Freeman Installer v2.2 for Windows 95 and NT

Freeman Installer v2.2 is a software package that gives you the ability to automatically install your Windows applications onto your users' system.

If you don't know how to use help, press F1 now.

For introductory information on how to use Freeman Installer, you should follow the four tutorials included in this package.

Other Useful Documentation

It is STRONGLY recommended that you follow the four tutorials and go through Q&A and the gotcha list:

• Tutorial 1	Basic skills.
• Tutorial 2	DLL install, INI entries, registry, selective install.
• Tutorial 3	Prompting for Yes/No, message box, conditional event, executing EXE, 256 color bitmap.
• Tutorial 4	Large file install, TT font, C++ code customization.
• Q&A	Common tasks.
• Gotcha's	Common traps and resolutions.

Non-Technical Information

Copyright & License

Disclaimer

Purchasing Freeman Installer!!!

What Files Can be Modified/Customized?

Acknowledgment

Basic concepts

Interpreted Installer vs Compiled Installer

Project Directory Structure

Auxiliary Files

Variables

File Version Checking

Command explanations

Edit | Events

Edit | General Information

Edit | Disk Descriptions

Edit | Files

Edit | Program Items

Edit | INI Entries

Edit | Registration Keys

Edit | Software Components

Edit | Background | Text

Edit | Background | Gadgets

Copyright & License

Freeman Installer

version 2.2

Disclaimer

Variables

Freeman Installer supports two kinds of variables. string variables and integer variables.

A variable is defined/created by assigning a value to it. You do this in the various events.

String variables can be used in almost all textual strings you enter in the constructor. That is, almost all string expressions are evaluated before they are used.   Some examples are file source directory, and prompts in the welcome dialog.

Integer variables can be used in all the places where an integer/boolean expression is expected. Some examples are event condition, and the minimum disk space required.

A variable name follows the C++ variable syntax. In fact, if you are going to work in compile mode, the variables are true C++ variables of type char [256] and int.

A string variable myvar can be referenced by $myvar or ${myvar}. To say "$", enter "$$".

An Integer variable is referenced the same way as you would in C/C++ program.

Since the $ can distinguish a string variable from an integer one, for simplicity, sometimes we simply say $myvar is the variable.

When you create a new install project, the constructor creates some events for you in which the following variables are defined/assigned:

$i	-----	Install target directory entered by the user
$s	-----	Install source directory, i.e., where install.exe was run from. Usually "a:"
$w	-----	Windows directory. Usually "c:\windows"
$y	-----	Windows system directory. Usually "c:\windows\system"
minspace	-----	The total disk space required by the current selection

File Version Checking

Version statement is a kind of resource whose purpose is to attach version info (version number, among some other things) to a file.

Version checking means when trying to copy a file onto the user's system, if there is already an existing file with the same name in the intended target directory, the version statement of two files are retrieved and compared against each other to determine which one is newer.

You can attach a version statement to executive files only (.EXE, .DLL, .DRV and so on). For your own data files, you can use INI version. Refer to the Q&A for more info.

What Files Can be Modified/Customized?

All the files copied, by the constructor, to the user directory can be modified. In addition, various versions of fimain.rtf found in the language specific sub-directories, such as c:\finstall\eng (English), c:\finstall\frn (French), c:\finstall\ger (German), can be modified too.

Files of special interests are:

fimain.rc	-----	Ad dialogs, bitmaps
fimain.rtf	-----	Online help of the installer
finstall.ico	-----	Icon of the installer

Acknowledgment

Compression Code

The compression code used in the constructor was derived from Info-Zip. There is no extra or hidden cost for the use of this code in my program, i.e., I only charge you for my own work, not for that of Info-Zip. You can get the original source code from ftp.uu.net /pub/archiving/zip/*.

Translation

The following people have kindly translated the English version into different languages:

Danish	-----	Niels Ull Jacobsen <null@diku.dk>
Dutch	-----	Andre van den Bos <bos@cs.utwente.nl>
French	-----	Jean-Francois Messier <messier@igs.net>
German	-----	Henning Stams <hstams@k.mup.de>
Italian	-----	Davide Rossi <rossi@cs.unibo.it>
Japanese	-----	Chris Undery <chrisu@eis.net.au>
Norwegian	-----	Eivind Bakkestuen <hillbilly@programmers.bbs.no>
Spanish	-----	Javier Cárdenas <74052,1565>
Sweden	-----	Ola Strandberg <olas@bahnhof.se>

Requirements

Mr. Jean-Francois Messier has been making many valuable suggestions pertaining to the requirements on the final product. Without these suggestions, Freeman Installer wouldn't be as easy to use, flexible and powerful as it is.

