WINSTALL 2.0

Network Application Installer for Windows

Documentation and Software

7319 Willow Avenue
Takoma Park, MD 20912
(301)270-4458

NOTE: Microsoft and Microsoft Windows are registered trademarks of the Microsoft Corporation. Micrografx is a registered trademark of the Micrografx corporation.

TABLE OF CONTENTS

TABLE OF CONTENTS, Continued

PREFACE:

New Features

For those users who have purchased or evaluated earlier versions of WINSTALL (formerly known as N.A.I.), here is a list summarizing the major enhancements in release 2.0:

Environment and Application-Specific Variables

In addition to the support provided for user-specific variables, WINSTALL now fully supports environment variables (up to 5) and also supports application-specific variables (up to 3) as well. WINSTALL can substitute any or all of these variables within the file modifications, in source and destination paths for files to copy, and in the command line, icon path, and working directories. Environment and user-specific variables can also be written to the WINSTALL network log file as well.

Additional File Modifications

WINSTALL now provides a means of automatically modifying information within the AUTOEXEC.BAT and CONFIG.SYS files, plus one other ASCII file per application .DAT file. These modifications are in addition to the earlier support for modifications to the WIN.INI and SYSTEM.INI files.

File Copy Improvements

The copying of files during installation has been improved in several ways. First of all, the speed of the copy operation has been vastly increased. Second, the destination files now have the same date and time as the source files. Finally, WINSTALL offers an array of options for handling copy operations when the destination file already exists.

WINSTALL Network Log

WINSTALL now offers the option of logging all its operations to a file on the server. WINSTALL will log all activities by users and administrators, including error messages and WINSTALL configuration changes. User-specific variables and environment variables can be logged along with each entry to help identify the user.

Special Handling of the WIN.INI [Fonts] Section

Because multiple programs modify the WIN.INI [Fonts] section, WINSTALL now accords the [Fonts] section the same special treatment it has always provided to the [Extensions], [OLE], and [Embedding] sections. This special treatment enables the REMOVE APPLICATION operation to remove only the entries in these sections that were added during the installation of the application, preserving any entries belonging to other applications.

Message Display Options

WINSTALL now provides the option of displaying a short message to the end user before or after an installation or removal operation.

Program Call Option

Release 2.0 extends the reach of the WINSTALL program by providing an option to automatically call any other program at the conclusion of an installation or removal. Called programs can include other Windows applications, DOS applications, upgrade or installation routines for specific software packages, even .BAT files. WINSTALL will even pass command line parameters to called programs.

Miscellaneous Enhancements and Improvements

A number of small improvements and enhancements have also been added to release 2.0, including the ability to create a working directory if it does not exist when an application is installed, and the ability to remove an empty working directory when an application is removed, the removal of empty directories when an application is removed, improved error checking throughout the program, and checking for valid DOS filenames where appropriate.

Compatibility with Prior Releases

WINSTALL 2.0 is backwards compatible with all prior releases of N.A.I. The executable programs in earlier versions of the program had different names and used somewhat different file formats. Earlier versions of the end user program were called NAI.EXE. The current version of the end user program is

WINSTALL.EXE. Earlier versions of the administrative program were called NAIMAINT.EXE; the current version is WINSTADM.EXE.

The WINSTALL.EXE program can read and execute application .DAT files created with all versions of the NAIMAINT.EXE program, so there is no need to immediately update application .DAT files as soon as you install the new version.

The WINSTADM.EXE administrative program can read application .DAT files created with all versions of the NAIMAINT.EXE program, but **application .DAT files created by WINSTADM.EXE are not compatible with any versions of NAI.EXE or NAIMAINT.EXE.**

The WINSTADM.EXE will, on startup, look for its own configuration file, WINSTADM.CFG. If it does not find that file, it will then look for the NAIMAINT.EXE configuration file, NAIMAINT.CFG. It can read NAIMAINT.CFG. If the working directory changes, however, WINSTADM.EXE will create a new configuration file, and NAIMAINT.EXE will be unable to use that file.

What WINSTALL (Network Application Installer) Is

One of the great advantages of a PC LAN file server is that administrators can provide their entire user community with access to a software program simply by installing a single copy of the program on the server. Unfortunately, many Microsoft Windows applications are not amenable to this approach, because Windows programs often require specific changes to the environment of each PC where the application is to run. For example, a Windows application install routine is likely to make changes to the user's WIN.INI file, to create special, application-specific .INI files, to create special .DLL files, etc. The application may not run, or may not run properly, without these changes. Until now, the only means of meeting the need for these local configuration changes was to make them manually to each PC on the LAN or else to run the application install routine on each PC on the LAN.

WINSTALL, the Network Application Installer, eliminates these problems, allowing a LAN administrator to install a Windows application only once, and providing end users on the LAN with the means of installing in their local workspace (or removing from their workspace) any Windows application with a click of the mouse. The convenient operation of WINSTALL makes quick work of adding new applications or upgrading installed applications to the latest release.

WINSTALL was initially designed to enable users to install applications which would run from the file server, but it can work equally well as a method of distributing full applications to each PC's local hard disk, especially now that the speed of the file copy operation has been increased.

In addition, whether you run your Windows applications locally or from the file server, WINSTALL's removal option provides end users with the most effective tool available to help keep their local Windows configurations up to date and free of extraneous, leftover files and information. This option alone may be reason enough to install WINSTALL on your file server.

An important feature of WINSTALL is that it provides the ability to install *any* Windows program, and *an unlimited number of Windows programs*, without purchasing any additional modules or waiting for their development. You simply install an application once, note the modifications it makes, and then create the necessary WINSTALL module yourself in minutes, using the WINSTALL Administration Program from within Windows! Your users are instantly provided with the means of installing the new application at the

click of a mouse.

WINSTALL is flexible enough to allow you to work the way you want to. You can adapt WINSTALL to your environment, no matter what your requirements are, thanks to features such as the ability to modify any ASCII file during the installation process, the ability to display custom messages to your end users at the begining and/or end of an installation (or removal) process, and the ability to call any other program, even a .BAT file, at the end of the installation or removal.

Although some new applications now include a node install program which makes the necessary configuration changes on the local workstation, WINSTALL provides you with better control over the configuration and the process, provides the end user with superior ease of use, and offers the unique ability to remove the entire application from the local configuration at any time, with a simple click of the mouse.

When WINSTALL installs an application, it performs these critical actions automatically:

1. Modifies any or all of these files as required:
   WIN.INI
   SYSTEM.INI
   AUTOEXEC.BAT
   CONFIG.SYS
   One additional ASCII file of your choice

2. Copies any needed files to the local disk.

3. Installs the appropriate icons in the local workspace.

4. Calls an external program, if configured to do so.

5. Displays a custom message to the user at the start and /or end, if so configured.

6. Updates the WINSTALL network log, if enabled.

7. Updates the user's private list of installed applications (file NAI.INI).

Some applications may require as little in the way of local configuration

changes as the single act of placing an icon in the local workspace, but WINSTALL is helpful even with these simple installations because it relieves the end user of having to know or enter anything concerning directories or executable file names; the icon can be installed in the workspace by a simple click of the mouse with no chance of typographical errors or other miscues, and the WINSTALL network log, if enabled, will track the installation for the administrator's records.

How WINSTALL Works

WINSTALL consists of two programs: WINSTALL.EXE, which the end user runs to install or remove programs, and WINSTADM.EXE, which the LAN administrator runs to configure WINSTALL.EXE to make applications available for automatic installation and removal.

WINSTALL.EXE

When a user clicks on the WINSTALL icon, the program WINSTALL.EXE reads (from the file WINAPPS.LST) the list of applications available to be installed or removed.

WINAPPS.LST also contains information on a number of WINSTALL configuration options. For example, it tells WINSTALL whether or not the *REMOVE APPLICATION* function is enabled, allowing users to remove applications as well as to install them. In addition, WINAPPS.LST contains the configured options for several other features, including environment variables, prompts for User-specific variables, configuration options concerning the WINSTALL network log, and instructions on what to do during a file copy operation when the destination file already exists.

When the user selects an application to install or remove, WINSTALL reads the data file (filename.DAT) for that application. The .DAT file tells WINSTALL what changes to make to the user's local configuration for the installation or removal of that application, as well as what other actions to take in conjunction with the installation or removal. For example, the application .DAT file may instruct WINSTALL to display a custom message before beginning the installation.

WINSTALL then automatically makes the changes, including placing the icon

on the user's workspace, and lets the user know that the installation was successful. A typical application can be completely installed in this manner in under 10 seconds!

WINSTADM.EXE

Five types of data files are associated with the WINSTALL programs:

WINAPPS.LST (The list of applications available for installation or removal.)

Application .DAT files (Instructions to WINSTALL.EXE for installation and removal of particular applications)

WINSTADM.CFG (the administrative program configuration file)

NAI.INI (the user's private WINSTALL log file)

any filename (the WINSTALL network log file)

All are ASCII text files, but each has its own specific format. To avoid requiring the LAN administrator to understand and remember the details of those formats (although they are fully described later in this document), and to avoid inadvertent errors in file formats, all these files are created and maintained by means of an additional program: WINSTADM.EXE (except for NAI.INI, which is created and maintained automatically by the end user program, WINSTALL.EXE, and the WINSTALL network log, which is updated by both programs as needed).

WINSTADM.EXE allows the LAN administrator to create new application .DAT files and modify or delete existing application .DAT files, at the same time updating WINAPPS.LST, the list of available applications that users see when they run WINSTALL.EXE. WINSTADM.EXE also permits the LAN administrator to enable or disable the *REMOVE APPLICATION* function in WINSTALL.EXE, to control the use of user-specific variables and environment variables, to configure the WINSTALL network log function, and to specify what to do during a copy file operation if the destination file already exists.

