[Previous](http://nsis.sourceforge.net/Docs/Chapter3.html) | [Contents](http://nsis.sourceforge.net/Docs/Contents.html) | [Next](http://nsis.sourceforge.net/Docs/Chapter5.html)

* [**Chapter 4:**Scripting Reference](http://nsis.sourceforge.net/Docs/Chapter4.html)
  + [Script File Format](http://nsis.sourceforge.net/Docs/Chapter4.html#fileformat)
  + [Variables](http://nsis.sourceforge.net/Docs/Chapter4.html#variables)
    - [User Variables](http://nsis.sourceforge.net/Docs/Chapter4.html#varuser)
    - [Other Writable Variables](http://nsis.sourceforge.net/Docs/Chapter4.html#varother)
    - [Constants](http://nsis.sourceforge.net/Docs/Chapter4.html#varconstant)
    - [Constants Used in Strings](http://nsis.sourceforge.net/Docs/Chapter4.html#varstrings)
  + [Labels](http://nsis.sourceforge.net/Docs/Chapter4.html#labels)
  + [Relative Jumps](http://nsis.sourceforge.net/Docs/Chapter4.html#jumps)
  + [Pages](http://nsis.sourceforge.net/Docs/Chapter4.html#pages)
    - [Ordering](http://nsis.sourceforge.net/Docs/Chapter4.html#pageoreder)
    - [Page Options](http://nsis.sourceforge.net/Docs/Chapter4.html#pageoptions)
    - [Callbacks](http://nsis.sourceforge.net/Docs/Chapter4.html#pagecallbacks_explain)
    - [Page](http://nsis.sourceforge.net/Docs/Chapter4.html#page)
    - [UninstPage](http://nsis.sourceforge.net/Docs/Chapter4.html#uninstpage)
    - [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex)
    - [PageExEnd](http://nsis.sourceforge.net/Docs/Chapter4.html#pageexend)
    - [PageCallbacks](http://nsis.sourceforge.net/Docs/Chapter4.html#pagecallbacks)
  + [Sections](http://nsis.sourceforge.net/Docs/Chapter4.html#sections)
    - [Section Commands](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectioncommands)
    - [Uninstall Section](http://nsis.sourceforge.net/Docs/Chapter4.html#UninstallSection)
  + [Functions](http://nsis.sourceforge.net/Docs/Chapter4.html#functions)
    - [Function Commands](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunctioncommands)
    - [Callback Functions](http://nsis.sourceforge.net/Docs/Chapter4.html#callbacks)
  + [Installer Attributes](http://nsis.sourceforge.net/Docs/Chapter4.html#instattribs)
    - [General Attributes](http://nsis.sourceforge.net/Docs/Chapter4.html#attribgen)
    - [Compiler Flags](http://nsis.sourceforge.net/Docs/Chapter4.html#flags)
    - [Version Information](http://nsis.sourceforge.net/Docs/Chapter4.html#versioninfo)
  + [Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#instr)
    - [Basic Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#basicinstructions)
    - [Registry, INI, File Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#registry)
    - [General Purpose Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#generalpurpose)
    - [Flow Control Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#flowcontrol)
    - [File Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#fileinst)
    - [Uninstaller Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#uninst)
    - [Miscellaneous Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#miscinst)
    - [String Manipulation Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#stringinst)
    - [Stack Support](http://nsis.sourceforge.net/Docs/Chapter4.html#stackinst)
    - [Integer Support](http://nsis.sourceforge.net/Docs/Chapter4.html#intinst)
    - [Reboot Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#rebootinst)
    - [Install Logging Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#installlog)
    - [Section Management](http://nsis.sourceforge.net/Docs/Chapter4.html#secmanage)
    - [User Interface Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#ui)
    - [Multiple Languages Instructions](http://nsis.sourceforge.net/Docs/Chapter4.html#langsinstructions)
  + [Multiple Languages](http://nsis.sourceforge.net/Docs/Chapter4.html#langs)
    - [Language Selection](http://nsis.sourceforge.net/Docs/Chapter4.html#langselection)
    - [LangDLL Plug-in](http://nsis.sourceforge.net/Docs/Chapter4.html#langdll)
    - [RTL Languages](http://nsis.sourceforge.net/Docs/Chapter4.html#rtl)
  + [Plug-in DLLs](http://nsis.sourceforge.net/Docs/Chapter4.html#plugindlls)
    - [Using Plug-in Commands](http://nsis.sourceforge.net/Docs/Chapter4.html#usingplug)
    - [Calling plug-ins manually](http://nsis.sourceforge.net/Docs/Chapter4.html#calldiskplug)
  + [Silent Installers/Uninstallers](http://nsis.sourceforge.net/Docs/Chapter4.html#silent)

**Chapter 4: Scripting Reference**

**4.1 Script File Format**

A NSIS Script File (.nsi) is just a text file with script code.

**Commands**

Commands lines are in the format 'command [parameters]'

File "myfile"

**Comments**

Lines beginning with ; or # are comments. You can put comments after commands. You can also use C-style comments to comment one or more lines.

; Comment

# Comment

# Comment \

Another comment line (see `Long commands` section below)

/\*

Comment

Comment

\*/

Name /\* comment \*/ mysetup

File "myfile" ; Comment

If you want a parameter to start with ; or # put it in quotes.

**Plug-ins**

To call a plug-in, use 'plugin::command [parameters]'. For more info see [Plug-in DLLs](http://nsis.sourceforge.net/Docs/Chapter4.html#plugindlls).

nsExec::Exec "myfile"

**Numbers**

For parameters that are treated as numbers, use decimal (the number) or hexadecimal (with 0x prepended to it, i.e. 0x12345AB), or octal (numbers beginning with a 0 and no x).

Colors should be set in hexadecimal RGB format, like HTML but without the #.

IntCmp 1 0x1 lbl\_equal

SetCtlColors $HWND CCCCCC

**Strings**

To represent strings that have spaces, use quotes:

MessageBox MB\_OK "Hi there!"

Quotes only have the property of containing a parameter if they begin the parameter. They can be either single quotes, double quotes, or the backward single quote.

You can escape quotes using $\:

MessageBox MB\_OK "I'll be happy" ; this one puts a ' inside a string

MessageBox MB\_OK 'And he said to me "Hi there!"' ; this one puts a " inside a string

MessageBox MB\_OK `And he said to me "I'll be happy!"` ; this one puts both ' and "s inside a string

MessageBox MB\_OK "$\"A quote from a wise man$\" said the wise man" ; this one shows escaping of quotes

It is also possible to put newlines, tabs etc. in a string using $\r, $\n, $\t etc. [More information...](http://nsis.sourceforge.net/Docs/Chapter4.html#varstrings)

**Variables**

Variables start with $. User variables should be declared.

Var MYVAR

StrCpy $MYVAR "myvalue"

[More information...](http://nsis.sourceforge.net/Docs/Chapter4.html#variables)

**Long commands**

To extend a command over multiple lines, use a backslash (\) at the end of the line. The next line will effectively be concatenated to the end of it. For example:

CreateShortcut "$SMPROGRAMS\NSIS\ZIP2EXE project workspace.lnk" \

"$INSTDIR\source\zip2exe\zip2exe.dsw"

MessageBox MB\_YESNO|MB\_ICONQUESTION \

"Do you want to remove all files in the folder? \

(If you have anything you created that you want \

to keep, click No)" \

IDNO NoRemoveLabel

Line extension for long commands works for comments as well. It can be a bit confusing, so it should be avoided.

# A comment \

still a comment here...

**Configuration file**

If a file named "nsisconf.nsh" in the config directory exists, it will be included by default before any scripts (unless the /NOCONFIG command line parameter is used). The config directory on Windows is the same directory as makensis.exe is in. On other platforms this is set at install time and defaults to $PREFIX/etc/. You can alter this at runtime, see [section 3.1.3](http://nsis.sourceforge.net/Docs/Chapter3.html#usageenvironment) for more information.

**4.2 Variables**

All variables are global and can be used in Sections or Functions. Note that, by default, variables are limited to 1024 characters. To extend this limit, [build NSIS](http://nsis.sourceforge.net/Docs/AppendixG.html#build) with a bigger value of the NSIS\_MAX\_STRLEN build setting or use the [special build](http://nsis.sourceforge.net/download/specialbuilds/).

**4.2.1 User Variables**

*$VARNAME*

User variables can be declared with the [Var](http://nsis.sourceforge.net/Docs/Chapter4.html#var) command. You can use these variables to store values, work with string manipulation etc.

**4.2.1.1 Var**

[/GLOBAL] var\_name

Declare a user variable. Allowed characters for variables names: [a-z][A-Z][0-9] and '\_'. All defined variables are global, even if defined in a section or a function. To make this clear, variables defined in a section or a function must use the /GLOBAL flag. The /GLOBAL flag is not required outside of sections and functions.

Var example

Function testVar

Var /GLOBAL example2

StrCpy $example "example value"

StrCpy $example2 "another example value"

FunctionEnd

**4.2.2 Other Writable Variables**

*$0, $1, $2, $3, $4, $5, $6, $7, $8, $9, $R0, $R1, $R2, $R3, $R4, $R5, $R6, $R7, $R8, $R9*

Registers. These variables can be used just like user variables, but are usually used in shared functions or macros. You don't have to declare these variables, so you won't get any name conflicts when using them in shared code. When using these variables in shared code, it's recommended that you use the stack to save and restore their original values. These variables can also be used for communication with plug-ins, because they can be read and written by the plug-in DLLs.

*$INSTDIR*

The installation directory ($INSTDIR is modifiable using [StrCpy](http://nsis.sourceforge.net/Docs/Chapter4.html#StrCpy), [ReadRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readregstr), [ReadINIStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readinistr), etc. - This could be used, for example, in the [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit) function to do a more advanced detection of install location).

Note that in uninstaller code, $INSTDIR contains the directory where the uninstaller lies. It does **not** necessarily contain the same value it contained in the installer. For example, if you write the uninstaller to $WINDIR and the user doesn't move it, $INSTDIR will be $WINDIR in the uninstaller. If you write the uninstaller to another location, you should keep the installer's $INSTDIR in the registry or an alternative storing facility and read it in the uninstaller.

*$OUTDIR*

The current output directory (set implicitly via [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) or explicitly via [StrCpy](http://nsis.sourceforge.net/Docs/Chapter4.html#StrCpy), [ReadRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readregstr), [ReadINIStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readinistr), etc)

*$CMDLINE*

The command line of the installer. The format of the command line can be one of the following:

* "full\path to\installer.exe" PARAMETER PARAMETER PARAMETER
* installer.exe PARAMETER PARAMETER PARAMETER
* For parsing out the PARAMETER portion, see [GetParameters](http://nsis.sourceforge.net/Docs/AppendixE.html#getparameters). If /D= is specified on the command line (to override the install directory) it won't show up in $CMDLINE.

*$LANGUAGE*

The identifier of the language that is currently used. For example, English is 1033. You can change this variable in [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.2.3 Constants**

Constants can also be used in the [InstallDir](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstalldir) attribute.

Note that some of the new constants will not work on every OS. For example, $CDBURN\_AREA will only work on Windows XP and above. If it's used on Windows 98, it'll be empty. Unless mentioned otherwise, a constant should be available on every OS.

*$PROGRAMFILES*, *$PROGRAMFILES32*, *$PROGRAMFILES64*

The program files directory (usually C:\Program Files but detected at runtime). On Windows x64, $PROGRAMFILES and $PROGRAMFILES32 point to C:\Program Files (x86) while $PROGRAMFILES64 points to C:\Program Files. Use $PROGRAMFILES64 when installing x64 applications.

*$COMMONFILES*, *$COMMONFILES32*, *$COMMONFILES64*

The common files directory. This is a directory for components that are shared across applications (usually C:\Program Files\Common Files but detected at runtime). On Windows x64, $COMMONFILES and $COMMONFILES32 point to C:\Program Files (x86)\Common Files while $COMMONFILES64 points to C:\Program Files\Common Files. Use $COMMONFILES64 when installing x64 applications.

*$DESKTOP*

The Windows desktop directory (usually C:\Windows\Desktop but detected at runtime). The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

*$EXEDIR*

The directory containing the installer executable (technically you can modify this variable, but it is probably not a good idea).

*$EXEFILE*

The base name of the installer executable.

*$EXEPATH*

The full path of the installer executable.

*${NSISDIR}*

A symbol that contains the path where NSIS is installed. Useful if you want to call resources that are in NSIS directory e.g. Icons, UIs etc.

When compiled with support for keeping makensis and the data in the same place (the default on Windows), it is in the same place as makensis, on other platforms it is set at compile time (See the INSTALL file for info). In both instances you can modify it at runtime by setting the NSISDIR environment variable. See [section 3.1.3](http://nsis.sourceforge.net/Docs/Chapter3.html#usageenvironment) for more info.

*$WINDIR*

The Windows directory (usually C:\Windows or C:\WinNT but detected at runtime).

*$SYSDIR*

The Windows system directory (usually C:\Windows\System or C:\WinNT\System32 but detected at runtime).

*$TEMP*

The system temporary directory (usually C:\Windows\Temp but detected at runtime).

*$STARTMENU*

The start menu folder (useful in adding start menu items using [CreateShortcut](http://nsis.sourceforge.net/Docs/Chapter4.html#createshortcut)). The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

*$SMPROGRAMS*

The start menu programs folder (use this whenever you want $STARTMENU\Programs). The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

*$SMSTARTUP*

The start menu programs / startup folder. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

*$QUICKLAUNCH*

The quick launch folder for IE4 active desktop and above. If quick launch is not available, simply returns the same as $TEMP.

*$DOCUMENTS*

The documents directory. A typical path for the current user is C:\Documents and Settings\Foo\My Documents. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is not available on Windows 95 with Internet Explorer 4 not installed.

*$SENDTO*

The directory that contains Send To menu shortcut items.

*$RECENT*

The directory that contains shortcuts to the user's recently used documents.

*$FAVORITES*

The directory that contains shortcuts to the user's favorite websites, documents, etc. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is not available on Windows 95 with Internet Explorer 4 not installed.

*$MUSIC*

The user's music files directory. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is available on Windows XP, ME and above.

*$PICTURES*

The user's picture files directory. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is available on Windows 2000, XP, ME and above.

*$VIDEOS*

The user's video files directory. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is available on Windows XP, ME and above.

*$NETHOOD*

The directory that contains link objects that may exist in the My Network Places/Network Neighborhood folder.

This constant is not available on Windows 95 with Internet Explorer 4 and Active Desktop not installed.

*$FONTS*

The system's fonts directory.

*$TEMPLATES*

The document templates directory. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

*$APPDATA*

The application data directory. Detection of the current user path requires Internet Explorer 4 and above. Detection of the all users path requires Internet Explorer 5 and above. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is not available on Windows 95 with Internet Explorer 4 and Active Desktop not installed.

*$LOCALAPPDATA*

The local (nonroaming) application data directory.

This constant is available on Windows 2000 and above.

*$PRINTHOOD*

The directory that contains link objects that may exist in the Printers folder.

This constant is not available on Windows 95 and Windows 98.

*$INTERNET\_CACHE*

Internet Explorer's temporary internet files directory.

This constant is not available on Windows 95 and Windows NT with Internet Explorer 4 and Active Desktop not installed.

*$COOKIES*

Internet Explorer's cookies directory.

This constant is not available on Windows 95 and Windows NT with Internet Explorer 4 and Active Desktop not installed.

*$HISTORY*

Internet Explorer's history directory.

This constant is not available on Windows 95 and Windows NT with Internet Explorer 4 and Active Desktop not installed.

*$PROFILE*

The user's profile directory. A typical path is C:\Documents and Settings\Foo.

This constant is available on Windows 2000 and above.

*$ADMINTOOLS*

A directory where administrative tools are kept. The context of this constant (All Users or Current user) depends on the [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) setting. The default is the current user.

This constant is available on Windows 2000, ME and above.

*$RESOURCES*

The resources directory that stores themes and other Windows resources (usually C:\Windows\Resources but detected at runtime).

This constant is available on Windows XP and above.

*$RESOURCES\_LOCALIZED*

The localized resources directory that stores themes and other Windows resources (usually C:\Windows\Resources\1033 but detected at runtime).

This constant is available on Windows XP and above.

*$CDBURN\_AREA*

A directory where files awaiting to be burned to CD are stored.

This constant is available on Windows XP and above.

*$HWNDPARENT*

The decimal HWND of the parent window.

*$PLUGINSDIR*

The path to a temporary folder created upon the first usage of a plug-in or a call to [InitPluginsDir](http://nsis.sourceforge.net/Docs/Chapter4.html#initpluginsdir). This folder is automatically deleted when the installer exits. This makes this folder the ideal folder to hold INI files for [InstallOptions](http://nsis.sourceforge.net/Docs/InstallOptions/Readme.html), bitmaps for the splash plug-in, or any other file that a plug-in needs to work.

**4.2.4 Constants Used in Strings**

*$$*

Use to represent $.

*$\r*

Use to represent a carriage return (\r).

*$\n*

Use to represent a newline (\n).

*$\t*

Use to represent a tab (\t).

**4.3 Labels**

Labels are the targets of [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto) instructions, or of the various branching instructions (such as [IfErrors](http://nsis.sourceforge.net/Docs/Chapter4.html#iferrors), [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox), [IfFileExists](http://nsis.sourceforge.net/Docs/Chapter4.html#iffileexists), and [StrCmp](http://nsis.sourceforge.net/Docs/Chapter4.html#strcmp)). Labels must be within a [Section](http://nsis.sourceforge.net/Docs/Chapter4.html#ssection) or a [Function](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunction). Labels are local in scope, meaning they are only accessible from within the [Section](http://nsis.sourceforge.net/Docs/Chapter4.html#ssection) or [Function](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunction) that they reside in. To declare a label, simply use:

*MyLabel:*

Labels cannot begin with a -, +, !, $, or 0-9. When specifying labels for the various instructions that require them, remember that both an empty string ("") and 0 both represent the next instruction (meaning no [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto) will occur). Labels beginning with a period (.) are global, meaning you can jump to them from any function or section (though you cannot jump to an uninstall global label from the installer, and vice versa).

**4.4 Relative Jumps**

Unlike labels, relative jumps are, as the name suggests, relative to the place they are called from. You can use relative jumps wherever you can use labels. Relative jumps are marked by numbers. +1 jumps to the next instruction (the default advancement), +2 will skip one instruction and go to the second instruction from the current instruction, -2 will jump two instructions backward, and +10 will skip 9 instructions, jumping to the tenth instruction from the current instruction.