Documentation

Mr. Todd C. Wilson <tc@galadriel.ecaetc.ohio-state.edu> has kindly proof-read and enhanced the readability of the tutorials, Q&A, and the online help file. It means that if you find some syntax or grammatical errors in the documentation, they are all mine :-)

Edit | Events

Overview

An event is an operation to be performed during the install. Some examples are copying files, executing an external program, converting an integer expression to a string expression, searching for a certain file.

Every event is associated with an integer expression condition. If that expression is evaluated to true (non-zero) at run time, that event will be executed. Otherwise, it will be skipped. Making use of this condition field we can act differently depending on, say, whether a file is found on the user's system or whether the user answers yes to our question..

For most events that return a status or exit value, the value is true (i.e., non-zero) if that event was executed successfully. Otherwise the value is false (i.e., 0). For example, for the "search for file" event, the return value is true when the file was found. Another example is the "get INI value" event that will return true when the entry is really there. However, there is one exception, i.e., the "compare two strings" event. This event will return 0 if the two strings are equal, a negative value if string 1 is less than string 2; a positive value if string 1 is greater than string 2.

Here is a list of all available event types:

Display welcome dialog

Ask user info

Display component selection dialog to let the user select which parts to install

Perform file operations (install, rename, delete, backup)

Create Program Manager items

Set INI entries

Set registration keys

Execute external program (wait until it terminates or otherwise)

Display install completion dialog

Execute a piece of C/C++ code

Modify autoexec.bat

Modify config.sys

Display a message box

Prompt for a string

Get install source directory

Get Windows directory

Get Windows system directory

Get temporary directory

Get CD ROM root directory

Get disk space needed by the current selection

Evaluate a string expression

Evaluate an integer expression

Compare two string expressions

Search for a file

Get the value of an INI entry

Get the value of a registration key

Convert an integer expression to a string
Convert a string expression to an integer
Check if a certain leaf component is selected
Split a full path into directory and file name
Prompt the user to choose Yes/No
Get Windows major and minor versions
Get CPU model
Check the presence of math processor
Check if Windows is running in 386 enhanced mode
Abort installation immediately

Edit | General Information

Application name

The name used to refer to your application. It is used in a couple of dialog boxes.

Where to put install log

The directory where the install log (filog.ini) will be created.

Minimum time to copy a file

Suppose that it is set to 1000 ms. If it only takes 600 ms to copy the file readme.txt, the installer will wait 400 ms before copying the next file. It is used to slow down the copying process to let the user know what is going on. It is particularly useful for small applications.

Putting 0 here will instruct the installer not to wait at all.

Divide files into blocks of this big

Suppose that it is set to 1024 bytes. If a file manual.hlp is 2050 bytes, the installer will copy it at three times, one block at a time. The block sizes are 1024, 1024, 2 bytes respectively.

Putting 0 here is equivalent to putting 2048.

Here blocks are used as a kind of time unit to control how often to check the abort button and how often to update the progress bar.

Check abort button when ?? blocks have been copied

Suppose that it is set to 2. The installer will check if the abort button has been pressed when 2 blocks have been copied. The smaller this value is, the quicker the response to the abort button is.

Setting it to 1 is usually the best. Checking the abort button virtually doesn't slow down anything.

Update progress bar when ?? blocks have been copied

Suppose that it is set to 2. The installer will update the progress bar when 2 blocks have been copied. The smaller this value is, the more frequently the progress bar is updated.

Use 3D controls

Let the installer use 3D control if ctl3dv2.dll already exists on the user's system. Otherwise, there is no effect.

Kill Program Manager if we started it to DDE with

If the shell does not support the shell DDE commands, the installer will load Program Manager to create the program items. This controls whether the installer should kill Program Manager or not when finishes creating the program items.

If you don't kill it, the user can have a chance to pick up those items he/she thinks useful and migrate them to his/her shell. But this might also give the user an impression that the install program doesn't clean up its own mess.

Use INI version. This version is ???

We use version statements for EXEs and DLLs. In contrast, we use INI version for your data files. There is no special requirement to use it, i.e., any file can be protected by INI version checking so that older versions will never overwrite newer versions.

To use INI version for your files, check this flag and enter the version of this package like 1.5.1.0 (reads v1.51). Then you can use INI version as the condition in the file dialog.