INSTALLATION:

User Files

The user files (WINSTALL.EXE, WINAPPS.LST, and all the application .DAT files) should all reside in the same directory on the file server. Users need not have write access to this directory.

Each user must have the files VBRUN100.DLL and DISKSTAT.DLL in the Windows directory, and the WINSTALL icon in the workspace.

LAN Administrator Files

WINSTADM.EXE and WINSTADM.CFG can reside in same directory as the user files, or, for better security, they can reside in another directory. (The *CURRENT WORKING DIRECTORY* field on the main screen of WINSTADM.EXE allows the LAN administrator to specify the location of the user files). The LAN administrator must have read, write, and delete privileges for the directory where the user files reside and for the directory where the maintenance files reside.

The LAN administrator must have VBRUN100.DLL in his local Windows directory and the WINSTADM icon in his workspace.

RUNNING WINSTALL:

Choosing an Application to Install

When the user clicks on the Network Application Installer icon, the WINSTALL.EXE program reads the file WINAPPS.LST, which lists all the applications available for installation. WINSTALL then presents the main screen, containing the full listing of all applications available for installation. To install an application, the user simply clicks once on the desired application in the list and then clicks on the *INSTALL APPLICATION* Button.

If WINSTALL discovers that the user has already installed this application, it will offer a choice of canceling the installation or performing a re-installation.

If the application is configured to present a custom message before the installation, WINSTALL displays it and waits for the user to click on the *OK* Button to proceed or the *CANCEL* Button to back out of the installation.

WINSTALL will display a completion bar to indicate its progress as it makes the necessary configuration changes and copies the required files to the user's local disk.

The final step in the installation is the placing of the application icon(s) in the user's Windows workspace. If the application .DAT file specifies the creation of a separate icon group, WINSTALL will create that group and place the icon(s) in it; if the application specifies that the icons be added to an existing group, WINSTALL will do that, creating the group if it does not already exist; otherwise, it will place the icons in whatever is the active group on the end user's workspace at the time of the installation.

If the application is configured to call another program at the end of the installation process, WINSTALL displays a message to the user that a called program is loading, and WINSTALL runs the called program.

If the application is configured to present a custom message after the installation, WINSTALL displays it and waits for the user to click on the *OK* Button.

Choosing an Application to Remove

To remove an application, the user clicks on the application in the application list and then clicks the *REMOVE APPLICATION* Button. At this point, WINSTALL reads the application .DAT file and then , after displaying a custom message (if specified in the application .DAT file), removes the appropriate WIN.INI lines, files, etc., from the user's local configuration, again displaying a completion bar to keep the user apprised of the progress.

If the application .DAT file specifies that the icons for the application belong in a separate icon group, WINSTALL will remove the group and any icons it contains from the end user's local workspace. On the other hand, if the application .DAT file specifies that the icons belong in an existing group or in whatever the active group is at the time of installation, WINSTALL will not remove any icons from the user's workspace; instead, at the end of the removal process, WINSTALL will advise the user that the icon may not have

been removed and will provide simple instructions on how to do so manually. At the end of the process, if a program is specified to be called, WINSTALL displays a message to the user that a called program is loading, and WINSTALL runs the called program. If a custom message is configured to be displayed at the end of the process, WINSTALL displays it and waits for the user to click the *OK* Button.

WINSTALL ADMINISTRATION:

Administrative Program Startup

When the LAN administrator clicks on the WINSTADM.EXE icon, the program looks for the file WINSTADM.CFG in the WINSTADM.EXE program directory. If this file is found, WINSTADM.EXE reads it to discover where the correct version of the file WINAPPS.LST is located. If WINSTADM.CFG is not present, then WINSTADM.EXE looks for WINAPPS.LST in the WINSTADM.EXE program directory. If WINAPPS.LST is not found, WINSTADM.EXE informs the LAN administrator, but proceeds anyway, providing the opportunity to create a new WINAPPS.LST or to change the working directory to one which does contain a WINAPPS.LST file.

WINSTADM.EXE reads the file WINAPPS.LST to build its list of installable applications before displaying the initial screen.

This screen offers the options of setting the working directory, configuring the user-specific variables, configuring the runtime options, plus several choices regarding the list of available applications: create a new application entry in the list, modify an existing application entry, insert an application into the list, or remove an existing application from the list.

Setting the Working Directory

To enable the security of keeping WINSTADM.EXE in a directory separate from the WINSTALL.EXE program, the file WINSTADM.CFG, which WINSTADM.EXE reads on startup, is kept in the same directory as WINSTADM.EXE. This file stores the location of the last working directory, the directory where WINSTALL.EXE, WINAPPS.LST, and the application .DAT files are kept. If this configuration file is missing, WINSTADM.EXE sets the working directory to the directory where WINSTADM.EXE itself is located.

The current working directory is displayed in a modifiable field on the WINSTADM.EXE main screen. When that directory is changed, WINSTADM.EXE immediately switches to the specified directory and loads the WINAPPS.LST file from that directory, if there is one. If it does not find a WINAPPS.LST file in the specified directory, it informs you and begins by presenting a blank list of applications. Any changes will result in the creation of a new WINAPPS.LST in whatever the working directory is at that time; the actual file will be created either when the WINSTADM.EXE program is exited, or when the working directory is changed.

Setting the User-Specific and Environment Variables

WINSTALL supports the use of up to three separate, global (non-application-specific), user-specific variables plus up to five separate environment variables during the installation and removal of applications. (WINSTALL also supports application-specific variables, which are stored in each application .DAT file and which are then used only in the installation or removal of that specific application. Application-specific variables are covered in their own section, below.) User-specific and environment variables allow installations to vary according to information that may be unique to each user. For example, if you want to install certain files for an application in a directory that is named differently for each user, WINSTALL's user-specific variable feature enables you to do so.

If you choose to implement the global user-specific variables, WINSTALL will prompt each user at runtime for the unique information you need. After receiving this information from the user, WINSTALL will use it during the installation or removal of applications by substituting the unique, user-specific information for special codes ($VAR codes) it encounters in the application .DAT files. Environment variables work in much the same way, except that the user is prompted only if the specified variable is not present in his environment.

User-specific variables are implemented in two steps: First, you must instruct WINSTALL what information to request from the user or to gather from the environment. Second, you tell WINSTALL, in each application .DAT file, what to do with that information.

To tell WINSTALL what information you need from the user or the environment, click on the *SET USER-SPECIFIC VARIABLES* Button on the

WINSTADM.EXE Main Screen. The User-Specific Variables screen will appear, with three blank data entry fields labeled *PROMPT 1, PROMPT 2*, and *PROMPT 3,* and 5 more blank data entry fields labeled *Environment Variable 1, Environment Variable 2,* etc.

You may enter information in any or all of these fields. For example, if you wanted to place certain application files in a directory named according to the user's network username, you could enter for *PROMPT 1*, the following text:

Username

If information has been entered for *PROMPT 1*, *PROMPT2*, or *PROMPT3*, then when an end user runs WINSTALL and selects the *INSTALL APPLICATION* or *REMOVE APPLICATION* option for the first time, WINSTALL will present a screen asking for the unique, user-specific information you specify, prompting him with whatever text you enter in the *PROMPT 1, PROMPT 2*, and/or *PROMPT 3* fields.

NOTE: Use of the user-specific variables feature is entirely optional: WINSTALL will only prompt for those fields where you have actually entered information, and if you do not enter information for any of the fields, WINSTALL will not request any information from the users and will not attempt to implement any user-specific installation parameters.

To continue the example above, if you entered the text shown above, then WINSTALL would present a screen instructing the user to enter his username.

When the user enters the requested information, WINSTALL will proceed with the installation or removal, substituting the text supplied by the user for the corresponding $VAR codes encountered within the application .DAT file.

Thus, the second step in implementing user-specific variables is to place the $VAR codes into the application .DAT files as required. WINSTALL will interpret $VAR codes wherever it finds them within text to be added to the WIN.INI and SYSTEM.INI files, to be added to the AUTOEXEC.BAT, CONFIG.SYS and other ASCII files, within directories and filenames specified in the FILES TO COPY fields of the Add/Modify Application screen, and in the directories and filenames in the ICONS TO INSTALL fields. Each of these areas is explained in detail in its own section, below.

Each of the three available user-specific variables has a different $VAR code. For the variable corresponding to *PROMPT 1*, you would enter *$VAR1$* (note both the leading and trailing *$* characters) in the application .DAT file. For the variables corresponding to *PROMPT 2* and *PROMPT 3*, you would enter *$VAR2$* and *$VAR3$*, respectively.

To continue the example, if you had instructed WINSTALL to prompt the end user for his network username, and he had entered *BWILSON* in response, then WINSTALL would substitute *BWILSON* for the $VAR code *$VAR1$* whenever that code appears during the installation. So, if you had indicated that a file should be copied to the destination directory *F:\$VAR1$*, WINSTALL would copy the files to the *F:\BWILSON* directory, creating it in the process, if necessary.

Likewise, if you had the line *DOC-PATH=F:\$VAR1$\DOCS* in the WIN.INI Additions section of an application, what WINSTALL would actually add to the user's WIN.INI file would, in the case of our example user, be this:

DOC-PATH=F:\BWILSON\DOCS

WINSTALL will only prompt the user for this information once during each execution of WINSTALL.EXE. If the user installs several programs without exiting WINSTALL.EXE, the program will remember the responses he provided for the first installation and use them whenever a subsequent application .DAT file contains a $VAR code. If the user exits WINSTALL.EXE, however, the information he provided is not retained and must be re-entered when the first application is installed or removed during subsequent executions of WINSTALL.EXE.