An instruction is every command that is executed at run-time, when the installer is running. [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox), [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto), [GetDLLVersion](http://nsis.sourceforge.net/Docs/Chapter4.html#getdllversion), [FileRead](http://nsis.sourceforge.net/Docs/Chapter4.html#FileRead), [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) are all instructions. [AddSize](http://nsis.sourceforge.net/Docs/Chapter4.html#saddsize), [Section](http://nsis.sourceforge.net/Docs/Chapter4.html#ssection), [SectionGroup](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectiongroup), [SectionEnd](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectionend), [SetOverwrite](http://nsis.sourceforge.net/Docs/Chapter4.html#asetoverwrite) (and everything under [Compiler Flags](http://nsis.sourceforge.net/Docs/Chapter4.html#flags)), [Name](http://nsis.sourceforge.net/Docs/Chapter4.html#aname),[SetFont](http://nsis.sourceforge.net/Docs/Chapter4.html#asetfont), [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring), are not instructions because they are executed at compile time.

Examples:

Goto +2

MessageBox MB\_OK "You will never ever see this message box"

MessageBox MB\_OK "The last message was skipped, this one should be shown"

Goto +4

MessageBox MB\_OK "The following message will be skipped"

Goto +3

MessageBox MB\_OK "You will never ever see this message box"

Goto -3

MessageBox MB\_OK "Done"

Note that [macro insertion](http://nsis.sourceforge.net/Docs/Chapter5.html#insertmacro) is not considered as one instruction when it comes to relative jumps. The macro is expanded before relative jumps are applied, and so relative jumps can jump into code inside an inserted macro. The following code, for example, will not skip the macro. It will show a message box.

!macro relative\_jump\_test

MessageBox MB\_OK "first macro line"

MessageBox MB\_OK "second macro line"

!macroend

Goto +2

!insertmacro relative\_jump\_test

**4.5 Pages**

Each (non-silent) NSIS installer has a set of pages. Each page can be a NSIS built-in page or a custom page created by a user's function (with [nsDialogs](http://nsis.sourceforge.net/Docs/nsDialogs/Readme.html) or [InstallOptions](http://nsis.sourceforge.net/Docs/InstallOptions/Readme.html) for example).

Using the script you can control the pages' order, appearance, and behavior. You can skip pages, paint them white, force the user to stay in a certain page until a certain condition is met, show a readme page, show custom designed pages for input and more. In this section, you will learn how to control all of the above.

There are two basic commands regarding pages, [Page](http://nsis.sourceforge.net/Docs/Chapter4.html#page) and [UninstPage](http://nsis.sourceforge.net/Docs/Chapter4.html#uninstpage). The first adds a page to the installer, the second adds a page to the uninstaller. On top of those two there is the [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) command which allows you to add a page to either one and with greater amount of options. [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex)allows you to set options to the specific page you are adding instead of using the default that's set outside of [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex).

**4.5.1 Ordering**

The page order is set simply by the order [Page](http://nsis.sourceforge.net/Docs/Chapter4.html#page), [UninstPage](http://nsis.sourceforge.net/Docs/Chapter4.html#uninstpage) and [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) appear in the script. For example:

Page license

Page components

Page directory

Page instfiles

UninstPage uninstConfirm

UninstPage instfiles

This code will tell NSIS to first show the license page, then the components selection page, then the directory selection page and finally the install log where sections are executed, just like in old installers. The uninstaller will first show the uninstall confirmation page and then the uninstallation log.

You can specify the same page type more than once.

For backwards compatibility with old NSIS scripts, the following installer pages will be added if no installer page commands are used: license (if [LicenseText](http://nsis.sourceforge.net/Docs/Chapter4.html#alicensetext) and [LicenseData](http://nsis.sourceforge.net/Docs/Chapter4.html#alicensedata) were specified), components (if [ComponentText](http://nsis.sourceforge.net/Docs/Chapter4.html#acomponenttext) was specified and there is more than one visible section), directory (if [DirText](http://nsis.sourceforge.net/Docs/Chapter4.html#adirtext) was specified) and instfiles. When there are no uninstaller page commands, the following uninstaller pages will be added: uninstall confirmation page (if [UninstallText](http://nsis.sourceforge.net/Docs/Chapter4.html#auninstalltext) was specified) and instfiles. This method is deprecated, converting scripts to use page commands is highly recommended because you can use the new standard language strings.

**4.5.2 Page Options**

Each page has its unique set of data that defines how it will look and act. This section describes what data each type of page uses and how you can set it. [Callback functions](http://nsis.sourceforge.net/Docs/Chapter4.html#pagecallbacks_explain) are described below and are not dealt with in this section.

The list below lists what commands affect the certain page type. Unless mentioned otherwise, these commands can be used both in and out of a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block. If used inside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block they will only affect the current page being set by [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex), else they will set the default for every other page.

*License page*

* [LicenseText](http://nsis.sourceforge.net/Docs/Chapter4.html#alicensetext)
* [LicenseData](http://nsis.sourceforge.net/Docs/Chapter4.html#alicensedata)
* [LicenseForceSelection](http://nsis.sourceforge.net/Docs/Chapter4.html#alicenseforceselection)

*Components selection page*

* [ComponentText](http://nsis.sourceforge.net/Docs/Chapter4.html#acomponenttext)

*Directory selection page*

* [DirText](http://nsis.sourceforge.net/Docs/Chapter4.html#adirtext)
* [DirVar](http://nsis.sourceforge.net/Docs/Chapter4.html#adirvar) - can only be used in [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex)
* [DirVerify](http://nsis.sourceforge.net/Docs/Chapter4.html#adirverify)

*Un/Installation log page*

* [DetailsButtonText](http://nsis.sourceforge.net/Docs/Chapter4.html#adetailsbuttontext)
* [CompletedText](http://nsis.sourceforge.net/Docs/Chapter4.html#acompletedtext)

*Uninstall confirmation page*

* [DirVar](http://nsis.sourceforge.net/Docs/Chapter4.html#adirvar) - can only be used in [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex)
* [UninstallText](http://nsis.sourceforge.net/Docs/Chapter4.html#auninstalltext)

To set the page caption use [Caption](http://nsis.sourceforge.net/Docs/Chapter4.html#acaption).

**4.5.3 Callbacks**

Each built-in page has three callback functions: the pre-function, the show-creation function and the leave-function. The pre-function is called right before the page is created, the show-function is called right after it is created and before it is shown and the leave-function is called right after the user has pressed the next button and before the page is left.

* The pre-function allows you to skip the page using [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort).
* The show-function allows you to tweak the page's user interface with [CreateFont](http://nsis.sourceforge.net/Docs/Chapter4.html#createfont), [SetCtlColors](http://nsis.sourceforge.net/Docs/Chapter4.html#setctlcolors), [SendMessage](http://nsis.sourceforge.net/Docs/Chapter4.html#sendmessage) and others.
* The leave-function allows you to force the user to stay on the current page using [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort).

A custom page only has two callback functions, one that creates it which is mandatory, and one leave-function that acts just like the leave-function for built-in pages.

Examples:

Page license skipLicense "" stayInLicense

Page custom customPage "" ": custom page"

Page instfiles

Function skipLicense

MessageBox MB\_YESNO "Do you want to skip the license page?" IDNO no

Abort

no:

FunctionEnd

Function stayInLicense

MessageBox MB\_YESNO "Do you want to stay in the license page?" IDNO no

Abort

no:

FunctionEnd

Function customPage

GetTempFileName $R0

File /oname=$R0 customPage.ini

InstallOptions::dialog $R0

Pop $R1

StrCmp $R1 "cancel" done

StrCmp $R1 "back" done

StrCmp $R1 "success" done

error: MessageBox MB\_OK|MB\_ICONSTOP "InstallOptions error:$\r$\n$R1"

done:

FunctionEnd

**4.5.4 Page**

custom [creator\_function] [leave\_function] [caption] [/ENABLECANCEL]

OR

internal\_page\_type [pre\_function] [show\_function] [leave\_function] [/ENABLECANCEL]

Adds an installer page. See the above sections for more information about built-in versus custom pages and about callback functions.

*internal\_page\_type* can be:

* *license* - license page
* *components* - components selection page
* *directory* - installation directory selection page
* *instfiles* - installation page where the sections are executed
* *uninstConfirm* - uninstall confirmation page

The last page of the installer has its cancel button disabled to prevent confusion. To enable it anyway, use */ENABLECANCEL*.

**4.5.5 UninstPage**

custom [creator\_function] [leave\_function] [caption] [/ENABLECANCEL]

OR

internal\_page\_type [pre\_function] [show\_function] [leave\_function] [/ENABLECANCEL]

Adds an uninstaller page. See the above sections for more information about built-in versus custom pages and about callback functions.

See [Page](http://nsis.sourceforge.net/Docs/Chapter4.html#page) for possible values of *internal\_page\_type*.

**4.5.6 PageEx**

[un.](custom|uninstConfirm|license|components|directory|instfiles)

Adds an installer page or an uninstaller page if the un. prefix was used. Every PageEx must have a matching [PageExEnd](http://nsis.sourceforge.net/Docs/Chapter4.html#pageexend). In a PageEx block you can set options that are specific to this page and will not be used for other pages. Options that are not set will revert to what was set outside the PageEx block or the default if nothing was set. To set the sub-caption for a page use [Caption](http://nsis.sourceforge.net/Docs/Chapter4.html#acaption) or [SubCaption](http://nsis.sourceforge.net/Docs/Chapter4.html#asubcaption) to set the default. To set the callback functions for a page set with PageEx use [PageCallbacks](http://nsis.sourceforge.net/Docs/Chapter4.html#pagecallbacks). See the above sections for more information about built-in versus custom pages.

Example usage:

PageEx license

LicenseText "Readme"

LicenseData readme.rtf

PageExEnd

PageEx license

LicenseData license.txt

LicenseForceSelection checkbox

PageExEnd

**4.5.7 PageExEnd**

Ends a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block.

**4.5.8 PageCallbacks**

([creator\_function] [leave\_function]) | ([pre\_function] [show\_function] [leave\_function])

Sets the callback functions for a page defined using [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex). Can only be used inside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block. See the above sections for more information about callback functions.

PageEx license

PageCallbacks licensePre licenseShow licenseLeave

PageExEnd

**4.6 Sections**

Each NSIS installer contains one or more sections. Each of these sections are created, modified, and ended with the following commands.

* Each section contains zero or more instructions.
* Sections are executed in order by the resulting installer, and if [ComponentText](http://nsis.sourceforge.net/Docs/Chapter4.html#acomponenttext) is set, the user will have the option of disabling/enabling each visible section.
* If a section's name is 'Uninstall' or is prefixed with 'un.', it's an uninstaller section.

**4.6.1 Section Commands**

**4.6.1.1 AddSize**

size\_kb

Tells the installer that the current section needs an additional "size\_kb" kilobytes of disk space. Only valid within a section (will have no effect outside of a section or in a function).

Section

AddSize 500

SectionEnd

**4.6.1.2 Section**

[/o] [([!]|[-])section\_name] [section\_index\_output]

Begins and opens a new section. If section\_name is empty, omitted, or begins with a -, then it is a hidden section and the user will not have the option of disabling it. If the section name is 'Uninstall' or is prefixed with 'un.', then it is a an uninstaller section. If *section\_index\_output* is specified, the parameter will be [!defined](http://nsis.sourceforge.net/Docs/Chapter5.html" \l "define) with the section index (that can be used for [SectionSetText](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsettext) etc). If the section name begins with a !, the section will be displayed as bold. If the /o switch is specified, the section will be unselected by default.

Section "-hidden section"

SectionEnd

Section # hidden section

SectionEnd

Section "!bold section"

SectionEnd

Section /o "optional"

SectionEnd

Section "install something" SEC\_IDX

SectionEnd

To access the section index, curly brackets must be used and the code must be located below the section in the script.

Section test1 sec1\_id

SectionEnd

Section test2 sec2\_id

SectionEnd

Function .onInit

SectionGetText ${sec2\_id} $0

MessageBox MB\_OK "name of ${sec2\_id}:$\n$0" # will correctly display 'name of 1: test2'

FunctionEnd

Function .onInit

SectionGetText ${sec2\_id} $0

MessageBox MB\_OK "name of ${sec2\_id}:$\n$0" # will incorrectly display 'name of ${sec2\_id}: test1'

# plus a warning stating:

# unknown variable/constant "{sec2\_id}" detected, ignoring

FunctionEnd

Section test1 sec1\_id

SectionEnd

Section test2 sec2\_id

SectionEnd

**4.6.1.3 SectionEnd**

This command closes the current open section.

**4.6.1.4 SectionIn**

insttype\_index [insttype\_index] [RO]

This command specifies which install types (see [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype)) the current section defaults to the enabled state in. Multiple [SectionIn](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectionin) commands can be specified (they are combined). If you specify RO as a parameter, then the section will be read-only, meaning the user won't be able to change its state. The first install type defined using [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) is indexed 1, the next 2 and so on.

InstType "full"

InstType "minimal"

Section "a section"

SectionIn 1 2

SectionEnd

Section "another section"

SectionIn 1

SectionEnd

**4.6.1.5 SectionGroup**

[/e] section\_group\_name [index\_output]

This command inserts a section group. The section group must be closed with [SectionGroupEnd](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectiongroupend), and should contain 1 or more sections. If the section group name begins with a !, its name will be displayed with a bold font. If /e is present, the section group will be expanded by default. If*index\_output* is specified, the parameter will be !defined with the section index (that can be used for [SectionSetText](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsettext) etc). If the name is prefixed with 'un.' the section group is an uninstaller section group.

SectionGroup "some stuff"

Section "a section"

SectionEnd

Section "another section"

SectionEnd

SectionGroupEnd

**4.6.1.6 SectionGroupEnd**

Closes a section group opened with [SectionGroup](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectiongroup).

**4.6.2 Uninstall Section**

A special [Section](http://nsis.sourceforge.net/Docs/Chapter4.html#ssection) named 'Uninstall' must be created in order to generate an uninstaller. This section should remove all files, registry keys etc etc that were installed by the installer, from the system. Here is an example of a simple uninstall section:

Section "Uninstall"

Delete $INSTDIR\Uninst.exe ; delete self (see explanation below why this works)

Delete $INSTDIR\myApp.exe

RMDir $INSTDIR

DeleteRegKey HKLM SOFTWARE\myApp

SectionEnd

The first [Delete](http://nsis.sourceforge.net/Docs/Chapter4.html#delete) instruction works (deleting the uninstaller), because the uninstaller is transparently copied to the system temporary directory for the uninstall.

Note that in uninstaller code, $INSTDIR contains the directory where the uninstaller lies. It does **not** necessarily contain the same value it contained in the installer.

**4.7 Functions**

Functions are similar to Sections in that they contain zero or more instructions. User functions are not called by the installer directly, instead they are called from Sections using the [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call) instruction. Callback functions will be called by the installer when a certain event occurs.

Functions must be declared outside of Sections or other Functions.

**4.7.1 Function Commands**

**4.7.1.1 Function**

[function\_name]

Begins and opens a new function. [Function](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunction) names beginning with "." (e.g. ".Whatever") are generally reserved for callback functions. [Function](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunction) names beginning with "un." are functions that will be generated in the Uninstaller. Hence, normal install Sections and functions cannot call uninstall functions, and the Uninstall Section and uninstall functions cannot call normal functions.

Function func

# some commands

FunctionEnd

Section

Call func

SectionEnd

**4.7.1.2 FunctionEnd**

This command closes the current open function.

**4.7.2 Callback Functions**

You can create callback functions which have special names, that will be called by the installer at certain points in the install. Below is a list of currently available callbacks:

**4.7.2.1 Install Callbacks**

**4.7.2.1.1 .onGUIInit**

This callback will be called just before the first page is loaded and the installer dialog is shown, allowing you to tweak the user interface.

Example:

!include "WinMessages.nsh"

Function .onGUIInit

# 1028 is the id of the branding text control

GetDlgItem $R0 $HWNDPARENT 1028

CreateFont $R1 "Tahoma" 10 700

SendMessage $R0 ${WM\_SETFONT} $R1 0

# set background color to white and text color to red

SetCtlColors $R0 FFFFFF FF0000

FunctionEnd

**4.7.2.1.2 .onInit**

This callback will be called when the installer is nearly finished initializing. If the '.onInit' function calls [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort), the installer will quit instantly.

Here are two examples of how this might be used:

Function .onInit

MessageBox MB\_YESNO "This will install. Continue?" IDYES NoAbort

Abort ; causes installer to quit.

NoAbort:

FunctionEnd

or:

Function .onInit

ReadINIStr $INSTDIR $WINDIR\wincmd.ini Configuration InstallDir

StrCmp $INSTDIR "" 0 NoAbort

MessageBox MB\_OK "Windows Commander not found. Unable to get install path."

Abort ; causes installer to quit.

NoAbort:

FunctionEnd

**4.7.2.1.3 .onInstFailed**

This callback is called when the user hits the 'cancel' button after the install has failed (if it could not extract a file, or the install script used the [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort) command).

Example:

Function .onInstFailed

MessageBox MB\_OK "Better luck next time."

FunctionEnd

**4.7.2.1.4 .onInstSuccess**

This callback is called when the install was successful, right before the install window closes (which may be after the user clicks 'Close' if [AutoCloseWindow](http://nsis.sourceforge.net/Docs/Chapter4.html#aautoclosewindow) or [SetAutoClose](http://nsis.sourceforge.net/Docs/Chapter4.html#setautoclose) is set to false).

Example:

Function .onInstSuccess

MessageBox MB\_YESNO "Congrats, it worked. View readme?" IDNO NoReadme

Exec notepad.exe ; view readme or whatever, if you want.

NoReadme:

FunctionEnd

**4.7.2.1.5 .onGUIEnd**

This callback is called right after the installer window closes. Use it to free any user interface related plug-ins if needed.

**4.7.2.1.6 .onMouseOverSection**

This callback is called whenever the mouse position over the sections tree has changed. This allows you to set a description for each section for example. The section id on which the mouse is over currently is stored, temporarily, in $0.

Example:

Function .onMouseOverSection

FindWindow $R0 "#32770" "" $HWNDPARENT

GetDlgItem $R0 $R0 1043 ; description item (must be added to the UI)

StrCmp $0 0 "" +2

SendMessage $R0 ${WM\_SETTEXT} 0 "STR:first section description"

StrCmp $0 1 "" +2

SendMessage $R0 ${WM\_SETTEXT} 0 "STR:second section description"

FunctionEnd

**4.7.2.1.7 .onRebootFailed**

This callback is called if [Reboot](http://nsis.sourceforge.net/Docs/Chapter4.html#reboot) fails. [WriteUninstaller](http://nsis.sourceforge.net/Docs/Chapter4.html#writeuninstaller), [plug-ins](http://nsis.sourceforge.net/Docs/Chapter4.html#plugindlls), [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) and [WriteRegBin](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregbin) should not be used in this callback.