Language

Want a Dutch installer? Just choose Dutch here and everything is done!

When you are asked if you want to delete all user files, before answering yes, please backup all of your customizations (dialogs, bitmaps, online help, C/C++ user code).

Compiler

If you don't plan to work in compile mode, don't touch it. Otherwise, select your favorite compiler here.

Reboot

Controls whether the installer should ask the user to reboot.

Don't reboot	-----	Don't reboot. Stay in Windows after completion
Reboot Windows	-----	Ask the user's permission to reboot Windows
Reboot DOS	-----	Ask the user's permission to restart the machine
Default behavior	-----	Reboot Windows if system.ini was modified. Reboot DOS if autoexec.bat or config.sys was modified.

Edit | Disk Descriptions

Overview

A disk description is just a string description about a disk. For example, the first   disk may have a disk description "Installation Disk 1" or "Main Program Disk". If a file is on the first disk, that file is associated with the disk description for the first disk. When the installer can not find a particular file, it will use the corresponding disk description as a (part of the) prompt to ask the user to insert the disk.

Usually you DON'T need to enter the disk descriptions yourself. The constructor will create the disk descriptions and associate the files to them properly when it is building the disk set. Since it has no knowledge on what your package is, it will simply sequential create "Installation Disk 1", "Installation Disk 2", ... and so on.

The purpose of this dialog is to let you edit them to fit your application best.

Edit | Files

Overview

In here basically you specify the set of files to be distributed and installed. Although we can also delete, rename, or backup files on the user's system.

The entries here will be executed sequentially.

The fastest way to specify the files is clicking on the "Pick Files" button to pick up the files in File Manager and then drop them into the list box. If you want to set the parameters for multiple entries at once, you can make use of the multiple selection capacity. Drag'n'drop is also supported within the file list which makes it very easy to rearrange the order.

Show full path

Show the full path rather than file name of the files in the list box.

Op type

Install	-----	Install a file onto the user's system
Rename	-----	Rename a file on the user's system
Delete	-----	Delete a file on the user's system
Backup	-----	Backup a file on the user's system

Condition

Execute this entry only when the condition is evaluated to true.

Edit | Program Items

Overview

In here you specify the set of program items to be created.

The items will be created in the order they are listed here.

Item

The name of the item.

Group

The name of the program group the item will belong to. If the group does not exist, it will be created automatically.

Command line

The command line of the item. Here are some examples,

`$i\csh.exe`	run csh.exe in the install target directory
`$i\csh.exe -e $i\script`	ditto but passing two parameters
`$w\winhelp.exe $i\manual.hlp`	run help on the manual help file
`$i\manual.hlp`	ditto

Icon path

The path pointing to the icon file for the item. Here are some examples,

`$i\csh.exe`	use an icon in csh.exe in the install target directory
`$w\winfile.exe`	use File Manager's icon

If you set it to empty, Program Manager will get the icon from the command line.

Icon index

0 means using the first icon in the icon file. 1 means the second icon and so on.

Working dir

The working directory (or startup directory) of the item.

Edit | INI Entries

Overview

In here you specify the set of INI entries to be created, set, or added.

File

The INI file (path). If it doesn't exist, it will be created (including all directories involved) automatically.
Here are some examples,

`$i\csh.ini`	File csh.ini in install target directory
`$w\win.ini`	win.ini (in Windows directory, of course)
`$w\system.ini`	system.ini (in Windows directory, of course)

Section

The section of the entry. If it doesn't exist, it will be created automatically.

Entry

The name (key) of the entry.

Value

The value of the entry.

What to do if same key exists

Set the old entry	-----	Replace the old value with the new value
Add an entry if not the same value	-----	If old value = new value, then do nothing; otherwise add a new entry. This is useful when adding device drivers where you want to add a device= line unless you already have one.
Add an entry no matter how	-----	Always add a new entry.

Edit | Registration Keys

Overview

In here you specify the set of registration keys to be created or set. For more info on how to use registration keys, refer to tutorial 2.

Key path

The path of the registration key.

Field

The name of the value. This can be empty if you want to set the key itself.

Value type

REG_SZ

REG_EXPAND_SZ

REG_DWORD

REG_BINARY

REG_MULTI_SZ

Edit | Software Components

Overview

A component represents a part of your package. If a component is selected by the end user, that part of the package will be installed.

