 - Delete a file

Deletes the file **Settings.ini**.

```
DELETE %DESTDIR\Settings.ini
```

### Example 4 - Delete a folder

Removes the folder **%DESTDIR\Temp** from the hard disk.

```
XREMOVEDIR %DESTDIR\Temp
```

### Example 5 - Delete a folder tree

Deletes all files and folders in the **%DESTDIR\Alfa** folder.

```
DELTREE %DESTDIR\Alfa
```

### Example 6 - Run a program

Runs a program with the filename "**App.exe**". The program is located in the main destination folder for the setup.

```
RUN %DESTDIR\App.exe
```

## Example 7 - Run an MSI installation

Displays an information message on the progress bar and installs **MYSetup.msi**. Visual Installer pauses its own installation until the installation of **MYSetup.msi** is completed. The installation is run silently.

```
MESSAGE My MSI installation is running
MSIEXEC %DESTDIR\MySetup.msi, /qn, WAIT
```

## Example 8 - Create a menu and add shortcuts

Creates a menu with the name "**Office Apps**" and adds two shortcuts ("**Calulator**" and "**Notepad**") to the menu.

```
CREATEMENU Office Apps
ADDICON_MENU Office Apps, Calculator, %DESTDIR\Calculat.exe
ADDICON_MENU Office Apps, Notepad, %DESTDIR\Editor.exe
```

## Example 9 - Create a menu and sub menu

Creates a menu with the name "**Office Apps**" and a sub menu with the name "**Documents**".

```
CREATEMENU Office Apps
CREATEMENU Office Apps\Documents
```

## Example 10 - Add a shortcut to the Startup menu

Adds a shortcut with the description text "**My Program**" to the **Startup** menu in Windows.

```
ADDICON_MENU_STARTUP Mitt program, %DESTDIR\MyApp.exe
```

## Example 11 - Register a DLL

Registers a DLL with the filename **MyLib.dll**.

```
SELFREGISTER %SYSDIR\MyLib.dll
```

## Example 12 - Error information dialog box

Turns on handling of error information dialog boxes.

```
SHOWERROR 1
```

## Example 13 - Remove files and folder at uninstallation

Shows how to log a file and folder for deletion at uninstallation.

```
UNINSTALL_DELETE %DESTDIR\Docs\Readme.txt
UNINSTALL_REMOVEDIR %DESTDIR\Docs
```

## Example 14 - Command interpreter

The example below executes the DIR command in Windows (and lists all files in the current folder).

```
CMD DIR *.*
```

## Example 15 - Conditions - OS  test

The example below shows how to only execute a script line if the setup program is run in Windows 7.

```
IF OS=WIN7
   RUN %DESTDIR\MyWin7App.exe
```

```
END IF
```

## Example 16 - Conditions - .NET test

The example below shows how to start an installation of .NET Framework if the requested version is not installed in Windows.

```
IF NET!=2.0
   MSGBOX Information, You must install .NET 2.0 in this computer before you can run this program. Press OK to start an installation of .NET 2.0., 2
   RUN \DOTNET\dotnetfx.exe
END IF
```

## Example 17 - Conditions - How to use ELSE

Visual Installer's scripting language also supports *else*. If the IF condition gives false, script lines that are between ELSE and END IF can be executed instead. Below is an example of how to use ELSE. If the operating system is Windows 10, the first RUN command (RUN %DESTDIR\MyWin10App.exe) will be executed. If the operating system is something else, the second RUN command (RUN %DESTDIR\MyGeneralApp.exe) will be executed.

```
IF OS=WIN10
   RUN %DESTDIR\MyWin10App.exe
ELSE
   RUN %DESTDIR\MyGeneralApp.exe
END IF
```

## Example 18 - Comments

If you want to add many script lines it can be a good idea to also add comments. A line with a comment always starts with two slashes (//). Example:

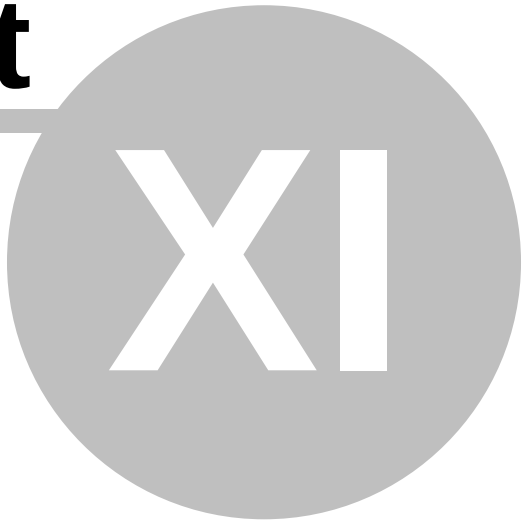
```
// Copy files to the system directory
COPY %DESTDIR\Modules\Module1.dll, %SYSDIR\MyModule1.dll
COPY %DESTDIR\Modules\Module2.dll, %SYSDIR\MyModule2.dll

// Delete old INI file
DELETE %WINDIR\MyApp.ini
```

## More examples

More examples are available on our tips & tricks pages for Visual Installer (see the **Advanced** section):

http://www.samlogic.net/visual-installer/tips/index-tips-tricks-visual-installer.htm

# Part



# More Details and Articles

In this chapter you will find some articles and more detailed information about some functions in the program.

## 11.1 Command Line Parameters

# Command Line Parameters

Visual Installer's editor (VI.exe) supports some command line parameters. With a command line parameter you can for example open Visual Installer with a specific VIP project file loaded from start or open Visual Installer in a special (silent) build mode where a setup package is created without any user interaction.

The following command line parameters are supported:

**VI.exe** ***<project file>*** **/BUILD /BUILDFOLDER:*<folder>* /LOG:*<log file>* /INTERNET** or **/CD** or **/USB**

Below are all parameters explained in detail:

| | | |
|---|---|---|
| ***<project file>*** | : | A complete file path to a project file (.VIP file). If the file path contain space characters you should enclose the path with quotes. |
| **/BUILD** | : | If this parameter is specified Visual Installer will operate in a special (silent) build mode. |
| **/BUILDFOLDER:** ***<folder>*** | : | If you want to build the setup package in another folder than specified in the VIP file, you can specify the folder path here. |
| **/LOG:*<log file>*** | : | Errors during build mode can be stored in a log file that you specify here. |
| **/INTERNET** | : | If you want to change media type from current project setting to Internet you can specify this parameter. |
| **/CD** | : | If you want to change media type from current project setting to CD/DVD you can specify this parameter. |
| **/USB** | : | If you want to change media type from current project setting to USB stick you can specify this parameter. |

All folder and file paths that you specify must be complete, including the drive letter (e.g. "C:\MyProjects\MySetup.vip"). The /BUILDFOLDER, /LOG, /INTERNET, /CD and /USB parameters are only supported if you also specify the /BUILD parameter, otherwise these parameters are ignored. All parameter names (switches) must be entered in uppercase (e.g. "/BUILD"). File and folder paths can be entered in any case.

The order of the parameters are also important. They must be entered in the order shown above. For example, the first parameter must be the file path to the project file (.VIP file), the second parameter must always be /BUILD, the third parameter must be /BUILDFOLDER etc.

### More information about the /BUILD parameter

If you specify the /BUILD parameter in the command line, Visual Installer will be run in a silent build mode. Visual Installer editor's user interface will be not be shown and no user interaction is possible. The build mode is very useful if you want to include Visual Installer in batch build processes. When the /BUILD mode parameter is specified, Visual Installer will open the project file, build the setup package and return automatically when finished.