If you enter information in any of the Environment Variable fields, then WINSTALL will look in the user's environment for an environment variable that matches what you have entered, and it will substitute for the appropriate $VAR code the value that it finds in the user's environment. The environment $VAR codes are *$ENVAR1$*, *$ENVAR2$*, etc.

So to modify the above example slightly, if you entered *USERNAME* in the Environment Variable 1 field, then WINSTALL would look for a *USERNAME=* string in the user's environment. If it was found, then WINSTALL would substitute for *$ENVAR1$* the value following *USERNAME=* in the user's environment. If it was not found, then if WINSTALL encountered the *$ENVAR1$* string within an application .DAT file, it would prompt the user to

enter his USERNAME and WINSTALL would then substitute for that string the information provided by the user.

NOTE: If you enter *SET* from the DOS prompt and press *RETURN*, DOS will list all the current environment variables and their values. So, in this example, if the user had *USERNAME=BWILSON* in his environment, WINSTALL would find it and substitute *BWILSON* for the *$ENVAR1$* string anywhere it is found in an application .DAT file.

If any of the fields on the User-Specific Variables Screen are changed, then a new WINAPPS.LST file will be created when WINSTADM.EXE is exited or the working directory is changed, whether or not the list of installed applications has changed.

Setting the Runtime Options

On the WINSTADM.EXE main screen is a *SET RUNTIME OPTIONS* Button. Clicking this button will bring up the Runtime Options Screen, where you are able to configure the WINSTALL network log, the Files to Copy options, and the WINSTALL Remove option.
If any of the Runtime Options settings are changed, then a new WINAPPS.LST file will be created on exit from WINSTADM.EXE, or when the working directory is changed, whether or not any changes were made to the application list itself.

SETTING THE WINSTALL NETWORK LOG OPTIONS

Log Filename Field

At the top of the screen is the WINSTALL Network Log Filename field. If you wish to enable the network logging feature, then enter in this field the full pathname of the file to which you want WINSTALL to log its activities. Note that all WINSTALL users must have read and write access to this file.

Log Enabled Checkbox

Beneath the Log Filename field is the Network Log Enabled Checkbox. If this box is checked and a valid DOS filename is entered in the Log Filename field, then the WINSTALL Network Log function is enabled.

Enabling/Disabling the Log

When the network log function is enabled, WINSTALL will place an entry in the log each time a user attempts to install or remove an application with WINSTALL, and WINSTADM.EXE will also add an entry each time an administrator makes a change to the WINSTALL configuration.

Items to Log

The WINSTALL Runtime Options Screen contains a listing of all user-specific and environment variables. If you have not entered information for any of these variables in the Set User-specific Variables Screen, then they are all grayed out and disabled. Any variables for which you have entered information appear in the list as the information you have entered, and they are not disabled. Clicking beside one of these to place an X in the corresponding checkbox will instruct WINSTALL to log that variable with each log entry.

Logfile Format

Log entries in general follow this format:

date,time=application installed/removed [error]
variable=value
variable=value

If user-specific variables or environment variables are being logged, then these items will appear on successive lines immediately following the main log entry. Main log entries are separated by blank lines.

The following is a short sample from a WINSTALL network log:

```
03-31-1992,12:20:10=Network Log Enabled
                    USERNAME=Benson

03-31-1992,12:22:16=F:\LANAPPS\WINAPPS.LST MODIFIED
                    Logfile changed from F:\WINLOG\WINSTALL.LOG to F:\WINLOG\
WINSTAL2.LOG
                    USERNAME=Benson

03-31-1992,13:23:50=CHARISMA.DAT REMOVED
                    USERNAME=Benson

03-31-1992,14:46:15=Designer Install ABORTED: File Copy or Delete Failed
                    USERNAME=Williams

03-31-1992,14:48:25=DESIGNER.DAT UPDATED
                    USERNAME=Benson

03-31-1992,14:52:42=Designer INSTALLED
                    USERNAME=Williams

04-02-1992,07:38:38=Working Directory changed from F:\LANAPPS\ to G:\TESTAPPS
                    USERNAME=Benson
```

SETTING THE FILES TO COPY OPTIONS

During most application installations, WINSTALL is instructed to copy certain files from the file server to the local hard disk or to the user's private area on the file server. As WINSTALL begins to copy a file, it checks to make sure that the destination file does not already exist. If WINSTALL finds that the destination file does already exist, it will take one of several courses of action, configurable here, on the Runtime Options Screen. Depending on which radio button you have selected, WINSTALL will take one of the following four actions:

1. If the destination file already exists, do not copy the file.

2. If the source file is older than the destination file, then do not copy the file.

3. If the source file is older than the destination file, then ask the user whether or not to copy the file.

4. Always copy the file.

ENABLING/DISABLING THE WINSTALL REMOVE OPTION

The final option available on the Runtime Options Screen is the *WINSTALL REMOVE OPTION* Checkbox, which enables or disables the *REMOVE APPLICATION* option in WINSTALL.EXE. When a new WINAPPS.LST file is created (on exit from WINSTADM.EXE or when the working directory is changed), the current setting of the *REMOVE OPTION* Checkbox determines whether or not the *REMOVE APPLICATION* option in WINSTALL is enabled or disabled.

When the *REMOVE APPLICATION* option is disabled, WINSTALL users will not see the *REMOVE* Button on their screen and will not be able to use WINSTALL.EXE to remove from their local configurations any of the listed applications.

Creating a New Application Entry or Modifying a Listed Application Entry

To create a new application entry in the list of available applications, simply click on the *CREATE NEW APPLICATION* Button. At this point, WINSTADM.EXE brings up the Add Form, which contains 4 blank data entry fields and eight data entry buttons.

To modify the entry for a listed application, click on the desired application in the list and then click on the *MODIFY LISTED APPLICATION* Button. WINSTADM.EXE will bring up the Modify Form, which is identical to the Add Form, except that the data entry fields will display whatever information was found in the application .DAT file for the application you have chosen to modify.

DATA ENTRY FIELDS

WINSTALL List Name

In the first field on the Add/Modify Form, *WINSTALL LIST NAME,* enter the name of the application exactly as you would have it appear in the WINSTALL application list.

.DAT File Name

The next field, *.DAT FILE NAME,* holds the DOS filename of the file which WINSTALL will use to store the changes needed to install this application. The extension *.DAT* will automatically be appended to the name entered here.

Files to Copy

If a Windows application requires that any files be present on the user's local hard disk, these should be specified in the *FILES TO COPY* fields. Enter in the *SOURCE* field the filename, including drive and full path, of the file to be copied to the local disk. DOS wild cards (* and ?) are supported, but please read the discussion of **Wild Card Deletions** in the **Reference Notes** section of this document for important implications of using wild cards.

If drive and/or path is omitted from the *SOURCE* field, WINSTALL will look only in the WINSTALL directory itself.

If a source file is not found during installation, WINSTALL will inform the user of that fact and abort the installation.

When you add a file to be copied to the list, NAIMAINT will check to see if the source file you have specified is, in fact, found where you have indicated it should be. If the file is not found, NAIMAINT will inform you of that fact, though it will not prevent you from adding it to the list.

Enter in the *DESTINATION* field the drive and path of the local copy of the file that WINSTALL will create at installation time (**NOTE:** omit the filename--WINSTALL will automatically create the destination file with the same name as the source file).

If drive and/or path is omitted from the *DESTINATION* field, WINSTALL will default to the drive where Windows is located and to the Windows directory.

If a specified destination directory is not present on the user's local hard disk, WINSTALL will create it automatically during the installation. During a REMOVE operation, WINSTALL will remove destination directories if they are empty after the files specified in the application .DAT file have been

removed.

If you need to place files in a directory or path that is unique to each user, you can do so by making use of WINSTALL's variable feature. If WINSTALL encounters the codes *$VAR1$*, *$VAR2$*, or *$VAR3$* anywhere in a destination path, then during the installation or removal process, it will substitute for that code whatever information it has obtained from the user. Similarly, if *$ENVAR1$*, *$ENVAR1$*, *$ENVAR3$*, *$ENVAR4$*, or *$ENVAR5$* appear, the appropriate environment variable information will be substituted. If you have a need for information specific to a particular application, WINSTALL provides application-specific variables for that purpose. The codes which instruct WINSTALL to substitute application-specific variable information are *$APPVAR1$*, *$APPVAR2$*, and *$APPVAR3$*. For a complete explanation of this feature, please see the **User-Specific and Environment Variables** section, above, and the **Application-Specific Variables** section, below.

NOTE: If the *DESTINATION* field is left blank (or if you enter *@WINDOWS* as the destination directory), WINSTALL will copy the file(s) to the user's Windows directory (for example, C:\WINDOWS), whatever it may be. If you enter *@SYSTEM* in the *DESTINATION* field, WINSTALL will copy the file(s) to the user's Windows System directory (for example, C:\WINDOWS\SYSTEM), whatever that may be.

The *SOURCE* and *DESTINATION* fields are arranged as two data entry fields, each above a list. The two lists are linked. Once a source and destination have been entered, they will drop together down into their respective lists. To add another pair, click on either data entry field and type in the desired information, or click the *ADD* Button. Once both fields have been completed, they are added to the lists automatically when a TAB or mouse click passes the focus to another object or field on the screen, when the *ADD* Button is clicked, or when the *OK* Button is clicked.