There are two kinds of components: leaf component and branch component. A leaf component contains some files, program items, INI entries, and registration keys. Selecting a leaf component means selecting those files, program items, ... included in it. A branch component contains some other components (leaf or branch). Selecting a branch component means selecting those components included in it. Any sub-component (including files, items, ...) can be included in more than 1 components, i.e., it can be shared. Every component has a name that will be displayed in the tree control to represent the component. Every component has a detailed description (<= 255 chars) that will be displayed in the "description" window on the right side when that component is highlighted. A component can be made exclusive to its sibling. In this case when it is selected, it will try to deselect all its siblings. This is useful when implementing the well known full-typical-min-custom install option because one can not select full install and typical install at the same time. For more info on how to do this, refer to tutorial 2.

A component is selected when the user clicks on the check box next to it. A component is highlighted when the user selects the component or clicks on the text part of the component. If a component is selected, it will be installed subsequently. If a component is highlighted, its long description will shown in the "description" window but it may or may not be installed. There is a progress bar in the component selection dialog box indicating the maximum disk space needed and the disk space needed by the current selection. When the user select/deselect a component, the progress bar will be updated accordingly.

If you have some components in your project and a file is not included in any of them, it won't be installed no matter how the user selects the components. This will be reported as a warning by the constructor.

To re-arrange the order of the components, just drag'n'drop.

Name

The name of the component.

Description

The detailed description of the component. Must be less than 255 chars long.

Is leaf component

It is leaf or a branch?

Is exclusive to siblings

When it is selected, will it deselect all its siblings?

Is expanded at startup

Should it show its sub-components initially?

Allow user to select

If the expression is evaluated to true (non-zero) at run time, then the component is enabled and can be selected or deselected. Otherwise, the component is disabled and grayed out. Disabling a component is useful when you need to force the user to install something or prohibit the user from installing something (such as in the case of crippled software).

Is pre-selected at startup

If the expression is evaluated to true (non-zero) at run time, then the component is selected initially.

Otherwise, it is not selected initially.

Sub-components

Here you select the sub-components for a component.

All potential sub-components are listed here. The actual sub-components are selected. If the currently selected component is a leaf component, the potential sub-components are all files, program items, INI entries, registration keys. If the currently selected component is a branch component, the potential sub-components are all components. Although all components are listed there, you MUST NOT select its ancestor or itself as its sub-component.

Show full path

In the sub-component list box, use the full names of the files, items and so on.

Edit | Background | Text

Overview

Here you can put some text objects on the gradient blue background. You can control their position, size, color, font, and special effects.

The position and size are all measured in a unit called "screen coordinate". The left top corner of the screen is (0, 0) while the right bottom corner of the screen is (1000, 1000).

```
      (0, 0)+-----------------+ (1000, 0)
            |                 |
            |    (500,500)    |
            |        +        |
            |                 |
            |                 |
   (0, 1000)+-----------------+ (1000, 1000)
```

For example, if you center your text object at (500, 500), it will appear at the center on your screen and on your end users' screens no matter if you have yours set to 1280 by 768 and they only have 640 by 480.

You can see it in action immediately by clicking on the "test" button.

Text

The text string to be drawn.

Font

Draw the text string in which font?

Bold

Use bold style?

Italic

Use italic style?

Color

Click on this button to choose the color for the text string.

Effect

Use which special effect.

None	-----	No special effect. Just draw the text itself
3D	-----	3D effect
Shadow	-----	Shadow effect

x, y

The (x, y) screen coordinates of the left top corner of the text cell.

w, h

The width and height (in screen coordinates) of the text cell.

Natural width

Use the current height to calculate the width so that the font looks the best/most natural.

Test

Draw the background (including the text objects and the gadgets) immediately to let you see the outcome. Clicking the mouse will end the test.

Edit | Background | Gadgets

Overview

Here you can put some gadgets on the gradient blue background.

There are four gadgets available: exit button, help button, minimize button, and CPU-RAM gauge.The position of the gadgets is measured in a unit called "screen coordinate". The left top corner of the screen is (0, 0) while the right bottom corner of the screen is (1000, 1000).

```
      (0, 0)+-----------------+ (1000, 0)
            |                 |
            |    (500,500)    |
            |        +        |
            |                 |
            |                 |
   (0, 1000)+-----------------+ (1000, 1000)
```

For example, if you put a button at (500, 500), it will appear at the center on your screen and on your end users' screens no matter if you have yours set to 1280 by 768 and they only have 640 by 480.

You can see it in action immediately by clicking on the "test" button.

Center x, Center y

The (x, y) screen coordinates of the center of the gadget.

Enabled

Show the gadget? If you don't want the gadget, you can turn it off here.

