# wInstall

## An Example of a Graphical Installation Program

This distribution disk contains a set of source files and documentation that you can use to build an application installation utility based on Microsoft® Windows™. These sources are a subset of the source code used to build the Windows retail Setup application.

This information consists of the following components:

- C and assembly-language source files. These files can be compiled to produce a sample installation utility or modified to produce your own customized installation utility. See the following section for additional information.

- This document, which provides an overview of the source files and briefly explains how to use them to build an installation utility.

The source files are organized in three directories as follows:

**\WINSTALL (root)**
The files in the root directory are the files you would probably want to modify if you were customizing your installation utility.

**\WINSTALL\WSLIB**
The files in the WSLIB subdirectory are the "core-level" files, which contain routines you probably won't need to modify in order to customize your utility. You would want to change these files only if you wanted to change some of the basic ways in which Install works.

**\WINSTALL\DEMODSKS**
The files in the DEMODSKS subdirectory will create for you a quick two disk demonstration of how the setup program works. Use MAKE.BAT to create the demonstration.

***IMPORTANT*** *The Setup source files are not to be considered as fully tested under all possible configurations. Do not assume that an application compiled from these sources will be free of problems. You are encouraged to use these sources to build an installation utility that will be included with your application; however, it is your responsibility to ensure that the utility is fully tested and debugged before shipping it. Microsoft Corporation does not assume any responsibility for the stability of the code included here.*

# Trying the Sample Installation Utility

The source code included with this distribution disk can be compiled to produce a sample installation utility, INSTALL.EXE. To get an idea of what a basic Windows-based installation utility looks like, compile and run INSTALL.EXE.

The information file APPINSTALL.INF as provided on this distribution disk is designed to install the components of this distribution disk onto your system. It also includes exhaustive comments that describe the various sections, and their formats. Another file, DEMODSK.INF is used to create the demonstration installation that you find in the DEMODSK subdirectory.

Follow these steps to compile the sample installation utility and create a set of distribution disks for it to work with:

If you have not already done so, run the INSTALL program from the distribution disk to install this product onto your system. You can specify any destination location for this installation, by default, it is C:\WINSTALL.

This installation should include a fully built version of the entire application, it also includes the RES, OBJ, MAP and SYM files that result from building it, so you do not need to re-build it at this time.

Later, you will want to re-build the entire project to verify that it can be fully constructed on your particular development system. To do this, delete all the OBJ and RES files, then use the MAKEALL.BAT batch job in the root directory to re-build all the components.

You have already seen what the final product can do, because the installation of this distribution disk uses it to install itself. If you would like to create a 'demo' disk set that shows you another example, then go to the DEMODSKS directory and use MAKE.BAT to create a two disk demo of the install program. This will copy the installation program (INSTALL.EXE) and an information file (APPINSTALL.INF) and several dummy files to the floppy disks.

To test the example Install program, choose Run from the File menu in Program Manager. (This is necessary in order to take advantage of the automatic installation of program groups). Specify A:\INSTALL (where A: is your floppy drive) for the name of the program you want to run.

The sample installation program illustrates the Dynamic Data Exchange (DDE) interface for adding groups and items to Program Manager. For more information, see Section 22.4.4, "Executing Commands in a Remote Application," in the Microsoft Windows Software Development Kit Guide to Programming.

## Using the Setup Sources

There are several ways you can use the included files to build an installation utility:

You can use the source files more or less as they are. This means making changes primarily to the APPSETUP.INF file, and changing other source files only as needed to fix any problems. The resulting installation utility will be similar to the sample utility INSTALL.EXE.

You can add functionality, such as additional dialogs and supporting logic, to the basic source files. This method can produce an installation utility that is tailored to your application's needs.

You can also modify or add to the "core" source files, which determine basic behavior of Install, such as the method it uses to copy application files, how it uses DDE to interact with Program Manager, and so on.

The next section explains each method.

