

Getting Started with VisualSetup 4.0

Every journey begins with a first step. In this document, we will walk you through the creation of a new project and introduce you to the VisualSetup program interface.

1.1 Starting a New Project

Everything has to start somewhere. In VisualSetup, the design process starts with the creation of a new project.

A project is simply the collection of files, shortcuts, registry's entries and settings and everything else that goes into building an installer. A typical project will contain all of the files that you want to distribute, some dialogs that inform or gather information from the user, and maybe a few actions to take care of any "extras" (such as storing the installation path in a registry key for future use by your patching tools).

The project file

Each file that you add to a VisualSetup project has individual settings that control where, when and how the file will be installed at run time. Likewise, each shortcut, registry's entry and any other project's item can be completely customized. Also, each dialog that you can display has its own properties that determine everything from the text that appears on the various parts of the screen to the color of the dialog itself. These settings are all stored in a single file called the project file. The project file contains all of the properties and settings of a project and the list of source files that need to be gathered up each time the project is built.

When you start a new project, VisualSetup's project wizard walks you through the first few steps of project creation. This helps you get your project started quickly without missing any of the basics.

Let's open the VisualSetup program and start a new project.

1) Open VisualSetup.

Use the Start menu to launch the VisualSetup program.

You will find VisualSetup under:

Start > Programs > VisualSetup 4.0

2) When the Welcome dialog appears, click on "New Project".

The Welcome dialog appears whenever you run VisualSetup. It not only welcomes you to the program; it also lets you easily create a new project, open an existing one, or restore the last project you worked on. (Restoring the last project automatically opens the project you were working on the last time you ran VisualSetup).

When you click on "New Project", the Welcome dialog closes and the project wizard appears.

3) Enter your information and click Next.

The first step of the project wizard asks you for four pieces of information related to your project. Simply enter your company name, product name, product version, and web address into the appropriate fields.

When you have entered all your information, click Next to move to the next step in the project wizard.

Tip: At any step in the project wizard, you can click Cancel to go straight to the program window with no files added and all of the default settings untouched.

4) Choose the language of your project.

VisualSetup was completely designed for multi-language projects. All setup projects are saved using the Unicode character set and all editors allows you define information that depends on a language. In this step, you must select the default language of your project.

Note: VisualSetup does not include the resources for all available languages. So, probably you will need to translate the resources by yourself or download the available translated resources from the VisualSetup web site. All language-dependent resources, like messages, dialogs and string tables, can be exported and imported from VisualSetup. Also, all dialogs can be saved as a dialog template to be used in other projects.

5) Define the setup appearance and license agreement.

VisualSetup allows you customize all the setup interface, including the background window. Choose between a setup program with background window (standard installation style) or without it (Windows Installer style). All users that will install your application must to accept a license agreement. In this step you can define the text file that will be used as your license agreement. Simply select a text file, that can be a plain-text file (.TXT) or a formatted text file (.RTF).

6) Provide the full path of the folder where your files are located, and click Next.

The next step in the project wizard is to specify the folder where your source files are located.

Note: In this case, "source files" refers to the original files on your local hard drive, i.e. the files that you want to create an installer for, and not files containing source code. VisualSetup does not need access to your source code files at all (unless you intend to distribute them in your installer).

If your product files are spread across several folders, choose the main folder that represents the core of your product. By default, all of the files and folders contained within the folder that you select will be included as well, so you want to choose the folder at the "top" of your product's folder hierarchy.

Once you have selected the folder, click Next to proceed to the next step.

7) Choose the third-party technologies you want to install with your application, and click Next.

Some projects use third-party technologies that must to be distributed with your application. The next step in the wizard is to select these technologies. You can include them to your project later, using the Runtimes Wizard.

8) Choose the type of your project.

VisualSetup allows you automatically import Visual Studio .NET and Visual Basic projects. Simply select your project type and the wizard will import all dependencies of your application.

9) Provide the full path of the main executable file.

The next step in the project wizard is to specify the path where your main executable is located. Check the option *Scan dependencies* if you want VisualSetup imports all files that the executable depends.

Note: Normally you do not need to scan the dependencies of the main executable, since all the true dependencies can be imported using the Wizards provided by VisualSetup. Almost dependencies your executable have are Operational System libraries that you must not distribute with your application.

Please read the section related to your project's type.

10) Select the optional features you want. Once you are ready, click Finish to apply your settings to your new project.

The final step of the project wizard lets you configure a number of optional features, such as whether to automatically provide an uninstaller.

When you click Finish, the project wizard closes and the new project is loaded into the design environment with the settings you chose. At this point, you could build the project and generate a basic installer for those files. Of course, you'd probably want to customize the screens before distributing it.