To delete a file from the list, click on either the source or destination list to select the desired pair, then click on the *REMOVE* Button.

To modify an existing source and destination pair, either select the pair by clicking on either one in the list, and then click on the *CHANGE* Button, or double-click on either entry in its list. Either method will remove the pair from their lists and place them in the data entry fields above, where they can be modified and added once again to the lists. As with adding new information, the modified information is added to the lists when a TAB or

mouse click passes the focus to another object or field, when the *ADD* Button is clicked, or when the *OK* Button is clicked.

APPLICATION-SPECIFIC VARIABLES BUTTON

The *APPLICATION-SPECIFIC VARIABLES VIEW/EDIT/LOAD* Button brings up a screen with three data entry fields. The three application-specific variables are analogous to the three user-specific variables, but they also have certain differences.

As with the user-specific variables, what you enter in the data entry fields on this screen will be used as prompts for the user at runtime. And, as with the user-specific variables, WINSTALL will substitute the information provided by the user for certain $VAR codes (in this case, *$APPVAR1$*, *$APPVAR2$*, and *$APPVAR3$*).

On the other hand, WINSTALL will ask the user for this information only if it is found in the .DAT file being used for the current installation or removal. And WINSTALL will not retain the information provided in response beyond the end of the single installation or removal process. Unlike the "global" user-specific and environment variable information, this information is "local" to the single application .DAT file in which it appears.

So, if you need a user's word processing document directory, for example, you could enter in the field for Prompt 1:

Your private word processing document directory

When a user runs WINSTALL to install the application, WINSTALL will present a screen such as the following, asking the user to please enter his private word processing directory. WINSTALL will then substitute the information the user provides in response for any occurrence of the string $APPVAR1$ in the .DAT file.

WIN.INI ADDITIONS BUTTON

When the *WIN.INI ADDITIONS VIEW/EDIT/LOAD* Button is clicked, WINSTADM.EXE displays the WIN.INI Additions Editor. In the editor, you can type in or load any changes to the WIN.INI file which must be applied to a user's configuration to install the application, up to 350 lines total.

NOTE: Different WIN.INI headings, and (sometimes even different lines within the same heading) are manipulated in different ways, according to their function within Windows. For details on how WINSTALL handles these additions, see the **Shared WIN.INI Headings** section, and **Special WIN.INI Heading**s section, below.

If the *VIEW/EDIT/LOAD* Button is clicked during a *MODIFY* operation, or if WIN.INI additions have been entered earlier during an *ADD* operation, the WIN.INI Editor will display the changes and allow their further modification.

This editor is a full ASCII editor with buttons to perform the following functions:

- Load text from a file
- Append text from a file
- Save text to a file
- Cut selected text to the Windows Clipboard
- Paste text from the Clipboard
- Delete all text in the editor
- Delete all text in the editor, except selected text
- Quit without saving changes
- Exit and save the changes in the application .DAT file.

The text you enter in the WIN.INI Additions Editor can have embedded in it the codes for any or all of the three types of variables: *$VAR1$*, *$VAR2$*, *$VAR3$*, *$ENVAR1$*, *$ENVAR2$*, *$ENVAR3$*, *$ENVAR4$*, *$ENVAR5$*, *$APPVAR1$*, *$APPVAR2$*, and/or *$APPVAR3$*. These codes instruct WINSTALL to substitute in their place whatever user-specific information it has obtained from the environment or from the end user at runtime. More information on these options is provided in the **User-Specific and Environment Variables** section, and in the **Application-Specific Variables** section, above.

WINSTALL treats certain headings within the WIN.INI file in a special way. For complete information on which headings these are and what the special treatment is, please see the **SPECIAL WIN.INI Headings** section, below.

SYSTEM.INI ADDITIONS BUTTON

When the *SYSTEM.INI ADDITIONS VIEW/EDIT/LOAD* Button is clicked, WINSTADM.EXE displays the SYSTEM.INI Additions Editor. This editor functions exactly like the WIN.INI Additions Editor, except that any changes you type in or load here will be applied to the user's SYSTEM.INI file when WINSTALL installs the application. The SYSTEM.INI Additions Editor will allow the entry of up to 100 lines of text to be added to the SYSTEM.INI file. The SYSTEM.INI Additions Editor will also substitute user-specific information for the three types of variable codes at runtime.

ICONS TO INSTALL BUTTON

WINSTALL needs three to five pieces of information in order to install an icon on the user's workspace. Pressing the *ICON VIEW/EDIT* Button brings up the Icons Screen, where this information can be entered.

Icon Group Radio Buttons

At the top of the screen is a group of three radio buttons permitting you to select where you want the icons for this application to be installed. By default, the *ADD TO CURRENT (ACTIVE) ICON GROUP* Button is selected. This choice will instruct WINSTALL to place the icons in whatever Program Manager group is active at runtime. The user can then move that icon to another group or leave it where it has been placed.

Selecting the *ADD TO EXISTING ICON GROUP* Button will cause a data entry field to appear beside the radio buttons. In this field, enter the name of the existing icon group where you want the icons placed. If WINSTALL does not find a group by this name in the user's configuration, it will create it.

NOTE: The icon group name is the text which appears in the icon group title bar on the Windows workspace.

To install all the icons for an application in their own separate icon group, click on the *CREATE SEPARATE ICON GROUP* Button and enter the desired group name in the data field which appears beside the radio buttons. WINSTALL will create this icon group in the user's configuration at install time, unless the group already exists, in which case WINSTALL will simply add the icons to this existing group.

NOTE: When WINSTALL is REMOVEing an application, its treatment of the icons varies, depending on which icon group radio button is selected for that application. If the application .DAT file instructs WINSTALL to create a separate icon group, then WINSTALL will delete that entire icon group when it removes the application. Otherwise, WINSTALL will NOT remove any icons; instead, it will conclude the removal process with a message to the user indicating that the icons may still be present in his workspace and providing simple instructions on how the user can remove those icons manually.

Because some Windows applications install more than one icon, the information required for installation of icons is presented in three fields which, like the *SOURCE* and *DESTINATION* Fields for *FILES TO COPY,* are presented as data entry fields above linked lists:

Description

In the *DESCRIPTION* Field, enter the text that you would like to appear below the icon in the user's workspace.

Command Line

In the *COMMAND LINE* Field, enter the drive and full path and filename of the executable file for the application. You may use the $*VAR* codes within this field to instruct WINSTALL to substitute user-specific information at runtime. For further details, see the **User-Specific and Environment Variables** section, and the **Application-Specific Variables** section, above.

Icon File Name

If the icon to be used for the application comes from other than the application's executable file, enter in the *ICON FILE NAME* Field the drive and full path and filename of the file from which to extract the icon. If this field is left blank, WINSTADM.EXE will automatically fill it in with the path and filename of the application's executable file. You may use the *$VAR* codes within this field, too, to instruct WINSTALL to substitute user-specific information at runtime. For further details, see the **User-Specific and Environment Variables** section, and the **Application-Specific Variables** section, above.

Working Directory

Beginning with release 3.1, Windows offers the option of specifying a working directory for each application. This directory is where the application will by default look for and save its documents or data files. When a user runs WINSTALL, the program checks to see what version of Windows the user is running. If the version is 3.1 or above, WINSTALL will install the directory entered in this field as the application's working directory. If the version is 3.0, WINSTALL will ignore any information in this field. As with the *Command Line* and *Icon File Name* fields, you may use the *$VAR* codes within this field to instruct WINSTALL to substitute user-specific information.

As it installs the icon, WINSTALL will check for the existence of the working directory, if one is specified in the application .DAT file. If that directory does not exist, WINSTALL will create it.

During a REMOVE operation, WINSTALL will remove the working directory if it is empty.

The same techniques for adding, removing, and modifying pairs of source filenames and destination directories apply to the sets of icon descriptions, command lines, icon file names, and working directories (see the **Files to Copy** section, above). The only differences are that the icon file name, if left blank, will automatically default to the path for the executable file name, and the working directory may be left blank if desired.

When these fields are complete, clicking the *OK* Button saves the changes and returns the Add/Modify Screen.

When the fields on the Add/Modify Screen have been filled in and the WIN.INI Changes and Icon Information have been entered, click the *OK* Button to create the new application .DAT file and update the list of applications.

PRE/POST OPTIONS BUTTON

When you click on the PRE/POST OPTIONS Button, WINSTADM.EXE presents the Pre/Post Options Screen, with 6 data entry fields for you to specify messages and actions for WINSTALL to perform before and/or after installing or removing applications.

Text Files to Display Fields

The Pre/Post Options screen contains 4 fields for the filenames of ASCII text files to display before installation, after installation, before removal, and/or after removal of the application. Enter in any or all of these fields the full path and filename of ASCII text files to display to the user at the specified point in the installation or removal process. You may choose to display as many or as few of these messages as you like. The only limitations are that every WINSTALL user must have read access to the files you specify, and the files should be no wider than about 60 characters (width limitations may vary due to Windows' proportional fonts) and no longer than 12 lines.

If a text file is to be displayed prior to installation or removal, the user will have a choice of clicking the OK Button to proceed with the operation or a CANCEL Button to back out of the operation.

If a text file is to be displayed at the conclusion of the installation or removal process, the user will have only an OK Button to remove the message from the screen. The conclusion text files take the place of the usual WINSTALL conclusion messages, so you should include in them the information that the installation or removal is complete, plus any directions you want the users to have concerning icon removal or placement, etc.