Example:

Function .onRebootFailed

MessageBox MB\_OK|MB\_ICONSTOP "Reboot failed. Please reboot manually." /SD IDOK

FunctionEnd

**4.7.2.1.8 .onSelChange**

Called when the selection changes on the [component page](http://nsis.sourceforge.net/Docs/Chapter4.html#pages). Useful for using with [SectionSetFlags](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsetflags) and [SectionGetFlags](http://nsis.sourceforge.net/Docs/Chapter4.html#sectiongetflags).

Selection changes include both section selection and installation type changes. The section id of the changed section is stored in $0. $0 is -1 if the installation type changed. You only get notifications for changes initiated by the user and only one notification per action even if the action also affected child sections and/or parent groups.

**4.7.2.1.9 .onUserAbort**

This callback is called when the user hits the 'cancel' button, and the install hasn't already failed. If this function calls [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort), the install will not be aborted.

Example:

Function .onUserAbort

MessageBox MB\_YESNO "Abort install?" IDYES NoCancelAbort

Abort ; causes installer to not quit.

NoCancelAbort:

FunctionEnd

**4.7.2.1.10 .onVerifyInstDir**

This callback enables control over whether or not an installation path is valid for your installer. This code will be called every time the user changes the install directory, so it shouldn't do anything crazy with [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox) or the likes. If this function calls [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort), the installation path in $INSTDIR is deemed invalid.

Example:

Function .onVerifyInstDir

IfFileExists $INSTDIR\Winamp.exe PathGood

Abort ; if $INSTDIR is not a winamp directory, don't let us install there

PathGood:

FunctionEnd

**4.7.2.2 Uninstall Callbacks**

**4.7.2.2.1 un.onGUIInit**

This callback will be called just before the first page is loaded and the installer dialog is shown, allowing you to tweak the user interface.

Have a look at [.onGUIInit](http://nsis.sourceforge.net/Docs/Chapter4.html#onguiinit) for an example.

**4.7.2.2.2 un.onInit**

This callback will be called when the uninstaller is nearly finished initializing. If the ' [un.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#unonInit)' function calls [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort), the uninstaller will quit instantly. Note that this function can verify and/or modify $INSTDIR if necessary.

Here are two examples of how this might be used:

Function un.onInit

MessageBox MB\_YESNO "This will uninstall. Continue?" IDYES NoAbort

Abort ; causes uninstaller to quit.

NoAbort:

FunctionEnd

or:

Function un.onInit

IfFileExists $INSTDIR\myfile.exe found

Messagebox MB\_OK "Uninstall path incorrect"

Abort

found:

FunctionEnd

**4.7.2.2.3 un.onUninstFailed**

This callback is called when the user hits the 'cancel' button after the uninstall has failed (if it used the [Abort command](http://nsis.sourceforge.net/Docs/Chapter4.html#abort) or otherwise failed).

Example:

Function un.onUninstFailed

MessageBox MB\_OK "Better luck next time."

FunctionEnd

**4.7.2.2.4 un.onUninstSuccess**

This callback is called when the uninstall was successful, right before the install window closes (which may be after the user clicks 'Close' if [SetAutoClose](http://nsis.sourceforge.net/Docs/Chapter4.html#setautoclose) is set to false)..

Example:

Function un.onUninstSuccess

MessageBox MB\_OK "Congrats, it's gone."

FunctionEnd

**4.7.2.2.5 un.onGUIEnd**

This callback is called right after the uninstaller window closes. Use it to free any user interface related plug-ins if needed.

**4.7.2.2.6 un.onRebootFailed**

This callback is called if [Reboot](http://nsis.sourceforge.net/Docs/Chapter4.html#reboot) fails. [WriteUninstaller](http://nsis.sourceforge.net/Docs/Chapter4.html#writeuninstaller), [plug-ins](http://nsis.sourceforge.net/Docs/Chapter4.html#plugindlls), [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) and [WriteRegBin](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregbin) should not be used in this callback.

Example:

Function un.onRebootFailed

MessageBox MB\_OK|MB\_ICONSTOP "Reboot failed. Please reboot manually." /SD IDOK

FunctionEnd

**4.7.2.2.7 un.onSelChange**

Called when the selection changes on the [component page](http://nsis.sourceforge.net/Docs/Chapter4.html#pages). Useful for using with [SectionSetFlags](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsetflags) and [SectionGetFlags](http://nsis.sourceforge.net/Docs/Chapter4.html#sectiongetflags).

Selection changes include both section selection and installation type changes. The section id of the changed section is stored in $0. $0 is -1 if the installation type changed. You only get notifications for changes initiated by the user and only one notification per action even if the action also affected child sections and/or parent groups.

**4.7.2.2.8 un.onUserAbort**

This callback is called when the user hits the 'cancel' button and the uninstall hasn't already failed. If this function calls [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort), the install will not be aborted.

Example:

Function un.onUserAbort

MessageBox MB\_YESNO "Abort uninstall?" IDYES NoCancelAbort

Abort ; causes uninstaller to not quit.

NoCancelAbort:

FunctionEnd

**4.8 Installer Attributes**

**4.8.1 General Attributes**

The commands below all adjust attributes of the installer. These attributes control how the installer looks and functions, including which pages are present in the installer, as what text is displayed in each part of each page, how the installer is named, what icon the installer uses, the default installation directory, what file it writes out, and more. Note that these attributes can be set anywhere in the file except in a [Section](http://nsis.sourceforge.net/Docs/Chapter4.html#ssection) or [Function](http://nsis.sourceforge.net/Docs/Chapter4.html#ffunction).

**Defaults are bold and underlined**

**4.8.1.1 AddBrandingImage**

(left|right|top|bottom) (width|height) [padding]

Adds a branding image on the top, bottom, left, or right of the installer. Its size will be set according to the width/height specified, the installer width/height and the installer font. The final size will not always be what you requested; have a look at the output of the command for the actual size. Because this depends on the installer font, you should use [SetFont](http://nsis.sourceforge.net/Docs/Chapter4.html#asetfont) before [AddBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#aaddbrandingimage). The default padding value is 2.

[AddBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#aaddbrandingimage) only adds a placeholder for an image. To set the image itself on runtime, use [SetBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#setbrandingimage).

AddBrandingImage left 100

AddBrandingImage right 50

AddBrandingImage top 20

AddBrandingImage bottom 35

AddBrandingImage left 100 5

**4.8.1.2 AllowRootDirInstall**

true|**false**

Controls whether or not installs are enabled to the root directory of a drive, or directly into a network share. Set to 'true' to change the safe behavior, which prevents users from selecting C:\ or \\Server\Share as an install (and later on, uninstall) directory. For additional directory selection page customizability, see [.onVerifyInstDir](http://nsis.sourceforge.net/Docs/Chapter4.html#onverifyinstdir).

**4.8.1.3 AutoCloseWindow**

true|**false**

Sets whether or not the install window automatically closes when completed. This is overrideable from a section using [SetAutoClose](http://nsis.sourceforge.net/Docs/Chapter4.html#setautoclose).

**4.8.1.4 BGFont**

[font\_face [height [weight] [/ITALIC] [/UNDERLINE] [/STRIKE]]]

Specifies the font used to show the text on the background gradient. To set the color use [BGGradient](http://nsis.sourceforge.net/Docs/Chapter4.html#abggradient). If no parameters are specified, the default font will be used. The default font is bold and italic Times New Roman.

**4.8.1.5 BGGradient**

[**off**|(topc botc [textcolor|notext])]

Specifies whether or not to use a gradient background window. If 'off', the installer will not show a background window, if no parameters are specified, the default black to blue gradient is used, and otherwise the top\_color or bottom\_color are used to make a gradient. Top\_color and bottom\_color are specified using the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). 'textcolor' can be specified as well, or 'notext' can be specified to turn the big background text off.

**4.8.1.6 BrandingText**

/TRIM(LEFT|RIGHT|CENTER) text

Sets the text that is shown (by default it is 'Nullsoft Install System vX.XX') at the bottom of the install window. Setting this to an empty string ("") uses the default; to set the string to blank, use " " (a space). If it doesn't matter to you, leave it the default so that everybody can know why the installer didn't suck. heh. Use /TRIMLEFT, /TRIMRIGHT or /TRIMCENTER to trim down the size of the control to the size of the string.

Accepts variables. If variables are used, they must be initialized on [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.8.1.7 Caption**

caption

When used outside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block: Sets the text for the titlebar of the installer. the By default, it is 'Name Setup', where Name is specified with the [Name](http://nsis.sourceforge.net/Docs/Chapter4.html#aname) instruction. You can, however, override it with 'MyApp Installer' or whatever. If you specify an empty string (""), the default will be used (you can however specify " " to achieve a blank string).

When used inside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block: Sets the subcaption of the current page.

Accepts variables. If variables are used, they must be initialized on [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.8.1.8 ChangeUI**

dialog ui\_file.exe

Replaces dialog (*IDD\_LICENSE*, *IDD\_DIR*, *IDD\_SELCOM*, *IDD\_INST*, *IDD\_INSTFILES*, *IDD\_UNINST* or *IDD\_VERIFY*) by a dialog with the same resource ID in ui\_file.exe. You can also specify 'all' as the dialog if you wish to replace all 7 of the dialogs at once from the same UI file. For some example UIs look at Contrib\UIs under your NSIS directory.

* *IDD\_LICENSE* must contain *IDC\_EDIT1* (RICHEDIT control).
* *IDD\_DIR* must contain *IDC\_DIR* (edit box), *IDC\_BROWSE* (button) and *IDC\_CHECK1* (checkbox).
* *IDD\_SELCOM* must contain *IDC\_TREE1* (SysTreeView32 control), and *IDC\_COMBO1* (combo box).
* *IDD\_INST* must contain *IDC\_BACK* (button), *IDC\_CHILDRECT* (static control the size of all other dialogs), *IDC\_VERSTR* (static), *IDOK* (button), and *IDCANCEL* (button). If an image control (static with *SS\_BITMAP* style) will be found in this dialog it will be used as the default for[SetBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#setbrandingimage).
* *IDD\_INSTFILES* must contain *IDC\_LIST1* (SysListView32 control), *IDC\_PROGRESS* (msctls\_progress32 control), and *IDC\_SHOWDETAILS* (button).
* *IDD\_UNINST* must contain *IDC\_EDIT1* (edit box).
* *IDD\_VERIFY* must contain *IDC\_STR* (static).

ChangeUI all "${NSISDIR}\Contrib\UIs\sdbarker\_tiny.exe"

**4.8.1.9 CheckBitmap**

bitmap.bmp

Specifies the bitmap with the images used for the checks of the component-selection page treeview.

This bitmap should have a size of 96x16 pixels, no more than 8bpp (256 colors) and contain six 16x16 images for the different states (in order: selection mask, not checked, checked, greyed out, unchecked & read-only, checked & read-only). Use magenta as mask color (this area will be transparent).

**4.8.1.10 CompletedText**

text

Replaces the default text ("Completed") that is printed at the end of the install if parameter is specified. Otherwise, the default is used.

Accepts variables. If variables are used, they must be initialized before the message is printed.

**4.8.1.11 ComponentText**

[text [subtext] [subtext2]]

Used to change the default text on the component page.

text: Text above the controls, to the right of the installation icon.

subtext: Text next to the installation type selection.

subtext2: Text to the left of the components list and below the installation type.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the components page is created.

**4.8.1.12 CRCCheck**

**on**|off|force

Specifies whether or not the installer will perform a CRC on itself before allowing an install. Note that if the user uses /NCRC on the command line when executing the installer, and you didn't specify 'force', the CRC will not occur, and the user will be allowed to install a (potentially) corrupted installer.

**4.8.1.13 DetailsButtonText**

show details text

Replaces the default details button text of "Show details", if parameter is specified (otherwise the default is used).

Accepts variables. If variables are used, they must be initialized before the install log (instfiles) page is created.

**4.8.1.14 DirText**

[text] [subtext] [browse\_button\_text] [browse\_dlg\_text]

Used to change the default text on the directory page.

text: Text above the controls, to the right of the installation icon.

subtext: Text on the directory selection frame.

browse\_button\_text: Text on the Browse button.

browse\_dlg\_text: Text on the "Browse For Folder" dialog, appears after clicking on "Browse" button.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the directory page is created.

**4.8.1.15 DirVar**

user\_var(dir input/output)

Specifies which variable is to be used to contain the directory selected. This variable should contain the default value too. This allows to easily create two different directory pages that will not require you to move values in and out of $INSTDIR. The default variable is $INSTDIR. This can only be used in [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) and for directory and uninstConfirm pages.

Var ANOTHER\_DIR

PageEx directory

DirVar $ANOTHER\_DIR

PageExEnd

Section

SetOutPath $INSTDIR

File "a file.dat"

SetOutPath $ANOTHER\_DIR

File "another file.dat"

SectionEnd

**4.8.1.16 DirVerify**

**auto**|leave

If `[DirVerify](http://nsis.sourceforge.net/Docs/Chapter4.html#adirverify) leave' is used, the Next button will not be disabled if the installation directory is not valid or there is not enough space. A flag that you can read in the leave function using [GetInstDirError](http://nsis.sourceforge.net/Docs/Chapter4.html#getinstdirerror) will be set instead.

PageEx directory

DirVerify leave

PageCallbacks "" "" dirLeave

PageExEnd

**4.8.1.17 FileErrorText**

file error text

Replaces the default text that comes up when a file cannot be written to. This string can contain a reference to $0, which is the filename ($0 is temporarily changed to this value). Example: "Can not write to file $\r$\n$0$\r$\ngood luck.".

Accepts variables. If variables are used, they must be initialized before [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) is used.

**4.8.1.18 Icon**

[path\]icon.ico

Sets the icon of the installer. Every icon in the icon file will be included in the installer. Use [UninstallIcon](http://nsis.sourceforge.net/Docs/Chapter4.html#auninstallicon) to set the uninstaller icon.

**4.8.1.19 InstallButtonText**

install button text

If parameter is specified, overrides the default install button text (of "Install") with the specified text.

Accepts variables. If variables are used, they must be initialized before the install button shows.

**4.8.1.20 InstallColors**

/windows | (foreground\_color background\_color)

Sets the colors to use for the install info screen (the default is 00FF00 000000. Use the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). Note that if "/windows" is specified as the only parameter, the default windows colors will be used.

**4.8.1.21 InstallDir**

definstdir

Sets the default installation directory. See the [variables section](http://nsis.sourceforge.net/Docs/Chapter4.html#variables) for variables that can be used to make this string (especially $PROGRAMFILES). Note that the part of this string following the last \ will be used if the user selects 'browse', and may be appended back on to the string at install time (to disable this, end the directory with a \ (which will require the entire parameter to be enclosed with quotes). If this doesn't make any sense, play around with the browse button a bit.

**4.8.1.22 InstallDirRegKey**

root\_key subkey key\_name

This attribute tells the installer to check a string in the registry, and use it for the install dir if that string is valid. If this attribute is present, it will override the [InstallDir](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstalldir) attribute if the registry key is valid, otherwise it will fall back to the [InstallDir](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstalldir) default. When querying the registry, this command will automatically remove any quotes. If the string ends in ".exe", it will automatically remove the filename component of the string (i.e. if the string is "C:\Program Files\Foo\app.exe", it will know to use "C:\Program Files\Foo"). For more advanced install directory configuration, set $INSTDIR in .onInit.

Language strings and variables cannot be used with [InstallDirRegKey](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstalldirregkey).

InstallDirRegKey HKLM Software\NSIS ""

InstallDirRegKey HKLM Software\ACME\Thingy InstallLocation

**4.8.1.23 InstProgressFlags**

[flag [...]]

Valid values for flag are "smooth" (smooth the progress bar) or "colored" (color the progress bar with the colors set by [InstallColors](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstallcolors). Examples: "InstProgressFlags" (default old-school windows look), "InstProgressFlags smooth" (new smooth look), "InstProgressFlags smooth colored" (colored smooth look whee). Note: neither "smooth" or "colored" work with [XPStyle](http://nsis.sourceforge.net/Docs/Chapter4.html#axpstyle) on when the installer runs on Windows XP with a modern theme.

**4.8.1.24 InstType**

install\_type\_name | /NOCUSTOM | /CUSTOMSTRING=str | /COMPONENTSONLYONCUSTOM

Adds an install type to the install type list, or disables the custom install type. There can be as many as 32 types, each one specifying the name of the install type. If the name is prefixed with 'un.' it is an uninstaller install type. The name can contain variables which will be processed at runtime before the components page shows. Another way of changing the [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) name during runtime is the [InstTypeSetText](http://nsis.sourceforge.net/Docs/Chapter4.html#sinsttypesettext) command. The difference is that with [InstTypeSetText](http://nsis.sourceforge.net/Docs/Chapter4.html#sinsttypesettext) you are saving your precious user variables. The first type is the default (generally 'Typical'). If the /NOCUSTOM switch is specified, then the "custom" install type is disabled, and the user has to choose one of the pre-defined install types. Alternatively, if the /CUSTOMSTRING switch is specified, the parameter will override the "Custom" install type text. Alternatively, if the /COMPONENTSONLYONCUSTOM flag is specified, the component list will only be shown if the "Custom" install type is selected.

Accepts variables for type names. If variables are used, they must be initialized before the components page is created.

**4.8.1.25 LicenseBkColor**

color | **/gray** | /windows

Sets the background color of the license data. Color is specified using the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). Default is '/gray'. You can also use the Windows OS defined color by using '/windows'.

**4.8.1.26 LicenseData**

licdata.(txt|rtf)

Specifies a text file or a RTF file to use for the license that the user can read. Omit this to not have a license displayed. Note that the file must be in the evil DOS text format (\r\n, yeah!). To define a multilingual license data use [LicenseLangString](http://nsis.sourceforge.net/Docs/Chapter4.html#licenselangstring).

If you make your license file a RTF file it is recommended you edit it with WordPad and not MS Word. Using WordPad will result in a much smaller file.

Use [LicenseLangString](http://nsis.sourceforge.net/Docs/Chapter4.html#licenselangstring) to show a different license for every language.

**4.8.1.27 LicenseForceSelection**

(checkbox [accept\_text] | radiobuttons [accept\_text] [decline\_text] | **off**)

Specifies if the displayed license must be accept explicit or not. This can be done either by a checkbox or by radiobuttons. By default the "next button" is disabled and will only be enabled if the checkbox is enabled or the right radio button is selected. If off is specified the "next button" is enabled by default.

LicenseForceSelection checkbox

LicenseForceSelection checkbox "i accept"

LicenseForceSelection radiobuttons

LicenseForceSelection radiobuttons "i accept"

LicenseForceSelection radiobuttons "i accept" "i decline"

LicenseForceSelection radiobuttons "" "i decline"

LicenseForceSelection off

**4.8.1.28 LicenseText**

[text [button\_text]]

Used to change the default text on the license page.

text: Text above the controls, to the right of the installation icon.

button\_text: Text on the "I Agree" button.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the license page is created.

