

Chief's Installer Pro for Windows

© 1994, 1995, Dr Abimbola A. Olowofoyeku (the African Chief)

Chief's Installer Pro for Windows is a shareware **INSTALLER** and **UNINSTALLER** for Windows. Please read this documentation, and the **disclaimer** section carefully before using the program.

Registration costs **£20.00** (U.K. Sterling) or **\$29** (US dollars). Please refer to the **registration** and **registration sites** sections below, for fuller information.

INTRODUCTION

FEATURES

FOREIGN LANGUAGE SUPPORT

THE INF FILE

RESERVED WORDS

EXISTING FILES

THE UNINSTALLER

COMMAND LINE OPERATION

REGISTRATION

REGISTRATION SITES

REGISTRATION FORM

DISCLAIMER

CREDITS

FEEDBACK

UPDATES

INTRODUCTION

Chief's Installer Pro (hereinafter sometimes referred to as "Install") is a program for the setup, installation, and "uninstallation" of Windows applications. It basically is an "off-the-shelf" installation suite. The program will copy files from floppy disks (or a directory on a hard disk) to the destination directory. If the files are compressed with Microsoft's **COMPRESS.EXE** they will be decompressed automatically. In this respect Install uses the routines in **LZEXPAND.DLL**. If the files are not compressed, they will simply be copied to the destination directories.

SETUP.EXE

The main executable is **INSTALL.EXE** - and that is all that needs to be run. Also supplied is **SETUP.EXE**, an optional loader for **INSTALL.EXE**. What **SETUP.EXE** does is to;

[a] display a "initializing install ..." message

[b] copy the necessary Install files (**INSTALL.EXE**, **WINSTALL.INF**, **WINSTALL.HLP**) and **WINSTALL.DLL**, **WINSTALP.DLL**, **WINSTALL.TXT**, **WINSTALL.BMP** (if they exist) to the TEMP directory. **CTL3DV2.DLL** (if found) will be copied to the Windows SYSTEM directory - but only if a copy does not already exist there, **and** no copy is currently loaded in memory.

For these purposes, all these files **may** be compressed on the installation disks - but in such cases, they must retain their real names (as above) - except **INSTALL.EXE**, which if compressed with the **-r** option, can be called **INSTALL.EX_** (the original name will be restored by **SETUP.EXE**). I would however suggest that the .DLL files and the .BMP file should not be compressed.

SETUP will also optionally copy **USER FILES** to the TEMP directory in the same operation. If you want any user files to be copied (because you are going to do something with them) the file names should be specified in a **\$TEMPDIR** line in **SETUPINF.INF**. You can have only one **\$TEMPDIR** line in that file, and this can only contain a maximum of 10 file names (separated with semi-colons). You should remember to delete such files with the **\$CLEANUP** command in **WINSTALL.INF**.

[c] load the copy of **INSTALL.EXE** from the TEMP directory, with the parameters necessary for it to work properly if run in this way

Is **SETUP.EXE** useful? "It depends" is the answer. If there is more than one installation diskette, then you **should** use it. It will save you the embarrassment of Windows trying to read from **INSTALL.EXE** after disk 2 (or whatever), and the disk in the floppy drive does not contain **INSTALL.EXE**. If there is only one installation diskette, it may help keep the user occupied while **INSTALL.EXE** is loading.

I have tried to make this installer as flexible and easy to use as possible. To use it, you only need to take the following steps;

1. Create your installation disk set by placing your application's files on them (up to 45 installation disks are supported).
2. You can compress the files with **COMPRESS.EXE** (it is immaterial whether you do so or not). If files are compressed and an underscore is used in the compressed file names, these names will only be converted to the original file names if they were compressed with the **-r** option.
3. Create an installation information file in ASCII format, using a text editor. The file should be called **WINSTALL.INF**, and should be in the format described below.
4. Run the Install program.

See also;

RESERVED WORDS

FEATURES

Below is a summary of the **features and restrictions** in Chief's Installer Pro.

1. You can only install into any combination of the following;
 - [a] ONE target directory, and up to 45 subdirectories under it
 - [b] the Windows directory
 - [c] the Windows SYSTEM directory
 - [d] the TEMP directory
2. Only a maximum of **45** installation disks are supported. Chief's Installer Pro will prompt for the disks as they are required.
3. Chief's Installer Pro will optionally offer to put the destination directory into the "PATH" statement in AUTOEXEC.BAT
4. Chief's Installer Pro will optionally create Program Manager group files, and icons for any supplied file(s) - .EXEs, README files, etc.
5. Chief's Installer Pro will optionally run any supplied program(s) during the installation, as part of the installation process.
6. Chief's Installer Pro will optionally run any supplied program(s) immediately after the installation is complete
7. The stipulated format of the WINSTALL.INF file must be followed strictly.
8. Chief's Installer Pro allows you to supply on-line for the installation. You have to create a Windows help file called WINSTALL.HLP. This help file can be accessed by clicking on the "HELP" button. A simple one is supplied. You can either use that, or create your own. If no WINSTALL.HLP file is found in the path, then the "HELP" button is removed from the Install window.
9. Chief's Installer Pro will check whether there is sufficient space on the target drive - based on information which you supply as to how much disk space your application requires. If you need extra temporary disk space for the installation, Install can check for this also.
10. Chief's Installer Pro will make entries in any INI (or other) file(s) specified by you. Up to 20 entries can be made. Do NOT use this feature to insert an entry which may already exist - the old entry will be deleted and replaced by the new one (e.g., do NOT use it for "DEVICE=" lines in SYSTEM.INI
11. Chief's Installer Pro provides support for non-English languages.
12. Chief's Installer Pro provides support for **PARTIAL INSTALLATIONS** of programs. In this respect, you can have up to **10** installation options.
13. Chief's Installer Pro provides support for displaying a **banner** in the background, and for specifying the text of the banner, the font to use used for it, the font size, the color of the text, the color of the background, and a brush to paint the background.
14. Chief's Installer Pro provides support for displaying a **bitmap** file, stretched to fill the screen.
15. Chief's Installer Pro will optionally check the disks being inserted by

the user to verify that they are the correct ones.

16. Chief's Installer Pro provides an **UNINSTALLER**, which can be used to uninstall any program that was installed with Chief's Installer Pro.

17. Chief's Installer Pro will check the target directory for the existence of any copy of each file being installed, and will prompt the user for over-write permission.

18. Chief's Installer Pro will check for **version information** in existing copies of shared binary files, and for date stamps in non-shared files.

19. Chief's Installer Pro will use the 3D dialog effects in **CTL3DV2.DLL** if a copy of that file is found. If CTL3DV2.DLL is not found, the program will try to use CTL3D.DLL. If that is also not found, then the program will use standard Windows dialogs (the absence of these files will NOT cause an error).

20. Chief's Installer Pro provides support for restarting Windows if any active shared file was overwritten. A dialog asking for confirmation appears automatically if any active shared file was overwritten during the installation. The text on this dialog can be changed by the \$RESTARTWIN-MESSAGE reserved word.

21. Chief's Installer Pro provides support for displaying a README file to the user when the installer is executed. The README file should be a plain ASCII file not bigger than 16kb. The file should be called **WINSTALL.TXT**. You can cause the contents of the file to be displayed automatically by setting **\$AUTO-CLICK-BUTTON** to 4 (otherwise the user will have to click on the "readme" button to display the text. If the file **WINSTALL.TXT** is not found, then the "readme" button will be removed at run time.

22. Chief's Installer Pro provides support for making entries into the the **Registration Database**, by the reserved word **\$REG-DATA**.

23. Chief's Installer Pro provides limited support for installing TRUETYPE fonts.

FOREIGN LANGUAGE SUPPORT

All the string messages presented to the user by the installer are in string tables. There are internal ENGLISH string tables in INSTALL.EXE, UNINSTALL.EXE, and SETUP.EXE. These will normally be used to display all the messages and information. However, Chief's Installer Pro provides 2 methods of changing/translating the string tables for use by the various .EXEs. One method is by compiling the string tables into DLLs (for INSTALL.EXE and UNINSTALL.EXE) and the other is by putting the string tables in an ASCII file (for SETUP.EXE).

INSTALL.EXE will always look for a dynamic link library file called **WINSTALL.DLL** from which to load the string tables. If this file is found, the string tables are read from it by Install at startup time. If you are going to use this DLL, it **MUST** be in the same directory as INSTALL.EXE. If WINSTALL.DLL is not found, Install will use the default (English) string tables inside INSTALL.EXE itself.

By the same token, **UNINSTALL.EXE** will also look for a DLL called **UNINST.DLL** in order to load the string tables from it. If you create your own UNINST.DLL, it **must** be in the same directory as UNINSTALL.EXE. If UNINST.DLL is not found, then UnInstall will use the default (English) string tables inside UNINSTALL.EXE itself.

Finally, **SETUP.EXE** will also always look for an ASCII file called **SETUPINF.INF** in order to load the string tables from it. Each of the strings must be numbered (by a hash, followed immediately by its numeric ID), and they must all be together in a section called **[SETUP]**. This is because SETUP.EXE will use the **GetPrivateProfileString** API to retrieve them. A sample of this file is provided - it mirrors the English string tables inside SETUP.EXE. If you are using an English language installation, you should delete this file, because you do not need it.

If you create your own SETUPINF.INF, it **must** be in the same directory as SETUP.EXE, it must be uncompressed, and it must be **exactly** in the same format as the sample that I have provided. If SETUPINF.INF is not found, then Setup will use the default (English) string tables inside SETUP.EXE itself.

What all this means is that you can change the language used by the installer by simply producing your own translations of the English string tables, and compiling them into the relevant DLLs (or putting them into SETUPINF.INF in the required form). For these purposes, I have provided copies of my resource scripts, and a sample SETUPINF.INF file. These serve as a guide to the string tables and the numeric IDs of the strings. Please do **NOT** change the numeric ID of any string.

If anybody produces a non-English translation of the script files, please

send me a copy, so that I can package non-English versions of with subsequent releases of the installer. If I package your translation, your name will make it into the "Credits" section for each version that contains your translation.

Please note this;

[a] NO checking is carried out to verify the contents of these string tables, or even that the string tables actually exist. Thus, if you edit the string tables and/or produce your own DLLs or SETUPINF.INF file, you are on your own.

[b] I can only vouch for the accuracy of the English version of any string table - and even then, only the version which was produced by me. If I package any non-English version of any file, please do not stone me if the translation is incorrect - but if you do find errors, please DO send me what you think is a correct translation of the string table.

THE INF FILE

The configuration file for the installation is called **WINSTALL.INF**. This is an ASCII file that has to be created with a text editor (e.g., the Windows NOTEPAD applet). Every line in the file should end with a carriage return plus a line feed - so please avoid using UNIX editors to create your INF files).

You need to follow the instructions on the format of this file carefully, otherwise, the installer will not work correctly. The best approach is to edit the sample files which I have provided. They contain sufficient comments for you to understand them. **WINSTALL.INF** is a standard **ASCII** file, in the following format;

1. Any line beginning with a ";", or "[" or "REM" is ignored
2. Empty lines are ignored
e.g., these lines will be ignored
[This line will be ignored]
; So will this line
REM so will the one just above me, and me as well!
3. Each line must not contain more than 220 characters
4. There are **RESERVED WORDS** for every valid entry, and these must be followed strictly.