Programs to Call Fields

WINSTALL offers you the opportunity to extend its functionality by providing the means of calling another program or batch file at the conclusion of any installation or removal process. Simply enter in the appropriate Program to Call data entry field the full path and filename (plus any desired command line parameters), and WINSTALL will call this program at the conclusion of the installation or removal process. This step will take place immediately after the display of the concluding message, if you have configured the .DAT file to display a concluding message.

AUTOEXEC.BAT ADDITIONS BUTTON

The *AUTOEXEC.BAT ADDITIONS VIEW/EDIT/LOAD* Button, like the corresponding button for WIN.INI Additions and SYSTEM.INI Additions, brings up a full function ASCII editor, but this screen has a number of other data fields as well.

AUTOEXEC.BAT Additions Editor

The editor allows the input of up to 25 lines of text for addition to the AUTOEXEC.BAT file. As with the text in the WIN.INI and SYSTEM.INI editors, this text can be typed in at the keyboard, pasted from the clipboard, or loaded from a file.

The Insert Additions Radio Buttons

The placement of the added text within the AUTOEXEC.BAT file is governed by the selection among the Insert Additions Radio Buttons. The added text can be placed at the beginning of the file or at the end of the file or before or after the first line containing a key text string.

To instruct WINSTALL to place the added text at the start or end of the file, click on the appropriate radio button.

To key the placement of the added text to existing text within the AUTOEXEC.BAT file, select either the *BEFORE LINE CONTAINING* or the *AFTER LINE CONTAINING* radio button. When you click to select either of these buttons, more fields will appear on the screen.

The Key Text Field and Alternate Placement Radio Buttons

When either the *BEFORE LINE CONTAINING* or the *AFTER LINE CONTAINING* Button is selected, a Key Text Field will appear to the right of the selected button. Enter in this field the text you want WINSTALL to find in order to properly locate the added lines within the AUTOEXEC.BAT file.

Below the Key Text field are the Alternate Placement Radio Buttons. Select the appropriate one of these three buttons to instruct WINSTALL what to do with the additions if the key text is not found in AUTOEXEC.BAT. You may tell WINSTALL to place the additions at the start of the file or at the end of the file or to omit the additions entirely.

The Boot Drive Radio Buttons

In order to locate the right AUTOEXEC.BAT file, WINSTALL needs to know what drive the end user booted from. The default choice is the C: drive, but you may specify any other drive or instruct WINSTALL to ask the user at install time by clicking on the corresponding button. If you click on *OTHER*, a small data entry field will appear to the right of the word *Other*. Enter in that field the letter for the end user's boot drive.

CONFIG.SYS ADDITIONS BUTTON

If you click on the *CONFIG.SYS ADDITIONS VIEW/EDIT/LOAD* Button, you will bring up yet another ASCII editor, this one identical in appearance and

function to the AUTOEXEC.BAT Additions Editor, except that changes entered here will be applied to the CONFIG.SYS file.

OTHER ADDITIONS BUTTON

Clicking on the OTHER ADDITIONS VIEW/EDIT/LOAD Button brings up an editor very similar to the AUTOEXEC.BAT and CONFIG.SYS Additions Editor, but with one difference. This editor allows you to enter up to 25 lines of text to be added to any ASCII file of your choice, so instead of a selection of Boot Drive Radio Buttons, you are presented with a Text Filename data entry field, where you can enter the full path and filename of the file to apply the additions to, and a checkbox to indicate whether or not you would like WINSTALL to create the file if it is not found.

NOTE: WINSTALL executes the instructions in the Other Additions section of the .DAT file *AFTER* it copies the files specified in the Files to Copy fields. Therefore, you can use this section to modify "generic" text files, including .INI files, that have been copied to the local disk as part of the same application installation process.

Inserting an Application into the List

When you click on the *INSERT APPLICATION INTO LIST* Button, WINSTALL presents a blank data field and a list box of application .DAT files in the Current Working Directory.

This feature enables you to add an existing .DAT file to the list. For example, if an application .DAT file had been created in another directory, it could be copied to the current working directory and it and its application added to WINAPPS.LST by using the *INSERT APPLICATION INTO LIST* Button. The text entered into the first field will appear in the *AVAILABLE APPLICATIONS* listing in WINSTALL (and also in the WINSTADM.EXE program), while the .DAT file selected from the list will provide WINSTALL with the information it needs to install or remove that application from the user's configuration.

SUGGESTIONS:

WINSTALL Configuration

Aleph Systems

You might consider running several copies of WINSTALL on the same file server. (This is a no-extra-cost configuration option, because WINSTALL is licensed by file server, not by the number of copies in use or by the number of nodes or users). For example, you might want to make certain applications available to everyone but other applications available only to certain users. To accomplish this, you could create two directories containing WINSTALL, each with a different WINAPPS.LST appropriate to the groups of users having access to it.

Using two directories, two copies of WINSTALL.EXE, and two separate WINAPPS.LST files would also allow you to control which applications users can remove and which they cannot remove; you would simply enable the *REMOVE APPLICATION* option in one WINAPPS.LST file but not in the other.

.DAT File Creation

Lately, some new Windows programs have included a file which describes all changes made by the installation process. These files make creating a .DAT file very easy.

In absence of such a file, it may be easiest to create a brand new Windows configuration on a drive which does not have one (or on a PC which does not have one, if you have the luxury of being able to reserve a PC for this type of work), and install the application onto the file server from that configuration. If you make a backup of the WIN.INI, SYSTEM.INI, AUTOEXEC.BAT, and CONFIG.SYS files and a printout of the WINDOWS, SYSTEM, and ROOT directories before installing the application, you can quickly and easily identify any changes by comparing them with what you find after the installation. Make a point of checking the file dates, especially in the SYSTEM directory; more and more applications are now updating during their installation processes some of the .DLL files which are shared by multiple applications.

For changes to the WIN.INI, SYSTEM.INI, AUTOEXEC.BAT, CONFIG.SYS, and other ASCII files, use the corresponding Additions Editors built into WINSTADM.EXE. An easy way to pull these changes into a .DAT file without retyping is to follow this procedure:

1. Run WINSTADM.EXE and select *CREATE NEW APPLICATION*.

2. At the Add Form, click on the *WIN.INI ADDITIONS VIEW/EDIT/LOAD* Button to enter the WIN.INI Additions Editor.

3. Click on the *LOAD FROM FILE* Button, and specify the WIN.INI file from the configuration used to load the application to the file server.

4. Once the WIN.INI file is loaded, select only the text which was added during the installation of the application.

5. Click on the *DELETE ALL EXCEPT SELECTED* Button, and you will have in the editor only those changes applicable to the application you are adding. (The original WIN.INI file is not modified).

This same procedure will work equally well for any changes to the SYSTEM.INI, AUTOEXEC.BAT, CONFIG.SYS, and OTHER files, using the corresponding *ADDITIONS VIEW/EDIT/LOAD* Button from the Add Form.

Application Updates

When a new version of an application you are using is released, add the new version to WINAPPS.LST, but leave the old version in place as well. (You can distinguish the new version by adding a version number to the end of the List Name, and by using a different .DAT filename). Instruct your users to use WINSTALL to remove the old version and then to install the new version. After you are satisfied that everyone has upgraded, remove the old version from WINAPPS.LST and from the file server.

If the upgrade would best be served by preserving some of the files or other data from the old version rather than wiping it all out and starting over, you can edit the old version's .DAT file and remove from it the information you want to remain after the REMOVE operation. If it is not listed in the .DAT file, then WINSTALL will not remove it.

Initial End User Windows Configurations

You can configure the Windows setup program to automatically install the WINSTALL icon in the Main program group (or any other program group) of every Windows installation. That way, you can let end users finish the job themselves by installing whatever Windows applications they need through WINSTALL

It is also very quick and easy to install local copies of Windows from a file server. Consult the Windows documentation for instructions on how to set Windows up to best allow this type of installation.

Three easy steps enable the Windows setup program to automatically install WINSTALL to each new Windows installation:

1. Copy the files VBRUN100.DLL and DISKSTAT.DLL into the directory on the file server, or onto one of the diskettes, from which you will install Windows. If you are going to be installing Windows from diskettes, note the number of the diskette to which you have copied these two .DLL files. If they do not fit onto any of the Windows diskettes, it will be simplest if you copy them manually into each Windows directory immediately after the Windows setup program completes.

2. In the same directory on the file server, (or on the first diskette, if you are installing Windows from diskettes) open the file SETUP.INF with the Windows Notepad or any other ordinary text editor.

   If you have copied the files VBRUN100.DLL and DISKSTAT.DLL to the Windows directory on the file server or to the first diskette, add the following line to the [win.apps] section:

   1:VBRUN100.DLL, "Visual Basic Runtime Module"
   1:DISKSTAT.DLL, "Art Krumsee's VB Extension"

   The first character in this line (the number 1 in the example above) specifies on which Windows diskette the setup program can expect to find the .DLL file. If you are installing Windows from a file server, diskettes will not be involved, so you can just use the number 1. If you will be installing Windows on each network node from diskettes, this number should correspond to the number of the diskette to which you have copied the .DLL file. In either case, these lines will assure that the required .DLL files are automatically copied into the Windows directory of each new Windows installation. If you have been unable to copy the .DLL files to one of the Windows diskettes, do not add this line.

3. Finally, add to the [Main] section, the line which will create the WINSTALL icon in the Program Manager *Main* Group (or in whatever group you want the WINSTALL icon to appear):

"WINSTALL",F:\NETAPPS\WINSTALL.EXE

For the path *F:\NETAPPS\,* substitute the drive and directory where you have installed WINSTALL and the application .DAT files.