**4.8.1.29 ManifestDPIAware**

**notset**|true|false

Declare that the installer is DPI-aware. A DPI-aware application is not scaled by the DWM (DPI virtualization) so the text is never blurry. NSIS does not scale the bitmap used by the tree control on the component page and some plugins might have compatibility issues so make sure that you test your installer at different DPI settings if you select *true*.

See [MSDN](http://msdn.microsoft.com/en-us/library/dd464660) for more information about DPI-aware applications.

**4.8.1.30 ManifestSupportedOS**

none|all|WinVista|**Win7|Win8|Win8.1**|Win10|{GUID} [...]

Declare that the installer is compatible with the specified Windows version(s). This adds a SupportedOS entry in the compatibility section of the application manifest. The default list of Win7+Win8+Win8.1 will probably be updated to include newer Windows versions in the future. *none* is the default if [RequestExecutionLevel](http://nsis.sourceforge.net/Docs/Chapter4.html#arequestexecutionlevel) is set to *none* for compatibility reasons.

You can read more about the changes in behavior on [MSDN](http://msdn.microsoft.com/en-us/library/windows/desktop/hh848036).

**4.8.1.31 MiscButtonText**

[back button text [next button text] [cancel button text] [close button text]]

Replaces the default text strings for the four buttons (< Back, Next >, Cancel and Close). If parameters are omitted, the defaults are used.

Accepts variables. If variables are used, they must be initialized in [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.8.1.32 Name**

name [name\_doubled\_ampersands]

Sets the name of the installer. The name is usually simply the product name such as 'MyApp' or 'CrapSoft MyApp'. If you have one or more ampersands (&) in the name, set the second parameter to the same name, only with doubled ampersands. For example, if your product's name is "Foo & Bar", use:

Name "Foo & Bar" "Foo && Bar"

If you have ampersands in the name and use a [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring) for the name, you will have to create another one with doubled ampersands to use as the second parameter.

Accepts variables. If variables are used, they must be initialized in [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.8.1.33 OutFile**

[path\]install.exe

Specifies the output file that the MakeNSIS should write the installer to. This is just the file that MakeNSIS writes, it doesn't affect the contents of the installer.

**4.8.1.34 RequestExecutionLevel**

**none**|user|highest|admin

Specifies the requested execution level for Windows Vista and higher. The value is embedded in the installer and uninstaller's XML manifest and tells Windows which privilege level the installer requires. *user* requests the a normal user's level with no administrative privileges. *highest* will request the highest execution level available for the current user and will cause Windows to prompt the user to verify privilege escalation. The prompt might request for the user's password. *admin* requests administrator level and will cause Windows to prompt the user as well. Specifying *none*, which is also the default, will keep the manifest empty and let Windows decide which execution level is required. Windows automatically identifies NSIS installers and decides administrator privileges are required. Because of this, *none* and *admin* have virtually the same effect.

It's recommended, at least by Microsoft, that every application is marked with a required execution level. Unmarked installers are subject to compatibility mode. Workarounds of this mode include automatically moving any shortcuts created in the user's start menu to all users' start menu. Installers that need not install anything into system folders or write to the local machine registry (HKLM) should specify *user* execution level.

More information about this topic can be found on [MSDN](http://msdn.microsoft.com/en-us/library/bb756929).

**4.8.1.35 SetFont**

[/LANG=lang\_id] font\_face\_name font\_size

Sets the installer font. Please remember that the font you choose must be present on the user's machine as well. Don't use rare fonts that only you have.

Use the /LANG switch if you wish to set a different font for each language. For example:

SetFont /LANG=${LANG\_ENGLISH} "English Font" 9

SetFont /LANG=${LANG\_FRENCH} "French Font" 10

There are two [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring)s named ^Font and ^FontSize which contain the font and font size for every language.

**4.8.1.36 ShowInstDetails**

**hide**|show|nevershow

Sets whether or not the details of the install are shown. Can be 'hide' to hide the details by default, allowing the user to view them, or 'show' to show them by default, or 'nevershow', to prevent the user from ever seeing them. Note that sections can override this using [SetDetailsView](http://nsis.sourceforge.net/Docs/Chapter4.html#setdetailsview).

**4.8.1.37 ShowUninstDetails**

**hide**|show|nevershow

Sets whether or not the details of the uninstall are shown. Can be 'hide' to hide the details by default, allowing the user to view them, or 'show' to show them by default, or 'nevershow', to prevent the user from ever seeing them. Note that sections can override this using [SetDetailsView](http://nsis.sourceforge.net/Docs/Chapter4.html#setdetailsview).

**4.8.1.38 SilentInstall**

**normal**|silent|silentlog

Specifies whether or not the installer should be silent. If it is 'silent' or 'silentlog', all sections that have the SF\_SELECTED flag are installed quietly (you can set this flag using [SectionSetFlags](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsetflags)), with no screen output from the installer itself (the script can still display whatever it wants, use[MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox)'s /SD to specify a default for silent installers). Note that if this is set to 'normal' and the user runs the installer with /S (case sensitive) on the command line, it will behave as if [SilentInstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentinstall) 'silent' was used. Note: see also [LogSet](http://nsis.sourceforge.net/Docs/Chapter4.html#logset).

See [section 4.12](http://nsis.sourceforge.net/Docs/Chapter4.html#silent) for more information.

**4.8.1.39 SilentUnInstall**

**normal**|silent

Specifies whether or not the uninstaller should be silent. If it is 'silent' or 'silentlog', the uninstall section will run quietly, with no screen output from the uninstaller itself (the script can still display whatever it wants, use [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox)'s /SD to specify a default for silent installers). Note that if this is set to 'normal' and the user runs the uninstaller with /S on the command line, it will behave as if [SilentUnInstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentuninstall) 'silent' was used. Note: see also [LogSet](http://nsis.sourceforge.net/Docs/Chapter4.html#logset).

See [section 4.12](http://nsis.sourceforge.net/Docs/Chapter4.html#silent) for more information.

**4.8.1.40 SpaceTexts**

[req text [avail text]]

If parameters are specified, overrides the space required and space available text ("Space required: " and "Space available: " by default). If 'none' is specified as the required text no space texts will be shown.

Accepts variables. If variables are used, they must be initialized before the components page is created.

**4.8.1.41 SubCaption**

[page\_number subcaption]

Overrides the subcaptions for each of the installer pages (0=": License Agreement",1=": Installation Options",2=": Installation Directory", 3=": Installing Files", 4=": Completed"). If you specify an empty string (""), the default will be used (you can however specify " " to achieve a blank string).

You can also set a subcaption (or override the default) using [Caption](http://nsis.sourceforge.net/Docs/Chapter4.html#acaption) inside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block.

Accepts variables. If variables are used, they must be initialized before the relevant page is created.

**4.8.1.42 UninstallButtonText**

text

Changes the text of the button that by default says "Uninstall" in the uninstaller. If no parameter is specified, the default text is used. See also [WriteUninstaller](http://nsis.sourceforge.net/Docs/Chapter4.html#writeuninstaller) (replaces UninstallEXEName).

Accepts variables. If variables are used, they must be initialized before the uninstall button shows.

**4.8.1.43 UninstallCaption**

caption

Sets what the titlebars of the uninstaller will display. By default, it is 'Name Uninstall', where Name is specified with the Name command. You can, however, override it with 'MyApp uninstaller' or whatever. If you specify an empty string (""), the default will be used (you can however specify " " to achieve a blank string).

Accepts variables. If variables are used, they must be initialized in [un.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#unonInit).

**4.8.1.44 UninstallIcon**

[path\]icon.ico

Sets the icon of the uninstaller.

**4.8.1.45 UninstallSubCaption**

page\_number subcaption

Sets the default subcaptions for the uninstaller pages (0=": Confirmation",1=": Uninstalling Files",2=": Completed"). If you specify an empty string (""), the default will be used (you can however specify " " to achieve a blank string).

You can also set a subcaption (or override the default) using [Caption](http://nsis.sourceforge.net/Docs/Chapter4.html#acaption) inside a [PageEx](http://nsis.sourceforge.net/Docs/Chapter4.html#pageex) block.

Accepts variables. If variables are used, they must be initialized before the relevant page is created.

**4.8.1.46 UninstallText**

text [subtext]

Specifies the texts on the uninstaller confirm page.

text: Text above the controls

subtext: Text next to the uninstall location

Accepts variables. If variables are used, they must be initialized before the uninstaller confirm page is created.

**4.8.1.47 WindowIcon**

**on**|off

Sets whether or not the installer's icon is being displayed.

**4.8.1.48 XPStyle**

on|**off**

Sets whether or not an XP manifest will be added to the installer. An XP manifest makes the installer controls use the new XP style when running on Windows XP. This affects the uninstaller too.

**4.8.2 Compiler Flags**

The following commands affect how the compiler generates code and compresses data. Unless otherwise noted, these commands are valid anywhere in the script, and effect every line below where each one is placed (until overridden by another command). They cannot be jumped over using[flow control insutrctions](http://nsis.sourceforge.net/Docs/Chapter4.html#flowcontrol).

For example, in the following script, blah.dat will never be overwritten.

${If} $0 == 0

SetOverwrite on

${Else}

SetOverwrite off

${EndIf}

File blah.dat # overwrite is always off here!

Instead, the following should be used.

${If} $0 == 0

SetOverwrite on

File blah.dat

${Else}

SetOverwrite off

File blah.dat

${EndIf}

**4.8.2.1 AllowSkipFiles**

**on**|off

This command specifies whether the user should be able to skip a file or not. A user has an option to skip a file if [SetOverwrite](http://nsis.sourceforge.net/Docs/Chapter4.html#asetoverwrite) is set to on (default) and the installer fails to open a file for writing when trying to extract a file. If *off* is used the ignore button which allows the user to skip the file will not show and the user will only have an option to abort the installation (Cancel button) or retry opening the file for writing (Retry button). If *on* is used the user will have an option to skip the file (error flag will be set - see [SetOverwrite](http://nsis.sourceforge.net/Docs/Chapter4.html#asetoverwrite)).

**4.8.2.2 FileBufSize**

buffer\_size\_in\_mb

This command sets the size of the compiler's internal file buffers. This command allows you to control the compiler's memory usage by limiting how much of a given file it will load into memory at once. Since the compiler needs both input and output, twice the memory size specified could be used at any given time for file buffers. This command does not limit the compression buffers which could take another couple of MB, neither does it limit the compiler's other internal buffers, but those shouldn't normally top 1MB anyway. Specifying a very small number could decrease performance. Specifying a very large number could exhaust system resources and force the compiler to cancel the compilation process. The default value is 32MB.

**4.8.2.3 SetCompress**

**auto**|force|off

This command sets the compress flag which is used by the installer to determine whether or not data should be compressed. Typically the [SetCompress](http://nsis.sourceforge.net/Docs/Chapter4.html#asetcompress) flag will affect the commands after it, and the last [SetCompress](http://nsis.sourceforge.net/Docs/Chapter4.html#asetcompress) command in the file also determines whether or not the install info section and uninstall data of the installer is compressed. If compressflag is 'auto', then files are compressed if the compressed size is smaller than the uncompressed size. If compressflag is set to 'force', then the compressed version is always used. If compressflag is 'off' then compression is not used (which can be faster).

Note that this option has no effect when solid compression is used.

**4.8.2.4 SetCompressor**

[/SOLID] [/FINAL] **zlib**|bzip2|lzma

This command sets the compression algorithm used to compress files/data in the installer. It can only be used outside of sections and functions and before any data is compressed. Different compression methods can not be used for different files in the same installer. It is recommended to use it on the very top of the script to avoid compilation errors.

Three compression methods are supported: ZLIB, BZIP2 and LZMA.

ZLIB (the default) uses the deflate algorithm, it is a quick and simple method. With the default compression level it uses about 300 KB of memory.

BZIP2 usually gives better compression ratios than ZLIB, but it is a bit slower and uses more memory. With the default compression level it uses about 4 MB of memory.

LZMA is a new compression method that gives very good compression ratios. The decompression speed is high (10-20 MB/s on a 2 GHz CPU), the compression speed is lower. The memory size that will be used for decompression is the dictionary size plus a few KBs, the default is 8 MB.

If */FINAL* is used, subsequent calls to [SetCompressor](http://nsis.sourceforge.net/Docs/Chapter4.html#asetcompressor) will be ignored.

If */SOLID* is used, all of the installer data is compressed in one block. This results in greater compression ratios.

**4.8.2.5 SetCompressorDictSize**

dict\_size\_mb

Sets the dictionary size in megabytes (MB) used by the LZMA compressor (default is 8 MB).

**4.8.2.6 SetDatablockOptimize**

**on**|off

This command tells the compiler whether or not to do datablock optimizations. Datablock optimizations have the compiler check to see if any data being added to the data block is already in the data block, and if so, it is simply referenced as opposed to added (can save a little bit of size). It is highly recommended to leave this option on.

**4.8.2.7 SetDateSave**

**on**|off

This command sets the file date/time saving flag which is used by the [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) command to determine whether or not to save the last write date and time of the file, so that it can be restored on installation. Valid flags are 'on' and 'off'. 'on' is the default.

**4.8.2.8 SetOverwrite**

**on**|off|try|ifnewer|ifdiff|lastused

This command sets the overwrite flag which is used by the [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) command to determine whether or not the file should overwrite any existing files that are present. If overwriteflag is 'on', files are overwritten (this is the default). If overwriteflag is 'off', files that are already present are not overwritten. If overwriteflag is 'try', files are overwritten if possible (meaning that if the file is not able to be written to, it is skipped without any user interaction). If overwriteflag is 'ifnewer', then files are only overwritten if the existing file is older than the new file. If overwriteflag is 'ifdiff', then files are only overwritten if the existing file is older or newer than the new file. Note that when in 'ifnewer' or 'ifdiff' mode, the destination file's date is set, regardless of what [SetDateSave](http://nsis.sourceforge.net/Docs/Chapter4.html#asetdatesave) is set to.

SetOverwrite off

File program.cfg # config file we don't want to overwrite

SetOverwrite on

**4.8.2.9 Unicode**

true|**false**

Generate a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode). It can only be used outside of sections and functions and before any data is compressed.

**4.8.3 Version Information**

**4.8.3.1 VIAddVersionKey**

[/LANG=lang\_id] keyname value

Adds a field in the Version Tab of the File Properties. This can either be a field provided by the system or a user defined field. The following fields are provided by the System:

* ProductName
* Comments
* CompanyName
* LegalCopyright
* FileDescription
* FileVersion
* ProductVersion
* InternalName
* LegalTrademarks
* OriginalFilename
* PrivateBuild
* SpecialBuild

The name of these fields are translated on the target system, whereas user defined fields remain untranslated.

VIAddVersionKey /LANG=${LANG\_ENGLISH} "ProductName" "Test Application"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "Comments" "A test comment"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "CompanyName" "Fake company"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "LegalTrademarks" "Test Application is a trademark of Fake company"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "LegalCopyright" "© Fake company"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "FileDescription" "Test Application"

VIAddVersionKey /LANG=${LANG\_ENGLISH} "FileVersion" "1.2.3"

**4.8.3.2 VIProductVersion**

[version\_string\_X.X.X.X]

Sets the Product Version in the VS\_FIXEDFILEINFO version information block.

VIProductVersion "1.2.3.4"

**4.8.3.3 VIFileVersion**

[version\_string\_X.X.X.X]

Sets the File Version in the VS\_FIXEDFILEINFO version information block (You should also set the FileVersion string with VIAddVersionKey so the information is displayed at the top of the Version Tab in the Properties of the file). If you don't provide a File Version the Product Version is used in the VS\_FIXEDFILEINFO block.

VIFileVersion "1.2.3.4"

**4.9 Instructions**

**4.9.1 Basic Instructions**

The instructions that NSIS uses for scripting are sort of a cross between PHP and assembly. There are no real high level language constructs, but the instructions themselves are (for the most part) high level, and you have handy string capability (i.e. you don't have to worry about concatenating strings, etc). You essentially have 25 registers (20 general purpose, 5 special purpose), and a stack.

**4.9.1.1 Delete**

[/REBOOTOK] file

Delete file (which can be a file or wildcard, but should be specified with a full path) from the target system. If /REBOOTOK is specified and the file cannot be deleted then the file is deleted when the system reboots -- if the file will be deleted on a reboot, the reboot flag will be set. The error flag is set if files are found and cannot be deleted. The error flag is not set from trying to delete a file that does not exist.

Delete $INSTDIR\somefile.dat

**4.9.1.2 Exec**

command

Execute the specified program and continue immediately. Note that the file specified must exist on the target system, not the compiling system. $OUTDIR is used for the working directory. The error flag is set if the process could not be launched. Note, if the command could have spaces, you should put it in quotes to delimit it from parameters. e.g.: Exec '"$INSTDIR\command.exe" parameters'. If you don't put it in quotes it will *not* work on Windows 9x with or without parameters.

Exec '"$INSTDIR\someprogram.exe"'

Exec '"$INSTDIR\someprogram.exe" some parameters'

**4.9.1.3 ExecShell**

action command [parameters] [SW\_SHOWDEFAULT | SW\_SHOWNORMAL | SW\_SHOWMAXIMIZED | SW\_SHOWMINIMIZED | SW\_HIDE]

Execute the specified program using ShellExecute. Note that action is usually "open", "print", etc, but can be an empty string to use the default action. Parameters and the show type are optional. $OUTDIR is used for the working directory. The error flag is set if the process could not be launched.

ExecShell "open" "http://nsis.sf.net/"

ExecShell "open" "$INSTDIR\readme.txt"

ExecShell "print" "$INSTDIR\readme.txt"

**4.9.1.4 ExecWait**

command [user\_var(exit code)]

Execute the specified program and wait for the executed process to quit. See [Exec](http://nsis.sourceforge.net/Docs/Chapter4.html#exec) for more information. If no output variable is specified [ExecWait](http://nsis.sourceforge.net/Docs/Chapter4.html#execwait) sets the error flag if the program executed returns a nonzero error code, or if there is an error. If an output variable is specified, [ExecWait](http://nsis.sourceforge.net/Docs/Chapter4.html#execwait) sets the variable with the exit code (and only sets the error flag if an error occurs; if an error occurs the contents of the user variable are undefined). Note, if the command could have spaces, you should put it in quotes to delimit it from parameters. e.g.: ExecWait '"$INSTDIR\command.exe" parameters'. If you don't put it in quotes it will *not* work on Windows 9x with or without parameters.

ExecWait '"$INSTDIR\someprogram.exe"'

ExecWait '"$INSTDIR\someprogram.exe"' $0

DetailPrint "some program returned $0"

**4.9.1.5 File**

[/nonfatal] [/a] ([/r] [/x file|wildcard [...]] (file|wildcard) [...] | /oname=file.dat infile.dat)

Adds file(s) to be extracted to the current output path ($OUTDIR).