Test

Draw the background (including the text objects and the gadgets) immediately to let you see the outcome. Clicking the mouse will end the test.

Edit | Advertising Dialogs

Overview

An "advertising dialog" is just a usual dialog that will be displayed on the background while the installer is copying the files.

You can use all the standard controls in an ad dialog. In addition, you can put a bitmap of up to 256 color there as an ordinary control. For more info, refer to tutorial 3 and the Q&A.

After creating the dialogs in c:\prj\usr\fimain.rc, you come to here to tell the constructor the names (ASCII resource ID's) of those dialogs. The dialogs will be displayed one by one sequentially.

Dialog resource name

The name, or rather, the ASCII resource ID of the ad dialogs. You must use ASCII ID in the resource file to identify the dialog box resource, i.e., integer IDs are not supported.

Paying via CompuServe

You can GO SWREG and pick up the appropriate product depending on the license type:

- product 7610 for an author license (US$57.5). This includes US$7.5 charged by CIS (not us!).
- product 7612 for an organization license (US$120).
- product 7613 for an author license upgrade from the 16bit version of Freeman Installer (US$28.75). This includes US$3.75 charged by CIS (not us!).

For other types of license, you need to pay with a credit card or a cheque.

Interpret | Run

Make the interpreted version of the package on the emulated floppy disk (c:\prj\int\run) and test run it from there.

Interpret | Build Disk Set

Create the interpreted version of the package on floppy disks.

Interpret | Make

Make the interpreted version of the package on the emulated floppy disk (c:\prj\int\run). Only out-dated files will be prepared again.

Interpret | Build

Build the interpreted version of the package on the emulated floppy disk (c:\prj\int\run). All files will be prepared again no matter they are out-dated or not.

Interpret | Purify

Delete all files on the emulated floppy disk for the interpreted package (c:\prj\int\run). Usually this command is followed by a make, build, or test run so that the emulated floppy disk is rebuilt. The net result is that any file that was once included but not in the final package will not stay in c:\prj\int\run.

Interpret | Zip

Create a zip file containing the whole interpreted package. The zip file will be put in c:\prj\int.

Interpret | Auxiliary Files

Overview

In here you specify the set of auxiliary files for the interpreted package.

Local path

The path of the file on your own machine.

Prepare

How the constructor prepare the file.

Compress	-----	Prepares the file by compressing it
Copy	-----	Prepares the file by copying it
None	-----	Don't prepare it at all

Filename

The file name of the file on the distribution disk.

Compile | Run

Make the compiled version of the package on the emulated floppy disk (c:\prj\com\run) and test run it from there.

Compile | Build Disk Set

Create the compiled   version of the package on floppy disks.

Compile | Make

Make the compiled version of the package on the emulated floppy disk (c:\prj\com\run). Only out-dated files will be prepared again.

Compile | Build

Build the compiled version of the package on the emulated floppy disk (c:\prj\com\run). All files will be prepared again no matter they are out-dated or not.

Compile | Purify

Delete all files on the emulated floppy disk for the compiled package (c:\prj\com\run). Usually this command is followed by a make, build, or test run so that the emulated floppy disk is rebuilt. The net result is that any file that was once included but not in the final package will not stay in c:\prj\com\run.

Compile | Zip

Create a zip file containing the whole interpreted package. The zip file will be put in c:\prj\com.

Compile | To Compiler

Launch the compiler and load the project for you. Since some compilers like BC3.1 will refuse to load if they are not the first instance, you may want to close the compiler first before choosing this command.

Compile | Auxiliary Files

Overview

In here you specify the set of auxiliary files for the compiled package.

Local path

The path of the file on your own machine.

Prepare

How the constructor prepare the file.

Compress	-----	Prepares the file by compressing it
Copy	-----	Prepares the file by copying it
None	-----	Don't prepare it at all

Filename

The file name of the file on the distribution disk.

Purchasing Freeman Installer

When to purchase?

The copy you are keeping is a demo (fully functioning) only. If, fortunately, you like it, you should purchase the REAL THING.

How much is the license fee?

Depends what you are going to do with it:

Author license	-----	US$50. You can use Freeman Installer with that all the applications written by you (alone or in co-operation with your colleagues). There is no limitation on quantity, distribution scope or time period. This kind of license is suitable for small software development companies and individual developers.
Organization license	-----	US$120. You can use Freeman Installer with all applications your organization produces. There is no limitation on quantity, distribution scope or time period. This kind of license is stuiable for medium to large software development companies.
Admin license	-----	US$1.5x(3**log(# of users)). You can use Freeman Installer with as many applications as you like, no matter who made them, as long as the distribution is limited to a certain group of users. This license of kind is suitable for Internet Service Providers (ISPs) and computer system administrators. If you'd like, you can get an admin license for unlimited number of users for US$200.
Application license	-----	The price of your application. This kind of license is suitable for freeware or applications cheaper than US$50.