AUTOCALC.EXE

Some assistance can be derived from the **AUTOCALC** utility in creating INF files. AUTOCALC can be run just to calculate disk space requirements - but it can also (optionally) do a lot more. For those who do not mind getting their hands dirty, running AUTOCALC.EXE with the **/FULL** parameter leads to a whole range of possibilities, e.g., generating a full template INF file, and attempting (not always successfully) to create your installation disks for you .

To use AUTOCALC in this **advanced** (read "complicated") way requires some careful setting up, and may not be everybody's cup of tea. The good news is that you are free to completely ignore AUTOCALC's "advanced" features (just pretend that they are not there - they were only added as an afterthought, for those who like to get into the thick of things). Please read **AUTOCALC.TXT** for full details (or simply ignore AUTOCALC if you so desire - I won't take it personally :)).

See also;
RESERVED WORDS

REGISTRATION

Chief's Installer Pro is distributed under the **Shareware** principle. It can be copied and distributed freely, as long as **ALL** the supplied files, including documentation (this file) are included, and **NO ATTEMPT** is made to modify any of the files.

The Shareware principle means that you get a chance to **EVALUATE** the program free of charge for a reasonable period of time (in the case of Chief's Installer Pro, **a maximum of 14 days**). It does not mean that you will NOT have to pay for the program.

This program is NOT crippled in any way, and there are NO nagging registration messages. What this means is that you now have the FULL version of Chief's Installer Pro. Nothing is disabled, there are no extra files, and there is no written manual. All the documentation is in the .HLP file (**CHIEF.HLP**) and the .WRI file (**CHIEF.WRI**).

I felt that releasing the full version in this way was necessary in order to enable people to fully evaluate the program, being that they will be seeing exactly what the program is. It also helps to ensure that when you do register, you do not have to wait for days or weeks to receive your "registered copy".

However, this approach also means that I am relying totally on people's honesty to register. Chief's Installer Pro is a tool to help programmers concentrate on their products by not having to worry about installation routines, thereby improving their productivity. A lot of time and effort has gone into this program, and I am not asking for much. If you find Chief's Installer Pro useful and would like to continue using it, or you would like to use it as the installation routine for your own programs, then I would encourage you to **please REGISTER your copy**.

BENEFITS OF REGISTRATION

1. Free upgrades to all future shareware releases of Chief's Install Pro, up till (but not including) the next MAJOR release. MAJOR upgrades will be numbered in whole numbers, and attract an upgrade fee of 50% of whatever is the prevailing registration fee. Minor upgrades will be numbered in .10 increments, and will be free.
2. Freedom to use Chief's Installer Pro as the installation program for an unlimited number of your own applications.
3. A clear conscience.
4. Support (via e-mail) for the program.
5. A chance to have an input into the features of future versions (I will not accept suggestions for new features from anyone who has not registered).
6. Not having to worry that I will discover that you have used the package as the installation routine for your program without registering

<g>

7. You will be very cool indeed.

REGISTRATION FEE:

£20.00 (U.K. STERLING)

\$29.00 (U.S.)

**\$34.00 (Canadian): British Columbia residents should add 7%
Sales Tax**

\$42.00 (Australian)

Kr190.00 (Danish)

**Please NOTE that ALL prices are subject to change WITHOUT
NOTICE.**

See also;

REGISTRATION SITES

RESERVED WORDS

Everything is done in Install through the use of **RESERVED WORDS**. Each reserved word begins with a dollar sign (\$) and determines a certain aspect of Install's behaviour. Below is a list of the reserved words and the methods of using them.

List of RESERVED WORDS;

\$TITLE
\$TARGET
\$SPACE
\$SYSDIR-SPACE
\$WINDIR-SPACE
\$SWAP-SPACE
\$DISK
\$DEST
\$WINDIR
\$SYSDIR
\$TEMPDIR
\$AUTO-REPLACE
\$INI
\$GROUP
\$ICON
\$PRE-EXEC
\$CLEANUP
\$EXEC
\$WINDOW
\$COPYBUFFER
\$WINDOW-BACKGROUND
\$TEXT-BACKGROUND
\$PAINTDIALOGS
\$SETCHECKBOX
\$NO-END-DIALOG
\$NO-PATH-DIALOG
\$SHOW-FILE-PERCENT
\$MAKE-UNINSTALL-LOG
\$USER-OPTION
\$OPTIONAL
\$BANNER-FONT
\$BANNER-FONT-SIZE
\$BANNER-MESSAGE
\$BANNER-TEXT-COLOR
\$BANNER-TEXT-BACKGROUND
\$BANNER-WINDOW-BRUSH
\$BANNER-SHADOW-COLOR

\$BITMAP
\$VERIFY-INSTALL-DISKS
\$AUTO-CLICK-BUTTON
\$CHECK-MY-DLL-VERSIONS
\$VERSION-INFO-TITLE
\$VERSION-INFO-MESSAGE
\$FINAL-MESSAGE
\$VERSION-COPY-ERROR-MESSAGE
\$RESTARTWIN-MESSAGE
\$README-BUTTON-TEXT
\$README-FONT
\$FORCE-OVERWRITE-OLDER-FILES
\$NO-CTL3D.DLL
\$REG-DATA
\$DIALOG-ICON
\$FONT
\$SOURCEDIR

\$TITLE

This is used to specify the name or title of your application. This is what will appear in the title of the installation program's window - You can have only ONE such line. If you don't supply any text for the **\$BANNER-MESSAGE** reserved word, this title will be used for the banner, with the words "Welcome to" prepended to it.

The Syntax is;

\$TITLE=<program title>

EXAMPLE:

\$TITLE=Great Program v1.20

See also;

\$BANNER-MESSAGE

\$TARGET

This is used to specify the name of the DEFAULT target directory for the installation. The user will be able to change this at run time. If the target directory does not exist Install will create it - but note that it will not create directories recursively. You can have only ONE such line.

By default, ALL the files will be installed into whatever is the target directory chosen by the user. You can however specify that certain files should be installed into subdirectories UNDER THE TARGET DIRECTORY TREE, or into the WINDOWS DIRECTORY, or into the WINDOWS SYSTEM DIRECTORY, or into the TEMP DIRECTORY.

To use this, you use the \$DEST, \$WINDIR, \$SYSDIR, \$TEMPDIR reserved words.

This reserved word can take **optional** extra parameters - details of an INI file **in the Windows directory** from which entries from a previous installation of your program can be obtained. The entry here should point to the directory into which the user installed any previous version. If such an entry is found, it will be used to replace the default one in your \$TARGET line. If no entry is found, the default will be used.

If the extra parameters are used, then the \$TARGET line **MUST** contain 4 entries, in the following format;

\$TARGET=Default;INI file name;Section;KeyName

"Section" corresponds to "ApplicationName" in the Windows API speak (i.e., the title of the relevant section in the INI file) and "KeyName" corresponds to it's ordinary meaning with regard to the **GetPrivateProfileString** API call, which is what is used to retrieve the entries from the INI file (see also the **\$INI** reserved word). The INI file name should **not** contain any path (just the filename only) - the program will only look for the file in the **Windows** directory.

e.g.,

\$TARGET=C:\CHIEPRO;CHIEFPRO.INI;ChiefPro;ChiefDir

Note that you will need to create the requisite entry with the **\$INI**

reserved word.

e.g.,

\$INI=\$WINDIR\CHIEFPRO.INI;ChiefPro;ChiefDir;\$DEST

The Syntax is;

\$TARGET=<default directory>[;INI file name;Section;KeyName]

EXAMPLES:

\$TARGET=C:\MYPROG

\$TARGET=C:\MYPROG;PROG.INI;ProgPriv;ProgDir

See also;

\$DEST

\$INI

\$SYSDIR

\$TEMPDIR

\$WINDIR

\$SPACE

Use this to specify the amount of disk space needed for the installation. The amount should be in **BYTES** and should only contain whole numbers (no spaces, no letters, and no decimals).

This information is used by Install to warn the users of the amount of space that they need to have free on their disks, and to show the progress of the installation in the "percentage meter". Install will check to see that the specified amount of space exists on the target drive before installation begins. If there is insufficient space, Install will abort with an error message.

There is no need for the number to correspond exactly with the actual required number of bytes - a difference of up to 2% of the size of your application (plus or minus) is allowed, and such differences will be catered for automatically. In fact, it is always good to add about 1% to the actual disk space needed - because of the vagaries of disk cluster sizes, it may be wise to over-estimate the disk space needed (a little bit of trial and error is in order here). You can have only ONE such line. The amount specified here should also take into account any disk space requirements specified in any \$OPTION lines.

NOTE: The bonus program **AUTOCALC.EXE** can be used to calculate the required space automatically. Please read **AUTOCALC.TXT** for fuller details.

The Syntax is;

\$SPACE=<required disk space>

EXAMPLE:

\$SPACE=2002003

See also;

\$USER-OPTION

\$OPTIONAL
\$SWAP-SPACE
\$SYSDIR-SPACE
\$WINDIR-SPACE

\$SYSDIR-SPACE

This reserved word is **optional**. It is only useful if you are using the **\$SYSDIR** command to install shared files into the Windows SYSTEM directory. Its purpose is to enable the installer to ascertain that there is sufficient space on the drive which holds the Windows SYSTEM directory (in cases where the user is installing the program onto another drive). The line should only contain the total size of the files that will be installed to the Windows SYSTEM directory.

This line does not in any way affect the entry that should be on the **\$SPACE** line, because they serve different (but sometimes overlapping) purposes. If the user is installing the program onto the same drive as that on which Windows is installed, this line is ignored at run time.

NOTE: The bonus program **AUTOCALC.EXE** can be used to calculate the required space automatically. Please read **AUTOCALC.TXT** for fuller details.

The Syntax is;

\$SYSDIR-SPACE=<space>

See also;

\$SPACE
\$SYSDIR
\$WINDIR-SPACE

\$WINDIR-SPACE

This reserved word is **optional**. It is only useful if you are using the **\$WINDIR** command to install shared files into the Windows directory. Its purpose is to enable the installer to ascertain that there is sufficient space on the drive which holds the Windows directory (in cases where the user is installing the program onto another drive). The line should only contain the total size of the files that will be installed to the Windows directory.

This line does not in any way affect the entry that should be on the **\$SPACE** line, because they serve different (but sometimes overlapping) purposes. If the user is installing the program onto the same drive as that on which Windows is installed, this line is ignored at run time.

NOTE: The bonus program **AUTOCALC.EXE** can be used to calculate the required space automatically. Please read **AUTOCALC.TXT** for

fuller details.

The Syntax is;

\$WINDIR-SPACE=<space>

See also;

\$SPACE

\$SYSDIR-SPACE

\$WINDIR

\$SWAP-SPACE

Use this to specify the amount of any temporary swap disk space needed for the installation. The amount should be in BYTES and should only contain whole numbers (no spaces, no letters, and no decimals).

This information is used by Install to warn the users of the amount of space that they need to have free on their disks - but it does not show in the "percentage meter". The installer will check on the drive which contains the TEMP directory to ensure that there is sufficient swap space. This obviously presumes that all your scratch and temporary files will be created in the TEMP directory. You can have only ONE such line.

EXAMPLE:

\$SWAP-SPACE=0

See also;

\$SPACE

\$SYSDIR-SPACE

\$WINDIR-SPACE

\$DISK

You use this reserved word to specify the disk(s) in the installation set, and the file(s) which should be copied from them. Each file name or file specification should be separated from the next one by a semi-colon.