* Note that the output file name is $OUTDIR\filename\_portion\_of\_file.
* Use /oname=X switch to change the output name. X may contain variables and can be a fully qualified path or a relative path in which case it will be appended to $OUTDIR set by [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath). When using this switch, only one file can be specified. If the output name contains spaces, quote the entire parameter, including /oname, as shown in the examples below.
* Wildcards are supported.
* If the /r switch is used, matching files and directories are recursively searched for in subdirectories. If just one path segment is specified (e.g. File /r something), the current directory will be recursively searched. If more than one segment is specified (e.g. File /r something\\*.\*), the last path segment will be used as the matching condition and the rest for the directory to search recursively. If a directory name matches, all of its contents is added recursively. Directory structure is preserved.
* Use the /x switch to exclude files or directories.
* If the /a switch is used, the attributes of the file(s) added will be preserved.
* The [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) command sets the error flag if overwrite mode is set to 'try' and the file could not be overwritten, or if the overwrite mode is set to 'on' and the file could not be overwritten and the user selects ignore.
* If the /nonfatal switch is used and no files are found, a warning will be issued instead of an error.

File something.exe

File /a something.exe

File \*.exe

File /r \*.dat

File /r data

File /oname=temp.dat somefile.ext

File /oname=$TEMP\temp.dat somefile.ext

File "/oname=$TEMP\name with spaces.dat" somefile.ext

File /nonfatal "a file that might not exist"

File /r /x CVS myproject\\*.\*

File /r /x \*.res /x \*.obj /x \*.pch source\\*.\*

**Note:** when using the */r* switch, both matching directories and files will be searched. This is always done with or without the use of wildcards, even if the given path perfectly matches one directory. That means, the following directory structure:

<DIR> something

file.dat

another.dat

<DIR> dir

something

<DIR> dir2

file2.dat

<DIR> another

<DIR> something

readme.txt

with the following *File* usage:

File /r something

will match the directory named *something* on the root directory, the file named *something* in the directory named *dir* and the directory named *something* in the directory named *another*. To match only the directory named *something* on the root directory, use the following:

File /r something\\*.\*

When adding *\\*.\**, it will be used as the matching condition and *something* will be used as the directory to search. When only *something* is specified, the current directory will be recursively searched for every file and directory named *something* and *another\something* will be matched.

**4.9.1.6 Rename**

[/REBOOTOK] source\_file dest\_file

Rename source\_file to dest\_file. You can use it to move a file from anywhere on the system to anywhere else and you can move a directory to somewhere else on the same drive. The destination file must not exist or the move will fail (unless you are using /REBOOTOK). If /REBOOTOK is specified, and the file cannot be moved (if, for example, the destination exists), then the file is moved when the system reboots. If the file will be moved on a reboot, the reboot flag will be set. The error flag is set if the file cannot be renamed (and /REBOOTOK is not used) or if the source file does not exist.

If no absolute path is specified the current folder will be used. The current folder is the folder set using the last [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) instruction. If you have not used [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) the current folder is [$EXEDIR](http://nsis.sourceforge.net/Docs/Chapter4.html#varother).

Rename $INSTDIR\file.ext $INSTDIR\file.dat

**4.9.1.7 ReserveFile**

[/nonfatal] [/r] [/x file|wildcard [...]] file [file...] | [/nonfatal] /plugin file.dll

Reserves a file in the data block for later use. Files are added to the compressed data block in the order they appear in the script. Functions, however, are not necessarily called in the order they appear in the script. Therefore, if you add a file in a function called early but put the function at the end of the script, all of the files added earlier will have to be decompressed to get to the required file. This process can take a long time if there a lot of files. [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit) is one such function. It is called at the very beginning, before anything else appears. If you put it at the very end of the script, extract some files in it and have lots of files added before it, the installer might take a very long time to load. This is where this command comes useful, allowing you to speed up the loading process by including the file at the top of the data block instead of letting NSIS seek all the way down to the bottom of the *compressed* data block.

Use /plugin to reserve a plugin in ${NSISDIR}\Plugins\\*.

See [File](http://nsis.sourceforge.net/Docs/Chapter4.html#file) for more information about the parameters.

**4.9.1.8 RMDir**

[/r] [/REBOOTOK] directory\_name

Remove the specified directory (fully qualified path with no wildcards). Without /r, the directory will only be removed if it is completely empty. If /r is specified, the directory will be removed recursively, so all directories and files in the specified directory will be removed. If /REBOOTOK is specified, any file or directory which could not have been removed during the process will be removed on reboot -- if any file or directory will be removed on a reboot, the reboot flag will be set. The error flag is set if any file or directory cannot be removed.

RMDir $INSTDIR

RMDir $INSTDIR\data

RMDir /r /REBOOTOK $INSTDIR

RMDir /REBOOTOK $INSTDIR\DLLs

Note that the current working directory can not be deleted. The current working directory is set by [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath). For example, the following example will not delete the directory.

SetOutPath $TEMP\dir

RMDir $TEMP\dir

The next example will succeed in deleting the directory.

SetOutPath $TEMP\dir

SetOutPath $TEMP

RMDir $TEMP\dir

**Warning:** using *RMDir /r $INSTDIR* in the uninstaller is not safe. Though it is unlikely, the user might select to install to the Program Files folder and so this command will wipe out the entire Program Files folder, including other programs that has nothing to do with the uninstaller. The user can also put other files but the program's files and would expect them to get deleted with the program. Solutions are [available](http://nsis.sourceforge.net/Uninstall_only_installed_files) for easily uninstalling only files which were installed by the installer.

**4.9.1.9 SetOutPath**

outpath

Sets the output path ($OUTDIR) and creates it (recursively if necessary), if it does not exist. Must be a full pathname, usually is just $INSTDIR (you can specify $INSTDIR if you are lazy with a single "-").

SetOutPath $INSTDIR

File program.exe

**4.9.2 Registry, INI, File Instructions**

In all of the below registry instructions use an empty string (just two quotes with nothing between them - "") as the key name to specify the default key which is shown as (Default) in regedit.exe.

If a full path is not specified for any of the INI handling instructions, the Windows directory will be used.

**4.9.2.1 DeleteINISec**

ini\_filename section\_name

Deletes the entire section [section\_name] from ini\_filename. If the section could not be removed from the ini file, the error flag is set. It does not set the error flag if the section could not be found.

WriteINIStr $TEMP\something.ini section1 something 123

WriteINIStr $TEMP\something.ini section1 somethingelse 1234

WriteINIStr $TEMP\something.ini section2 nsis true

DeleteINISec $TEMP\something.ini section1

**4.9.2.2 DeleteINIStr**

ini\_filename section\_name str\_name

Deletes the string str\_name from section [section\_name] from ini\_filename. If the string could not be removed from the ini file, the error flag is set. It does not set the error flag if the string could not be found.

WriteINIStr $TEMP\something.ini section1 something 123

WriteINIStr $TEMP\something.ini section1 somethingelse 1234

DeleteINIStr $TEMP\something.ini section1 somethingelse

**4.9.2.3 DeleteRegKey**

[/ifempty] root\_key subkey

Deletes a registry key. If /ifempty is specified, the registry key will only be deleted if it has no subkeys (otherwise, the whole registry tree will be removed). Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). The error flag is set if the key could not be removed from the registry (or if it didn't exist to begin with).

DeleteRegKey HKLM "Software\My Company\My Software"

DeleteRegKey /ifempty HKLM "Software\A key that might have subkeys"

**4.9.2.4 DeleteRegValue**

root\_key subkey key\_name

Deletes a registry value. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). The error flag is set if the value could not be removed from the registry (or if it didn't exist to begin with).

DeleteRegValue HKLM "Software\My Company\My Software" "some value"

**4.9.2.5 EnumRegKey**

user\_var(output) root\_key subkey index

Set user variable $x with the name of the 'index'th registry key in root\_key\Subkey. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). Returns an empty string if there are no more keys, and returns an empty string and sets the error flag if there is an error.

StrCpy $0 0

loop:

EnumRegKey $1 HKLM Software $0

StrCmp $1 "" done

IntOp $0 $0 + 1

MessageBox MB\_YESNO|MB\_ICONQUESTION "$1$\n$\nMore?" IDYES loop

done:

**4.9.2.6 EnumRegValue**

user\_var(output) root\_key subkey index

Set user variable $x with the name of the 'index'th registry value in root\_key\Subkey. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). Returns an empty string and sets the error flag if there are no more values or if there is an error.

StrCpy $0 0

loop:

ClearErrors

EnumRegValue $1 HKLM Software\Microsoft\Windows\CurrentVersion $0

IfErrors done

IntOp $0 $0 + 1

ReadRegStr $2 HKLM Software\Microsoft\Windows\CurrentVersion $1

MessageBox MB\_YESNO|MB\_ICONQUESTION "$1 = $2$\n$\nMore?" IDYES loop

done:

**4.9.2.7 ExpandEnvStrings**

user\_var(output) string

Expands environment variables in *string* into the user variable *$x*. If an environment variable doesn't exist, it will not be replaced. For example, if you use "%var%" and var doesn't exists, the result will be "%var%". If there is an error, the variable is set to empty, and the error flag is set.

ExpandEnvStrings $0 "WINDIR=%WINDIR%$\nTEMP=%TEMP%"

**4.9.2.8 FlushINI**

ini\_filename

Flushes the INI file's buffers. Windows 9x keeps all changes to the INI file in memory. This command causes the changes to be written to the disk immediately. Use it if you edit the INI manually, delete it, move it or copy it right after you change it with [WriteINIStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeinistr), [DeleteINISec](http://nsis.sourceforge.net/Docs/Chapter4.html#deleteinisec) or[DeleteINStr](http://nsis.sourceforge.net/Docs/Chapter4.html#deleteinistr).

WriteINIStr $TEMP\something.ini test test test

FlushINI $TEMP\something.ini

Delete $TEMP\something.ini

**4.9.2.9 ReadEnvStr**

user\_var(output) name

Reads from the environment string "name" and sets the value into the user variable $x. If there is an error reading the string, the user variable is set to empty, and the error flag is set.

ReadEnvStr $0 WINDIR

ReadEnvStr $1 TEMP

**4.9.2.10 ReadINIStr**

user\_var(output) ini\_filename section\_name entry\_name

Reads from entry\_name in [section\_name] of ini\_filename and stores the value into user variable $x. The error flag will be set and $x will be assigned to an empty string if the entry is not found.

ReadINIStr $0 $INSTDIR\winamp.ini winamp outname

**4.9.2.11 ReadRegDWORD**

user\_var(output) root\_key sub\_key name

Reads a 32 bit DWORD from the registry into the user variable $x. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). The error flag will be set and $x will be set to an empty string ("" which is 0) if the DWORD is not present. If the value is present, but is not a DWORD, it will be read as a string and the error flag will be set.

ReadRegDWORD $0 HKLM Software\NSIS VersionBuild

**4.9.2.12 ReadRegStr**

user\_var(output) root\_key sub\_key name

Reads from the registry into the user variable $x. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). The error flag will be set and $x will be set to an empty string ("") if the string is not present. If the value is present, but is of type REG\_DWORD, it will be read and converted to a string and the error flag will be set.

ReadRegStr $0 HKLM Software\NSIS ""

DetailPrint "NSIS is installed at: $0"

**4.9.2.13 WriteINIStr**

ini\_filename section\_name entry\_name value

Writes entry\_name=value into [section\_name] of ini\_filename. The error flag is set if the string could not be written to the ini file.

WriteINIStr $TEMP\something.ini section1 something 123

WriteINIStr $TEMP\something.ini section1 somethingelse 1234

WriteINIStr $TEMP\something.ini section2 nsis true

**4.9.2.14 WriteRegBin**

root\_key subkey key\_name valuedata

This command writes a block of binary data to the registry. Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). Valuedata is in hexadecimal (e.g. DEADBEEF01223211151). The error flag is set if the binary data could not be written to the registry. If the registry key doesn't exist it will be created.

WriteRegBin HKLM "Software\My Company\My Software" "Binary Value" DEADBEEF01223211151

**4.9.2.15 WriteRegDWORD**

root\_key subkey key\_name value

This command writes a dword (32 bit integer) to the registry (a user variable can be specified). Valid values for root\_key are listed under [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr). The error flag is set if the dword could not be written to the registry. If the registry key doesn't exist it will be created.

WriteRegDWORD HKLM "Software\My Company\My Software" "DWORD Value" 0xDEADBEEF

**4.9.2.16 WriteRegStr**

root\_key subkey key\_name value

Write a string to the registry. See [WriteRegExpandStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregexpandstr) for more details.

WriteRegStr HKLM "Software\My Company\My Software" "String Value" "dead beef"

**4.9.2.17 WriteRegExpandStr**

root\_key subkey key\_name value

Write a string to the registry. *root\_key* must be one of:

* *HKCR* or *HKEY\_CLASSES\_ROOT*
* *HKLM* or *HKEY\_LOCAL\_MACHINE*
* *HKCU* or *HKEY\_CURRENT\_USER*
* *HKU* or *HKEY\_USERS*
* *HKCC* or *HKEY\_CURRENT\_CONFIG*
* *HKDD* or *HKEY\_DYN\_DATA*
* *HKPD* or *HKEY\_PERFORMANCE\_DATA*
* *SHCTX* or *SHELL\_CONTEXT*

If *root\_key* is *SHCTX* or *SHELL\_CONTEXT*, it will be replaced with *HKLM* if [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) is set to *all* and with *HKCU* if [SetShellVarContext](http://nsis.sourceforge.net/Docs/Chapter4.html#setshellvarcontext) is set to *current*.

The error flag is set if the string could not be written to the registry. The type of the string will be REG\_SZ for [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr), or REG\_EXPAND\_STR for [WriteRegExpandStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregexpandstr). If the registry key doesn't exist it will be created.

WriteRegExpandStr HKLM "Software\My Company\My Software" "Expand String Value" "%WINDIR%\notepad.exe"

**4.9.3 General Purpose Instructions**

**4.9.3.1 CallInstDLL**

dllfile function\_name

Calls a function named *function\_name* inside a NSIS extension DLL, a plug-in. See the [example plugin](http://nsis.sourceforge.net/Examples/Plugin/) for how to make one. Extension DLLs can access the stack and variables. Note: To automatically extract and call plug-in DLLs, use a plug-in command instead of [CallInstDLL](http://nsis.sourceforge.net/Docs/Chapter4.html#callinstdll).

Push "a parameter"

Push "another parameter"

CallInstDLL $INSTDIR\somedll.dll somefunction

For easier plug-in handling, use the new [plug-in call syntax](http://nsis.sourceforge.net/Docs/Chapter4.html#plugindlls).

**4.9.3.2 CopyFiles**

[/SILENT] [/FILESONLY] filespec\_on\_destsys destination\_path [size\_of\_files\_in\_kb]

Copies files from the source to the destination on the installing system. Useful with $EXEDIR if you want to copy from installation media, or to copy from one place to another on the system. You might see a Windows status window of the copy operation if the operation takes a lot of time (to disable this, use /SILENT). The last parameter can be used to specify the size of the files that will be copied (in kilobytes), so that the installer can approximate the disk space requirements. On error, or if the user cancels the copy (only possible when /SILENT was omitted), the error flag is set. If /FILESONLY is specified, only files are copied.

Fully-qualified path names should always be used with this instruction. Using relative paths will have unpredictable results.

CreateDirectory $INSTDIR\backup

CopyFiles $INSTDIR\\*.dat $INSTDIR\backup

**4.9.3.3 CreateDirectory**

path\_to\_create

Creates (recursively if necessary) the specified directory. The error flag is set if the directory couldn't be created.

You should always specify an absolute path.

CreateDirectory $INSTDIR\some\directory

**4.9.3.4 CreateShortcut**

[/NoWorkingDir] link.lnk target.file [parameters [icon.file [icon\_index\_number [start\_options [keyboard\_shortcut [description]]]]]]

Creates a shortcut 'link.lnk' that links to 'target.file', with optional parameters 'parameters'. The icon used for the shortcut is 'icon.file,icon\_index\_number'; for default icon settings use empty strings for both icon.file and icon\_index\_number. start\_options should be one of: *SW\_SHOWNORMAL*,*SW\_SHOWMAXIMIZED*, *SW\_SHOWMINIMIZED*, or an empty string. keyboard\_shortcut should be in the form of 'flag|c' where flag can be a combination (using |) of: *ALT*, *CONTROL*, *EXT*, or *SHIFT*. c is the character to use (a-z, A-Z, 0-9, F1-F24, etc). Note that no spaces are allowed in this string. A good example is "ALT|CONTROL|F8". $OUTDIR is used for the working directory. You can change it by using [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) before creating the Shortcut or use /NoWorkingDir if you don't need to set the working directory. description should be the description of the shortcut, or comment as it is called under XP. The error flag is set if the shortcut cannot be created (i.e. either of the paths (link or target) does not exist, or some other error).

CreateDirectory "$SMPROGRAMS\My Company"

CreateShortcut "$SMPROGRAMS\My Company\My Program.lnk" "$INSTDIR\My Program.exe" \

"some command line parameters" "$INSTDIR\My Program.exe" 2 SW\_SHOWNORMAL \

ALT|CONTROL|SHIFT|F5 "a description"

**4.9.3.5 GetDLLVersion**

filename user\_var(high dword output) user\_var(low dword output)

Gets the version information from the DLL (or any other executable containing version information) in "filename". Sets the user output variables with the high and low dwords of version information on success; on failure the outputs are empty and the error flag is set. The following example reads the DLL version and copies a human readable version of it into $0:

GetDllVersion "$INSTDIR\MyDLL.dll" $R0 $R1

IntOp $R2 $R0 / 0x00010000

IntOp $R3 $R0 & 0x0000FFFF

IntOp $R4 $R1 / 0x00010000

IntOp $R5 $R1 & 0x0000FFFF

StrCpy $0 "$R2.$R3.$R4.$R5"

**4.9.3.6 GetDLLVersionLocal**

localfilename user\_var(high dword output) user\_var(low dword output)

This is similar to [GetDLLVersion](http://nsis.sourceforge.net/Docs/Chapter4.html#getdllversion), only it acts on the system building the installer (it actually compiles into two [StrCpy](http://nsis.sourceforge.net/Docs/Chapter4.html#StrCpy) commands). Sets the two output variables with the DLL version information of the DLL on the build system. Use [!getdllversion](http://nsis.sourceforge.net/Docs/Chapter5.html" \l "ppgetdllversion) if you need to use the values with[VIProductVersion](http://nsis.sourceforge.net/Docs/Chapter4.html#viproductversion).

**4.9.3.7 GetFileTime**

filename user\_var(high dword output) user\_var(low dword output)

Gets the last write time of "filename". Sets the user output variables with the high and low dwords of the timestamp on success; on failure the outputs are empty and the error flag is set.