What are included in the real product?

1. License to let you distribute Freeman Installer with your application(s).
2. Installer and class libraries without the "demo" notice on the background.
3. Life time technical support (via email or mail only).
4. Source code to scramble.h and scramble.cpp (on your request) so that your application can extract and decrypt the user name and company name entered during the installation.

How to purchase?

1. Double click on the order form icon and fill it out.
2. Email it to us. If you don't have email access, fax instead.
3. Pay the license fee via PsL,, CompuServe, or with a cheque.

What can I do if I regret purchasing?

You can get a 100% refund (excluding shipping & handling) if you destroy the software you received before distributing it.

How to contact us?

All queries, bug reports, suggestions, and payments are welcomed and should be directed to:

Freeman-Teresa Software
GPO Box 712
Broadway, NSW 2007
Australia

Internet email: freemant@wr.com.au
CompuServe email: 100351,3364
Fax: 61-2-2116314
NEVER fax during the night SYNEY LOCAL time!!!

Note: Email is the very much prefered way to contact us. Fax only when absolutely necessary.

File Operation: Install

Overview

You specify the details about the file to be installed.

In most of the cases you only need to take care of the "description" field. Others fields can be left to the constructor.

For private DLLs, you need to set "source dir" field to $i, set "uninstall" field to "uninstall".

These three fields are enough for most purposes.

Local Path

The path of the selected file on YOUR OWN computer.

Source directory

The source directory of the file. If source directory is a: and the file name is readme.txt, the installer will read from the file a:\readme.txt.

If the file resides in the same directory as the installer, you should put $s here. If the file resides in a sub directory of that called doc, you should put $s\doc here.

File

Source filename.

Target directory

The target directory of the file. If target directory is c:\csh\doc and the file name is readme.txt, the installer will write to the file c:\csh\doc\readme.txt. You don't need to bother about whether c:\csh and c:\csh\doc exist or not. If necessary, the installer will create them automatically.

If the file should reside in the install target directory, you should put $i here. If the file should reside in a sub directory of that called doc, you should put $i\doc here. If the file is a shared file like ctl3dv2.dll, you should put $y here.

File

Target filename.

Description

The description for the file. It will be displayed in the dialog box while the file is being copied.

File size

The size (in bytes) of the file. If the file will be compressed, this is the original size.

Font file

If the file is marked as a font file, the installer will create a font (register with Windows) out of it.

Last file

If the file is marked as a last file, the constructor will proceed to prepare the next disk even though the current disk is still not full.

Disk

The disk description for the file. See Edit | Disk Descriptions.

Condition

None	-----	Install the file, no questions asked
Version info	-----	Compare the version info (statements). Install the later version only
INI version	-----	Compare the INI version. Install the later version only
File time stamp	-----	Compare the time stamps. Install the newer version only
File size	-----	Compare the file size. Install the bigger version only

Prepare

How the constructor prepares the file for us.

Compress	-----	Prepare the file by compressing it
Copy	-----	Prepare the file by copying it
None	-----	Don't prepare the file for us

Create

How the installer create the file on the user's system.

Create	-----	Create the file. If one already exists, overwrite it
Append	-----	Append to the file. If it doesn't exist, create one.

Transfer

How the installer transfers the bytes from the source file to the target file.

Decompress	-----	Decompress the file
Copy	-----	Copy the file

Uninstall

How the uninstaller uninstalls the file.

Uninstall	-----	Uninstall the file
None	-----	Don't uninstall the file. Let it survive

Pick Files

Launch File Manager to let you select the files.

File Operation: Rename

Overview

You specify the details on the file to be renamed. It can be used to move the file across different directories or drives.

Source Dir + File

The old filename.

Target Dir + File

The new filename.

Description

Describes what the file is.

Fatal error if source file doesn't exist

Control whether the installer should abort if the source file doesn't exist.

File Operation: Delete

Overview

You specify the details on the file to be deleted. It's OK if the file doesn't exist.

Dir + File

The file to be deleted.

Description

Describes what the file is.

File Operation: Backup

Overview

You specify the details on the file to be backed up. It's OK if the file doesn't exist.