Installing Windows Applications to Run Locally

Heavy network use and a need for better performance may encourage you to load Windows applications on each PC's local hard disk. WINSTALL can make quick work of this type of installation, too, saving many hours of tedious support effort. The same procedures suggested for installing Windows applications on a network drive and running them from networked PCs will work for installing a separate copy of each application on each PC. The .DAT file will differ, of course, in what files are specified to be copied to the local disk, and in the path to the executable file and icon, but the principles remain the same. Obviously, you will be copying many more files, so each installation will take a bit longer than if the applications are to be run from the file server. In either situation, though, WINSTALL can enable all your end users to do the job themselves--quickly and easily, without the possibility of confusion.

REFERENCE NOTES:

Backup Files

WIN.INI

When a user runs WINSTALL, the program makes a backup copy of the WIN.INI file, named WININI.TMP. When the program concludes, if it has changed the WIN.INI file, it renames WININI.TMP to WININI.BAK (first deleting any existing file named WININI.BAK). If it has not changed the WIN.INI file, it deletes WININI.TMP.

AUTOEXEC.BAT, CONFIG.SYS, and Other ASCII Files

In each case, if WINSTALL modifies one of these files, it preserves the original with a .BAK extension, deleting in the process any earlier .BAK files if they exist.

Application .DAT Files

If a new application .DAT file is created as a result of modifications to an existing .DAT file, the original .DAT file is preserved with a .BAK extension. (Again, if an earlier .BAK file exists, it is deleted to allow the current version to use that file name).

WINAPPS.LST

When the file WINAPPS.LST is updated, the previous version is retained with a .BAK extension. As with WIN.INI and the Application .DAT files, only the last version of WINAPPS.LST is preserved as WINAPPS.BAK.

File Formats

Formats for all data files associated with WINSTALL.EXE and WINSTADM.EXE are fully described here. All of these files are flat ASCII files, so they can be edited manually with the Windows Notepad, or any other ordinary text editor. However, the information provided here is intended for reference only; these files are created and maintained by the WINSTADM.EXE program (except for NAI.INI, which is created and maintained automatically by the WINSTALL.EXE program) and under normal circumstances should require no manual editing whatsoever. Because of the many parameters they contain and the intricate nature of the modifications which WINSTALL makes to a user's configuration, it is strongly recommended that these files be manually edited only in an emergency and even then only by someone who well understands how WINSTALL works.

WINAPPS.LST

```
;================================================
;           FILE WINAPPS.LST
; WINDOWS APPLICATIONS LIST FOR WINSTALL rel. 2.0
;================================================
;
; WINSTALL Runtime Options:
REMOVE=ENABLED
NETLOG=ENABLED
LOGFILE=F:\WINSTLOG\WINSTALL.LOG
LOGITEMS=VAR1 VAR2 ENV1 ENV3
COPYFILES=ALWAYS

; User-specific information:
$VAR1=First Name
```

```
$VAR2=Last Name
$VAR3=
$ENVAR1=USERNAME
$ENVAR2=GROUPNAME
$ENVAR3=NODENAME
$ENVAR4=
$ENVAR5=

; Format for application list is:

; Application List Name|Application .DAT file name

Charisma|CHARISMA
CorelDraw|CORELDRW
Designer 3.1|DESIGNER
Excel 4.0|EXCEL
Instant Org Chart|INSTORG
Milestones, Etc.|MILES3
Packrat|PACKRAT
Powerpnt|POWERPNT
Project for Windows|PROJECT
Word for Windows 2.0|WFW

;=================================================

; WINSTALL 2.0 (Network Application Installer for Windows)

;    (c) Copyright 1991, 1992 by Aleph Systems
;               All rights reserved.

;     7319 Willow Avenue, Takoma Park, MD 20912

;                   (301)270-4458
```

```
;================================================
; End of file WINAPPS.LST
;================================================
```

Notes:

Blank lines and lines beginning with ; are ignored.

Before the list of applications, the runtime options lines must appear. The first two options are *REMOVE=* and *NETLOG=*. The only recognized values for these two options are *ENABLED* and *DISABLED*. The *REMOVE=* setting determines whether or not the *REMOVE* Button appears in the WINSTALL program, enabling users to remove listed applications from their local configurations. The *NETLOG=* setting determines whether or not the WINSTALL network log is enabled.

The *LOGITEMS=* option lists all the global user-specific variables and environment variables which are to be logged to the WINSTALL network log. User-specific variables are here specified as *VARn*, where n is the number of the variable (between 1 and 3). Environment variables are specified as *ENVn*, where n is the number of the variable (between 1 and 5). The items are separated by spaces.

The *COPYFILES=* option determines how WINSTALL is to handle copy operations where the destination file already exists. The value must be one of the following:

NODESTINATION (copy only if the destination file does NOT already exist)

OLDDESTINATION (if the destination file is newer than the source file, then do not copy)

PROMPT (if the destination file is newer than the source file, then ask the user)

ALWAYS (always do the copy)

User-specific variables are indicated in the WINAPPS.LST file by the *$VAR1=*, *$VAR2=*, and *$VAR3=* lines. If the line is empty following the "=", then the variable is not in use. If a variable is in use, then the text which follows the "=" will be used by the WINSTALL.EXE program to prompt the user for the

specific information needed at installation time.

Environment variables are indicated in the WINAPPS.LST file by the *$ENVAR1=*, *$ENVAR2=*, *$ENVAR3=*, *$ENVAR4=*, and *$ENVAR5=* lines. If the line is empty following the "=", then the variable is not in use. If a variable is in use, then the text which follows the "=" will be used by the WINSTALL.EXE program to search the user's environment for the specific information needed at installation time. If the text is not found in the environment, then WINSTALL will prompt the user for it.

In the application list section of WINAPPS.LST, each line represents one application, with the display name of the application (what the user sees in the list of available applications) first, followed by a vertical bar and then the filename (no path, no extension) of the data file containing the information needed to install the program.

Application .DAT Files

```
;====================================================================
; File DESIGNER.DAT (WINSTALL rel. 2.0 .DAT file for DESIGNER 3.1)
;====================================================================

; Application-Specific Variables:
@APPVARS
$APPVAR1=
$APPVAR2=
$APPVAR3=
PREINSTALL=
POSTINSTALL=
PREREMOVE=F:\WINSTALL\PREREM.TXT
POSTREMOVE=
INSTALLSHELL=
REMOVESHELL=

@WIN.INI
[Extensions]
drw=designer.exe ^.drw
shw=mgxslide.exe ^.shw

[Micrografx]
Designer 3.1=F:\DESIGNER
Libraries 4.0=F:\MGXLIBS
CLIPART=F:\MGXCLIP
Libraries=F:\MGXLIBS

[Clipboard Formats]
MGX_DRAW=0,"Micrografx Picture"
MGX_PICT=0,"In*a*Vision or Windows Draw Picture"

[Micrografx Outline Fonts]
BSBEZIER=F:\MGXLIBS\BITFONTS,MGXBITBZ,BSBEZ.FTM,0,15
BSSPEEDO=F:\MGXLIBS\SPDFONTS,MGXBITSP,BSSPD.FTM,0,20
URWBEZIER=F:\MGXLIBS\URWFONTS,MGXURWBZ,URWBEZ.FTM,0,6
```

```
[MGXTIFF]
Format=64
Device=1
Compression=2

[MGX_DRW_OUTPUT]
CPI=480

[CGMIN]
CPI=4052
TypeOfCGM=Harvard Graphics
ShowProfile=0

[Designer 3.1]
Warning=1
AutoPaste=0
Format=64
AutoPaste=0

@SYSTEM.INI

@AUTOEXEC.BAT
BOOT=C
PLACEMARKER=
KEY=SMARTDRV
ALTPLACEMARKER=
[START AUTOEXEC.BAT]
```

```
@CONFIG.SYS
BOOT=C
PLACEMARKER=
KEY=
ALTPLACEMARKER=
[START CONFIG.SYS]

@FILES
F:\DESIGNER\DESIGNER.INI @WINDOWS

@OTHER
FILE=
CREATE=ENABLED
PLACEMARKER=
KEY=
ALTPLACEMARKER=
[START OTHER]

@ICONS
XGROUP=Main
F:\DESIGNER\DESIGNER.EXE,Designer,F:\DESIGNER\DESIGNER.EXE,,,,F:\$VAR1$\DATA

;=====================================================

; WINSTALL 2.0: Network Application Installer for Windows

;   (c) Copyright 1991, 1992 by Aleph Systems
;              All rights reserved.

;   7319 Willow Avenue, Takoma Park, MD 20912

;                  (301)270-4458

;=====================================================
; End of File DESIGNER.DAT
;=====================================================
```

Notes:

Blank lines and lines beginning with ; are ignored.
Eight headings appear in the .DAT file , and they should appear in this order:

@APPVARS
@WIN.INI
@SYSTEM.INI
@AUTOEXEC.BAT
@CONFIG.SYS
@FILES
@OTHER
@ICONS

Beneath the *@APPVARS* heading are listed the following items:

$APPVAR1=
$APPVAR2=

$APPVAR3=
PREINSTALL=
POSTINSTALL=
PREREMOVE=
POSTREMOVE=

INSTALLSHELL=
REMOVESHELL=

Each item will be listed, but if no text follows the equal sign, then the feature or variable will go unused.

The *$APPVARn* keys specify the application-specific variables to be used with the installation and removal of this application. Any text following the equal sign will be used to prompt the user for the information needed. If nothing follows the equal sign, then the variable is not in use, and the appearance of its $VAR code later in the .DAT file will cause an installation or removal failure.