**4.9.3.8 GetFileTimeLocal**

localfilename user\_var(high dword output) user\_var(low dword output)

This is similar to [GetFileTime](http://nsis.sourceforge.net/Docs/Chapter4.html#getfiletime), only it acts on the system building the installer (it actually compiles into two [StrCpy](http://nsis.sourceforge.net/Docs/Chapter4.html#StrCpy) commands). Sets the two output variables with the file timestamp of the file on the build system.

**4.9.3.9 GetFullPathName**

[/SHORT] user\_var(output) path\_or\_file

Assign to the user variable $x, the full path of the file specified. If the path portion of the parameter is not found, the error flag will be set and $x will be empty. If /SHORT is specified, the path is converted to the short filename form. However, if /SHORT is not specified, the path isn't converted to its long filename form. To get the long filename, call GetLongPathName using the System plug-in. Note that GetLongPathName is only available on Windows 98, Windows 2000 and above.

StrCpy $INSTDIR $PROGRAMFILES\NSIS

SetOutPath $INSTDIR

GetFullPathName $0 ..

DetailPrint $0 # will print C:\Program Files

GetFullPathName /SHORT $0 $INSTDIR

DetailPrint $0 # will print C:\Progra~1\NSIS

StrCpy $0 C:\Progra~1\NSIS

System::Call 'kernel32::GetLongPathName(t r0, t .r1, i ${NSIS\_MAX\_STRLEN}) i .r2'

StrCmp $2 error +2

StrCpy $0 $1

DetailPrint $0 # will print C:\Program Files\NSIS, where supported

**4.9.3.10 GetTempFileName**

user\_var(output) base\_dir

Assign to the user variable $x, the name of a temporary file. The file will have been created, so you can then overwrite it with what you please. The name of the temporary file is guaranteed to be unique. If to want the temporary file to be created in another directory than the Windows temp directory, specify a base\_dir. [Delete](http://nsis.sourceforge.net/Docs/Chapter4.html#delete) the file when done with it.

GetTempFileName $0

File /oname=$0 something.dat

# do something with something.dat

Delete $0

**4.9.3.11 SearchPath**

user\_var(output) filename

Assign to the user variable $x, the full path of the file named by the second parameter. The error flag will be set and $x will be empty if the file cannot be found. Uses [SearchPath](http://nsis.sourceforge.net/Docs/Chapter4.html#searchpath)() to search the system paths for the file.

**4.9.3.12 SetFileAttributes**

filename attribute1|attribute2|...

Sets the file attributes of 'filename'. Valid attributes can be combined with | and are:

* *NORMAL* or *FILE\_ATTRIBUTE\_NORMAL* (you can use 0 to abbreviate this)
* *ARCHIVE* or *FILE\_ATTRIBUTE\_ARCHIVE*
* *HIDDEN* or *FILE\_ATTRIBUTE\_HIDDEN*
* *OFFLINE* or *FILE\_ATTRIBUTE\_OFFLINE*
* *READONLY* or *FILE\_ATTRIBUTE\_READONLY*
* *SYSTEM* or *FILE\_ATTRIBUTE\_SYSTEM*
* *TEMPORARY* or *FILE\_ATTRIBUTE\_TEMPORARY*

The error flag will be set if the file's attributes cannot be set (i.e. the file doesn't exist, or you don't have the right permissions). You can only set attributes. It's not possible to unset them. If you want to remove an attribute use NORMAL. This way all attributes are erased. This command doesn't support wildcards.

**4.9.3.13 RegDLL**

dllfile [entrypoint\_name]

Loads the specified DLL and calls DllRegisterServer (or entrypoint\_name if specified). The error flag is set if an error occurs (i.e. it can't load the DLL, initialize OLE, find the entry point, or the function returned anything other than ERROR\_SUCCESS (=0)).

Use [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) to set the current directory for DLLs that depend on other DLLs that are now in the path or in the Windows directory. For example, if foo.dll depends on bar.dll which is located in $INSTDIR use:

SetOutPath $INSTDIR

RegDLL $INSTDIR\foo.dll

**4.9.3.14 UnRegDLL**

dllfile

Loads the specified DLL and calls DllUnregisterServer. The error flag is set if an error occurs (i.e. it can't load the DLL, initialize OLE, find the entry point, or the function returned anything other than ERROR\_SUCCESS (=0)).

**4.9.4 Flow Control Instructions**

**4.9.4.1 Abort**

user\_message

Cancels the install, stops execution of script, and displays user\_message in the status display. Note: you can use this from [Callback functions](http://nsis.sourceforge.net/Docs/Chapter4.html#callbacks) to do special things. [Page callbacks](http://nsis.sourceforge.net/Docs/Chapter4.html#pages) also uses [Abort](http://nsis.sourceforge.net/Docs/Chapter4.html#abort) for special purposes.

Abort

Abort "can't install"

**4.9.4.2 Call**

function\_name | :label\_name | user\_var(input)

Calls the function named *function\_name*, the label named *label\_name*, or a variable that specifies an address. An address is returned by [GetCurrentAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getcurrentaddress), [GetFunctionAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getfunctionaddress) or [GetLabelAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getlabeladdress). A call returns when it encounters a [Return](http://nsis.sourceforge.net/Docs/Chapter4.html#return) instruction. Sections and functions are automatically ended with a [Return](http://nsis.sourceforge.net/Docs/Chapter4.html#return) instruction. Uninstall functions cannot be called from installer functions and sections, and vice-versa.

Function func

Call :label

DetailPrint "#1: This will only appear 1 time."

label:

DetailPrint "#2: This will appear before and after message #1."

Call :.global\_label

FunctionEnd

Section

Call func

Return

.global\_label:

DetailPrint "#3: The global label was called"

SectionEnd

**4.9.4.3 ClearErrors**

Clears the error flag.

ClearErrors

IfErrors 0 +2

MessageBox MB\_OK "this message box will never show"

**4.9.4.4 GetCurrentAddress**

user\_var(output)

Gets the address of the current instruction (the [GetCurrentAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getcurrentaddress)) and stores it in the output user variable. This user variable then can be passed to [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call) or [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto).

Function func

DetailPrint "function"

IntOp $0 $0 + 2

Call $0

DetailPrint "function end"

FunctionEnd

Section

DetailPrint "section"

DetailPrint "section"

GetCurrentAddress $0

Goto callFunc

DetailPrint "back to section"

Return

callFunc:

Call func

DetailPrint "section end"

SectionEnd

**4.9.4.5 GetFunctionAddress**

user\_var(output) function\_name

Gets the address of the function and stores it in the output user variable. This user variable then can be passed to [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call) or [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto). Note that if you [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto) an address which is the output of [GetFunctionAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getfunctionaddress), your function will never be returned to (when the function you Goto'd to returns, you return instantly).

Function func

DetailPrint "function"

FunctionEnd

Section

GetFunctionAddress $0 func

Call $0

SectionEnd

**4.9.4.6 GetLabelAddress**

user\_var(output) label

Gets the address of the label and stores it in the output user variable. This user variable then can be passed to [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call) or [Goto](http://nsis.sourceforge.net/Docs/Chapter4.html#goto). Note that you may only call this with labels accessible from your function, but you can call it from anywhere (which is potentially dangerous). Note that if you [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call) the output of [GetLabelAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getlabeladdress), code will be executed until it Return's (explicitly or implicitly at the end of a function), and then you will be returned to the statement after the [Call](http://nsis.sourceforge.net/Docs/Chapter4.html#call).

label:

DetailPrint "label"

GetLabelAddress $0 label

IntOp $0 $0 + 4

Goto $0

DetailPrint "done"

**4.9.4.7 Goto**

label\_to\_jump\_to | +offset| -offset| user\_var(target)

If label is specified, goto the label 'label\_to\_jump\_to:'.

If +offset or -offset is specified, jump is relative by offset instructions. Goto +1 goes to the next instruction, Goto -1 goes to the previous instruction, etc.

If a user variable is specified, jumps to absolute address (generally you will want to get this value from a function like [GetLabelAddress](http://nsis.sourceforge.net/Docs/Chapter4.html#getlabeladdress)). Compiler flag commands and [SectionIn](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectionin) aren't instructions so jumping over them has no effect.

Goto label

Goto +2

Goto -2

Goto $0

**4.9.4.8 IfAbort**

label\_to\_goto\_if\_abort [label\_to\_goto\_if\_no\_abort]

If abort is called it will "return" true. This can happen if the user chose abort on a file that failed to create (or overwrite) or if the user aborted by hand. This function can only be called from the leave function of the instfiles [page](http://nsis.sourceforge.net/Docs/Chapter4.html#page).

Page instfiles "" "" instfilesLeave

Function instfilesLeave

IfAbort 0 +2

MessageBox MB\_OK "user aborted"

FunctionEnd

**4.9.4.9 IfErrors**

jumpto\_iferror [jumpto\_ifnoerror]

Checks and clears the error flag, and if it is set, it will goto jumpto\_iferror, otherwise it will goto jumpto\_ifnoerror. The error flag is set by other instructions when a recoverable error (such as trying to delete a file that is in use) occurs.

ClearErrors

File file.dat

IfErrors 0 +2

Call ErrorHandler

**4.9.4.10 IfFileExists**

file\_to\_check\_for jump\_if\_present [jump\_otherwise]

Checks for existence of file(s) file\_to\_check\_for (which can be a wildcard, or a directory), and Gotos jump\_if\_present if the file exists, otherwise Gotos jump\_otherwise. If you want to check to see if a file is a directory, use [IfFileExists](http://nsis.sourceforge.net/Docs/Chapter4.html#iffileexists) DIRECTORY\\*.\*

IfFileExists $WINDIR\notepad.exe 0 +2

MessageBox MB\_OK "notepad is installed"

**4.9.4.11 IfRebootFlag**

jump\_if\_set [jump\_if\_not\_set]

Checks the reboot flag, and jumps to jump\_if\_set if the reboot flag is set, otherwise jumps to jump\_if\_not\_set. The reboot flag can be set by [Delete](http://nsis.sourceforge.net/Docs/Chapter4.html#delete) and [Rename](http://nsis.sourceforge.net/Docs/Chapter4.html#rename), or manually with [SetRebootFlag](http://nsis.sourceforge.net/Docs/Chapter4.html#setrebootflag).

IfRebootFlag 0 noreboot

MessageBox MB\_YESNO "A reboot is required to finish the installation. Do you wish to reboot now?" IDNO noreboot

Reboot

noreboot:

**4.9.4.12 IfSilent**

jump\_if\_silent [jump\_if\_not]

Checks the silent flag, and jumps to jump\_if\_silent if the installer is silent, otherwise jumps to jump\_if\_not. The silent flag can be set by [SilentInstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentinstall), [SilentUninstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentuninstall), [SetSilent](http://nsis.sourceforge.net/Docs/Chapter4.html#setsilent) and by the user passing /S on the command line.

IfSilent +2

ExecWait '"$INSTDIR\nonsilentprogram.exe"'

**4.9.4.13 IntCmp**

val1 val2 jump\_if\_equal [jump\_if\_val1\_less] [jump\_if\_val1\_more]

Compares two integers val1 and val2. If val1 and val2 are equal, Gotos jump\_if\_equal, otherwise if val1 < val2, Gotos jump\_if\_val1\_less, otherwise if val1 > val2, Gotos jump\_if\_val1\_more.

IntCmp $0 5 is5 lessthan5 morethan5

is5:

DetailPrint "$$0 == 5"

Goto done

lessthan5:

DetailPrint "$$0 < 5"

Goto done

morethan5:

DetailPrint "$$0 > 5"

Goto done

done:

**4.9.4.14 IntCmpU**

val1 val2 jump\_if\_equal [jump\_if\_val1\_less] [jump\_if\_val1\_more]

Compares two unsigned integers val1 and val2. If val1 and val2 are equal, Gotos jump\_if\_equal, otherwise if val1 < val2, Gotos jump\_if\_val1\_less, otherwise if val1 > val2, Gotos jump\_if\_val1\_more. Performs the comparison as unsigned integers.

**4.9.4.15 MessageBox**

mb\_option\_list messagebox\_text [/SD return] [return\_check jumpto] [return\_check\_2 jumpto\_2]

Displays a [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox) containing the text "messagebox\_text". mb\_option\_list must be one or more of the following, delimited by |s (e.g. MB\_YESNO|MB\_ICONSTOP).

* *MB\_OK* - Display with an OK button
* *MB\_OKCANCEL* - Display with an OK and a cancel button
* *MB\_ABORTRETRYIGNORE* - Display with abort, retry, ignore buttons
* *MB\_RETRYCANCEL* - Display with retry and cancel buttons
* *MB\_YESNO* - Display with yes and no buttons
* *MB\_YESNOCANCEL* - Display with yes, no, cancel buttons
* *MB\_ICONEXCLAMATION* - Display with exclamation icon
* *MB\_ICONINFORMATION* - Display with information icon
* *MB\_ICONQUESTION* - Display with question mark icon
* *MB\_ICONSTOP* - Display with stop icon
* *MB\_USERICON* - Display with installer's icon
* *MB\_TOPMOST* - Make messagebox topmost
* *MB\_SETFOREGROUND* - Set foreground
* *MB\_RIGHT* - Right align text
* *MB\_RTLREADING* - RTL reading order
* *MB\_DEFBUTTON1* - Button 1 is default
* *MB\_DEFBUTTON2* - Button 2 is default
* *MB\_DEFBUTTON3* - Button 3 is default
* *MB\_DEFBUTTON4* - Button 4 is default

Return\_check can be 0 (or empty, or left off), or one of the following:

* *IDABORT* - Abort button
* *IDCANCEL* - Cancel button
* *IDIGNORE* - Ignore button
* *IDNO* - No button
* *IDOK* - OK button
* *IDRETRY* - Retry button
* *IDYES* - Yes button

If the return value of the [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox) is return\_check, the installer will Goto jumpto.

Use the /SD parameter with one of the return\_check values above to specify the option that will be used when the installer is silent. See [section 4.12](http://nsis.sourceforge.net/Docs/Chapter4.html#silent) for more information.

MessageBox MB\_OK "simple message box"

MessageBox MB\_YESNO "is it true?" IDYES true IDNO false

true:

DetailPrint "it's true!"

Goto next

false:

DetailPrint "it's false"

next:

MessageBox MB\_YESNO "is it true? (defaults to yes on silent installations)" /SD IDYES IDNO false2

DetailPrint "it's true (or silent)!"

Goto next2

false2:

DetailPrint "it's false"

next2:

**4.9.4.16 Return**

Returns from a function or section.

Function func

StrCmp $0 "return now" 0 +2

Return

# do stuff

FunctionEnd

Section

Call func

;"Return" will return here

SectionEnd

**4.9.4.17 Quit**

Causes the installer to exit as soon as possible. After [Quit](http://nsis.sourceforge.net/Docs/Chapter4.html#quit) is called, the installer will exit (no callback functions will get a chance to run).

**4.9.4.18 SetErrors**

Sets the error flag.

SetErrors

IfErrors 0 +2

MessageBox MB\_OK "this message box will always show"

**4.9.4.19 StrCmp**

str1 str2 jump\_if\_equal [jump\_if\_not\_equal]

Compares (case insensitively) str1 to str2. If str1 and str2 are equal, Gotos jump\_if\_equal, otherwise Gotos jump\_if\_not\_equal.

StrCmp $0 "a string" 0 +3

DetailPrint '$$0 == "a string"'

Goto +2

DetailPrint '$$0 != "a string"'

**4.9.4.20 StrCmpS**

str1 str2 jump\_if\_equal [jump\_if\_not\_equal]

Same as [StrCmp](http://nsis.sourceforge.net/Docs/Chapter4.html#strcmp), but case sensitive.

**4.9.5 File Instructions**

**4.9.5.1 FileClose**

handle

Closes a file handle opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen).

**4.9.5.2 FileOpen**

user\_var(handle output) filename openmode

Opens a file named "filename", and sets the handle output variable with the handle. The openmode should be one of "r" (read) "w" (write, all contents of file are destroyed) or "a" (append, meaning opened for both read and write, contents preserved). In all open modes, the file pointer is placed at the beginning of the file. If the file cannot be opened, the handle output is set to empty, and the error flag is set.

If no absolute path is specified the current folder will be used. The current folder is the folder set using the last [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) instruction. If you have not used [SetOutPath](http://nsis.sourceforge.net/Docs/Chapter4.html#setoutpath) the current folder is [$EXEDIR](http://nsis.sourceforge.net/Docs/Chapter4.html#varother).

FileOpen $0 $INSTDIR\file.dat r

FileClose $0

**4.9.5.3 FileRead**

handle user\_var(output) [maxlen]

Reads a string (ANSI characters) from a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). The string is read until either a newline (or carriage return newline pair) occurs, or until a null byte is read, or until maxlen is met (if specified). By default, strings are limited to 1024 characters (a special build with larger NSIS\_MAX\_STRLEN can be compiled or downloaded). If the end of file is read and no more data is available, the output string will be empty, and the error flag will be set.

**Unicode:**DBCS text is supported but conversion output is limited to UCS-2/BMP, surrogate pairs are not supported. The system default ANSI codepage (ACP) is used during the conversion.

ClearErrors

FileOpen $0 $INSTDIR\file.dat r

IfErrors done

FileRead $0 $1

DetailPrint $1

FileClose $0

done:

**4.9.5.4 FileReadUTF16LE**

handle user\_var(output) [maxlen]

This function is only available when building a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode).

Reads a string (UTF-16LE characters) from a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). The string is read until either a newline (or carriage return newline pair) occurs, or until a null wide-character is read, or until maxlen is met (if specified). By default, strings are limited to 1024 characters (a special build with larger NSIS\_MAX\_STRLEN can be compiled or downloaded). If the end of file is read and no more data is available, the output string will be empty, and the error flag will be set.

ClearErrors

FileOpen $0 $INSTDIR\file.dat r

IfErrors done

FileReadUTF16LE $0 $1

DetailPrint $1

FileClose $0

done:

**4.9.5.5 FileReadByte**

handle user\_var(output)

Reads a byte from a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). The byte is stored in the output as an integer (0-255). If the end of file is read and no more data is available, the output will be empty, and the error flag will be set.

ClearErrors

FileOpen $0 $INSTDIR\file.dat r

IfErrors done

FileReadByte $0 $1

FileReadByte $0 $2

DetailPrint "$1 $2"

FileClose $0

done:

**4.9.5.6 FileReadWord**

handle user\_var(output)

This function is only available when building a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode).

Reads a word (2-bytes) from a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). The word is stored in the output as an integer (0-65535). If the end of file is read and no more data is available, the output will be empty, and the error flag will be set.

ClearErrors

FileOpen $0 $INSTDIR\file.dat r

IfErrors done

FileReadWord $0 $1

FileReadWord $0 $2

DetailPrint "$1 $2"

FileClose $0

done:

**4.9.5.7 FileSeek**

handle offset [mode] [user\_var(new position)]

Seeks a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). If mode is omitted or specified as SET, the file is positioned to "offset", relative to the beginning of the file. If mode is specified as CUR, then the file is positioned to "offset", relative to the current file position. If mode is specified as END, then the file is positioned to "offset", relative to the end of the file. If the final parameter "new position" is specified, the new file position will be stored to that variable.