Dir + File

The file to be backed up.

Description

Describes what the file is.

Advise user

How the installer advises the user of the backup.

Do it quietly	-----	Don't advise the user
Advise user before backup	-----	Advise the user before backup
Advise user after backup	-----	Advise the user after backup

Value Type: REG_SZ

The value is an asciiz string.

String expression

Enter the intended value here. This expression will be evaluated first and then be used to set the registration key.

Value Type: REG_EXPAND_SZ

The value is an expand string (something like %PATH%).

String expression

Enter the intended value here. This expression will be evaluated first and then be used to set the registration key.

Value Type: REG_DWORD

The value is a DWORD.

Integer expression

Enter the intended value here. This expression will be evaluated first and then be used to set the registration key. However, you can't use hexadecimal value here. For example, 0x0001 is an invalid expression but 1 is OK. You can use integer variables and integer operations here (like myvar + 20).

Value Type: REG_BINARY

The value is a block of binary bytes.

Path to binary file

Enter the path to your binary file here. During the install the installer will retrieve the contents of this file and use it to set the registration key. If that binary file is of no use after the installation, it is a good idea to create an auxiliary file entry for it and to refer to by something like $t\mybin.dat.

Value Type: REG_MULTI_SZ

The value is a sequence of asciiz strings with the last one terminated by an extra null. However you don't don't the null characters directly. You simply put your strings in the multi-line edit control. The installer will convert it to a real REG_MULTI_SZ during the installation.

Edit

Bring up the multi-line editor to let you input or modify the REG_MULTI_SZ.

Project Options

These are the initial options for the project. All of them can be changed later. However, the initial event list, initial file list and initial auxiliary file list are determined by the options you select here. If you change any of the options later, you may have to manually adapt the event list, file list and auxiliary file list yourself. Therefore, you should consider carefully before clicking on "OK".

If you plan to save user info into the installer itself, set registration keys, you are better off selecting them now.

Call it setup.exe rather than install.exe

So that your users run the installer by typing "setup" rather than "install".

Save user info with installer

Saving the user information with installer will associate the user installing the package with the disk set for good unless it is cracked later.

Save user info with application

Save the user information into $i\filog.ini. The user information is not associated with the disk set.

Use 3D controls

Make use of ctl3d32.dll if it can be found on the user's system.

Support un-install

Make your application uninstallable at the cost of 78K floppy disk space.

Set registration keys

Does you package need to set registration keys?

Modify config.sys

Does you package need to modify config.sys?

Modify autoexec.bat

Does you package need to modify autoexec.bat?

Enable exit, help, minimize buttons and CPU-RAM gauge

Do you want the various buttons and gauge in the background?

Language

What an Italian installer? Choose "Italian" here!

Compiler

If you don't plan to work in compile mode, don't touch it. Otherwise, select your favorite compiler here.

Auxiliary Files

An auxiliary file is a file included in the package that won't be installed onto the user's system and the only purpose of it is to install other files. One example is fimain.exe (the installer). Another example is fimain.hlp (online help file). The interpreted installer uses a particular set of auxiliary files while the compiled installer uses another set. If you want to specify the auxiliary files (usually to add your own), you can use the command Interpret | Auxiliary Files. and Compile | Auxiliary Files respectively.

All auxiliary files will be copied/compressed onto the first distribution disk. When the user installs the package by typing install or setup, install.exe or setup.exe will copy/decompress the auxiliary files into a temporary directory on the user's system, and then launch the first auxiliary file as the main installer. This is why fimain.exe is almost always the first auxiliary file.

Warning: You must not put another file other than fimain.exe as the first auxiliary file unless you have a compelling reason to do so (e.g., you may have developed a custom main installer that will in turn launch fimain.exe sometime later.)

You can add (usually append) your own auxiliary files there. This is useful when you need access to some files during the installation but not after the installation. One example is when you need to unzip a zip file during the installation. Another example is that you need to set a binary value to a registry key. To access your own auxiliary files you need the value of the temporary directory. This can be achieved with the "get temp directory" event.

Interpreted Installer vs Compiled Installer

When you use the commands on the Interpret menu, you are using the interpreted installer. When you use the commands on the Compiled menu, you are using the compiled installer. The interpreted installer (c:\prj\usr\fimaini.exe) is a EXE pre-built by us. It contains every functionality supported by this package, no matter your installation actually makes use of it or not. It is supposed to function with your install.inf (or setup.inf). During the installation, it will read your .inf and execute the events as specified there The compiled installer (c:\prj\usr\fimainc.exe) is supposed to be generated by your C++ compiler, from the source code (c:\prj\usr\fiuser.cpp) spitted out by the constructor that is tailored to your installation, plus the class lib shipped with Freeman Installer. Since fiuser.cpp already contains all the information about your installation, there is no need to ship your .inf with the compiled installer, i.e., the compiled installer can perform the operations with further reference to your .inf.