Error can be handled by using the /LOG parameter. If an error occurs during the build mode, VI.exe will

return the exit code 1 instead of 0 (VI.exe returns always 0 at success), and a detailed error message is written to the log file.

### Examples

Below are some examples of how to use Visual Installer's command line parameters:

```
VI.exe  C:\MyProjects\MySetup.vip

VI.exe  C:\MyProjects\MySetup.vip  /BUILD

VI.exe  C:\MyProjects\MySetup.vip  /BUILD  /BUILDFOLDER:C:\MyCreatedSetups\Setup1

VI.exe  C:\MyProjects\MySetup.vip  /BUILD  /BUILDFOLDER:C:\MyCreatedSetups\Setup1
/LOG:C:\MyLogs\Err.log

VI.exe  C:\MyProjects\MySetup.vip  /BUILD  /BUILDFOLDER:C:\MyCreatedSetups\Setup1
/LOG:C:\MyLogs\Err.log  /INTERNET
```

### More information

More information about the command line parameters is available on this web page on our web site:

Tip: Visual Installer's command line parameters

11.2 Copy settings (the 'Inst' column)

## Copy settings (the 'Inst' column)

In the file list table (in the **File list** tab) there is column with the name **Inst**. This column shows copy settings for every file in the file list. A special character shows which copy setting that are set to every file. Below is a list with available copy settings:

**D** : Check file date before copying the file.
**V** : Check version number before copying the file.
**N** : Never replace an existsing file.
**W** : Replace also a write-protected file.
**X** : Warn if the file already exists on the hard disk.
**A** : If the file is in use (active) and locked, replace it when the computer is restarted.
**B** : Create a backup of the file before it is replaced.
**R** : Register the file in the system.
**P** : Write-protect the file after installation.
**E** : If the version number of the source and destination file is equal, do not copy.
***** : Never uninstall this file.

You can change a file's copy settings by choosing the menu item **List - File copy options**.

**Dialog Box - Copy options for file**

11.3 DEP files

## DEP files

A **DEP** file (dependency file) is a file that contains information about which other files a specific file is dependent of. In Visual Installer you can open a specific DEP file, and the Visual Installer editor will then add all files that are referenced in the DEP file. You can open a DEP file via the menu item **List - Add files via**

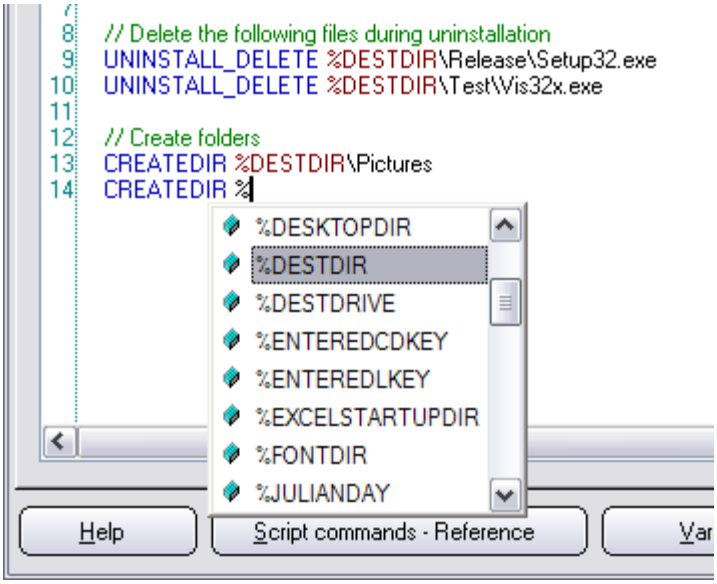
**DEP file**.

## 11.4 IntelliSense

# IntelliSense

Visual Installer supports **IntelliSense** in some parts of the editor. IntelliSense means that a list of available commands and variables can be shown when needed, and you can choose a command or variable in an easy way from the list.