You can use the wildcard character "*" in this respect.

The Syntax is;

\$DISK n = <filenames(s)>

where:

n = numbers from 1 to 45

<filename(s)> = the file specifications

the easiest thing to do would be to just specify "*.*" - to copy all the files - but you can be more specific.

NOTES:

If you specify a file that does not exist on the disk, it will just be ignored.

A maximum of 30 file specifications is allowed for each disk in the installation set. Note that each line cannot be longer than 220 characters in total. Since this might mean that all the files you want to specify for a disk might not fit on one line, you can either put all the file specifications for each disk on a single \$DISK line, or you can split them up into up to ten \$DISK lines for better readability. For example, for DISK 1 of the installation set, you can either put all the files on one "\$DISK1=" line (if they will fit on one line) or you can have as many as ten "\$DISK1=" lines, each line listing different file specifications to make up your DISK 1).

In most cases, judicious use of wildcard characters should mean that you only need one line for each disk in your installation set (e.g., if you use something like: **\$DISK1=*.EX;*.HL;*.TXT**). However, if you wish to be more specific about the files on each disk, the flexibility is available to spread each "\$DISK" across many lines.

Chief's Installer Pro will prompt the user for each disk in the installation set. However, unless you turn on disk verification with the **\$VERIFY-INSTALL-DISKS** reserved word, no attempt will be made to check that the disk being inserted is the correct one. **Please note this point.**

Please ensure that the \$DISK lines are numbered consecutively, otherwise there may be problems - for example, don't jump from "\$DISK1=" to "\$DISK3=" (the problem here is - where is \$DISK 2? - if you put "\$DISK2=" after "\$DISK3=", then there may be problems numbering the disks.)

Compressed files with underscores in the filename will have the filenames expanded into the name of the original files **ONLY if** the original files were compressed with the **-r** option

VERY IMPORTANT NOTE: Please be very careful with the way wildcard characters are used, **especially** if your program spans more than one disk. For example, it is very convenient to use *.* for all the disks in your installation set, and while this will be okay if your program is only going to be installed from floppy disks, imagine the chaos, if somebody copies all the files to a directory on the hard disk before installing, or if a CD-ROM distributor puts your program in a directory on a CD-ROM. You will have the same files ("*.*) being installed over and over again, for each disk on the installation set, and your users will not be impressed.

Thus, unless you are **absolutely certain** that your program will only ever be installed from floppy disks, you need to be selective in your use of wildcard characters - at the least, to make sure that if all your program's files are installed from a single source directory, none of the files which belong to one disk can be confused with files belonging to another disk. In this wise, it may be advisable to place files on each disk according to type and/or extension (e.g., \$DISK1=*.EX;*.HL_ : \$DISK2=*.DL;*.VB_ - etc., etc).

EXAMPLES:

\$DISK 1 = *.*

\$DISK 2 = *.DLL;*.HLP;*.DRV;WS*.*;*.EX_

\$DISK 2 = EXPAND.*;COMPRESS.EXE;FRÉD.EXE;CHIEF.EXE

\$DISK 3 = HELP.DOC;*.FFF

See also;

\$VERIFY-INSTALL-DISKS

\$DEST

[i] Where ever this appears **at the beginning of a line**, the following take place;

(a) "\$DEST" is replaced with the target directory selected by the user. For example, entry of "\$DEST\BIN=PROG.EXE", if the user installed to "C:\NEWPROG", becomes "C:\NEWPROG\BIN=PROG.EXE"

(b) anything after the "=" sign is taken as the file(s) to be installed into that directory (**instead of into the target directory**). There can be up to 30 file names, each separated by a semi-colon. Some **limited** use of wildcards is allowed here - if you want to use wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.TXT;*.INI;*.DLL;*.EXE**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible.

The sub directories will be created when necessary - but note that the order in which they are specified may be important - if there are deep levels of nesting, the ones higher up the tree **must** be specified first.

Please **NOTE** that in the case of files compressed with the -r switch, you should use the real (original) names of the **uncompressed** files, and **NOT** the names of the compressed files. For example, if the file **MYPROG.DLL** was compressed to **MYPROG.DL_**, you should put **MYPROG.DLL** on this line. The compressed filenames are only allowed on **\$DISK** lines.

You can have up to 45 such lines.

[ii] Where ever this appears **elsewhere in a line**, the "\$DEST" is replaced with the target directory selected by the user. This use of the \$DEST reserved word is only useful in the "\$ICON", "\$INI", and "\$EXEC" lines.

The Syntax is;

\$DEST=<filename(s)>

EXAMPLES:

\$DEST\BIN=*.EXE;*.DLL;WINSTALL.INF

\$DEST\HELP=*.HLP;*.TXT;*.WRI

\$DEST\SAMPLES=SAMPLE1.INF;SAMPLE2.INF;SAMPLE3.INF

See also;

\$EXEC

\$ICON

\$INI

\$TARGET

\$SYSDIR

\$TEMPDIR

\$WINDIR

\$WINDIR

[i] Whenever this appears at the beginning of a line, the files on that line are installed to the Windows directory (instead of the target directory). There can be up to **ten** \$WINDIR lines. Each line can contain a maximum of **30** file names, each separated by a semi-colon. Some **limited** use of wildcards is allowed here - if you want to use wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.INI;*.EXE**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible. Note that each line cannot be longer than 220 characters, and that the total number of files to be installed into the Windows directory cannot exceed **120**.

Please **NOTE** that in the case of files compressed with the -r switch, you should use the real (original) names of the **uncompressed** files, and **NOT** the names of the compressed files. For example, if the file **MYPROG.DLL** was compressed to **MYPROG.DL_**, you should put **MYPROG.DLL** on this line. The compressed filenames are only allowed on **\$DISK** lines.

[ii] Where ever this appears elsewhere in a line, the "\$WINDIR" is replaced with the Windows directory. This use of the \$WINDIR reserved word is only useful in the "\$ICON", "\$INI", and "\$EXEC" lines.

EXAMPLES:

\$WINDIR=PROG1.EXE;PROG2.EXE;PROG2.EXE;*.INI

\$WINDIR=RATTER.EXE;RETTO.EXE;DRAT.EXE

\$WINDIR=ROTTTO.INI;ROUTER.INI;TROUBLE.INI

See also;

\$DEST

\$SYSDIR

\$TARGET

\$TEMPDIR

\$SYSDIR

[i] Whenever this appears at the beginning of a line, the files on that line are installed to the Windows SYSTEM directory (instead of the target directory). There can be up to **ten** \$SYSDIR lines. Each line can contain a maximum of **30** file names, each separated by a semi-colon. Some **limited** use of wildcards is allowed here - if you want to use wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.VBX;*.DRV;*.DLL;*.TTF**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible.

When wildcards are used in this way, **any conflict is resolved in the following order: [1] \$SYSDIR, [2] \$WINDIR, [3] \$TEMPDIR, [4] \$DEST**. This means for example that, if you use "*.DLL" in a \$SYSDIR line, and then you put something like PROG.DLL in a \$DEST line, the file PROG.DLL will still be installed into the Windows SYSTEM directory, because the \$DEST entry is resolved last. Please note this point.

Note also that each line cannot be longer than 220 characters, and that the total number of files to be installed to the Windows SYSTEM directory cannot exceed **120**.

Please **NOTE** that in the case of files compressed with the -r switch, you should use the real (original) names of the **uncompressed** files, and **NOT** the names of the compressed files. For example, if the file **MYPROG.DLL** was compressed to **MYPROG.DL_**, you should put **MYPROG.DLL** on this line. The compressed filenames are only allowed on **\$DISK** lines.

[ii] Where ever this appears elsewhere in a line, the "\$SYSDIR" is replaced with the Windows SYSTEM directory. This use of the \$SYSDIR reserved word is only useful in the "\$ICON", and "\$EXEC" lines.

EXAMPLES:

\$SYSDIR=PROG1.DLL;PROG2.DLL;PROG2.DLL;MYDRV.DRV

\$SYSDIR=RATTER.DLL;RETTO.DLL;DRAT.DRV;*.VBX;*.TTF

\$SYSDIR=ROTTA.DRV;ROUTER.DRV;TROUBLE.DRV

See also;

\$DEST

\$TARGET

\$TEMPDIR

\$WINDIR

\$TEMPDIR

[i] Whenever this appears at the beginning of a line, the files on that line are installed to the TEMP directory (instead of the target directory). There can be up to 30 file names, each separated by a semi-colon. Some **limited** use of wildcards is allowed here - if you want to use

wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.TMP;*.\$\$\$**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible. You can have only ONE such line.

[ii] Where ever this appears elsewhere in a line, the "\$TEMPDIR" is replaced with the TEMP directory. This use of the \$TEMPDIR reserved word is only useful in the "\$INI", and "\$EXEC" lines.

See also;

\$DEST

\$TARGET

\$SYSDIR

\$WINDIR

\$AUTO-REPLACE

Use this to specify any files that should be replace automatically if they already exist (i.e., without first prompting the user for confirmation).

This reserved word if effective for matching files which exist in the target directory, and which are NOT newer (by their date stamp) than the files being installed. If the existing file has got a more recent date stamp than the one being installed, then the user WILL be prompted before it is overwritten.

There can be up to 20 \$AUTO-REPLACE lines, and up to 30 file names on each line (separated by semi-colons). Some **limited** use of wildcards is allowed here - if you want to use wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.TXT;*.INI;*.DLL;*.EXE**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible.

Please **NOTE** that in the case of files compressed with the -r switch, you should use the real (original) names of the **uncompressed** files, and **NOT** the names of the compressed files. For example, if the file **MYPROG.DLL** was compressed to **MYPROG.DL_**, you should put **MYPROG.DLL** on this line. The compressed filenames are only allowed on \$DISK lines.

The Syntax is;

\$AUTO-REPLACE=<filename(s)>

EXAMPLES:

\$AUTO-

REPLACE=PROG1.EXE;PROG1.DLL;PROG4.DLL;MYPROG.INI

\$AUTO-REPLACE=RATTER.RAT;RETTO.RET;DRAT.DRA

\$AUTO-REPLACE=ROTTOT.ROT;ROUTER.RUT;TROUBLE.HUT

See also;

EXISTING FILES

\$FORCE-OVERWRITE-OLDER-FILES

\$SKIP-IDENTICAL-FILES

\$INI

This is used to specify any ASCII files that configuration information should be written into. Normally, these will be INI files of some sort, but they can be any file, as long as any such file is in ASCII format. You have up to 20 \$INI lines - and each line **MUST** be in the format prescribed below;

Each line must contain at least 4 entries - each separated with a semi-colon

[a] the first entry is the NAME of the file to be written to - a full path must be supplied - otherwise, the file is presumed to be in the WINDOWS directory. You can use "\$DEST" here, to specify files in the directory tree of the target directory. If the file does not exist, it is created.

[b] the second entry is the title of the section ("Application name" in Windows API speak) that should contain the entry.

[c] the third entry is the name of the entry you wish to make ("Keyname" in Windows API speak)

[d] the fourth entry is the string that you wish to associate with the entry.