# Building a Simple Installation Utility

The simplest way to provide an installation utility for your application is to use the included source files more or less as they are. The only changes you need to make are modifications to APPSETUP.INF, and any changes to other source files, such as INSTALL.C, that are needed to fix problems you find during testing.

***IMPORTANT*** *The included source code is intended as a resource and not as a complete, tested product. If you use the source files as they are, it is your responsibility to test the resulting application and fix any problems before shipping your installation utility.*

**To build a simple installation utility, follow these steps:**

- Determine how much disk space your application requires.
- In APPSETUP.INF, use the mindiskspace statement to tell Install the minimum amount of space your application needs.
- Determine the number of distribution disks your application will use.
- In APPSETUP.INF, change the information in the [disks] section so that it defines each distribution disk.
- Determine which files will be shipped on each disk, and the destination directory for each file (relative to the main installation directory).
- In APPSETUP.INF, change the information in the [appcopy.app] section so that it defines the information for your application's files and destination directories.
- In APPSETUP.INF, change the information in the [progman.grp] section so that it specifies your application's command line and icon.
- Compile INSTALL.EXE by executing the MAKEALL.BAT batch file in the \SETUP directory.
- Copy the files INSTALL.EXE and APPSETUP.INF to the root directory of your application's first distribution disk.
- Test Install thoroughly by running it in conjunction with your APPSETUP.INF file and your application's distribution disks.
- Be sure to test Install in each of the three Windows operating modes (real, standard and 386 enhanced). Also, be sure to test all portions of the application and code.
- See the comments in the file APPSETUP.INF for details on how to change the sections and statements in that file.

# Building a Customized Installation Utility

You can create a customized installation utility by making changes to certain Install source files. In general, you should need only to change files in the main Install directory, rather than in the WSLIB subdirectory. (However, you might need to adjust some of the code in the WSLIB subdirectory if you encounter problems in those files during testing.)

**NOTE:**
To alter the "building blocks" of Install, such as the way it copies files, or the appearance and behavior of the "gas gauge" graphic that indicates current progress, you can modify one or more of the files in the WSLIB subdirectory. Each file includes comments that describe the functions they contain. See "Contents of the Source Files," later in this document, and the comments in each file, for additional information on the files in the WSLIB directory.

**The following are some ways in which you might want to customize your installation utility:**

- Adding a dialog box that asks the user whether or not to install specific elements of your application. For example, you might want to let the user choose whether or not to install a tutorial that takes up a lot of disk space.

- Adding logic that calculates conditional disk-space requirements, depending on options the user previously selected.

- Adding dialog boxes that allow the user to supply additional information you need in order to install your application. For example, if your application includes device drivers for various types of scanners, you might want to ask the user which type of scanner he or she is using, and perhaps ask additional questions about the scanner's configuration.

- Adding a Help system, if you think the user will have questions during installation. (To add a help system, use the Help-development tools that come with the Windows SDK.)

- Adding logic to support file decompression.

- Adding more sophisticated algorithms for calculating the percentage of files copied.

Note that most customization will require you to add code to the file INSTALL.C. (If you add a dialog box, you would add the dialog-boxdescription to INSTALL.RC, and add the functions that support the dialog box to INSTALL.C.)

***IMPORTANT*** *Use caution in changing the code-segment attributes in the module-definition file. Because Install is intended to be run from a floppy disk, all its code segments and resources must be PRELOAD FIXED. If the module-definition (.DEF) file does not define all segments as PRELOAD FIXED, Install will fail when the user changes floppy disks. This is because, without the PRELOAD FIXED segment attributes, Install will attempt to execute code or load resources from the executable file INSTALL.EXE on the floppy disk. If the user changes floppy disks, Install will be unable to find INSTALL.EXE on the second disk.*

# Designing a Good Installation Program

When building your installation utility, whether you customize the included files or create your own utility using these files as a base, you should keep in mind that consistency between applications is one of the things that makes Windows applications easier to learn and use. One goal in creating any Windows-based application should be to make it consistent with other Windows applications. This section lists and explains guidelines to keep in mind while designing and building your installation utility.