You can open the list with commands and variables by pressing **Ctrl - Space bar** on your keyboard. And to choose a particular command or variable in the list, you can press the space bar. You can also choose a command or variable with the mouse button.




If you know the first letter in the command you can enter the letter before you press **Ctrl - Space bar**. By pressing a letter, only commands start starts with that letter will be shown. All variables always starts with a **%** character, so if you press the **%** character, only variables will be shown in the list.

The following parts of the Visual Installer editor support IntelliSense:

- The **Registry** tab
- The **INI files** tab
- The script editor (that is opened via the menu item **Special - Execute script commands**)
- The **Register files** dialog box (that is opened via the menu item **List - Register files**)

## 11.5 Language files

# Language files

A language file is a text file that contains general texts and messages that will be shown during the setup process. You can open a language file in a text editor, for example in Notepad or Microsoft Word, and you can also edit the contents using the text editor.

### Same format as INI files

A language file is formatted in the same manner as an INI file in Windows. The language file contains

sections, value names and values. Below we show how a section is formatted:

```
[Section]
Object1 = Text for object 1
Object2 = Text for object 2
```

The example below shows how a section that contains texts for the **Destination Folder** dialog box look like:

```
[Dialog_DestinationDir]
Title = Destination Folder
Text = Please choose destination folder for your program. This is the directory where the program and its
support files are installed to.
FieldTitle = &Folder:
GroupSizeInfo = Space
SpaceReq = Required:
SpaceFree = Free:
```

If you want to change the text for an object in the dialog box, you can change the text to the right of the equal sign. The text to the left of the equal sign must never be altered.

### Special characters that can be used in texts

A **&** character in a text will underline the character to the right of the **&** character. This character tells Visual Installer that the underlined character will be a keyboard shortcut for jumping to the specific dialog box control. If the user presses **ALT** and the character, the control will get the focus, or if it is a button, the button will be pressed. For example, if you specify the text "**&Folder**", the control will get the focus when the user presses **ALT F**.

Other special characters that can be used is the character combination **%s**. When Visual Installer encounters these characters in a string, **%s** will be replaced with a text string which contents depends of the circumstances. For example if a file that should be installed already exists and the settings are that Visual Installer must warn, the following text string is shown in a message box:

```
Msg_FileExists = The file '%s' already exists! Do you want to replace it?
```

The **%s** characters will be replaced with the actual name of the file. For example like this:

**The file 'Readme.doc' already exists! Do you want to replace it?**

### Where are langauge files located?

A language file has normally the filename extension **.LNG** and they are normally located in the following folder in the C: drive:

| | |
|---|---|
| In Windows XP (or older): | **\Program Files\SamLogic\Visual Installer\Language** |
| In Windows Vista: | **\Document\Visual Installer\Languages Files** |
| In Windows 7: | **\My Document\Visual Installer\Languages Files** |
| In Windows 8: | **\My Document\Visual Installer\Languages Files** |
| In Windows 10: | **\My Document\Visual Installer\Languages Files** |

## 11.6 Microsoft .NET Framework

# Microsoft .NET Framework

Some programs requires that Microsoft .NET Framework is installed in the computer to run. It is also important that the correct version of .NET Framework is installed. If the installed version of .NET Framework is too old, the program may not run.

On Microsoft's web site you can read how to install a specific version of .NET Framework. Click on the links below to get more information:

**.NET Framework 1.1 Deployment Guide**
http://msdn2.microsoft.com/en-us/library/ms994339.aspx

**.NET Framework 2.0 Deployment Guide**

http://msdn2.microsoft.com/en-us/library/aa480237.aspx

**.NET Framework 3.0 Deployment Guide**
http://msdn.microsoft.com/en-us/library/aa480173.aspx

**.NET Framework 3.5 Deployment Guide**
http://msdn.microsoft.com/en-us/library/cc160716.aspx

**.NET Framework 4.0 Deployment Guide**
http://msdn.microsoft.com/en-us/library/ee942965(v=vs.100).aspx

**.NET Framework 4.5 Deployment Guide**
http://msdn.microsoft.com/en-us/library/ee942965(v=vs.110).aspx

11.7 More information about license keys

# More information about license keys

A license key, also called an installation key, is a kind of password that the user must enter in the setup program to be able to install the files. This is a kind of basic copy protection that assure that nobody that do not have a license key can install the files. If you distribute many programs using a CD, DVD or USB flash drive you can use the license key to assure that the user can install the software that he/she has bought a license for.