[e] you can have an **optional** fifth entry **NO-REPLACE**. Use this to signify that an existing entry should not be replaced. By default, an existing entry in an INI file will be replaced by the ones specified in the \$INI lines. Using **NO-REPLACE** as the fifth entry in a \$INI line will ensure that while an entry will be made if none already exists, old entries will be left intact.

The Syntax is;

\$INI=<Filename>;<Section>;<KeyName>;<String>[;NO-REPLACE]

EXAMPLES:

\$INI=\$DEST\MYPROG.INI;CONFIG;STARTUP;PROG.EXE - FE=XDS.XCL

\$INI=WIN.INI;EXTENSIONS;GFD;\$DEST\BIN\GFD2.EXE ^.GFD

\$INI=C:\AUTOEXEC.BAT;MYPROG;SET PROGDIR;\$DEST;NO-REPLACE

See also;

\$DEST

\$SYSDIR

\$WINDIR

\$GROUP

This is used to specify the DEFAULT name of the Program Manager Group in which the icons will be created. This can be the name of an existing group (e.g., "Accessories", "Main", etc.) in which case, the items will just be added to the ones already in that group. However, you may specify a completely new group. If this does not exist, it will be created.

You can have only ONE such line - but you can specify other group names for different icons in the \$ICON reserved word.

\$GROUP can also take an extra (and optional) parameter - the word **AUTO** or the word **DISABLE**. If used, this parameter should appear after the group name and should be separated from the group's name by a semi-colon.

1. **AUTO** - means create a group automatically - do not allow the user to uncheck the "Create Program Manager Group" checkbox.

2. **DISABLE** - means do NOT create any group at all - and do not allow the user to specify that a group should be created.

If either of these parameters is used, then the checkbox will not be presented. In none of them is used, then the checkbox will be presented and the user will have a choice. Both of these options deny the user a choice in the matter (i.e., either the group will be created automatically or it will not be created at all, regardless of what the user may want).

The Syntax is;

\$GROUP=<groupname>[;parameter]

EXAMPLES:

\$GROUP=My Program

\$GROUP=My Program;AUTO

\$GROUP=My Program;DISABLE

See also;

\$ICON

\$ICON

This is used to specify the names of the files for which you want Program Manager icons to be created. There can be a maximum of 45 icons.

Each \$ICON line should contain only **ONE entry**. This is the name of

the file to create an icon for (this could be a program file plus a parameter or any other file). This should be followed by a semi-colon, and after the semi-colon, the title that Program Manager should give to the icon; and (optionally), preceded by a semi-colon, the name of any other group (i.e., if different from the one in the \$GROUP reserved word) that the icon should be created in; and (optionally), the name of the .ICO file to use for the file.

If no group is specified on this line, then the one pointed to by the \$GROUP reserved word will be used.

If no external .ICO file is specified, then Program Manager will use the first icon it finds in the specified file, or if the file has no icon, then a default icon will be used.

If you specify the name of an external .ICO file, then the full path name of the icon file must be provided, AND, that path MUST be the same as the path of the file that a Program Manager icon is being created for. What this means is that the full path of that file must be the first thing on the \$ICON line (i.e., you cannot specify an executable, and then the file as an argument to that executable).

Secondly, if you specify an external .ICO file, then you MUST also specify the group in which the icon will be created (i.e., there must be 4 entries on the \$ICON line in such cases). In this case, you can simply put \$GROUP as the group name.

The Syntax is;

\$ICON=<filename>;<title>[;<group>;<.ICO file>]

EXAMPLES:

\$ICON=\$DEST\MYMAIN.EXE;Cool Prog v1.20

\$ICON=BACKUP.EXE;Backup Applet;Accessories

\$ICON=\$DEST\MYPROG.HLP;My help file;\$GROUP;\$DEST\PROG.ICO

\$ICON=\$DEST\README.TXT;Readme file;\$GROUP;\$DEST\TEXT.ICO

\$ICON=NOTEPAD.EXE REGISTER.TXT;Registration documentation

See also;

\$DEST

\$GROUP

\$SYSDIR

\$WINDIR

\$PRE-EXEC

This line is **optional**. It specifies the name(s) of any program(s) that should be run during the installation, as part of the installation process. These programs will be run immediately after the files have been copied from the disks. Install will **try** to wait for these programs to

terminate, before continuing. Such attempted waiting will work for Windows programs, but will fail if used to run **DOS programs** under OS/2.

There can be only ONE such line, but it may contain up to 5 programs, each separated with a semi-colon.

The Syntax is;

\$PRE-EXEC=<program name> [parameters] [<other program>]

EXAMPLE:

\$PRE-EXEC=\$WINDIR\EXPAND.EXE \$DEST\REE.BI_;\$TEMPDIR\GAGOFF.EXE > NUL

See also;

\$DEST

\$EXEC

\$SYSDIR

\$TEMPDIR

\$WINDIR

\$CLEANUP

This is optional. It specifies the name(s) of any temporary files(s) that should be deleted after the installation. Such deletions (if any) will be done immediately after any \$PRE-EXEC lines have executed and returned. If there is no \$PRE-EXEC line, then the deletions will be done after the \$INI lines have been processed. If there are no \$INI lines, then the deletions will be immediately after the \$DISK lines have been processed. If the specified files do not exist, they are simply ignored.

There can be up to 20 \$CLEANUP lines. Each line should contain only ONE entry. The only other reserved words permissible on these lines are \$DEST and \$TEMPDIR. I will never add \$WINDIR or \$SYSDIR to this list, because I can see no reason why one would want to delete files from people's Windows or SYSTEM directories. You can use wildcard characters here, but note that the program will NOT accept "*.*".

Please use this reserved word with care. I accept no responsibility for any problems caused by using it.

The \$TEMPDIR can be used here with the Install program files in cases where you choose to use SETUP.EXE as a loader. This way, Install can cleanup the files which have been copied by SETUP.EXE to the TEMP directory.

The Syntax is;

\$CLEANUP=<filespecs>

EXAMPLES:

\$CLEANUP=\$TEMPDIR\TMP*.*

\$CLEANUP=\$DEST\TEMPFIL.INI
\$CLEANUP=\$TEMPDIR\INSTALL.EXE
\$CLEANUP=\$TEMPDIR\WINSTAL*.*

See also;

\$DEST

\$TEMPDIR

\$EXEC

This line is **optional**. It specifies the name(s) of any program(s) that should be run immediately after the installation is completed (with any optional parameters to be passed to the programs). You can have only ONE such line, but you can put as many as 5 programs on this line, each separated by a semi-colon.

The Syntax is;

\$EXEC=<program name> [parameters] [;<other program>]

EXAMPLE:

\$EXEC=CONFIG.EXE -DIR=C:\TEMP; MYPROG1.EXE;
MYPROG2.EXE -NEW

See also;

\$DEST

\$PRE-EXEC

\$SYSDIR

\$TEMPDIR

\$WINDIR

\$WINDOW

This line is **optional**. It should specify whether you want Install to start up maximized or not. If the entry here is **MAXIMIZE** then Install will start maximized - otherwise it will just start normally. You can have only ONE such line.

The Syntax is;

\$WINDOW=MAXIMIZE

\$COPYBUFFER

This line is **optional**. It sets the size of the buffer used by Install to copy the files. The buffer size should be a whole number, representing the number of BYTES to be used. This number **MUST** be between 2048 and 32760. If it is set lower than 2048, then Install will replace the supplied value with 2048 - and if it is set higher than 32760, then Install will use 32760.

The higher the buffer, the faster the files are copied. However, the

buffer size also dictates;

[a] the frequency with which the "percent" meter is updated

[b] the frequency with which Install will "yield" the CPU and allow Windows to do other things (each time COPYBUFFER bytes are copied, Install "yields" for 128 milliseconds).

Therefore, if the number is set too high, the percent meter will not be updated frequently enough, and the display might look odd. If, on the other hand, the setting is very low, then the percent meter will be updated frequently, but the file copying will become much slower.

The DEFAULT value is 8190, and this will be used if this setting is left empty. I suggest a setting of 16384 (i.e., 16kb) as a good setting which adequately compromises between speed of copying, and the frequency of the progress bar's being updated.

The Syntax is;

\$COPYBUFFER=<buffersize>

EXAMPLE:

\$COPYBUFFER=4095

\$WINDOW-BACKGROUND

This line is **optional**. It can be used to set the background color of the main window of Chief's Installer Pro. The default is to have a light gray background for the main dialog, and a white background for other dialogs (the light gray will also be used for other dialogs if you use the \$PAINTDIALOGS command). Because the background is a Windows brush handle, the only valid values for this setting are 0, 1, 2, 3, or 4.

0 = **White Brush**

1 = **Light gray Brush**

2 = **Gray Brush**

3 = **Dark gray Brush**

4 = **Black Brush**

If you use this to change the window background, be sure to also set the \$TEXT-BACKGROUND (below) to an appropriate setting. For example, if this setting is 2 (dark gray) then the text background should be set to 128,128,128 (so that the window and text backgrounds should match).

The Syntax is;

\$WINDOW-BACKGROUND=<value>

EXAMPLE:

\$WINDOW-BACKGROUND=1

See also;

\$TEXT-BACKGROUND

\$TEXT-BACKGROUND

This line is **optional**. It can be used to set the background color of the **text** in the main window of Chief's Installer Pro. The default is to have a light gray background. Unless there is a pressing need to use another color, the color used here should be the same as that used for the \$WINDOW-BACKGROUND.

The color used here can either be either;

[a] one long integer value (you can use hexadecimal values in Pascal notation) - see below for explanation

or

[b] three values represent RGB (red, green, blue) values.

If using RGB values, they should be separated by commas, or semi-colons (e.g: 128,128,128 - for a dark gray background)

If using a hexadecimal value (those that begin with \$00 and then are followed by **SIX** values). The SIX values here represent Blue, Green, Red - or reversed RGB. In this respect, **FF** turns the color to full intensity, **00** turns it off, and any other value varies the intensity.

Note that the color that results from any value depends on the display driver of the user (particularly the number of colors). For a 256 color setup, you can use the following **EXAMPLE** values;

1. **White** : \$00FFFFFF
2. **White** : 255,255,255
3. **Black** : \$00000000
4. **Black** : 0,0,0
5. **Dark Gray** : \$00808080
6. **Dark Gray** : 128,128,128
7. **Red** : \$000000FF
8. **Red** : \$255,0,0
9. **Blue** : \$00FF0000
10. **Blue** : 0,0,255
11. **Light Cyan** : \$00FFFF00
12. **Green** : \$0000FF00
13. **Yellow** : \$0000FFFF
14. **Magenta** : \$00FF00FF
15. **Light Gray** : \$006F9FFF
17. **Light Gray** : 192,192,192
16. **Gray** : \$00C0C0C0

The Syntax is;

\$TEXT-BACKGROUND=<color value>

EXAMPLES:

\$TEXT-BACKGROUND=192,192,192

\$TEXT-BACKGROUND=128,128,128

\$TEXT-BACKGROUND=\$00FFFFFF

See also;

\$WINDOW-BACKGROUND
\$BANNER-TEXT-COLOR
\$BANNER-TEXT-BACKGROUND
\$BANNER-WINDOW-BRUSH

\$PAINTDIALOGS

This line is **optional**. It is for use in those cases when you want the status dialogs to be painted with the same text and background colors as the main Install window (the default is that the status "percent" dialogs have a white background). This line takes no parameter.