ClearErrors

FileOpen $0 $INSTDIR\file.dat r

IfErrors done

FileSeek $0 -5 END

FileRead $0 $1

DetailPrint $1

FileClose $0

done:

**4.9.5.8 FileWrite**

handle string

Writes an ANSI string to a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). If an error occurs writing, the error flag will be set.

(If you are building a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode), the function converts the string to ANSI/MBCS. The system default ANSI codepage (ACP) is used during the conversion)

ClearErrors

FileOpen $0 $INSTDIR\file.dat w

IfErrors done

FileWrite $0 "some text"

FileClose $0

done:

**4.9.5.9 FileWriteUTF16LE**

handle string

This function is only available when building a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode).

Writes a Unicode (UTF-16LE) string to a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). If an error occurs writing, the error flag will be set.

ClearErrors

FileOpen $0 $INSTDIR\file.dat w

IfErrors done

FileWriteUTF16LE $0 "some text"

FileClose $0

done:

**4.9.5.10 FileWriteByte**

handle string

Writes the integer interpretation of 'string' to a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). Of course you can enter the integer value directly. The following code writes a "Carriage Return / Line Feed" - Enter to the file.

FileWriteByte file\_handle "13"

FileWriteByte file\_handle "10"

If an error occurs while writing, the error flag will be set. Note that the low byte of the integer is used, i.e. writing 256 is the same as writing 0, etc.

**4.9.5.11 FileWriteWord**

handle string

This function is only available when building a [Unicode installer](http://nsis.sourceforge.net/Docs/Chapter1.html#intro-unicode).

Writes the integer interpretation of 'string' as a WORD (2-bytes, range: 0-65535) to a file opened with [FileOpen](http://nsis.sourceforge.net/Docs/Chapter4.html#FileOpen). Of course you can enter the integer value directly. The following code writes a "Carriage Return / Line Feed" - Enter to the file.

FileWriteWord file\_handle "13"

FileWriteWord file\_handle "10"

If an error occurs while writing, the error flag will be set. Note that the low WORD of the integer is used, i.e. writing 65536 is the same as writing 0, etc.

**4.9.5.12 FindClose**

handle

Closes a search opened with [FindFirst](http://nsis.sourceforge.net/Docs/Chapter4.html#FindFirst).

**4.9.5.13 FindFirst**

user\_var(handle output) user\_var(filename output) filespec

Performs a search for 'filespec', placing the first file found in filename\_output (a user variable). It also puts the handle of the search into handle\_output (also a user variable). If no files are found, both outputs are set to empty, and the error flag is set. Best used with [FindNext](http://nsis.sourceforge.net/Docs/Chapter4.html#FindNext) and [FindClose](http://nsis.sourceforge.net/Docs/Chapter4.html#FindClose). Note that the filename output is without path.

FindFirst $0 $1 $INSTDIR\\*.txt

loop:

StrCmp $1 "" done

DetailPrint $1

FindNext $0 $1

Goto loop

done:

FindClose $0

**4.9.5.14 FindNext**

handle user\_var(filename\_output)

Continues a search began with [FindFirst](http://nsis.sourceforge.net/Docs/Chapter4.html#FindFirst). handle should be the handle\_output\_variable returned by [FindFirst](http://nsis.sourceforge.net/Docs/Chapter4.html#FindFirst). If the search is completed (there are no more files), filename\_output is set to empty, and the error flag is set. Note that the filename output is without path.

**4.9.6 Uninstaller Instructions**

**4.9.6.1 WriteUninstaller**

[Path\]exename.exe

Writes the uninstaller to the filename (and optionally path) specified. Only valid from within an install section or function, and requires that you have an uninstall section in your script. See also Uninstall configuration. You can call this one or more times to write out one or more copies of the uninstaller.

WriteUninstaller $INSTDIR\uninstaller.exe

**4.9.7 Miscellaneous Instructions**

**4.9.7.1 GetErrorLevel**

user\_var(error level output)

Returns the last error level set by [SetErrorLevel](http://nsis.sourceforge.net/Docs/Chapter4.html#seterrorlevel) or -1 if it was never used.

GetErrorLevel $0

IntOp $0 $0 + 1

SetErrorLevel $0

**4.9.7.2 GetInstDirError**

user\_var(error output)

Use in the leave function of a directory page. Reads the flag set if '[DirVerify leave](http://nsis.sourceforge.net/Docs/Chapter4.html#adirverify)' is used. Possible values:

0: No error

1: Invalid installation directory

2: Not enough space on installation drive

!include LogicLib.nsh

PageEx directory

DirVerify leave

PageCallbacks "" "" dirLeave

PageExEnd

Function dirLeave

GetInstDirError $0

${Switch} $0

${Case} 0

MessageBox MB\_OK "valid installation directory"

${Break}

${Case} 1

MessageBox MB\_OK "invalid installation directory!"

Abort

${Break}

${Case} 2

MessageBox MB\_OK "not enough free space!"

Abort

${Break}

${EndSwitch}

FunctionEnd

**4.9.7.3 InitPluginsDir**

Initializes the plug-ins dir ([$PLUGINSDIR](http://nsis.sourceforge.net/Docs/Chapter4.html#varconstant)) if not already initialized.

InitPluginsDir

File /oname=$PLUGINSDIR\image.bmp image.bmp

**4.9.7.4 Nop**

Does nothing.

**4.9.7.5 SetErrorLevel**

error\_level

Sets the error level of the installer or uninstaller to *error\_level*. See [Error Levels](http://nsis.sourceforge.net/Docs/AppendixD.html#errorlevels) for more information.

IfRebootFlag 0 +2

SetErrorLevel 4

**4.9.7.6 SetRegView**

**32**|64|lastused

Sets the registry view affected by [registry commands](http://nsis.sourceforge.net/Docs/Chapter4.html#registry). On Windows x64 there are two views. One for 32-bit applications and one for x64 applications. By default, 32-bit applications running on x64 systems under WOW64 have access only to the 32-bit view. Using SetRegView 64 allows the installer to access keys in the x64 view of the registry.

Affects [DeleteRegKey](http://nsis.sourceforge.net/Docs/Chapter4.html#deleteregkey), [DeleteRegValue](http://nsis.sourceforge.net/Docs/Chapter4.html#deleteregvalue), [EnumRegKey](http://nsis.sourceforge.net/Docs/Chapter4.html#enumregkey), [EnumRegValue](http://nsis.sourceforge.net/Docs/Chapter4.html#enumregvalue), [ReadRegDWORD](http://nsis.sourceforge.net/Docs/Chapter4.html#readregdword), [ReadRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readregstr), [WriteRegBin](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregbin), [WriteRegDWORD](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregdword), [WriteRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregstr) and [WriteRegExpandStr](http://nsis.sourceforge.net/Docs/Chapter4.html#writeregexpandstr).

Does not affect [InstallDirRegKey](http://nsis.sourceforge.net/Docs/Chapter4.html#ainstalldirregkey). Instead, the registry can be read using [ReadRegStr](http://nsis.sourceforge.net/Docs/Chapter4.html#readregstr) in [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

SetRegView 32

ReadRegStr $0 HKLM Software\Microsoft\Windows\CurrentVersion ProgramFilesDir

DetailPrint $0 # prints C:\Program Files (x86)

SetRegView 64

ReadRegStr $0 HKLM Software\Microsoft\Windows\CurrentVersion ProgramFilesDir

DetailPrint $0 # prints C:\Program Files

Function .onInit

SetRegView 64

ReadRegStr $INSTDIR HKLM Software\NSIS ""

SetRegView 32

FunctionEnd

**4.9.7.7 SetShellVarContext**

**current**|all

Sets the context of $SMPROGRAMS and other shell folders. If set to 'current' (the default), the current user's shell folders are used. If set to 'all', the 'all users' shell folder is used. The all users folder may not be supported on all OSes. If the all users folder is not found, the current user folder will be used. Please take into consideration that a "normal user" has no rights to write in the all users area. Only admins have full access rights to the all users area. You can check this by using the UserInfo plug-in. See Contrib\UserInfo\UserInfo.nsi for an example.

Note that, if used in installer code, this will only affect the installer, and if used in uninstaller code, this will only affect the uninstaller. To affect both, it needs to be used in both.

SetShellVarContext current

StrCpy $0 $DESKTOP

SetShellVarContext all

StrCpy $1 $DESKTOP

MessageBox MB\_OK $0$\n$1

**4.9.7.8 Sleep**

sleeptime\_in\_ms

Pauses execution in the installer for sleeptime\_in\_ms milliseconds. sleeptime\_in\_ms can be a variable, e.g. "$0" or a number, i.e. "666".

DetailPrint "sleeping..."

Sleep 3000

DetailPrint "back to work"

**4.9.8 String Manipulation Instructions**

**4.9.8.1 StrCpy**

user\_var(destination) str [maxlen] [start\_offset]

Sets the user variable $x with str. Note that str can contain other variables, or the user variable being set (concatenating strings this way is possible, etc). If maxlen is specified, the string will be a maximum of maxlen characters (if maxlen is negative, the string will be truncated abs(maxlen) characters from the end). If start\_offset is specified, the source is offset by it (if start\_offset is negative, it will start abs(start\_offset) from the end of the string).

StrCpy $0 "a string" # = "a string"

StrCpy $0 "a string" 3 # = "a s"

StrCpy $0 "a string" -1 # = "a strin"

StrCpy $0 "a string" "" 2 # = "string"

StrCpy $0 "a string" "" -3 # = "ing"

StrCpy $0 "a string" 3 -4 # = "rin"

**4.9.8.2 StrLen**

user\_var(length output) str

Sets user variable $x with the length of str.

StrLen $0 "123456" # = 6

**4.9.9 Stack Support**

**4.9.9.1 Exch**

[user\_var | stack\_index]

When no parameter is specified, exchanges the top two elements of the stack. When a parameter is specified and is a user variable, exchanges the top element of the stack with the parameter. When a parameter is specified and is a positive integer, [Exch](http://nsis.sourceforge.net/Docs/Chapter4.html#Exch) will swap the item on the top of the stack with the item that is specified by the offset from the top of the stack in the parameter. If there are not enough items on the stack to accomplish the exchange, a fatal error will occur (to help you debug your code :).

Push 1

Push 2

Exch

Pop $0 # = 1

Push 1

Push 2

Push 3

Exch 2

Pop $0 # = 1

StrCpy $0 1

Push 2

Exch $0 # = 2

Pop $1 # = 1

**4.9.9.2 Pop**

user\_var(out)

Pops a string off of the stack into user variable $x. If the stack is empty, the error flag will be set.

Push 1

Pop $0 # = 1

**4.9.9.3 Push**

string

Pushes a string onto the stack. The string can then be Popped off of the stack.

Push "a string"

**4.9.10 Integer Support**

**4.9.10.1 IntFmt**

user\_var(output) format numberstring

Formats the number in "numberstring" using the format "format", and sets the output to user variable $x. Example format strings include "%08X" "%u"

IntFmt $0 "0x%08X" 195948557

IntFmt $0 "%c" 0x41

**4.9.10.2 IntOp**

user\_var(output) value1 OP [value2]

Combines value1 and (depending on OP) value2 into the specified user variable (user\_var). OP is defined as one of the following:

* *+* ADDs value1 and value2
* *-* SUBTRACTs value2 from value1
* *\** MULTIPLIEs value1 and value2
* */* DIVIDEs value1 by value2
* *%* MODULUSs value1 by value2
* *|* BINARY ORs value1 and value2
* *&* BINARY ANDs value1 and value2
* *^* BINARY XORs value1 and value2
* *>>* RIGHT SHIFTs value1 by value2
* *<<* LEFT SHIFTs value1 by value2
* *~* BITWISE NEGATEs value1 (i.e. 7 becomes 4294967288)
* *!* LOGICALLY NEGATEs value1 (i.e. 7 becomes 0)
* *||* LOGICALLY ORs value1 and value2
* *&&* LOGICALLY ANDs value1 and value2

IntOp $0 1 + 1

IntOp $0 $0 + 1

IntOp $0 $0 << 2

IntOp $0 $0 ~

IntOp $0 $0 & 0xF

**4.9.11 Reboot Instructions**

**4.9.11.1 Reboot**

Reboots the computer. Be careful with this one. If it fails, [.onRebootFailed](http://nsis.sourceforge.net/Docs/Chapter4.html#onrebootfailed) is called. In any case, this instruction never returns, just like [Quit](http://nsis.sourceforge.net/Docs/Chapter4.html#quit).

MessageBox MB\_YESNO|MB\_ICONQUESTION "Do you wish to reboot the system?" IDNO +2

Reboot

**4.9.11.2 SetRebootFlag**

true|false

Sets the reboot flag to either true or false. The flag's value can be read using [IfRebootFlag](http://nsis.sourceforge.net/Docs/Chapter4.html#ifrebootflag).

SetRebootFlag true

IfRebootFlag 0 +2

MessageBox MB\_OK "this message box will always show"

**4.9.12 Install Logging Instructions**

**4.9.12.1 LogSet**

on|**off**

Sets whether install logging to $INSTDIR\install.log will happen. $INSTDIR must have a value before you call this function or it will not work. Note that the *NSIS\_CONFIG\_LOG* build setting must be set (scons NSIS\_CONFIG\_LOG=yes) on compile time (it is not by default) to support this. See[Building NSIS](http://nsis.sourceforge.net/Docs/AppendixG.html#build) for more information about recompiling NSIS.

**4.9.12.2 LogText**

text

If installer logging is enabled, inserts text "text" into the log file.

IfFileExists $WINDIR\notepad.exe 0 +2

LogText "$$WINDIR\notepad.exe exists"

**4.9.13 Section Management**

**4.9.13.1 SectionSetFlags**

section\_index section\_flags

Sets the section's flags. The flag is a 32 bit integer. The first bit (lowest) represents whether the section is currently selected, the second bit represents whether the section is a section group (don't modify this unless you really know what you are doing), the third bit represents whether the section is a section group end (again, don't modify), the fourth bit represents whether the section is shown in bold or not, the fifth bit represents whether the section is read-only, the sixth bit represents whether the section group is to be automatically expanded, the seventh bit is set for section groups which are partially selected, the eighth bit is internally used for partially selected section group toggling and the ninth bit is used for reflecting section name changes. The error flag will be set if an out of range section is specified.

Each flag has a name, prefixed with `SF\_`:

!define SF\_SELECTED 1

!define SF\_SECGRP 2

!define SF\_SECGRPEND 4

!define SF\_BOLD 8

!define SF\_RO 16

!define SF\_EXPAND 32

!define SF\_PSELECTED 64

For an example of usage please see the [one-section.nsi](http://nsis.sourceforge.net/Examples/one-section.nsi) example.

For more useful macros and definitions, see Include\Sections.nsh.

Section test test\_section\_id

SectionEnd

Function .onInit

# set section 'test' as selected and read-only

IntOp $0 ${SF\_SELECTED} | ${SF\_RO}

SectionSetFlags ${test\_section\_id} $0

FunctionEnd

**4.9.13.2 SectionGetFlags**

section\_index user\_var(output)

Retrieves the section's flags. See above for a description of the flag. The error flag will be set if an out of range section is specified.

Section test test\_section\_id

SectionEnd

Function .onSelChange

# keep section 'test' selected

SectionGetFlags ${test\_section\_id} $0

IntOp $0 $0 | ${SF\_SELECTED}

SectionSetFlags ${test\_section\_id} $0

FunctionEnd

**4.9.13.3 SectionSetText**

section\_index section\_text

Sets the description for the section section\_index. If the text is set to "" then the section will be hidden. The error flag will be set if an out of range section is specified.

Section "" test\_section\_id

SectionEnd

Function .onInit

# change section's name to $WINDIR

SectionSetText ${test\_section\_id} $WINDIR

FunctionEnd

**4.9.13.4 SectionGetText**

section\_index user\_var(output)

Stores the text description of the section section\_index into the output. If the section is hidden, stores an empty string. The error flag will be set if an out of range section is specified.

Section test test\_section\_id

SectionEnd

Function .onInit

# append $WINDIR to section's name

SectionGetText ${test\_section\_id} $0

StrCpy $0 "$0 - $WINDIR"

SectionSetText ${test\_section\_id} $0

FunctionEnd

**4.9.13.5 SectionSetInstTypes**

section\_index inst\_types

Sets the install types the section specified by section\_index defaults to the enabled state in. Note that the section index starts with zero. Every bit of inst\_types is a flag that tells if the section is in that install type or not. For example, if you have 3 install types and you want the first section to be included in install types 1 and 3, then the command should look like this:

SectionSetInstTypes 0 5

because the binary value for 5 is "00000101". The error flag will be set if the section index specified is out of range.

Section test test\_section\_id

SectionEnd

Function .onInit

# associate section 'test' with installation types 3 and 4

SectionSetInstTypes ${test\_section\_id} 12

FunctionEnd

**4.9.13.6 SectionGetInstTypes**

section\_index user\_var(output)

Retrieves the install types flags array of a section. See above explanation about [SectionSetInstTypes](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsetinsttypes) for a description of how to deal with the output. The error flag will be set if the section index specified is out of range.

Section test test\_section\_id

SectionEnd

Function .onInit

# associate section 'test' with installation types 5, on top of its existing associations

SectionGetInstTypes ${test\_section\_id} $0

IntOp $0 $0 | 16

SectionSetInstTypes ${test\_section\_id} $0

FunctionEnd

**4.9.13.7 SectionSetSize**

section\_index new\_size

Sets the Size of the section specified by section\_index. Note that the Index starts with Zero. The Value for Size must be entered in KiloByte and supports only whole numbers.

Section test test\_section\_id

SectionEnd

Function .onInit

# set required size of section 'test' to 100 bytes

SectionSetSize ${test\_section\_id} 100

FunctionEnd

**4.9.13.8 SectionGetSize**

section\_index user\_var

Gets the Size of the section specified by section\_index and stores the value in the given User Variable. Note that the Index starts with Zero.

Section test test\_section\_id

SectionEnd

Function .onInit

# increase required size of section 'test' by 100 KiB

SectionGetSize ${test\_section\_id} $0

IntOp $0 $0 + 100

SectionSetSize ${test\_section\_id} $0

FunctionEnd

**4.9.13.9 SetCurInstType**

inst\_type\_idx

Sets the current [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype). inst\_type\_idx should be between 0 and 31. The Error Flag is **not** set if an out of range [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) was used.

**4.9.13.10 GetCurInstType**

user\_var

Get the current [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) and stores it in user\_var. If the first install type is selected, 0 will be put in user\_var. If the second install type is selected, 1 will be put in user\_var, and so on. The value of ${NSIS\_MAX\_INST\_TYPES} (32 by default) means that the user selected a custom set of sections. Note that simply selecting "Custom" in the drop-down menu is not enough to trigger this. The value is calculated by the sections actually selected.