The license key that the user enters is stored in the variable %ENTEREDLKEY. For backward compatibility reasons the license key is also stored in the %ENTEREDCDKEY variable.

11.8 More information about main folders

# More information about main folders

The main folder for the setup package is specified in the **Main folder** text box in the **File list** tab. This folder can be changed by the end-user, if you show the **Destination folder** dialog box in the setup program. The main folder is stored in the %DESTDIR variable, and can be used in various parts of Visual Installer. It is common to use the %DESTDIR variable as a part of a destination folder for a file.

### The %PROGRAMFILES variable

There is a special variable that can be used with the main folder. The name of the variable is %PROGRAMFILES, and this variable will always contain the folder path to the **Program Files** folder in Windows, usually: **C:\Program Files**. If you are going to install files to the **Program Files** folder, we recommend you to use this variable. Below we show an example of how the variable can be used:

%PROGRAMFILES\SamLogic\Visual Installer

By using the %PROGRAMFILES variable, you also assure that files are installed in folders with correct bitness in 64 bit systems. In 64 bit systems there exists two **Program Files** folder (one for 32 bit files and one for 64 bit files), but Visual Installer can decide which folder is the correct one if you use the %PROGRAMFILES variable.

### The %MYDOCUMENTS and %MYPICTURES variables

Two other variables that can be used with the main folder are the %MYDOCUMENTS and %MYPICTURES variables. They will contain the folder path to the **My Documents** folder and the **My Pictures** folder in the system.

### Other variables

Other variables that can be used with the main folder are the %APPDATADIR and

%APPDATADIR_ALLUSERS variables. We don't recommend to use these variables with the main folder, but if it is absolutely necessary it is possible. Also the the %REG1 - %REG5 variables can be used with the main folder.

**Variables**

11.9 SHA-1 or SHA-2 (which hash algorithm to use)

## SHA-1 or SHA-2 (which hash algorithm to use)

When you code sign an installation you can choose which hash algorithm to use. You can choose to use SHA-1 or SHA-2, or both. We recommend you to choose *both* (the **Use SHA-1 and SHA-2** option in the **Setup options** dialog box) if your software sometimes will be installed in older Windows, like Windows XP. If the minimum system requirements for your software is Windows 7, you can choose SHA-2.

It is not recommended to choose only SHA-1 because newer Windows (like Windows 10) does not consider this hash algorithm safe anymore, and extra warning messages may be shown in for example Windows 10, when the installation starts or when the software is downloaded. If you have a Visual Installer project file that uses the SHA-1 setting, we recommend you to switch to the **Use SHA-1 and SHA-2** setting instead.

When the **Use SHA-1 and SHA-2** setting is chosen, Visual Installer will first code sign the setup package using the SHA-1 algorithm, thereafter Visual Installer will code sign the setup package using the SHA-2 algorithm.