The Syntax is;

\$PAINTDIALOGS

\$SETCHECKBOX

This line is **optional**. It automatically checks the checkbox titled "create Program Manager item". This line takes no parameter.

The Syntax is;

\$SETCHECKBOX

\$NO-END-DIALOG

This line is **optional**, and is not very useful. All it does is to suppress the final dialogs which inform the user about whether the installation was successful or not, and that the installation is completed. The default behaviour of Install is to present these dialogs to the user. Use this reserved word to disable that feature.

If this feature is used, the warning dialog that comes up if the size of the files actually installed is less than 98% of the size stated in the \$SPACE reserved word is also disabled. This line takes no parameter.

The Syntax is;

\$NO-END-DIALOG

\$NO-PATH-DIALOG

This line is **optional**. It disables the dialog box which asks the user whether the target directory should be added to the PATH statement in AUTOEXEC.BAT. The default behaviour is to present this dialog. Use this reserved word to disable that feature. This line takes no parameter.

\$SHOW-FILE-PERCENT

This line is **optional**. What it does is to show a small percent meter for the progress of each individual file being installed (i.e., in addition to the large percent meter which shows the progress of the whole

installation process). This reserved word takes no parameter.

The Syntax is;

\$SHOW-FILE-PERCENT

\$MAKE-UNINSTALL-LOG

This line is **optional**. What it does is to cause Chief's Installer Pro to create a log file of all the changes it is making to the system. This file is a binary file (to prevent tampering with it) and is called **UNINSTAL.LOG**. It is created in the target directory, and should be left there. This file will be used by the **UNINSTALLER** to uninstall the program, if the user so wishes.

If the user is installing over an existing installation and a copy of UNINSTAL.LOG already exists, Chief's Installer Pro will just add the new information to the end of the existing one. This may result in some information being duplicated in the file, but will not lead to any strange result. The uninstaller is smart enough to handle any duplicated information.

This reserved word can take as an **optional** parameter the name of the file to use as the LOG file for **UNINSTAL.EXE**. The parameter should be separated with a semi-colon, and should contain a filename only (**no path**). If no filename is provided, the default name UNINSTAL.LOG will be used. If a filename is used here, it **MUST** be supplied as a **SECOND** parameter to UNINSTAL.EXE

This reserved word can also take another **optional** parameter - the word **OVERWRITE**. When this is used, the installer marks the LOG file so that UNINSTAL.EXE will overwrite every file and directory which it has deleted, so that they cannot be undeleted. This parameter should be used with **great care**.

The Syntax is;

\$MAKE-UNINSTALL-LOG[;logfilename][;OVERWRITE]

EXAMPLES:

\$MAKE-UNINSTALL-LOG

\$MAKE-UNINSTALL-LOG;OVERWRITE

\$MAKE-UNINSTALL-LOG;VER2.LOG

\$MAKE-UNINSTALL-LOG;VER2.LOG;OVERWRITE

See also;

[THE UNINSTALLER](#)

\$USER-OPTION

The user option lines are **optional**. By default, Chief's Installer Pro will install **all** the files which are specified in the **\$DISK** lines. However, sometimes, the user will only want to install the binaries, or the

documentation, or the libraries, or any other partial installation.

The **\$USER-OPTION** reserved word gives you the means of providing user-selectable installation options, for various parts of your program. So you can split your program's installation into program files, help files, libraries, dictionaries, bitmaps, etc., etc., and the user will be given a dialog with check boxes which allows him to choose, or just to install everything. This means that your users will now have the facility for incremental installation of different parts of your program.

There can be up to **10** \$USER-OPTION lines, each of them specifying a different optional part of your program. If you specify a user option, you **must** also use the **\$OPTIONAL** reserved word (see below) to specify the files which make up that user option. In such cases, appropriate check boxes will appear.

Each \$USER-OPTION line must contain the title of the option (this is the text that will appear beside it's check box), followed by a semi-colon, and then the amount of disk space (in bytes) which the option will require. This amount will be added to the amount specified in the **\$SPACE** reserved word, such that if you make all the different parts of your installation optional, then the \$SPACE line must specify 0 (zero) as the required disk space.

NOTE: The bonus program **AUTOCALC.EXE** can be used to calculate the required space automatically. Please read **AUTOCALC.TXT** for fuller details.

\$USER-OPTION lines can also take an extra (and optional) parameter - the word **UNCHECKED**. If used, it must be put last, separated from the size of the option's files by a semi-colon. If it is used, the checkbox for the option is not checked when the installer starts. The user can check it afterwards.

The Syntax is;

\$USER-OPTION n = <title>;<disk space needed>[;UNCHECKED]

where:

n = any number from 1 to 10

<title> = the text to show beside the option's check box

<disk space needed> = the amount of disk space required by the option (in bytes)

EXAMPLES:

\$USER-OPTION1=Program files;171000

\$USER-OPTION2=Optional DLL files;10000

\$USER-OPTION3=Optional executables;104000

\$USER-OPTION4=Readme files;14384;UNCHECKED

See also;

\$OPTIONAL

\$SPACE

\$OPTIONAL

This reserved word is used to specify the files that make up any user-selectable installation options specified with the **\$USER-OPTION** reserved word. Each line should specify a list the files that make up the particular option number, each file name separated from the next one by a semi-colon. Some **limited** use of wildcards is allowed here - if you want to use wildcards, then it must be an asterix, followed by a dot, and then the full extension of the files - e.g., ***.TXT;*.INI;*.DLL;*.EXE**. Great care must be taken not to confuse the program when using wildcards in this way. Careless use of wildcards might lead to files going where they were not meant to go. It is better to name individual files whenever possible. A maximum of 30 file specifications is allowed on each line - but note that each line cannot be longer than 220 characters in total.

There can be up to 10 \$OPTIONAL lines for **each** option specified by a \$USER-OPTION line. This facility is to allow for situations where all the file names will not fit on one line. The fact that you can have up to 10 lines for each option number, and that each line can contain up to 30 file names, means that you can in theory have up to 300 files making up each option. However, please do not go overboard with this, because **each** file name on **each** \$OPTIONAL line has to be checked against **every** file being installed, to see whether it should be installed or not. Therefore, if there are too many files in the \$OPTIONAL lines, the installation process will be slowed down (this might not be a problem on machines with fast CPUs).

The Syntax is;

\$OPTIONAL n = <filenames>

where:

n = any number from 1 to 10 (corresponding to the relevant \$USER-OPTION)

<filenames> = the files which make up the option - each separated by a semi-colon

EXAMPLES:

\$OPTIONAL1=INSTALL.EXE;WINSTALL.HLP;INSTALL.TXT;WINSTALL.INF

\$OPTIONAL1=SAMPLE1.INF;SAMPLE2.INF;SAMPLE3.INF;SAMPLE4.INF

\$OPTIONAL2=ENGLISH.dll;dansk.dll;deutsch.dll

\$OPTIONAL3=UNINSTAL.EXE;SETUP.EXE;

\$OPTIONAL4=*.WRI;*.TXT;*.DOC;*.PS;READ.ME

See also;

\$USER-OPTION

\$BANNER-FONT

This reserved word specifies the font to use for the **banner** text that will be displayed in the background of Chief's Installer Pro's dialog window (on the Windows desktop). Most of the Windows **TRUETYPE** fonts can be used here. The font will be in bold faced characters, and will be italicised.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and it's presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If the named font does not exist on the system, then Windows will try to use a substitute font, or at the least, a COURIER font. If this line is empty, then Chief's Installer Pro will default to using the TrueType **TIMES NEW ROMAN** font.

The Syntax is;

\$BANNER-FONT=

EXAMPLE:

\$BANNER-FONT=ARIAL

See also;

[\\$BANNER-FONT-SIZE](#)

[\\$BANNER-MESSAGE](#)

[\\$BANNER-TEXT-COLOR](#)

[\\$BANNER-TEXT-BACKGROUND](#)

[\\$BANNER-SHADOW-COLOR](#)

[\\$BANNER-WINDOW-BRUSH](#)

[\\$WINDOW](#)

\$BANNER-FONT-SIZE

This reserved word specifies the "point" size of the font used to display the banner. The size should be a whole number.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and it's presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If this line is empty, Chief's Installer Pro defaults to using **35 point**. Note that you should be careful to cater for the smallest display resolutions (practically, 640x480 displays). Therefore the font size

should be small enough for the banner message to fit in a standard VGA screen).

The Syntax is;

\$BANNER-FONT-SIZE=

EXAMPLE:

\$BANNER-FONT-SIZE=45

See also;

\$BANNER-FONT

\$BANNER-MESSAGE

\$BANNER-TEXT-COLOR

\$BANNER-TEXT-BACKGROUND

\$BANNER-SHADOW-COLOR

\$BANNER-WINDOW-BRUSH

\$WINDOW

\$BANNER-MESSAGE

This line specifies the message to be displayed as the banner for your installation. This message is displayed in the banner window, on the Windows desktop. The message should be short enough to fit on one line, taking into account the font being used, and its size.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and its presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If this line is empty, Chief's Installer Pro will default using the title of your program as specified on the **\$TITLE** line, and the words "Welcome to" will be prepended to that title.

The Syntax is;

\$BANNER-MESSAGE=<banner message>

EXAMPLE:

\$BANNER-MESSAGE=This is a Great Program!

See also;

\$BANNER-FONT

\$BANNER-FONT-SIZE

\$BANNER-TEXT-COLOR

\$BANNER-TEXT-BACKGROUND

\$BANNER-SHADOW-COLOR

\$BANNER-WINDOW-BRUSH

\$TITLE

\$WINDOW

\$BANNER-TEXT-COLOR

This reserved word specifies the color to be used for the banner text. This color can be either one long integer value (TColorRef in Windows) or three RGB values. The values that can be used here are the same as those that can be used in the **\$TEXT-BACKGROUND** reserved word. Please see the documentation on it for further details.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and its presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If this line is empty, install defaults to using a white text color (**\$00FFFFFF**).

The Syntax is;

\$BANNER-TEXT-COLOR=<color value>

EXAMPLE:

\$BANNER-TEXT-COLOR=\$00C0C0C0

See also;

\$BANNER-FONT

\$BANNER-FONT-SIZE

\$BANNER-MESSAGE

\$BANNER-TEXT-BACKGROUND

\$BANNER-WINDOW-BRUSH

\$BANNER-SHADOW-COLOR

\$TEXT-BACKGROUND

\$WINDOW

\$BANNER-TEXT-BACKGROUND

This reserved word specifies the color to be used for the banner text background. This color can be either one long integer value (TColorRef in Windows) or three RGB values. The values that can be used here are the same as those that can be used in the **\$TEXT-BACKGROUND** reserved word. Please see the documentation on it for further details.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and its presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE**

reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If this line is empty, install defaults to using a blue text background (**\$00800000**).

The Syntax is;

\$BANNER-TEXT-BACKGROUND=<color value>

EXAMPLE:

\$BANNER-TEXT-BACKGROUND=\$00800000

See also;

\$BANNER-FONT

\$BANNER-FONT-SIZE

\$BANNER-MESSAGE

\$BANNER-TEXT-COLOR

\$BANNER-SHADOW-COLOR

\$BANNER-WINDOW-BRUSH

\$TEXT-BACKGROUND

\$WINDOW

\$BANNER-SHADOW-COLOR

This reserved word is **optional**. It is used to give the banner text a "shadow". The color value here is in the same format as **\$BANNER-TEXT-COLOR** and **\$BANNER-TEXT-BACKGROUND**. The shadow is disabled by default (by giving the shadow color a default value less than 0). If the value is 0 or higher, then the shadow becomes enabled, and the **\$BANNER-TEXT-BACKGROUND** line becomes disabled.