A good Windows-based installation utility has a Windows user interface consistent with other well-designed Windows applications. It is simple and easy to use.

Most Windows-based installation utilities follow the same general course of action. This order can simplify the installation process for the user, and typically does not need to interrupt file copying to ask the user additional questions. Most utilities follow this general order:

The utility can be started simply by choosing the Run command from Program Manager's File menu, and specifying the utility's filename. (Normally, no command-line switches or arguments are necessary.)

After starting, the utility displays an introductory dialog that tells the user the name of the application being installed.

The utility then asks the user to specify a destination directory.

The utility checks that directory to make sure there is enough disk space. If not, it asks the user to specify a different directory or exit the utility.

(Optional) The utility asks the user to specify any other information it needs to complete installation. For example, an application that comes with sample documents might ask the user if he or she wants to install them.

The utility copies the application files from the distribution disk(s) to the destination directory. While doing so, the utilty:

- Prompts the user to change disks, if necessary.
- Shows the user what percentage of files have been copied, if there are many or large files.
- The utility adds the application's program icon(s) to Program Manager's Windows Applications group, or creates a new group and then adds the icon(s) to that group.
- The utility informs the user that installation is complete, and offers the choice of starting the application or returning to Program Manager.

# Simplifying the Installation Process

If you build your own installation utility, try to keep it as simple as possible. The simpler the installation process, the less likely the user will make mistakes that require him or her to repeat the entire process. This section provides some guidelines for simplifying the utility.

**Keep Dialog Boxes Simple**

Throughout the installation program, use a simple dialog box with only a Cancel button. Try to keep dialog boxes to a standard size and screen location. Keep dialog text to a minimum.

**Get Any Questions Out of the Way First**

Ask the user any necessary questions about component objects before copying any files. For example, if your application includes an optional tutorial that takes up extra disk space, ask the user if he or she wants to install the tutorial before the file-copying process begins.

**Design Disk Contents to Avoid Unnecessary Disk Swapping**

If your application files come on several distribution disks, you might want to design the disk contents carefully so that the user does not have to swap disks repeatedly. The best approach allows the user to insert each disk once.

**Keep the User Informed About What is Happening**

The APPSETUP.INF file lets you define strings that the utility will display while copying a particular file. You can use these strings to display either the filename or a description of that file.

You can use the GAUGE.C graphic to keep the user informed about the progress of the installation. This graphic is a bar similar to the one used by the Windows 3.0 Setup utility.

The utility should tell the user when installation is complete.

# Topics and Considerations not covered in this example

Every application is a little different in what it's needs are during installation. This distribution disk contains a very simple version of an installation program, and it is not meant to address all issues that your application might want to, or need to consider. Some items that might be appropriate to you particular needs are:

- Modifications of AUTOEXEC.BAT and CONFIG.SYS
  Changes to either of these files should be kept to a minimum. Windows applications should not depend on standard DOS environment variables, or other settings in these files if at all possible. There are some instances where certain minimum settings need to be verified, for example the FILES= setting. If your application places a higher requirement on this value beyond what is required to run Windows, then your installation program should detect this, and present to the user the option of directly modifying the appropriate file or creating a new file (with a different name) that shows what the file should look like. In either case, the original file should not be removed from the system, and full information on how to return to the original configuration should be presented to the user.

- Modifications of WIN.INI
  Win.INI is not to proper location for storing application specific operation values. It is primarily for information that directly affects or addresses issues about the Windows system, its operation, and the drivers it loads. In previous versions of Windows, this was the only location that was provided for storing application information, but starting with Windows 3.0, each application can create and maintain its own .INI file with private information that it can store and recall from one session to another.