**More information**
Additional information about SHA-1 and SHA-2, and which algorithm to choose, is available in this article on our web site:

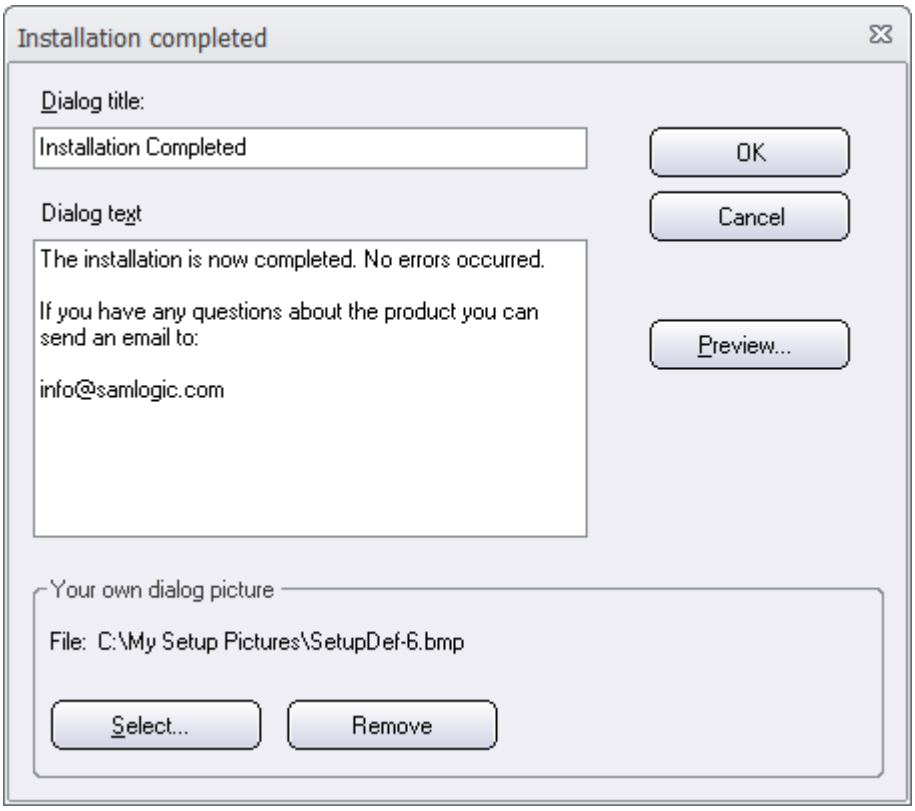
**SHA-1 and SHA-2 hash algorithms**

11.10 Using an e-mail address and web address in a dialog box

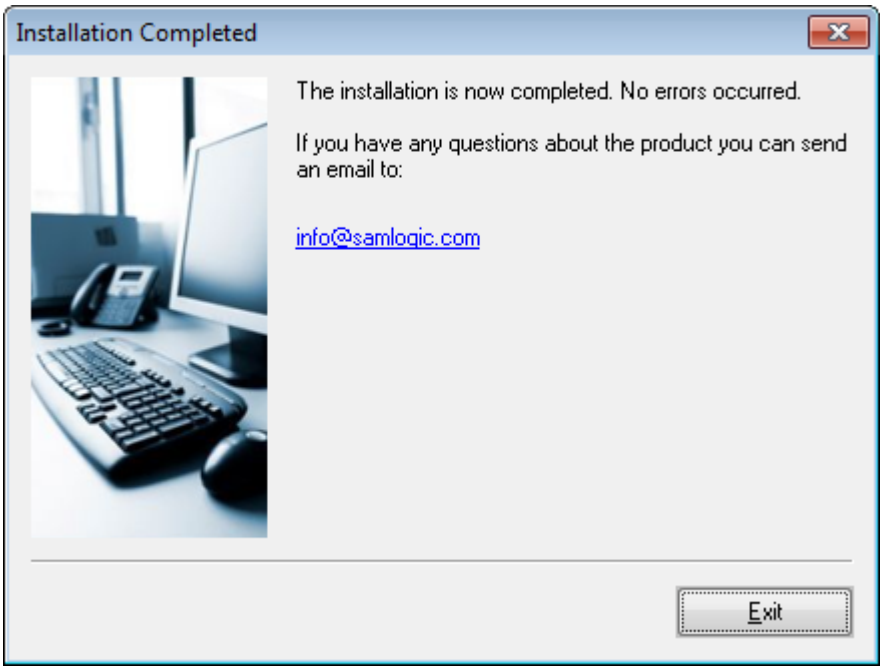
## Using an e-mail address and web address in a dialog box

It is possible to include a clickable e-mail address and web address in many of the setup dialog boxes in Visual Installer. When the user clicks on the e-mail address or web address, the user's e-mail client or web browser is opened, with the e-mail address pre-filled or the web page opened.