The Syntax is;

\$BANNER-SHADOW-COLOR=<color value>

EXAMPLE:

\$BANNER-SHADOW-COLOR=255,0,0

See also;

\$BANNER-FONT

\$BANNER-FONT-SIZE

\$BANNER-TEXT-COLOR

\$BANNER-TEXT-BACKGROUND

\$BANNER-SHADOW-COLOR

\$BANNER-WINDOW-BRUSH

\$TITLE

\$WINDOW

\$BANNER-WINDOW-BRUSH

This reserved word specifies the color to be used to paint the

background of the banner window. The painting is not done as a straight color. Rather, it **starts** as the color you specify (at the top of the screen), and gradually changes, until it reaches **black** (at the bottom of the screen). This presents a pleasant visual effect. This is even more so if you use the same or nearly the same color as the one used in the **\$BANNER-TEXT-BACKGROUND** line.

This color can be either one long integer value (TColorRef in Windows) or three RGB values. The values that can be used here are the same as those that can be used in the **\$TEXT-BACKGROUND** reserved word. Please see the documentation on it for further details.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and its presence is not needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used. This is because when Chief's Installer Pro's main window is maximized, the banner is not displayed at all (for obvious reasons).

If this line is empty, install defaults to using a blue color (**\$00800000**).

The Syntax is;

\$BANNER-WINDOW-BRUSH=<color value>

EXAMPLE:

\$BANNER-WINDOW-BRUSH=100010

See also;

\$BANNER-FONT

\$BANNER-FONT-SIZE

\$BANNER-MESSAGE

\$BANNER-TEXT-COLOR

\$BANNER-TEXT-BACKGROUND

\$BANNER-SHADOW-COLOR

\$TEXT-BACKGROUND

\$WINDOW

\$BITMAP

This reserved word specifies a Windows bitmap file to display in the banner window. The bitmap will be stretched to fill the screen, and therefore the painting will often be slow. The banner text will then be displayed on top of the bitmap. If a bitmap file is specified the painting of the banner window background, as specified in the **\$BANNER-WINDOW-BRUSH** line will not take place, since the bitmap will be occupying the whole screen. Only standard Windows .BMP files are supported.

This line is **optional**. It also depends on the file **WINSTALP.DLL**. That file contains all the **banner** functionality, and its presence is not

needed for Chief's Installer Pro to function (you just won't get any banner). If the file is not found by Chief's Installer Pro, this line will have no effect. It is also ineffective if the **\$WINDOW=MAXIMIZE** reserved word is used.

\$BITMAP can take an extra optional parameter **NORMAL**. If used, this should appear **AFTER** the name of the bitmap file, separated by a semi-colon (e.g., **\$BITMAP=winstall.bmp;normal**). This parameter disables the stretching of the bitmap, and the bitmap will be displayed in its normal size, centered on the screen. In such cases also, the main dialog will be hidden once the "Start Install" button is clicked - so that more of the bitmap will be visible.

This bitmap line is there only as an added extra. The "bitblitting" is often very slow and so many not be ideal in many cases. But it is there anyway. You may simply ignore it. If the line is empty, then Chief's Installer Pro will by default look for a file called **WINSTALL.BMP** to use for the background bitmap. If **WINSTALL.BMP** is not found, then the program simply use the banner window brush value to paint the background of the banner window.

The Syntax is;

\$BITMAP=<bitmap file name>[;NORMAL]

EXAMPLES:

\$BITMAP=MYPROG.BMP

\$BITMAP=WINSTALL.BMP;NORMAL

See also;

\$BANNER-MESSAGE

\$BANNER-TEXT-COLOR

\$BANNER-TEXT-BACKGROUND

\$BANNER-SHADOW-COLOR

\$BANNER-WINDOW-BRUSH

\$WINDOW

\$VERIFY-INSTALL-DISKS

This line is **optional**. By default, when the user is prompted to insert a particular numbered disk (e.g., "Please insert disk 4 in drive"), Chief's Installer Pro does not perform any check to verify that the disk being inserted is actually the correct one. You can however use this reserved word to force Chief's Installer Pro to perform these checks.

When this reserved word is used, Chief's Installer Pro will check each installation disk (from disk 2 onwards) to verify that it is the correct disk. This check is performed by looking for a file on the disk, which corresponds to the **\$DISK** being installed, but with the extension **.DSK**. Thus for example, if Chief's Installer Pro asked for **disk 4** to be inserted in the drive, it will check for the existence of a file called **\$DISK4.DSK** on any disk that is inserted. If the file exists, then this is

taken as the correct disk, and the installation continues. If the file is not found on the disk, then the user is prompted to insert the disk again, and this will go on until either the correct disk is inserted, or the user clicks on "cancel".

The contents of the **.DSK** file are irrelevant. It can be an empty file - but the file must exist. If you use this reserved word, you have to create a **.DSK** file on each of the installation disks (from disk 2 onwards).

This line takes no parameter.

See also;

\$DISK

\$AUTO-CLICK-BUTTON

The main window of Chief's Installer Pro has got four push buttons, labelled (in English) "Start Install", "Abort", "Help", and "View Read Me". You can use this reserved word to send a mouse click to any one of these buttons. When you send a mouse click in this way, the effect is exactly as if the user had clicked on that push button with the left mouse button. This will activate whatever the push button is supposed to do.

The line takes one parameter - the ID of the button to send the mouse click to. For this purpose, **1=Start Install**, **2=Abort**, **3=Help**, and **4=View Read Me**. This line is useful for example, for starting the installation without giving the user any opportunity to make any selections or choices, or for clicking on the "Help" or "Readme" button so that your help file or your readme file (WINSTALL.TXT) will be loaded automatically (i.e., to force your users to read your documentation).

This line is optional.

The Syntax is;

\$AUTO-CLICK-BUTTON=<button ID>

EXAMPLE:

\$AUTO-CLICK-BUTTON=1

See also;

COMMAND LINE OPERATION

\$NO-CTL3D.DLL

This line is **optional**, and is probably not very useful. It is for the purpose of giving people the option of dispensing with the use of CTL3DV2.DLL. If this line is found, the Chief's Installer Pro will not use the 3D dialog effects in CTL3DV2.DLL. The question is "why would anyone want to do this?". The answer is that some Windows video drivers are buggy and might not necessarily want to co-exist peacefully

with CTL3DV2.DLL in all circumstances.

Not mentioning any names, but I know of one company which produces buggy Windows drivers for their display cards which sometimes fall into the category described above. I personally do not use this reserved word, and it may indeed be unnecessary to use it. However, I think that it is good to have the option.

This line takes no parameter.

\$RESTARTWIN-MESSAGE

Chief's Installer Pro provides support for restarting Windows if any active shared file was overwritten. A dialog asking for confirmation appears automatically if any active shared file was overwritten during the installation. The text on this dialog can be changed by this reserved word. If this reserved word is not used, a default message is used, which tells the user that at least once active DLL has been replaced, and that the user should restart Windows immediately.

The message on this line can be up to 200 characters. Obviously, that is too wide for a dialog box. Therefore I have decided to support **one** formatting control here. You can insert carriage returns at any point in the message by using the "newline" code (i.e., **\n**).

This line can be used to DISABLE the dialog that asks if the user wants to restart Windows. To disable the feature, use **\$RESTARTWIN-MESSAGE=DISABLE**.

Please note that if you disable this feature in this way, it is up to you to inform your user that Windows must be restarted when an active DLL has been overwritten. How you will ascertain this fact is beyond me. I have only included this feature because a user asked for it. If you use it, you are on your own.

EXISTING-FILES

Chief's Install Pro will check in the target directories for existing copies of every file being installed. If no copy of the file exists, then the installation will proceed. If the file exists, the existing copy and the copy being installed will both be checked to see which one is newer.

Chief's Install Pro uses two methods of deciding whether or not a file is older than another. In the case of **shared binary files** (i.e., the ones that go into the Windows and/or the Windows SYSTEM directory), the version information in the files will first be compared. If there is no version information in the files, then their date stamps of the file will be compared. In the case of other files, only the date stamps will be compared (except in the case of proprietary DLLs if you use the **\$CHECK-MY-DLL-VERSIONS** reserved word).

When Chief's Install Pro has ascertained which of the two copies of a file is newer, what happens next depends on the choices you made in

your INF file, and/or the choices made by your user. Normally, Chief's Install Pro will simply display a dialog box informing the user that a copy (or a newer copy) of the file already exists in the target directory, and then show the user the details of the two copies.

You can decide in advance that certain files specified by you should be over-written automatically (**\$AUTO-REPLACE**) or that all **older** versions of files should be over-written automatically (**\$FORCE-OVERWRITE-OLDER-FILES**). In this case, a file is regarded as "older" if it is **not newer**. In the case of files which both have version information, if the version number is the same (e.g., they are both 1.1) then the date stamps will be used to decide which is "older". If the date stamps are the same, then the one that already exists in the target directory is treated as "older" than the one being installed. In the case of files without version information, if both files have the same date stamp, then the existing copy is still regarded as "older".

SHARED FILES;

For the purposes of the installation, Chief's Install Pro will regard a file as a **shared** file only if the following conditions apply;

[1] the file is being installed into the Windows or Windows SYSTEM directory, and

[2] the file's extension is either; [a] **.DLL** or [b] **.EXE** or [c] **.VBX** or [d] **.OCX** or [e] **.DRV** or [f] **.CPL**

See also;

[\\$AUTO-REPLACE](#)

[\\$CHECK-MY-DLL-VERSIONS](#)

[\\$FORCE-OVERWRITE-OLDER-FILES](#)

[\\$SKIP-IDENTICAL-FILES](#)

[\\$VERSION-INFO-TITLE](#)

[\\$VERSION-INFO-MESSAGE](#)

[\\$VERSION-COPY-ERROR-MESSAGE](#)

\$CHECK-MY-DLL-VERSIONS

This line is **optional**. By default, when another copy of a **.DLL** file already exists in the target directory, Chief's Install Pro will check for version information in that DLL only if the target directory is the Windows directory or the Windows SYSTEM directory. That is, only **SHARED** DLLs will be normally checked for version information. DLLs which are going into your application's directory for example will only be checked for their date stamps.

If you want **ALL** DLL files to be checked for their version information (i.e., regardless of their destination directory), then you should use this reserved word. This line takes no parameter.

See also;

EXISTING FILES

\$VERSION-INFO-TITLE

This line is **optional**. By default, when Chief's Install Pro is reporting the version information on an existing copy of a shared file, the version number is reported under the heading "File Version". You can use this reserved word to change that string to something else. This is really useful only for those who want to display that string in a language other than English. If you change this string, please try to make the replacement as short as possible.

The Syntax is;

\$VERSION-INFO-TITLE=<title>

EXAMPLE:

\$VERSION-INFO-TITLE=Product Version Number

See also;

EXISTING FILES

\$VERSION-INFO-MESSAGE

\$VERSION-INFO-MESSAGE

This line is **optional**. When Chief's Install Pro has retrieved the version information on an already existing copy of a shared file, a dialog informs the user that a copy of the file already exists and then asks for over-write permission. By default, this dialog will not contain any further explanation of the situation, and will not make any recommendation as to the course of action to be taken.