**4.9.13.11 InstTypeSetText**

inst\_type\_idx text

Sets the Text of the specified [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype). If the Text is empty than the [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) is removed. By using a previously unused inst\_type\_idx number you can create new InstTypes. To add/remove Sections to this new [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype) see [SectionSetInstTypes](http://nsis.sourceforge.net/Docs/Chapter4.html#sectionsetinsttypes). Unlike [SectionIn](http://nsis.sourceforge.net/Docs/Chapter4.html#ssectionin) the index is zero based, which means the first install type's index is 0.

InstType a

InstType b

Function .onInit

# set first installation type's name to $WINDIR

InstTypeSetText 0 $WINDIR

# set second installation type's name to $TEMP

InstTypeSetText 1 $TEMP

FunctionEnd

**4.9.13.12 InstTypeGetText**

inst\_type\_idx user\_var

Gets the Text of the specified [InstType](http://nsis.sourceforge.net/Docs/Chapter4.html#ainsttype).

InstType a

InstType b

Function .onInit

InstTypeGetText 0 $0

DetailPrint $0 # prints 'a'

InstTypeGetText 1 $0

DetailPrint $0 # prints 'b'

FunctionEnd

**4.9.14 User Interface Instructions**

**4.9.14.1 BringToFront**

Makes the installer window visible and brings it to the top of the window list. If an application was executed that shows itself in front of the installer, a [BringToFront](http://nsis.sourceforge.net/Docs/Chapter4.html#BringToFront) would bring the installer back in focus.

Recent Windows versions restrict the setting of foreground windows. If the user is working with another application during installation, the user may be notified using a different method.

**4.9.14.2 CreateFont**

user\_var(handle output) face\_name [height] [weight] [/ITALIC] [/UNDERLINE] [/STRIKE]

Creates a font and puts its handle into user\_var. For more information about the different parameters have a look at [MSDN's page about the Win32 API function CreateFont()](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/fontext_8fp0.asp).

You can get the current font used by NSIS using the ^Font and ^FontSize [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring)s.

!include WinMessages.nsh

GetDlgItem $0 $HWNDPARENT 1

CreateFont $1 "Times New Roman" "7" "700" /UNDERLINE

SendMessage $0 ${WM\_SETFONT} $1 1

**4.9.14.3 DetailPrint**

user\_message

Adds the string "user\_message" to the details view of the installer.

DetailPrint "this message will show on the installation window"

**4.9.14.4 EnableWindow**

hwnd (1|0)

Enables or disables mouse and keyboard input to the specified window or control. Possible states are 0 (disabled) or 1 (enabled).

GetDlgItem $0 $HWNDPARENT 1

EnableWindow $0 0

Sleep 1000

EnableWindow $0 1

**4.9.14.5 FindWindow**

user\_var(hwnd output) windowclass [windowtitle] [windowparent] [childafter]

Searches for a window. Behaves like the win32 FindWindowEx(). Searches by windowclass (and/or windowtitle if specified). If windowparent or childafter are specified, the search will be restricted as such. If windowclass or windowtitle is specified as "", they will not be used for the search. If the window is not found, the user variable returned is 0. To accomplish old-style [FindWindow](http://nsis.sourceforge.net/Docs/Chapter4.html#findwindow) behavior, use [FindWindow](http://nsis.sourceforge.net/Docs/Chapter4.html#findwindow) with [SendMessage](http://nsis.sourceforge.net/Docs/Chapter4.html#sendmessage).

FindWindow $0 "#32770" "" $HWNDPARENT

FindWindow $0 "my window class" "my window title"

**4.9.14.6 GetDlgItem**

user\_var(output) dialog item\_id

Retrieves the handle of a control identified by item\_id in the specified dialog box dialog. If you want to get the handle of a control on the inner dialog, first use FindWindow user\_var(output) "#32770" "" $HWNDPARENT to get the handle of the inner dialog.

GetDlgItem $0 $HWNDPARENT 1 # next/install button

**4.9.14.7 HideWindow**

Hides the installer.

**4.9.14.8 IsWindow**

HWND jump\_if\_window [jump\_if\_not\_window]

If HWND is a window, Gotos jump\_if\_window, otherwise, Gotos jump\_if\_not\_window (if specified).

GetDlgItem $0 $HWNDPARENT 1

IsWindow $0 0 +3

MessageBox MB\_OK "found a window"

Goto +2

MessageBox MB\_OK "no window"

**4.9.14.9 LockWindow**

on|off

*LockWindow on* prevents the main window from redrawing itself upon changes. When *LockWindow off* is used, all controls that weren't redrawn since *LockWindow on* will be redrawn. This makes the pages flickering look nicer because now it flickers a group of controls at the same time, instead of one control at a time. The individual control flickering is more noticeable on old computers.

**4.9.14.10 SendMessage**

HWND msg wparam lparam [user\_var(return value)] [/TIMEOUT=time\_in\_ms]

Sends a message to HWND. If a user variable $x is specified as the last parameter (or one before the last if you use /TIMEOUT), the return value of [SendMessage](http://nsis.sourceforge.net/Docs/Chapter4.html#sendmessage) will be stored to it. Note that when specifying 'msg' you must just use the integer value of the message. If you wish to send strings use "STR:a string" as wParam or lParam where needed.

* *WM\_CLOSE* 16
* *WM\_COMMAND* 273
* *WM\_USER* 1024

Include WinMessages.nsh to have all of Windows messages defined in your script.

To send a string param, put STR: before the parameter, for example: "STR:Some string".

Use /TIMEOUT=time\_in\_ms to specify the duration, in milliseconds, of the time-out period.

!include WinMessages.nsh

FindWindow $0 "Winamp v1.x"

SendMessage $0 ${WM\_CLOSE} 0 0

**4.9.14.11 SetAutoClose**

true|false

Overrides the default auto window-closing flag (specified for the installer using [AutoCloseWindow](http://nsis.sourceforge.net/Docs/Chapter4.html#aautoclosewindow), and false for the uninstaller). Specify 'true' to have the install window immediately disappear after the install has completed, or 'false' to make it require a manual close.

**4.9.14.12 SetBrandingImage**

[/IMGID=item\_id\_in\_dialog] [/RESIZETOFIT] path\_to\_bitmap\_file.bmp

Sets the current bitmap file displayed as the branding image. If no IMGID is specified, the first image control found will be used, or the image control created by [AddBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#aaddbrandingimage). Note that this bitmap must be present on the user's machine. Use File first to put it there. If /RESIZETOFIT is specified the image will be automatically resized (very poorly) to the image control size. If you used [AddBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#aaddbrandingimage) you can get this size, by compiling your script and watching for [AddBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#aaddbrandingimage) output, it will tell you the size. [SetBrandingImage](http://nsis.sourceforge.net/Docs/Chapter4.html#setbrandingimage) will not work when called from .onInit!

**4.9.14.13 SetDetailsView**

show|hide

Shows or hides the details, depending on which parameter you pass. Overrides the default details view, which is set via [ShowInstDetails](http://nsis.sourceforge.net/Docs/Chapter4.html#ashowinstdetails).

**4.9.14.14 SetDetailsPrint**

none|listonly|textonly|both|lastused

Sets mode at which commands print their status. None has commands be quiet, listonly has status text only added to the listbox, textonly has status text only printed to the status bar, and both enables both (the default). For extracting many small files, textonly is recommended (especially on win9x with smooth scrolling enabled).

SetDetailsPrint none

File "secret file.dat"

SetDetailsPrint both

**4.9.14.15 SetCtlColors**

hwnd [/BRANDING] [text\_color] [transparent|bg\_color]

Sets a background color and the text color for a static control, edit control, button or a dialog. *text\_color* and *bg\_color* don't accept variables. Use [GetDlgItem](http://nsis.sourceforge.net/Docs/Chapter4.html#getdlgitem) to get the handle (HWND) of the control. To make the control transparent specify "transparent" as the background color value. You can also specify /BRANDING with or without text color and background color to make the control completely gray (or any other color you choose). This is used by the branding text control in the MUI.

FindWindow $0 "#32770" "" $HWNDPARENT

GetDlgItem $0 $0 1006

SetCtlColors $0 0xFF0000 0x00FF00

**Warning:** setting the background color of check boxes to "transparent" may not function properly when using [XPStlye](http://nsis.sourceforge.net/Docs/Chapter4.html#axpstyle) on. The background may be completely black, instead of transparent, when using certain Windows themes.

**4.9.14.16 SetSilent**

silent | normal

Sets the installer to silent mode or normal mode. See [SilentInstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentinstall) for more information about silent installations. Can only be used in [.onInit](http://nsis.sourceforge.net/Docs/Chapter4.html#oninit).

**4.9.14.17 ShowWindow**

hwnd show\_state

Sets the visibility of a window. Possible show\_states are the same as [Windows ShowWindow](http://msdn2.microsoft.com/en-us/library/ms633548) function. SW\_\* constants are defined in [Include\WinMessages.nsh](http://nsis.sourceforge.net/usr/local/share/nsis/Include/WinMessages.nsh).

!include WinMessages.nsh

GetDlgItem $0 $HWNDPARENT 1

ShowWindow $0 ${SW\_HIDE}

Sleep 1000

ShowWindow $0 ${SW\_SHOW}

**4.9.15 Multiple Languages Instructions**

**4.9.15.1 LoadLanguageFile**

language\_file.nlf

Loads a language file for the construction of a language table. All of the language files that come with NSIS are in [Contrib\Language Files](http://nsis.sourceforge.net/usr/local/share/nsis/Contrib/Language%20files)

After you have inserted the language file ${LANG\_langfile} will be defined as the language id (for example, ${LANG\_ENGLISH} will be defined as 1033). Use it with [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring), [LicenseLangString](http://nsis.sourceforge.net/Docs/Chapter4.html#licenselangstring), LangDLL and [VIAddVersionKey](http://nsis.sourceforge.net/Docs/Chapter4.html#viaddversionkey).

**4.9.15.2 LangString**

name language\_id string

Defines a multilingual string. This means its value may be different (or not, it's up to you) for every language. It allows you to easily make your installer multilingual without the need to add massive switches to the script.

Each language string has a name that identifies it and a value for each language used by the installer. They can be used in any runtime string in the script. To use a language string all you need to add to the string is $(LangString\_name\_here) where you want the [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring) to be inserted.

**Notes:**

* Unlike defines that use curly braces - {}, language strings use parenthesis - ().
* If you change the language in the .onInit function, note that language strings in .onInit will still use the detected language based on the user's default Windows language, because the language is initialized after .onInit.
* Always set language strings for every language in your script.
* If you set the language ID to 0 the last used language by [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring) or [LoadLanguageFile](http://nsis.sourceforge.net/Docs/Chapter4.html#loadlanguagefile) will be used.

**Example of usage:**

LangString message ${LANG\_ENGLISH} "English message"

LangString message ${LANG\_FRENCH} "French message"

LangString message ${LANG\_KOREAN} "Korean message"

MessageBox MB\_OK "A translated message: $(message)"

**4.9.15.3 LicenseLangString**

name language\_id license\_path

Does the same as [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring) only it loads the string from a text/RTF file and defines a special LangString that can be used only by [LicenseData](http://nsis.sourceforge.net/Docs/Chapter4.html#alicensedata).

LicenseLangString license ${LANG\_ENGLISH} license-english.txt

LicenseLangString license ${LANG\_FRENCH} license-french.txt

LicenseLangString license ${LANG\_GERMAN} license-german.txt

LicenseData $(license)

**4.10 Multiple Languages**

As of version 2 NSIS fully supports multiple languages. The interface of one installer can support multiple languages.

Use [LoadLanguageFile](http://nsis.sourceforge.net/Docs/Chapter4.html#loadlanguagefile) for every language to load the default interface texts and language properties.

The default interface texts can easily be changed using instructions like [ComponentText](http://nsis.sourceforge.net/Docs/Chapter4.html#acomponenttext) etc.

You can also use the contents of the standard language strings in your own strings (for example, $(^Name) contains the installer's name set using the [Name](http://nsis.sourceforge.net/Docs/Chapter4.html#aname) instruction). The names of all standard language strings are listed as comments just above the strings in the language files. The language files are located in [Contrib\Language Files](http://nsis.sourceforge.net/usr/local/share/nsis/Contrib/Language%20files).

To create your own language strings, use [LangString](http://nsis.sourceforge.net/Docs/Chapter4.html#langstring).

For an example of an installer with multiple languages, see [languages.nsi](http://nsis.sourceforge.net/Examples/languages.nsi).

**4.10.1 Language Selection**

When the installer starts up it goes through these steps to select the interface language:

1. Get user's default Windows UI language
2. Find a perfect match for the language
3. If there is no perfect match, find a primary language match
4. If there is no match, use the first language defined in the script (make sure your first language is a common one like English)
5. If the language variable [$LANGUAGE](http://nsis.sourceforge.net/Docs/Chapter4.html#varconstant) has changed during .onInit, NSIS goes through steps 2 to 4 again.

**4.10.2 LangDLL Plug-in**

The LangDLL plug-in allows you to give the user an option to choose the language of the installer. Just push the language id (${LANG\_langfile}) and its name for every language in your installer, then the number of languages pushed, the caption, and the text that tells the user to select the language, call the plug-in function named LangDialog, pop the returned value into $LANGUAGE and you're good to go. If the user clicks on the cancel button the return value will be "cancel".

For an example of usage see [languages.nsi](http://nsis.sourceforge.net/Examples/languages.nsi).

**4.10.3 RTL Languages**

RTL languages are languages that are written from right to left (e.g. Arabic and Hebrew). NSIS fully supports RTL languages. In the language file there is a place to specify if the language is RTL or not. To find out at runtime if the current language is RTL or not, check the value of the $(^RTL) language string. It will be 1 if the language is RTL and 0 otherwise. This can be useful when using plug-ins that create dialogs, they usually have RTL settings too.

**4.11 Plug-in DLLs**

The abilities of the NSIS scripting language can be extended by utilising functionality provided in a DLL file. Probably the best known example of this is the InstallOptions.dll bundled with every NSIS release.

When the NSIS compiler starts it scans the plug-ins directory for DLLs and makes a list of the plug-ins found and their exported functions. During compilation if a sequence such as fred::flintstone is encountered where the compiler expected to find a language keyword the compiler will look through this list. If a list entry specifies that fred.dll exports function flintstone NSIS will pack the fred.dll file into the created installer binary.

During execution of a plug-in command NSIS will unpack the necessary DLL to a temporary folder ($PLUGINSDIR), push all of the arguments specified (right-to-left order), and then execute the DLL function.

**4.11.1 Using Plug-in Commands**

A plug-in call looks like this:

InstallOptions::dialog "ini\_file\_location.ini"

All parameters are pushed onto the stack (in this case, the plug-in function only needs one parameter). Some plug-in commands may not need any parameters on the stack, others might require more of them. To use a plug-in command you will need to read the documentation for the plug-in so that you know what parameters its functions require.

**4.11.2 Calling plug-ins manually**

If you want to call a plug-in that is stored on user's hard drive or somewhere else, use [CallInstDLL](http://nsis.sourceforge.net/Docs/Chapter4.html#callinstdll). Almost all plug-ins provide installer functionality, so using plug-in commands is way easier. Using [CallInstDLL](http://nsis.sourceforge.net/Docs/Chapter4.html#callinstdll) can be useful when you have created plug-ins that should be linked to a certain version of your application and are being copied to the installation folder.

**4.12 Silent Installers/Uninstallers**

Silent installers are installers which require no user intervention and have no user interface. The user doesn't see any dialog and isn't asked any questions. This is useful for network administrators who wish to install or uninstall something without user intervention so they can perform the operation quickly over any number of computers. It is also useful for other developers who wish to embed another installer in their own and collect all of the required information on their installer instead of showing two installers.

NSIS installers and uninstallers can be both silent and not silent. When an installer or an uninstaller is silent, not all callback functions are called. [.onGUIInit](http://nsis.sourceforge.net/Docs/Chapter4.html#onguiinit), [.onGUIEnd](http://nsis.sourceforge.net/Docs/Chapter4.html#onguiend), their uninstaller equivalents and any callback related to a specific page or page type will not be called.

There are several methods to make an installer or an uninstaller silent:

1. [SilentInstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentinstall) and [SilentUninstall](http://nsis.sourceforge.net/Docs/Chapter4.html#asilentuninstall)
2. [SetSilent](http://nsis.sourceforge.net/Docs/Chapter4.html#setsilent)
3. Passing /S on the command line (case sensitive)

To check if the installer/uninstaller is silent use [IfSilent](http://nsis.sourceforge.net/Docs/Chapter4.html#ifsilent).

To make sure your installer will be silent when it needs to, you should check with [IfSilent](http://nsis.sourceforge.net/Docs/Chapter4.html#ifsilent) before each command that might require user intervention or create a window. The [MessageBox](http://nsis.sourceforge.net/Docs/Chapter4.html#messagebox) command, which is the most common culprit in silent installers, has the /SD switch to set a default answer for silent installers. If you want your installer/uninstaller to be able to be completely silent you should use this switch. All internal NSIS message boxes have defaults for silent installers. The [silent.nsi example](http://nsis.sourceforge.net/Examples/silent.nsi) demonstrates all aspects of this topic.

Since the directory page can not be shown on silent installers, the user has an option to specify the installation directory on the command line (this also works on non-silent installers/uninstallers). To do that, the user uses the /D switch as in the following example:

foo.exe /S /D=C:\Program Files\Foo

If your installer/uninstaller requires some more information that can not be gathered when silent, you can allow the user to specify that information on the command line and process it in .onInit. You can use [GetOptions](http://nsis.sourceforge.net/Docs/AppendixE.html#getoptions).

!include FileFunc.nsh

!insertmacro GetParameters

!insertmacro GetOptions

Function .onInit

${GetParameters} $R0

ClearErrors

${GetOptions} $R0 /USERNAME= $0

FunctionEnd

The above example will copy the value the user passes on after /USERNAME= into $0. This allows the user to specify the required information on the command line instead of using the interactive user interface. The user can use:

foo.exe /S /USERNAME=Bar /D=C:\Program Files\Foo

or:

foo.exe /S /USERNAME=string with spaces /D=C:\Program Files\Foo

or:

foo.exe /S /USERNAME="string with spaces" /D=C:\Program Files\Foo

If your installer/uninstaller requires a lot of information and you want it to be able to be silent, you should allow the user to pass on a path to an answers file. This would be much more comfortable than writing all of the information on the command line.

[Previous](http://nsis.sourceforge.net/Docs/Chapter3.html) | [Contents](http://nsis.sourceforge.net/Docs/Contents.html) | [Next](http://nsis.sourceforge.net/Docs/Chapter5.html)