The advantages of the interpreted installer are:

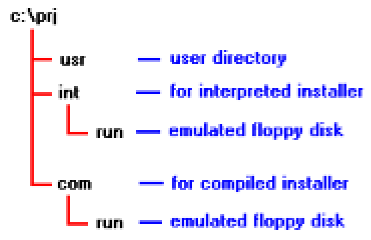
- You don't need a C++ compiler.
- You don't need to do a compilation every time you test run your installation and therefore you can have faster feedbacks.

The advantages of the compiled installer are:

- You can insert C/C++ code in the constructor for customizations.
- You will have a smaller installer. The difference in size is in the magnitude of 50K.

Project Directory Structure

The directory structure under a particular project is like:


c:\prj
usr — user directory
int — for interpreted installer
run — emulated floppy disk
com — for compiled installer
run — emulated floppy disk

User Directory

The user directory, c:\prj\usr, holds files that are initially put there by the constructor and then handed over to you and subject to your modifications (customizations). Some examples are:

• fimaini.exe	------	Interpreted installer.
• fimainc.exe	------	Compiled installer.
• fimain.rc	------	Resource file. You put your ad dialogs and bitmaps there.
• finstall.ico	------	The sunset icon. You can use your own icon to replace its contents.
• fiuser.cpp	------	C++ Source code spitted out by the constructor.
• fiuser.h	------	Header file where you can put declarations of your own C/C++ functions.
• fimain.lib	------	Class lib to be linked with fiuser.cpp to generate the compiled installer.
• fimain.cpp	------	Online help file for the installer.

Emulated floppy disk (Run directory) for the interpreted installer

When you test run or make the interpreted installer, the constructor will compress or copy the files into c:\prj\int\run as if it were a:\ and then launch c:\prj\int\run\install.exe. This is why this directory is called the emulated floppy disk. It is the hard disk image of a:\. Therefore, if the project directory is on a network server, your user will be able to start the installation by running install.exe in the run directory.

Emulated floppy disk (Run directory) for the compiled installer

c:\prj\com\run is the emulated floppy disk for the compiled installer. It plays a role similar to c:\prj\int\run.

Interpret | Bind Resources

Bind c:\prj\usr\fimain.rc (resources) to c:\prj\usr\fimaini.exe (interpreted installer). Since your ad dialogs, bitmaps and non-English text strings are all stored in fimain.rc, they won't take effect until resource binding occurs. Usually you don't need to issue this command yourself since resource binding is carried out automatically during make, build, or test run. However, if for some reasons the changes you made to fimain.rc don't take effect, you may try using this command.

Paying via PsL

You can order with MasterCard, Visa, American express or Discover credit card from Public Software Library (PsL) by contacting:

Toll-free:	1-800-2424-PSL
Phone:	(713)524-6394
Fax:	(713)524-6398
CompuServe:	71355,470
Internet:	71355.470@compuserve.com

Provide PsL with the following info:

Product number:	11888
Product name:	Freeman Installer
Platform:	Win31 / Win95-NT
License type:	Author / Organization / Admin / Application
Amount payable:	License fee + US$5 PsL fee
Name:	eg, Bill Gates
Email address:	eg, billg@microsoft.com
Fax number:	
Postal adress:	

Note:

- The total amount payable includes a $5 order taking fee charged by PsL (not us!).
- If you're entitled to a discount, simply calculate the reduced license fee, add US$5 and tell PsL the final total amount. You don't need to tell them anything about the discount.
- The people at PsL do NOT know anything about our product except pricing. All queries should be sent to us.

Paying by Cheque

The following instructions must be followed strictly. Any deviation will result in lengthy delay and difficulty in processing your order.

1. Make a cheque for the amount of the license fee. All major currencies are accepted with Australian dollars very much prefered. If it's in non-Australian fund, please add US$10 that will be charged by the bank. In short:

Non-Australian fund:	exchange rate x US$??? + US$10
Australian fund:	exchange rate x US$???

2. Make the cheque payable and addressed to:

   Freeman-Teresa Software
   GPO Box 712
   Broadway, NSW 2007
   Australia

3. Include your order form.