You can use this reserved word to provide some explanation and/or a recommended course of action. If anything appears on this line, it will be added to the dialog.

The message on this line can be up to 200 characters. Obviously, that is too wide for a dialog box. Therefore I have decided to support **one** formatting control here. You can insert carriage returns at any point in the message by using the "newline" code (i.e., **\n**).

The Syntax is;

\$VERSION-INFO-MESSAGE=<recommendation/explanation>

EXAMPLE:

\$VERSION-INFO-MESSAGE=You should click on "NO" \n if the target file is NEWER.

See also;

EXISTING FILES

\$VERSION-INFO-TITLE

\$FINAL-MESSAGE

If you want to give your user any final message (after the installation is complete) then you can put that message on this line. The message will be displayed in a dialog box at the tail end of the installation - just after the **\$EXEC** line is executed.

The message on this line can be up to 200 characters. Obviously, that is too wide for a dialog box. Therefore I have decided to support **one** formatting control here. You can insert carriage returns at any point in the message by using the "newline" code (i.e., **\n**).

The Syntax is;

\$FINAL-MESSAGE=<message>

EXAMPLE:

\$FINAL-MESSAGE=Please shut down all applications and restart Windows.

\$VERSION-COPY-ERROR-MESSAGE

This line is **optional**. By default, when Chief's Install Pro is unable to successfully install a file, all that the user will get is an error message that there was an error writing to the file, and then the installation will proceed with the other files.

In the case of **shared files**, the problem may be that the file is currently in use and therefore cannot be over-written. With **shared files**, a temporary copy will normally exist in the TEMP directory (and will **NOT** have been deleted by the installer if the attempt to install it was unsuccessful). Chief's Install Pro will therefore by default display a message advising the user to copy the file manually after closing Windows.

Note that the situation described above will only exist, if [a] the file is a shared file, and, [b] a copy of it already exists in the Windows or Windows SYSTEM directory, and, [c] an attempt to install over the existing copy was unsuccessful.

You may want to change this message described above to something that suits you, and this reserved word allows you to do that. A carriage return will automatically be added at the end of this message, followed by the full pathname of the temporary copy of the file.

Note that because the last thing that appears in the dialog box is the full path name of the temporary file in the TEMP directory, if you use this line to change the error message, you need to express it in such a way that it leads up to the file name.

The message on this line can be up to 200 characters. Obviously, that is too wide for a dialog box. Therefore I have decided to support **one** formatting control here. You can insert carriage returns at any point in the message by using the "newline" code (i.e., **\n**).

The Syntax is;

\$VERSION-COPY-ERROR-MESSAGE<message>

EXAMPLE:

\$VERSION-COPY-ERROR-MESSAGE=Please copy it from the TEMP directory later \n. A temporary copy exists as:

See also;

EXISTING FILES

\$FORCE-OVERWRITE-OLDER-FILES

This line is **optional**. By default, when a copy of the file being installed already exists in the target directory, Chief's Install Pro will ask the user whether the existing copy should be over-written or not. This will be the case even when the existing file is an older version which really ought to be replaced. This could be a bit of a nuisance sometimes, and so you might want older versions of files to be replaced automatically. You use this reserved word to achieve that.

Note that this reserved word is different from the **\$AUTO-REPLACE** reserved word, in that, this one applies to all files, while the former applies only to selected files.

This line takes no parameter.

See also;

EXISTING FILES

\$AUTO-REPLACE

\$CHECK-MY-DLL-VERSIONS

\$SKIP-IDENTICAL-FILES

\$SKIP-IDENTICAL-FILES

This reserved word is **optional**. It causes a file to be skipped if a copy of it already exists in the target directory, and that copy is exactly the same version as the copy on the installation disk. In order to decide whether two files are exactly the same version, their date/time stamps, file sizes, and version information (if the files are shared DLLs) are compared. If there is any discrepancy in any of these, the files are treated as not being the same, and will not be skipped.

The comparisons work correctly in all my tests - but if you are going to use this feature, **please test it thoroughly** with your particular set of files.

See also;

\$AUTO-REPLACE

\$FORCE-OVERWRITE-OLDER-FILES

\$README-BUTTON-TEXT

Chief's Installer Pro provides support for displaying a README file to the user before the installation begins. The readme file should be a plain ASCII file, should not be larger than 8192 bytes, and should be called **WINSTALL.TXT**. A button with the caption "View Read Me" is presented for this purpose. You can change the caption (text) on the "readme" button with this reserved word. The text used here must not be longer than 20 characters.

You can cause the contents of the file to be displayed automatically by setting **\$AUTO-CLICK-BUTTON** to 4 (otherwise the user will have to click on the "readme" button to display the text). If the file WINSTALL.TXT is not found, then the "readme" button will be removed at run time.

The Syntax is;

\$README-BUTTON-TEXT=<button caption>

EXAMPLE:

\$README-BUTTON-TEXT=&Installation Notes

See also;

\$README-FONT

\$README-FONT

This reserved word is **optional**. It is used to change the font in which the text in the "Readme" dialog is displayed, from a proportional font (MS Sans Serif, 9 point) to a FIXED or MONO spaced font (Courier 8 point).

The Syntax is;

\$README-FONT=FIXED

See also;

\$README-BUTTON-TEXT

\$REG-DATA

This reserved word is **optional**. It provides (limited) support for making entries into the Registration Database. You can have up to 20 **\$REG-DATA** lines, and each line can contain only a single entry. The lines should only contain the keys/sub-keys that you want to create (e.g., associating a file extension with your program, etc). The installer will prepend **HKEY_CLASSES_ROOT** to each of your entries, so you should **NOT** use that keyword at all.

Note that this feature depends on the presence of **REGEDIT.EXE** on the user's system.

Entries made in the registration database in this way will be removed by the uninstaller if and when the user chooses to uninstall the program.

The Syntax is;

\$REG-DATA=<subkey>=<value>

EXAMPLES:

\$REG-DATA=ChiefPro = Chief's Installer Pro

\$REG-DATA=.inf = ChiefPro

\$REG-DATA=.chf = ChiefPro

**\$REG-DATA=ChiefPro\shell\open\command = \$DEST\install.exe
%1**

**\$REG-DATA=ChiefPro\shell\print\command = \$DEST\
install.exe /p %1**

\$REG-DATA=ChiefPro\protocol\StdFileEditing\verb\0 = Edit

**\$REG-DATA=ChiefPro\protocol\StdFileEditing\server = \$DEST\
install.exe**

\$DIALOG-ICON

This reserved word is **optional**. It can be used to change the icon that is displayed on the installer's dialogs. The reserved word takes one parameter - a number which corresponds to that of the required icon. The icons must be in a DLL called **WINSTALC.DLL**, and must be given given **numeric names from 3 onwards** (e.g., 3, 4, 5, 6, etc.). Icons 1 and 2 (5.25 and 3.5 inch icons) are inside INSTALL.EXE itself. There can be up to 250 icons in **WINSTALC.DLL**, but only one can be used. If this reserved word is not used, the the default will be the pink 5.25" icon.

The Syntax is;

\$DIALOG-ICON=<icon number>

EXAMPLE:

\$DIALOG-ICON=4

\$FONT

This reserved word is **optional**. It is used to install TRUETYPE fonts. The reserved word takes 2 parameters. The first is the name of the font file (xxx.TTF) which will be used, and the second is the name or description of the font. Note that the description of the font must be accurate - exactly as it appears in the Windows Control Panel. The program will try to create a **.FOT** file in the Windows SYSTEM directory, and make the necessary entries in the Windows INI files. If the font is already installed, it will simply be reinstalled again.

You can have up to 20 **\$FONT** lines. Note that you need to direct the font files to the Windows SYSTEM directory with the **\$SYSDIR** command.

The Syntax is;

\$FONT=<fontfile.TTF>;

EXAMPLES:

\$FONT=ARIALBD.TTF;Arial Bold (True Type)

\$FONT=CHIEFBD.TTF;Bold Chief (True Type)

\$SOURCEDIR

This reserved word is **optional**, and not very useful. It points to the directory from which the program was actually installed. It is not valid at the beginning of a line, and can only be used in the **\$INI**, **\$PRE-EXEC**, and **\$EXEC** lines.

EXAMPLE:

\$INI=\$DEST\PROG.INI;History;SourceDir;\$SOURCEDIR

See also;

[\\$INI](#)

[\\$EXEC](#)

[\\$PRE-EXEC](#)

COMMAND LINE OPERATION

Chief's Installer Pro normally operates in an interactive way. When the program **INSTALL.EXE** is run, a dialog box will be presented to the user, from where the user can select options, click on a button to start the installation, etc. While this is sufficient in most cases, there are situations in which you might want to use your own "pre-installer" (e.g., instead of my own **SETUP.EXE** or for any other purpose). Note that this option is **not** open to you if you are using **SETUP.EXE**.

For such situations, Chief's Installer Pro provides you with the flexibility of running **INSTALL.EXE** with command line parameters. There are three types of parameters that the program can take, and you can use one, or all, or some, or none of them, in any combination.

The first is **/\$TARGET=<target directory>**. When this parameter is used, Chief's Installer Pro will assume that all the options which you have enabled in your INF file have been accepted, and will by-pass the first dialog - the installation will start straight away, without the user having the opportunity to select or unselect any of the options manually. This is useful if you want to ensure that the installation is carried out in a particular way (e.g., to ensure a standard setup on all computers in your company). If you are using this parameter, it should be the **first** one that is supplied.

Another parameter you can supply is **the name of the INF file** to use for the installation. Chief's Installer Pro defaults to **WINSTALL.INF**. You can however specify another file name for this purpose. If this is used, it should be the first parameter (if the **/\$TARGET=** parameter is not used) or the second parameter (if **/\$TARGET=** is used). This parameter will be taken as **paramstr(1)** or **argv[1]**, because **INSTALL.EXE** does not include the **/\$TARGET=** switch in the count of command line parameters.

The final parameter which you can supply is the **source directory**. You cannot use this parameter without using the one which specifies the name of the INF file. If this parameter is used, it should be the last one. If you specify an INF file, it is advisable to also use this parameter to specify the source directory for the installation. This parameter will be taken as **paramstr(2)** or **argv[2]**.

Note that when you choose to run Chief's Installer Pro in this way, the program will faithfully do whatever you say, and will not necessarily verify any of these parameters. This option is provided for added flexibility - but if you use it, you are on your own, and it is up to you to make sure that your program does all the necessary authentication of the parameters you are passing.

In my opinion, it is far better to run Chief's Installer Pro in the normal way, but to allow the installer to click on the "START INSTALL" button automatically, by using the **\$AUTO-CLICK-BUTTON** reserved word (with a parameter of **1**). That way, all the normal internal checks would have been carried out, and you would be able to use SETUP.EXE.