Text following the equal sign on the *PREINSTALL* line specifies the full path and filename of an ASCII text file to display at the start of the installation process. The text following the equal sign on the P*OSTINSTALL* line specifies a text file to display following the installation process. The *PREREMOVE* and *POSTREMOVE* lines fulfill the same function for the removal process.

Text following the equal sign on the *INSTALLSHELL* and *REMOVESHELL* lines specifies the full path and filename of an executable file (.BAT, .COM, or .EXE) to be called by WINSTALL.EXE at the conclusion of the installation or removal process.

Any changes to the user's WIN.INI file must appear below the *@WIN.INI* heading, exactly as they are to appear in the user's WIN.INI file. If there are no changes to WIN.INI, the heading must still appear, but it would be followed on the next uncommented line by the *@SYSTEM.INI* heading.

Any changes to the user's SYSTEM.INI file must appear below the *@SYSTEM.INI* heading, exactly as they are to appear in the user's SYSTEM.INI file. If there are no changes to SYSTEM.INI, the heading should still appear, but it would be followed on the next uncommented line by the *@AUTOEXEC.BAT* heading.

The next heading, *@AUTOEXEC.BAT*, has four key values and a subheading indicating the beginning of the actual changes to be installed.

The *BOOT*= line specifies the user's boot drive. This value can be any letter of the alphabet or *PROMPT*, the latter meaning that WINSTALL.EXE should ask the user for his boot drive.

The *PLACEMARKER=* line specifies the location within the AUTOEXEC.BAT file where the changes should be placed. This value can be *START*, *END*, *BEFORE*, or *AFTER*. *START* will place the additional text at the very beginning of the file; *END* will place it at the very end. *BEFORE* and *AFTER* instruct WINSTALL.EXE to key the placement either before or after the first line containing the key text, specified after the equal sign on the *KEY=* line.

The *ALTPLACEMARKER=* line instructs WINSTALL.EXE as to where the changes should go if the *PLACEMARKER=* line is set to either *BEFORE* or *AFTER* and the key text is not found in the AUTOEXEC.BAT file. Valid entries for the *ALTPLACEMARKER=* line are *START*, *END*, and *OMIT*.

Following the *ALTPLACEMARKER=* line is the subheading, *[START AUTOEXEC.BAT]*. Any additions to the AUTOEXEC.BAT file will follow this subheading, exactly as they are to appear in the user's AUTOEXEC.BAT file after installation. $VAR codes are valid within this text.

If the installation will make no changes to AUTOEXEC.BAT, these lines should all still appear, but no text would follow the equal signs, and no text would follow the *[START AUTOEXEC.BAT]* subheading. The subheading would be followed on the next uncommented line by the *@CONFIG.SYS* heading.

The format of the *@CONFIG.SYS* section is identical to that of the *@AUTOEXEC.BAT* section. This section describes changes to be made to the user's CONFIG.SYS file. The four key values which appear here are the same as those for the *@AUTOEXEC.BAT* section, and they permit the same valid entries for each. All of the actual changes to be made to the user's CONFIG.SYS file appear following the *[START CONFIG.SYS]* subheading. $VAR codes are valid within this text.

Any files which must be copied to the user's local hard disk must appear below the *@FILES* heading, in the following format:

sourcefile destinationdirectory

DOS wildcards (* and ?) are OK, but please see the discussion of **Wildcard Deletions** later in the **Reference Notes** section. The *sourcefile* must include drive and path. The *destinationdirectory* assumes the drive where Windows was started, unless another drive is specified in the path. Any directory in the *destinationdirectory* will automatically be created on the

user's local drive, if necessary.

If the *destinationdirectory* is omitted, or if *@WINDOWS* appears in its position, the file(s) will be copied to the user's Windows directory (where his WIN.INI file is found--usually C:\WINDOWS). If *@SYSTEM* appears in the *destinationdirectory* position, then the file(s) will be copied to his Windows\System directory, wherever that may be (usually C:\WINDOWS\SYSTEM).

If you need to place a user-specific directory into the *destinationdirectory*, WINSTALL provides full support for user-specific, environment, and application-specific variables. Within the *destinationdirectory*, you would insert the appropriate $VAR code, and WINSTALL will substitute at runtime the corresponding variable information. See the **User-Specific and Environment Variables** section, and the **Application-Specific Variables** section, above, for complete details.

If no files need to be copied, the heading must still appear, but it would be followed on the next uncommented line by the *@OTHER* heading. The *@OTHER* section describes changes to be made to any ASCII file in the end user's configuration, including files just copied there by the instructions in the *@FILES* section. This section has five key values and a subheading. The first key line, *FILE=*, contains the full path and filename of the ASCII file to modify. The second key, *CREATE=*, instructs WINSTALL as to whether or not to create this file if it is not found. Valid entries are *ENABLED* (meaning to create the file if not found) and *DISABLED* (meaning to ignore the *@OTHER* section entirely if the file is not found).

The other three keys, *PLACEMARKER=*, *KEY=*, and *ALTPLACEMARKER=*, are identical to the corresponding keys in the *@AUTOEXEC.BAT* and *@CONFIG.SYS* sections in both their function and their acceptable values.

The *[START OTHER]* subheading precedes the actual additions to be made to the specified ASCII file, which appear here exactly as they are to be entered in the user's file. $VAR codes are valid within this text.

The *@ICONS* section follows the *@OTHER* section. Information for each icon to be created in the user's workspace must follow the *@ICONS* heading. If the icons are to go in a separate icon group, then the name of that group should appear in the first uncommented line beneath the *@ICONS* heading, in the following format:

GROUP=groupname

If the icons are to go into an existing icon group, then the name of that group should appear in the first uncommented line beneath the *@ICONS* heading, in the following format:

XGROUP=groupname

The *GROUP=* or XGROUP= line is optional. If it is omitted, WINSTALL will install the icon(s) in the icon group which is current (active) at runtime.

Each icon to be placed must have a line in this final section of the .DAT file. The format for the icon information is as follows:

executablefilename,displayname,iconpath,,,,workingdirectory

The *executablefilename* should include a full, specific path to the program file to be executed when the icon is double-clicked. $VAR codes are valid within the *executablefilename*.

The *displayname* is whatever should appear beneath the icon in the user's workspace.

The *iconpath* indicates the file from which to draw the icon; it may be different from the executable file. $VAR codes are valid within the *iconpath*. Only the comma should separate the items (no spaces or tabs).

The *workingdirectory* specifies the default data directory for the application (in version 3.1 and above of Windows--if the end user is running an earlier version of Windows, the *workingdirectory* information is ignored). $VAR codes are valid within the *workingdirectory*. (**NOTE:** the *workingdirectory* must be preceded by exactly 4 commas) .

NOTE: If any of these items are omitted, it is important that the commas appear anyway. Only the commas should separate the items (no spaces or tabs).
At installation time, WINSTALL will create the *workingdirectory* on the user's disk if it does not already exist. If WINSTALL is used to remove the application, it will remove the *workingdirectory* from the user's disk if that directory is empty.

The group specified in the *GROUP=* line will be automatically created by WINSTALL, and all icons specified will be created in this group. If a group is specified in the X*GROUP=* line instead of a *GROUP=* line, that group will be automatically created by WINSTALL if it does not already exist, and all icons specified will be created in this group. If no *GROUP=* or *XGROUP=* line appears, the icons will be created in the group which is current (active) in the user's Windows workspace at the time when the installation is performed. While it is technically not necessary to include any icon information (the *@ICONS* heading must still appear in the .DAT file, though), the user would have no way to run the program after installation if no icon was installed.

WINSTADM.CFG

```
;=====================================================
; File WINSTADM.CFG (WINSTADM.EXE Config file)

; Comments begin with '

; Blank lines are ignored
;=====================================================
; File WINSTADM.CFG (WINSTADM.EXE rel 2.0 Config file)

; Comments begin with ;

; Blank lines are ignored
;=====================================================
DATDIR=F:\WINSTALL\
;=====================================================

; WINSTALL 2.0: Network Application Installer for Windows

;   (c) Copyright 1991, 1992 by Aleph Systems
;              All rights reserved.

;   7319 Willow Avenue, Takoma Park, MD 20912

;                   (301)270-4458

;=====================================================
; End of File WINSTADM.CFG
;=====================================================
```

Notes:

Blank lines and lines beginning with ; are ignored.

The only valid configuration option is the location of the working directory, specified in the *DATDIR=* line. All other lines must either be blank or begin

with a semicolon.

NAI.INI

```
[Programs]
Designer 2.0=10:03:23,01/07/92
Excel 3.0=14:02:45,02/14/92
PowerPoint=14:03:15,02/14/92
Word for Windows 2.0=14:04:00,02/14/92
```

Notes:

Blank lines and lines beginning with ; are ignored.

This file is a standard format Windows application initialization file. In the current release, the only valid heading is *[Programs]*. Each line beneath that heading has to the left of the equal sign the Application List Name of the application installed by WINSTALL To the right of the equal sign is the time of installation, followed by a comma and the date of installation.

In the current release, this file is used only during the WINSTALL Install Application function, to determine whether or not an application has been installed previously. Later releases will employ this file for the purpose of automatically determining when an application upgrade is needed.