The following setup dialog boxes support a clickable e-mail address and web address: **Welcome**, **General information**, **General information 2**, **Start installation** and **Installation completed**. When you want to add an e-mail address or web address to one of these mentioned setup dialog boxes, you need to insert the e-mail address / web address last in the text (or on a separate line) in the text that you enter in the **Dialog text** multiline editor. Example:




During the installation, the setup dialog box will look like:




As you can see, the e-mail address that you entered has been converted to a clickable link.

11.11 Why register a file?

# Why register a file?

Some components and DLL files require that they are registered before they are used. If these kind of file are not registered, they will not function as expected. When a file is registered, some important information will be stored in the Windows Registry.

In Visual Installer you can inform the setup program to register a file via the **Copy options for file** dialog box (check the **Register file** option in this dialog box). You can also use the **SELFREGISTER** script command or open the **Register files** dialog box (via the menu item **List - Register files**).

11.12 Windows Vista / Windows 7

# Windows Vista / Windows 7

**Windows Vista** and **Windows 7** functions very different compared to Windows XP and earlier, and there are some important things to consider when creating setup packages that should be able to install properly in these operating systems. In this technical article on our web site, you can get more information about installation programs in Windows Vista and Windows 7:

**Creating an Installation Program for Windows Vista and Windows 7**

# Part XII

12 Miscellaneous

# Miscellaneous

In this chapter you find system requirements for Visual Installer and information about how to get support for the program.

## 12.1 Support

# Support

If you have any problems with Visual Installer you can send an e-mail to **support@samlogic.com**.

In the e-mail, try to give detailed information about your problem. And if possible or suitable, take screen dumps when the problem occurs and attach the screen dump with your e-mail. If the screens dumps are in .BMP format, try to compress them using a ZIP tool.

## 12.2 System Requirements

# System Requirements

### Operating Systems

The Visual Installer 2020 editor can be run in the following operating systems:

**- Windows 2000**
**- Windows XP**
**- Windows Vista**
**- Windows 7**
**- Windows 8 and 8.1**
**- Windows 10**

The created setup programs can be run in the following operating systems:

- **Windows 98**
- **Windows NT 4.0, SP3** or later
- **Windows 2000**
- **Windows XP**
**- Windows Vista**
**- Windows 7**
**- Windows 8 and 8.1**
**- Windows 10**
- **Windows Server 2003-2019**

### 32/64 bit

SamLogic Visual Installer 2020 (both the editor and the created setup programs) can be run on both 32-bit Windows and 64-bit Windows.

### Graphic card

The graphic card must support at least 256 colors.

### Memory

- Editor: 512 MB of free RAM

- Setup program: 256 MB of free RAM

# Part XIII

# How To Use The Online Help

When you need help in a dialog box or tab you simply press the **F1** key on your keyboard. In some cases you can also press a **Help** or **?** button in the dialog box or window.

# Part XIV

# About Visual Installer 2020

**SamLogic Visual Installer 2020** version **11.8**, Copyright © by **SamLogic**

Created and programmed by:
***Mika Larramo***

Documentation:
***Mika Larramo***

Administration:
***Anders Persson***

**SamLogic**
**Box 102**
**S-135 23 TYRESO**
**SWEDEN**

**Phone:** **+46 8 531 83 900**
**Fax:** **+46 8 531 88 403**

**e-mail:** **info@samlogic.com**
**Internet:** **www.samlogic.net**