The Syntax is;

INSTALL.EXE [/\$target=<target dir>] [<INF filename> <source dir>]

EXAMPLES:

INSTALL.EXE /\$target=C:\CHIEFPRO

INSTALL.EXE /\$target=C:\CHIEFPRO C:\TEMP\CHIEF.INF

INSTALL.EXE /\$target=C:\CHIEFPRO C:\TEMP\CHIEF.INF A:

INSTALL.EXE A:\CHIEF2.INF

INSTALL.EXE B:\CHIEF2.INF B:

See also;

[\\$AUTO-CLICK-BUTTON](#)

THE UNINSTALLER

Many Windows programs are easy to install, but most are not so easy to remove, because of INI files and DLLs thrown all over the place. Users who wish to uninstall their programs face a hazardous task in which they may delete the wrong files, or remove the wrong entries in INI files. This may then make Windows unusable, necessitating an expensive re-install of Windows.

What this means is that many Windows users are reluctant to try out new programs on their systems because of the hassles of removing the programs if they don't want them any more. **This may mean that people will never ever get to see your wonderful program :).** Some other people have to spend a lot of money on commercial uninstallers, which attempt to snoop round the system. The fact that they are trying to undo someone else's work means that this is often a hit-or-miss affair - sometimes leading to an expensive program not

getting the job done properly. **Enter the UNINSTALLER!**

Chief's Installer Pro includes an "uninstall" program (**UNINSTAL.EXE**). Which will undo anything that Chief's Installer Pro did. If you used the **\$MAKE-UNINSTALL-LOG** reserved word in your WINSTALL.INF file, Chief's Installer Pro will create a log file called **UNINSTAL.LOG** in the target directory. This file contains details of every change made to the system by Chief's Installer Pro.

The user can subsequently "uninstall" your program by running **UNINSTAL.EXE**. The uninstall program will read the log file **UNINSTAL.LOG** and use its contents to undo everything it did during the installation. This includes deleting the installed files, any directories created by Chief's Installer Pro, any Program Manager groups or icons created by Chief's Installer Pro, any entries made into INI files by Chief's Installer Pro, etc.

Note that, unless the INI files are in your program's home directory, the INI files themselves will **NOT** be deleted. Only the entries made into them by Chief's Installer Pro will be deleted. This might result in "orphan" INI files - i.e., INI files with nothing inside them. I believe that this is preferable to deleting all the INI files themselves. This is because entries could have been made into existing INI files, and deleting such files will be disastrous. Therefore, I leave it to users to delete any orphan INI files manually ("better safe than sorry" is the motto here - and, also, "beware Murphy's law").

Shared files (DLL, VBX, and DRV files) installed into the Windows or Windows SYSTEM directory are a special case. If a copy of such files already existed when the installation was running, no log will be made of them, and the uninstaller will NOT delete them (this is because they were obviously not put there by the installer). In cases where no copy of the file was found at install time, the file will be logged, and the uninstaller will delete it - but after asking the user for confirmation. This is because although the file was put there by the installer, it may (after the user has installed other programs) be needed by other programs. This is very often the case with Visual Basic applications, and applications which use BWCC.DLL and/or CTL3Dxx.DLL.

I believe that the uninstaller is a good marketing point for your program. First, there is the perception that a person who provides an uninstaller with his program must be very confident about the program itself. Also, users have nothing to lose by trying your program, since removing it is simply a matter of clicking on the icon for the uninstaller, and then supplying the home directory of the program to be uninstalled. If the uninstaller does not find the log file in that directory, it aborts with an error message. If the file is found, the user is given ONE opportunity to confirm that he or she does really want to uninstall the program.

The Uninstaller can take optional parameters. The first is the home directory of the program to be uninstalled. This means that you can

pass your program's directory (**\$DEST**) as a parameter to **UNINSTAL.EXE** when you are creating your program's icons with the **\$ICON** command.

e.g: **\$ICON=\$DEST\UNINSTAL.EXE \$DEST;Uninstall my Program!**

The second parameter that the uninstaller can take is the name of the LOG file to use for the uninstall. This parameter is normally optional (the program will default to UNINSTAL.LOG). But note that if a filename was supplied as a parameter to the \$MAKE-UNINSTAL-LOG reserved word, then this parameter becomes **MANDATORY** here (i.e., you must supply that filename as a **second** parameter to UNINSTAL.EXE when you create the icons with the **\$ICON** reserved word). **Please note this point.**

e.g: **\$ICON=\$DEST\UNINSTAL.EXE \$DEST VER2.LOG;Uninstall my Program!**

Support for non-English languages is provided for the uninstaller by the use of string tables. These can be compiled into a DLL which must be called **UNINST.DLL**. If this file is not found at run time (it must be in the same directory as UNINSTAL.EXE) then the default English string table inside UNINSTAL.EXE will be used.

A copy of the English language version of the resource script (**UNINST.RC**) is provided for you to translate to your chosen language. Please note that if you choose to create your own translations and put them in the DLL, you are on your own.

The uninstaller will optionally uninstall the program in such a way that the deleted files cannot be undeleted. To enable this feature, use the parameter **OVERWRITE** in the **\$MAKE-UNINSTALL-LOG** line. Note that when this is used, uninstalled files and directories cannot be undeleted, no matter what is attempted. They are first deleted, then they are overwritten by a 1 byte file, and then that file is deleted. Any such file that can be undeleted (in most cases, no file can be undeleted) will only delete to a file containing 1 byte (of garbage data). Therefore, it is **not** recommended that this feature should be used. It is only there because of a specific request, and if you use it, you are on your own.

See also;

[\\$ICON](#)

[\\$MAKE-UNINSTALL-LOG](#)

REGISTRATION SITES

Below are the registration sites for Chief's Installer Pro. Please fill the REGISTRATION FORM below.

REGISTRATION FORM

YOU CAN SEND THE REGISTRATION FEE TO ANY OF THE FOLLOWING REGISTRATION SITES;

COMPUSERVE

On-line registration is available under the SWREG scheme. If you **GO SWREG**, the Registration ID is **4024**.

CANADA, and the UNITED STATES

Minds Edge Productions Inc.
P. O. Box 211
3456 Dunbar Street
Vancouver, BC V6S 2C2
Canada

Internet: karsai@mep.com
Fidonet: 1:153/709
BBS: 1-604-737-7026 19.2 ZyXEL Login: GUEST
WWW: <http://www.wimsey.com/~karsai/>

Fee: \$29.00 (US funds)
or: \$34.00 (Canadian funds)

Method of payment: Checks, Money Orders

Make cheques/money orders payable to: "Minds Edge Productions Inc.". British Columbia residents should add 7% sales Tax.

UNITED STATES

TODD MERRIMAN
Software Toolz, Inc.
8030 Pooles Mill Dr.
Ball Ground,
GA 30107
U.S.A.

Fax: 404-887-5960
Internet: software@toolz.atl.ga.us

Fee: \$29.00 (US funds)

Method of payment: Checks, Money Orders, Visa, Mastercard, American Express.

EUROPE

HENRIK MOERK
Survival BBS
P.O.Box 1538
DK-2700 Bronshoj
Denmark

FIDO: 2:231/306
Internet: Lene@vax.psl.ku.dk
Hmk@research.novo.dk

Fee: Kr190.00 (Danish funds)

**Method of payment: Cheques, Eurocheques, Money Orders
GIRO: 1-207-4247**

Make cheques/money orders payable to: "HENRIK MOERK".

AUSTRALIA, NEW ZEALAND, ASIA, AND THE FAR EAST

DAVID PERKOVIC
DP Computing
P.O.Box 712
Noarlunga Center
SA 5168
Australia

Internet: perkovic@cleese.apana.org.au
dpc@mep.com
Tel: +61 8 326 4364

Fee: \$42.00 (Australian funds)

Method of payment: Cheques, Money Orders

Make cheques/money orders payable to: "DP Computing".

UNITED KINGDOM, IRELAND, EUROPE

John Barton
57 Baddeley Green Lane

Baddeley Green
Stoke on Trent
Staffs, ST2 7JL
ENGLAND.

NOTE: Please write "Chief's Installer Pro REGISTRATION" clearly on the envelope.

Internet: laa12@keele.ac.uk

chief@mep.com

Compuserve: 100415,3414

Fee: £20.00 (U.K. funds; or equivalent)

plus: £3.00 (only if sending a foreign cheque)

Method of payment: Cheques, Eurocheques, Money Orders

To register Chief's Installer Pro, please PRINT and FILL IN the following Registration FORM.

NOTE: Please write "Chief's Installer Pro REGISTRATION" clearly on the envelope.

Please specify your CURRENT version of Chief's Installer Pro.

TO:

**I wish to REGISTER my copy of "Chief's Installer Pro".
My current version is _____
I am paying the REGISTRATION FEE of _____**

ADD Tax (if applicable) _____
(See info on the registration sites to see if they collect tax)

Total FEE: _____
I am paying by **Cheque/Money Order/Credit Card** (delete as inappropriate)

NAME

ADDRESS

POST/ZIP

CODE

E-MAIL

How did you get your copy of Chief's Installer Pro?

IF PAYING BY CREDIT CARD, PLEASE SEND THE FOLLOWING DETAILS;
(NOTE: Not all sites accept credit cards so please refer to the list of REGISTRATION SITES)

CARD ISSUER

CARD NUMBER

DATE OF ISSUE
EXPIRY DATE
SIGNATURE
DATE

CREDITS

Many thanks to the following:

1. **Claus Ziegler**, ZieglerSoft, Denmark - a great Windows guru! Thanks for everything, and for the Danish translations of the string tables.
2. **Joachim Rehmet** and **Juergen Kneifel** - for the German translations of the string tables.
3. **Borland International** - for producing a wonderful Windows Pascal compiler (the bugs are mine). **Now how about that nice OS/2 Pascal compiler? :)=**
4. **Drs. Bob Swart** - for the Dutch translations of the string tables.
5. **Gary W.Rohn** - for sharing the source code to your FI program.

DISCLAIMER

I do NOT warantee ANYTHING concerning any of the programs or files which make up "Chief's Installer Pro for Windows". I accept NO RESPONSIBILITY for ANY LOSS OR DAMAGE of ANY kind, including, but not limited to, losses of a physical, mental, social, financial, marital, or of whatever nature, resulting from the use, or the purported use of Chief's Installer Pro for Windows", or any of the files in the package, for any purpose whatsoever. I do not even warantee that the programs will not kill you. You use Chief's Installer Pro for Windows ENTIRELY AT YOUR OWN RISK, and you supply it to your customers, friends, family, acquaintances, or enemies, ENTIRELY AT YOUR OWN RISK.

If these terms are NOT acceptable to you, then you have no licence to use or test Chief's Installer Pro, and you should DELETE all the program's files from all your disks immediately AND PERMANENTLY.

FEEDBACK

Okay. I am keen to obtain feedback, especially from registered users. I also welcome suggestions for features. I cannot promise to implement every suggestion, but at least, I will consider the ideas. If you have any comments, ideas, suggestions, etc., or you just want to tell me how wonderful the program is :) then please feel free to contact me by e-mail. I will try to respond if a response is appropriate.

You can contact me by e-mail at the following internet addresses:

laa12@potter.cc.keele.ac.uk

chief@mep.com

UPDATES

This program is being constantly updated. I will endeavour to release bug fixes as often as I receive bug reports and fix them. However, it is rather difficult to spread the word about new releases and updates. There are a number of internet ftp sites which are not usually busy and to which I can therefore upload new versions. You might want to check these places from time to time.

FTP SITES (and directories)

ftp.demon.co.uk /pub/ibmpc/windows/chief/pro

micros.hensa.ac.uk /micros/ibmpc/win/e/e022

COMPUSERVE

WINSHARE LIB 4

MSBASIC LIB 2