Icon Groups

If an application .DAT file specifies that a separate icon group be created, then the *REMOVE* operation of WINSTALL will automatically remove that group and all its icons from the user's workspace. If the .DAT file specifies installing the icons into an existing icon group or into the icon group which is current (active) at runtime, then the *REMOVE* operation will make no attempt to remove any icons. Instead, when the rest of the removal process is complete, WINSTALL will inform the user that the icon(s) may remain in the workspace and may have to be removed manually. Simple instructions for removing an icon are included in this message.

Note that during the installation of an application, the CREATE SEPARATE ICON GROUP and ADD TO EXISTING ICON GROUP work the same way. That is, if the icon group does not exist, WINSTALL will create it and install the icons in it. On the other hand, if the specified icon group *does* exist, then WINSTALL will simply add the icons to those already in the specified group.

The difference between these two choices arises only when WINSTALL removes the application. The *REMOVE* operation for any application specifying CREATE SEPARATE ICON GROUP will remove the entire icon group and all the icons it contains, whether they are part of the application being removed or not. In contrast, the *REMOVE* operation for an application specifying ADD TO EXISTING ICON GROUP will remove no icons or icon groups. Instead, it will behave exactly the same as the ADD TO CURRENT (ACTIVE) ICON GROUP option: it will inform the user that the icons may have to be removed manually, and it will at the same time provide some brief instructions on how to do so.

Reinstallation Check

When a user clicks the *INSTALL APPLICATION* Button in WINSTALL, the program checks the user's local configuration to determine whether or not the selected application is already installed. WINSTALL keeps in the user's Windows directory the file NAI.INI, a log of all applications it has installed and the date and time it installed them. If WINSTALL does not find an application listed as having been installed in the NAI.INI file--and does not encounter a match between the [Extensions] heading in the WIN.INI additions section of the application .DAT file and the user's WIN.INI file, it assumes that the application has not been installed. If WINSTALL believes that the application is currently installed, it will give the user the option of aborting the installation or reinstalling the application.

Because of idiosyncrasies in various Windows applications and the way that they share files and WIN.INI headings, WINSTALL assumes that an application is NOT currently installed unless it finds the application listed as having been installed in the NAI.INI file or it finds in the WIN.INI [Extensions] section the extensions specified in the application .DAT file. If neither of these conditions exists, WINSTALL will assume that the application is not installed.

If WINSTALL reinstalls an application (whether or not it is aware that it is reinstalling), the affected WIN.INI sections will be entirely rewritten according to the information in the .DAT file, and the files specified in the .DAT file will replace any by the same name which are already located on the user's local disk, and new icons will be added to the user's workspace.

Shared WIN.INI Headings

Certain applications share WIN.INI headings with other applications. For example, a number of Microsoft applications use the [Microsoft Help] heading in WIN.INI, while a number of applications from Micrografx, Inc. share the [Micrografx] heading. If a user has installed in his configuration two applications which share a heading, this heading may cause temporary problems if WINSTALL is used to remove one of the two applications.

In such a case, the remaining application will find that the heading it needs is missing from WIN.INI. If you suspect that WINSTALL has removed a shared WIN.INI heading, try reinstalling the remaining application. The reinstallation will replace the missing heading and the remaining application should then work as before.

Special WIN.INI Headings

For most WIN.INI headings, WINSTALL treats the entire heading and all the lines beneath it as a single unit, installing it or removing it whole cloth. However, WINSTALL treats five WIN.INI headings in a special fashion. The *[Windows]*, *[Extensions]*, *[Embedding]*, *[OLE]*, and *[Fonts]* headings are all known to be shared. Therefore, WINSTALL inserts or deletes from these headings only the information which the application .DAT file specifies belongs under them. The result is that for these five headings, each application .DAT file will affect the only the lines it specifies; all other lines beneath these headings will remain untouched.

In addition, two lines within the *[Windows]* heading are handled in a unique fashion. The *RUN=* and *LOAD=* lines are never removed or replaced. Instead, if an application .DAT file specifies text to be inserted in one of these two lines, the WINSTALL install application process will simply add it to what is already there (if anything); likewise, the remove application process will remove from these lines only the text specified in the application .DAT file. All other text on these lines will be left intact.

SYSTEM.INI Headings

For SYSTEM.INI headings, WINSTALL inserts or deletes only the information which the application .DAT file specifies belongs under them. The result is

that for the SYSTEM.INI file, each application .DAT file will affect the only the individual lines it specifies; all other lines will remain untouched.

Wild Card Deletions

WINSTALL supports the use of DOS wildcards (* and ?) in the specification of files to be copied. However, this support has important safety implications for WINSTALL's application removal feature. WINSTALL includes special safeguards to prevent the application removal feature from inadvertently causing damage to a user's Windows configuration or other areas of his local PC. Nevertheless, the application removal feature is very powerful, and it is important for a LAN administrator to understand how it operates in order to be certain that it works as intended. Remember that the application removal feature can be disabled from within the WINSTADM.EXE program.

If a user clicks on the *REMOVE APPLICATION* Button in WINSTALL, the program will attempt to remove from the local configuration whatever files the application .DAT file specifies as required to be copied to the local configuration during installation of the application. If the source files are specified by complete filenames, they are simply deleted one by one, by complete filename. This method is completely safe. However, if these files are specified by means of wildcards, WINSTALL follows a cautious procedure for their removal, to avoid removing any files other than those intended:

1. WINSTALL first looks for the individual filenames in the specified source directory on the file server. If files matching the file specification are found in the source directory, then WINSTALL uses these names in the removal process: it deletes each of these filenames one by one from the user's local configuration. In other words, if the .DAT file specifies F:\FONTS*.FNT as the source file and C:\FONTS\ as the destination directory, WINSTALL will look in the F:\FONTS\ directory for files with the .FNT extension. If it finds, say, FONT1.FNT, FONT2.FNT, and FONT3.FNT, then it will perform three separate deletions on the local drive, equivalent to the following series of DOS commands: DEL C:\FONTS\FONT1.FNT, DEL C:\FONTS\FONT2.FNT, and DEL C:\FONTS\FONT3.FNT.
2. With certain exceptions for reasons of safety (see #3, below), if the file server does not contain any files matching the source file specification in the .DAT file, then WINSTALL performs a wild card deletion of files matching the specification on the local configuration. For example, if the source file specification reads F:\FONTS*.FNT but the F: drive holds no

files with the .FNT extension in the \FONTS\ directory (perhaps because the application has been removed from the file server), then WINSTALL will perform on the local drive the equivalent of the following DOS command: DEL \FONTS*.FNT.

3. *If the destination directory is the WINDOWS or SYSTEM directory or the ROOT directory of any drive, then WINSTALL will not delete files specified in the .DAT file by the use of wildcards unless it can discover the individual filenames from the file server, as discussed in #1, above.* In other words, **if** the .DAT file specifies files by the wildcard method **and** the source files are not found in the source directory **and** the destination for those files is the WINDOWS directory, or the SYSTEM directory, or a ROOT directory, **then** WINSTALL will **not** delete the files.

WINSTALL Capacities and Limitations

The WINSTALL program includes the following limitations:

Maximum applications per WINAPPS.LST file	100
Maximum WIN.INI additions per application .DAT file	350 lines
Maximum separate WIN.INI headings per application .DAT file	20
Maximum WIN.INI [Windows] additions per application .DAT file	5 lines
Maximum WIN.INI [Extensions] additions per application .DAT file	50 lines
Maximum WIN.INI [Embedding] additions per application .DAT file	50 lines
Maximum WIN.INI [OLE] additions per application .DAT file	50 lines
Maximum WIN.INI [Fonts] additions per application .DAT file	75 lines
Maximum SYSTEM.INI additions per application .DAT file	100 lines
Maximum AUTOEXEC.BAT additions per application .DAT file	25 lines

Maximum CONFIG.SYS additions per application .DAT file lines	25
Maximum OTHER ASCII file additions per application .DAT file	25 lines
Maximum number of icons per application .DAT file	20

LICENSE AGREEMENT

Disclaimer

This program is provided without any express or implied warranties whatsoever. Because of the diversity of conditions and hardware under which this program may be used, no warranty of fitness for a particular purpose is offered. The user is advised to test the program thoroughly before relying on it. The user must assume the entire risk of using the program. The manufacturer assumes no liability of any kind.

Licensing Information

This copyrighted program is NOT free. This is an evaluation copy: you are permitted to ***test*** it in your network to see if it will prove useful to you. ***If you do actually put it into use***, you are legally obligated to license your copy from the manufacturer, Aleph Systems. You are encouraged to pass copies of this evaluation program along to others for their evaluation, but please include all of the files, and please encourage others to license their copies as well. This program can save you enormous amounts of time, effort, and money: if you use WINSTALL to install even one application on most of the nodes in an average sized network, you will have saved enough time to easily cover the expense of a license.

An invoice/order form is provided for your convenience.

To keep licensing as simple and equitable as possible, this program is licensed on a per-file server basis. For the purposes of this license, a server is defined as any computer from which the program is made available to be run. In other words, even if you are not running WINSTALL from a dedicated file server (for example, in a peer-to-peer network which has no dedicated file servers), any computer from which the WINSTALL program is made available is considered a file server and requires its own WINSTALL license. You may elect to run multiple copies of this program on a single server; in this case, you still need only a single license. However, if you install a copy of this program on a second file server, you are obligated to purchase an additional license to run WINSTALL on that second file server.

For pricing information on licenses for 10 servers or more or on upgrades

from previous releases, please contact Aleph Systems directly, by mail at 7319 Willow Avenue, Takoma Park, MD 20912, by CompuServe mail at 71371,635, or by telephone at (301)270-4458.

Anyone distributing this program for any kind of remuneration must first contact Aleph Systems for authorization.