1.1.1 Visual Studio .NET Project

VisualSetup can import all the dependencies of your Visual Studio .NET project or just check if the .NET Framework is installed. If you choose to install your application dependencies, select the Visual Studio .NET project file. If you choose to check the .NET Framework version, type the minimum version your application needs to run.

1.1.2 Visual Basic Project

In this step, just provide the path of your Visual Basic project. VisualSetup supports Visual Basic 5.0 and 6.0 project's files. Select the Visual Basic version you are using and click Next.

1.2 Make Sure You Have the Latest Version

VisualSetup has the built-in ability to check the Internet to see if there is an update available. Before we start exploring the program interface, let's use this feature to make sure you have the latest version of the program.

1) Choose *Help > Check for Update*.

VisualSetup updater will open. It will check if you have installed the last version of VisualSetup and download the necessary files if it is necessary. It is strongly recommended to check for updates regularly.

1.3 Learning the Interface

Now that you have VisualSetup started and you have made sure that you're using the latest version, it's time to get comfortable with the program interface itself.

1) Explore the VisualSetup program window.

The VisualSetup program window is divided into a number of different parts. At the top of the window, just under the title bar, is the program menu. You can click on this program menu to access various commands, settings and tools.

Below the program menu are a number of toolbars. The buttons on these toolbars give you easy access to many of the commands that are available in the program menu.

Tip: You can create your own custom toolbars or edit the existing ones by choosing *Tools > Customize*.

Most of the program window is taken by the current editor, that can be the Files editor, Shortcuts editor, Dialog editor or any other. VisualSetup has a completed integrated interface; all editors work in a very similar form. The common actions you can do in the editors are: add new items, remove the selected items and edit the select items' properties.

At the very bottom of the window, a status bar reflects your interaction with the program and offers a number of informative readouts.

The rest of the program window is made up of individual sub-windows known as *panes*. Each pane can be docked, tabbed, pinned, resized, dragged, and even made to float on top of the design environment. Some important panes are displayed by default in the interface:

- Files and Shortcuts panes displays the folders structure of your setup project, just select the folder you want to see its contents in the Files/Shortcuts editor.
- Registry pane displays the Registry's keys structure of your setup project, select a Registry's key to see its contents in the Registry editor.
- Setup Dialogs pane displays all setup dialogs of your application, right-click this pane and, using the popup menu, add, remove and edit the setup dialogs.
- Properties pane shows the properties of the selected item of the current editor, use it to change the project's items properties.

The Project Manager pane provides easy access to the parts of VisualSetup you will use the most. It provides an alternative to the program menu for accessing the various parts of the VisualSetup environment. This pane, by default, is collapsed on the left.

2) Close the Project Manager pane.

You can close a pane by clicking on the little x on its title bar.

3) Choose *View > Project Manager* to open the Project Manager pane again.

All of the panes can be toggled on or off in the View menu. When you choose *View > Project Manager*, the Project Manager pane is restored to the same position it occupied before you closed it.

4) Make the output pane taller by dragging its top edge up.

You can resize panes by dragging their edges. In this case, you want to drag the part "between" the output pane and the file list...specifically, the little bit of window surface below the file list and above the output pane's title bar. As you begin to drag the edge of a pane, a line will appear to show where the edge will move to when you release the mouse button.

When panes are docked alongside each other, dragging an edge will resize both panes at the same time. As you make the output pane taller, you will be making the file list shorter at the same time.

5) Double-click on the output pane's title bar to un-dock it.

The output pane is docked by default. A docked pane is seamlessly integrated into the program window. You can "un-dock" it from the program window by double-clicking on its title bar.

When you un-dock the output pane, it floats above the program window, and the file list expands to fill in the space that the output pane left behind.

6) Drag the output pane to the bottom edge of the program window to dock it again.

You can move panes around by dragging them by their title bars. As you move a pane, an outline shows you the general area where the panel will end up. If you drag the pane near the edge of the program window, or near another pane, the outline will "snap" to show you how the pane can be docked, tabbed, or otherwise combined with the target area.

You can pin or unpin a pane by clicking the little pin icon on the pane's title bar.

Panes remember their positions even after you unpin them. If you unpin a pane, and then pin it again, it will return to the position it had before it was unpinned.

1.4 Getting Help

If you still have questions after reading the user's guide, there are many self-help resources at your disposal. Here are some tips on how to quickly access these self-help resources.

1) Press the F1 key.

The online help is only a key press away! VisualSetup comes with an extensive online program reference with information on every object, action, and feature in the program.