- Creation or Modifications of a private .INI file
  Several procedures are provided for storing application specific information that can allow a persistance of operation from one session to another. This information can contain information about the size and location that the user prefers the application to be displayed at, previous files that were opened, information about locations of additional data files, or virtually anything else that the application might need. Please refer to your programming documentation for a full discussion of these functions, and their proper usage. In the Microsoft Windows SDK Reference Volume 1, this information can be located on pp. 3-10, under the section: "3.9 Initialization-File Functions"

- Recovery from a cancelation of installation
  The possibility should be considered that for one reason or another the user cancels the installation of your application. Your installation should be constructed in such a way that a premature cancelation does not in any way affect the functionality of the users machine. In most cases, there shouldn't be any problem, but if you are modifying the CONFIG.SYS, or AUTOEXEC.BAT files, it is possible that the user could turn off the system at any time during this process. You should also consider the effect of installing your application on top of a partial, or aborted installation. If the user properly aborts installation (eg: tells the Install application to cancel the install), then you might want to consider providing the option of fully removing whatever parts of the installation had been completed..

- Deinstallation of software

The possibility exists, that at some time, for whatever reason, the user might want to remove you application from their system. You may want to provide a means of allowing the user a method of doing this in such a way that all the various datafiles and INI settings are properly removed from the system so that they do not clutter up the users disk or configuration.

- Updates

  You might want to consider how your installation might best be able to handle the situation of installing an update in the future. Do you want to check the existing version of various files and only copy the parts of the update that are new? Do you need to run some sort of conversion tool that will convert existing data files into a format that will be useable by the new version? Can you leverage system information based on the previous installation so that you can ask the user fewer, or less confusing questions about their system?

- Network installation

  Many companies are wanting to be able to install software from their network. This is something you should consider as a possibility, and if you wish to support it, make sure that you installation program will properly understand this and not present a confusing interface to the user that assumes it is being installed from a floppy drive.

## Contents of the Source Files

This section lists and briefly explains the source files in the SETUP directory and its subdirectories.

**The WINSTALL directory contains the following files:**

**APPSETUP.INF**
Information file. Defines installation disks and files, destination directories, and other installation information. This is the file you must edit in order to build a simple installation program.

**INSTALL.C**
Main module. Calls functions that perform the following tasks:
- Copying files while displaying progress information.
- Parsing the .INF file.
- Creating a new Program Manager group and installing icons into the group using DDE with Program Manager.

**INSTALL.DEF**
Module-definition file.

**INSTALL.EXE**
Executable utility, built from the code included with this document.

**INSTALL.H**
Header file.

**INSTALL.ICO**
Icon file.

**INSTALL.RC**
Resource script file.

**MAKEALL.BAT**
Batch file you can run to compile INSTALL.EXE.

**MAKEFILE**
Make file.

**The WSLIB subdirectory contains the following files:**

**COPY.C**
Code that copies files.

**DOS.ASM**
Miscellaneous utility functions needed to build INSTALL.LIB.

**GAUGE.C**
Graphic that shows the percentage of files that have been copied.

**GAUGE.H**
Header file for GAUGE.C.

**INFPARSE.C**
Reads the file APPSETUP.INF and uses information in it at run time.

**MAKEFILE**
Make file.

**PROGDDE.C**
Installs the program icon in Program Manager using DDE.

**PROGDDE.H**
Header file for PROGDDE.C.

**SULIB.H**
Required header file for building INSTALL.LIB.

**WINENV.H**
Required header file for building INSTALL.LIB.

**WPRINTF.H**
Required header file for building INSTALL.LIB.

**WS.H**
Required header file for building INSTALL.LIB.

**WSRC.H**
Required header file for building INSTALL.LIB.

**WPRINTF.LIB**
Needed for linking and building INSTALL.LIB and INSTALL.EXE.

The DEMODISKS subdirectory contains a set of files that you can use to create demonstration disks for use with the sample installation program.

All the tools required to build this source code can be found in the Microsoft Windows version 3.0 SDK and Microsoft C6.0 Compiler. Other compilers may work as well, but some modifications to the sources provided may be necessary.