In fact, whenever possible, pressing F1 will actually bring you directly to the appropriate topic in the online help.

Note: You can also access the online help system by choosing *Help > Contents*.

There are three ways to navigate the online help system: you can find the appropriate topic using the table of contents, or with the help of the keyword index, or by searching through the entire help system for a specific word or phrase.

2) Choose *Help > Dynamic Help*.

Dynamic Help is a context sensitive help, displayed inside the VisualSetup interface. It can guide you to do the common actions for each editor you are using.

3) Navigate through the VisualSetup community forums.

VisualSetup is used by thousands of people worldwide. Many users enjoy sharing ideas and tips with other users. The online forums can be an excellent resource when you need help with a project or run into a problem that other users may have encountered. Visit our on-line forum at <http://epocketsetup.forumup.com>.

4) Request technical support.

If you cannot find the answer for your question in the resources above, please contact us sending an e-mail message to support@e-pocketsetup.com. We will answer your message as soon as possible.

1.5 Setting Preferences

There are a number of preferences that you can configure to adjust the VisualSetup design environment to suit your work style. Let's have a look at some of them.

1) Choose *Tools > Editor's Preferences*.

The editor's preferences allow you to change settings that affect the project file and the environment. For example, you can configure when the Dynamic Help will be updated. Also you can change the fonts used in the interface to improve the accessibility of VisualSetup, and choose either to use the project wizard to create new

projects or to simply start with a new, blank project.

2) Save and restore the environment layout.

You can save and restore the environment state. It can be very useful when you want a different environment for each user or project. Choose *Layout* in the Editor's Preferences dialog. To save the layout, type a layout name and click *Save* button. To apply an existent layout to the interface, select the layout you want and click *Apply Layout* button.

3) Include those tools you normally use in the Tools menu.

VisualSetup allows you customize the Tools menu including those tools you normally use during the setup development process, like your favorite picture editor, icon editor and other applications. Simply type a name for the tool you want to include and then select the application file that will be called when you select the tool in the Tools menu.

1.6 Planning Your Installation

The first step in creating a successful software installation is to plan it out.

In fact, planning your installation is one of the most important steps in designing a professional installer. Knowing what your installer needs to accomplish in advance will give you a clear goal to aim for and a solid plan to follow.

Tip: As with most things, the more planning you do now, the less repairing you'll have to do later. Investing some time in planning at the start can save you a lot of time in the long run.

There are several things you need to know in order to create an installer:

- What files do you need to distribute?
- Where do your files need to be installed?
- What system changes need to be made?
- What information do you need from the user?

This section will provide you with the information you need to answer these questions as quickly and accurately as possible.

1.6.1 What Files Do You Need to Distribute?

The first step in preparing your installation is to determine exactly what files your software requires for proper operation.

There are four basic types of files that you may need to distribute: program files, configuration files, operating system components, and shared application resources.

Program Files

Program files are the "main files" of your application. These files are essential to your application and are only useful in the context of your application. They can be executables, help files, documents, templates, or any other data files that your application requires. Program files usually make up the bulk of your software.

Configuration Files

Configuration files are used to store startup options, user settings, and other configuration options for your software. Information can be read from and written to these files during the normal operation of your application. These files are often referred to as "config" files, and come in many different formats, from traditional "INI" files (which adopt the same internal structure as Windows .ini files) to modern XML files.

Operating System Components

These files are usually included with the Windows operating system, but you may want to distribute the newest versions of certain files, or you may have developed custom system files of your own. These files are generally DLLs, OCX components, or hardware drivers like SYS files and VxDs.

Shared Application Resources

These are files that may be shared by more than one application, such as ActiveX controls, OCX components, and DLL files.

Some of the files that you need to distribute will be obvious, such as the main executable and help files. Others may be less apparent, such as DLLs and ActiveX controls installed in the Windows directories of your development system.

Note: Many of today's development tools require that you distribute runtime support files along with your application. Please consult your development tool's documentation to determine what files you need to distribute with your software.

Often an executable in your application absolutely requires other files in order to work properly. These "absolutely required" files are known as dependency files.

Dependency Files

Dependency files are external support files that an executable requires for proper operation. In other words, they are external files that a program file "depends on" in order to function properly. Dependency files may include INI files, DLLs, ActiveX controls, OCX components, or any other support file type. Although it's generally preferable to install dependency files in the same folder as the program that needs them, they are often installed in other locations, such as the Windows system directories.

Tip: VisualSetup has a built-in dependency scanner that you can use to identify dependency files and add them to your project. You can access this feature by choosing *Tools > Dependencies Scanner Wizard*.

1.6.2 Preparing the Directory Structure

The ultimate goal of a good installer is to get your software onto the user's system in an easy and accurate manner. Although VisualSetup ensures both you and your users' ease of use, the accuracy of the installation itself is largely up to you. Whether your files are installed in a structure in which they can function properly depends largely on where you tell VisualSetup to install those files.

The best way to ensure an accurate installation is to prepare the software in its finished state on your development system before you begin creating the installer. This means setting up the entire directory structure and all of the file locations exactly as you want them to appear on the user's system.

The end result should be that your software is fully installed on your development system exactly as you want it installed on the user's system.

Not only does this make it easy to test your software in its intended directory structure, but it also makes the process of designing the installer much quicker and easier for you. All you'll have to do is add your application folder onto the VisualSetup files editor, and the files and sub-folders will be recreated exactly the same on the user's system.

Tip: You should fully test your software in its "final" structure before creating the installer.

1.6.3 Where Do Your Files Need To Be Installed?

Once you have your software organized, you need to determine exactly where each file needs to go on the user's system. Although VisualSetup does a lot of this work for you by maintaining directory structures when you add files to your project, there are still some files that may need to be directed to different locations on the user's system.

Here are some guidelines to help you determine where you should install your files on the user's system. Once again, there are four basic types of files that you may need to distribute: program files, configuration files, operating system components, and shared application resources.

Installing Program Files

Program files should be installed in a folder that the user chooses during the installation process. Throughout

this manual and in the VisualSetup program, this folder is referred to as the installation folder. The path to the installation folder is represented by the variable `%InstallDir%`.

It's okay to install program files in sub-folders within the application folder--in fact, organizing your program files into sub-folders is a very good idea. Your files don't all have to be in the application folder itself; the folder that the user chooses can be used as a common application folder, with sub-folders for all of your program files.

If the users aren't given an opportunity to choose an installation folder, the path that you provided as the default value for `%InstallDir%` will be used. You can provide a default path for the installation folder on the Project Information section of the Project Properties dialog. (To access this tab, choose *Project > Project Properties* from the menu.)

Installing Configuration Files

In the past, initialization or "INI" files were often installed in the Windows folder (`%WinDir%`). This is definitely not necessary or even beneficial in all cases. Unless your application shares a configuration file with other programs, it's best to install the file in the installation folder along with your program files.

You should avoid installing any files in the Windows folder, or in any other folders where critical operating system files are stored, unless it is absolutely required by your application.

Installing Operating System Components

Operating system components, such as DLL or OCX files, are traditionally installed in the WINDOWS\SYSTEM folder (`%SysDir%`). If your application doesn't need to share these files with other applications, it's best to install them in your installation folder (`%InstallDir%`) instead.

Installing Shared Application Resources

Shared application resources, such as some ActiveX controls or DLL files, are generally installed in the WINDOWS\SYSTEM folder (`%SysDir%`).

Note: Many DLL and OCX files are COM servers, also known as "OLE components" and "ActiveX controls." As such, they will need to be registered with the operating system before they will be available for use by your application.

VisualSetup will automatically try to detect files that require registration when you add them to your project. You can also manually indicate files that you want VisualSetup to register by using the ActiveX property of the file.

Keep in mind that some DLL and OCX files have dependency files themselves, and these dependency files need to be distributed and possibly also registered before the DLL or OCX files can be used. You should consult the documentation for your components to determine what dependencies they have.

Note: Many OCX and DLL controls ship with a dependency file. The dependency file typically has the same filename as the control with a .DEP extension, e.g., Control.ocx would have a dependency file named Control.dep. These dependency files can be opened and viewed with a text editor (such as Notepad) and can tell you a lot about what, if any, dependencies the control may have.

Tip: You can also use Setup Factory's built-in dependency scanner to identify any dependency files your components may have. You can access the dependency scanner by choosing *Tools > Dependencies Scanner Wizard*.

1.6.4 What System Changes Need To Be Made?

The next step is to think about what changes, if any, need to be made to the user's system. This can include changes to system files (such as WIN.INI, CONFIG.SYS, SYSTEM.INI and AUTOEXEC.BAT), changes to the Registry, and even such things as installing and registering fonts. All of these changes can be handled easily using VisualSetup actions.

1.6.5 What Information Do You Need from the User?

The final step in planning your software installation is to consider what information you will need from the user, and what information you will need from the user's system. For example, in order to personalize your software or register it to a specific user, you may want to ask the user to provide their name on a User Information dialog. Or, you may want to ask the user for a serial number on a Verify Serial Number dialog.

Tip: You can also use actions to prompt the user for information (Get User Information action). See the help file for a complete list of the actions available in VisualSetup 4.0.