A Web Link can be added to the Windows 95/Windows NT Start menu by creating a .URL file and creating a link from the Start menu.

The following example N_Dist script adds links to the Norman web sites on the Start menu.

The technique used is to store the details of each URL file to be built in an INI-file and create the links using the REGISTER command.   The links will appear as a group named 'Web Links' under the programs group or optionally, on the primary Start menu.

The script would be invoked from the server login script with the command:

```
\\<Server_Name>\<Share>\ndist\N_Dist.exe \\<Server_Name>\<Share>\ndist\url.nxd $Profile=
%USERPROFILE% /q
```

This example reflects a Windows NT, Windows 95/Windows NT server/client environment.   It is assumed that an Administrator install of **Norman Virus Control** has been performed, consequently the directory NVCADMIN and its subdirectories, including NDIST, exist on the server.

The variable $Profile is set on the command line for use when a Windows NT workstation logs on.   See the topic Using Environment variables for further details on setting/using variables.

The INI-file and the N_Dist script are located in the NVCADMIN\NDIST directory on the server.

The INI-file is in the following format:

URL.INI

```
[URLS]
nURL=<URL>
nText=<Text to appear on the Start menu>
nFile=<URL filename>
```

Where 'n' is a sequential number starting at 1 for each URL definition set.

A typical example would be:

```
[URLS]

1URL=http://www.norman.com.au
1Text=Norman on the Web (Australia)
1File=NRM061.URL

2URL=http://www.norman.de
2Text=Norman on the Web (Germany)
2File=NRM049.URL

3URL=http://www.norman.nl
3Text=Norman on the Web (Netherlands)
3File=NRM031.URL

4URL=http://www.norman.no
4Text=Norman on the Web (Norway)
4File=NRM047.URL

5URL=http://www.norman.com
5Text=Norman on the Web (USA)
5File=NRM001.URL
```

The script recursively reads the INI-file and creates as many URL files and links as the are definition sets.

URL.NXD

```
NXD BEGIN

; Set script variables
   set $Source='\\<Server>\<Share>\NDIST'
   set $Target='C:\MY DOCUMENTS\URL'
   set $Folder='Web Links'
   set $Count=1
   set $Move='Yes'

; Check workstation OS
#CheckOS
   if $System='WIN16'
```

```
        goto #End
     if $System='WIN95'
        goto #Win95
     if $System='WINNT'
        goto #WinNT

  ; Set Win 95 variable
  #Win95
     set $URL_Dir=$WinDir+'\START MENU\Web Links'
     goto #MakeDir

  ;Set Win NT variables
  #WinNT
     set $WininiDir=$Profile
     set $URL_Dir=$Profile+'\START MENU\Web Links'
     goto #MakeDir

  ; Make new directories
  #MakeDir
     if !exist $Target
        makedir $Target
     if $Move ! 'Yes'
        goto #Check
     if !exist $URL_Dir
        makedir $URL_Dir
     goto #Check

  ; Completion test then check if URL file exists.
  #Check
     getini $Check $Source+'\URL.INI' [URLS] $Count+'File'
     if $Check=''
        goto #Move
     if exist $Target+'\'+$Check
        goto #Next
     goto #GetURL

  ; Create .URL file and link.
  #GetURL
     getini $Name $Source+'\URL.INI' [URLS] $Count+'URL'
     getini $Text $Source+'\URL.INI' [URLS] $Count+'Text'
     getini $File $Source+'\URL.INI' [URLS] $Count+'File'
     setini $Target+'\'+$File 'InternetShortcut' URL $Name
     setini $Target+'\'+$File 'InternetShortcut' Title $Text
     register $Folder $Target+'\'+$File $Text

  ; Increment counter and check next link.
  #Next
     increment $Count
     goto #Check

  ; Move Links to new location.
  #Move
     if $Move ! 'Yes'
        goto #End
     if exist $WininiDir+'\START MENU\PROGRAMS\WEB LINKS\*.LNK'
        copy $WininiDir+'\START MENU\PROGRAMS\WEB LINKS\*.LNK' $URL_Dir+'\*.LNK'
     if exist $WininiDir+'\START MENU\PROGRAMS\WEB LINKS\*.LNK'
        delete $WininiDir+'\START MENU\PROGRAMS\WEB LINKS\*.LNK'
     if exist $WininiDir+'\START MENU\PROGRAMS\WEB LINKS\*.LNK'
        delete $WininiDir+'\START MENU\PROGRAMS\WEB LINKS' directory

  #End

  NXD END
```

How the script works

❖ The workstation operating system is checked at the top of the script. If it is not Windows 95, processing is transferred to the #End label and the script terminates.

❖ The #SetVars section sets the script variables which are:

$Source Points to the location of the NDIST directory on the server.

$Target Defines the location of the .URL files on the workstation.

$Folder   Defines the name of the folder to be added to the Start menu.

$Count   Initializes the counter.   This is incremented during each iteration of the script.

$Move   Defines if the Web links are to be moved from the Programs menu to the primary Start menu.

- The #CheckOS section determines the workstation operating system.   If the workstation is running Windows 3.x, processing is transferred to the #End label and the script terminates.
- If the workstation is running Windows 95, processing is transferred to the #Win95 label where the $URL_Dir variable is set.
- If the workstation is running Windows NT, processing is transferred to the #WinNT label where the variable $WininiDir is set to the value of the $Profile variable that was set on the command line that executed the script (the server logon script).   The $URL_Dir variable is set to point to the users 'Profiles' directory.   This ensures that each uses gets the URL's added to their Start menu.
- The $URL_Dir variable defines name of the directory to contain the links if they are to be added to the primary Start menu.   If the $Move variable (see above) is set to No, this directory is not created.
- The #MakeDir section creates new directories on the workstation if they do not exist.
- The #Check section uses the GETINI command to get the name of the URL file to create from the INI-file named URL.INI.   If the variable $Check contains a nul value, the last URL has been processed, and script processing is transferred to the #Move label.   Otherwise, the IF command is used to check if the URL file exists.   If it does, script processing is transferred to the #Next label.   If not, processing is transferred to the #GetURL label.
- The #GetURL section uses the GETINI command to get the data required to build the URL file from the INI-file named URL.INI.   The SETINI command is then used to build the URL file using the values stored in the variables. A typical URL file looks like this:

```
[InternetShortcut]
URL=http://www.norman.no
Title=Norman on the web
```

The REGISTER command is used to create a link to the URL file.   By default, N_Dist will create the new folder defined in the $Folder variable (if it does not exist) under the Programs folder on the Start menu.

- The #Next section uses the INCREMENT command to increment the integer value stored in the variable $Count, then transfers script processing back to the #Check label where the next URL to be built is checked.
- The #Move section uses the IF command to determine if the links are to be moved up to the primary Start menu or remain under the Programs menu.   If the $Move variable is set to Yes, the links are moved.

Notes:

- This example uses the Norman web sites but, any combination of web links can be created by defining them in the INI-file.
- So far, this has only been tested in the Windows NT Server, Windows 95/Windows NT workstation environments, however it can be modified to work in other environments.   For example, in a Netware server environment use the variable LOGIN_NAME to set the N_Dist variable $Profile. when running the N_Dist script from the Netware login script.   A typical example might be:

```
#server_name/volume:\path\n_dist.exe \\server_name\volume\path\url.nxd $Profile=LOGIN_NAME /q
```

For more information on running a script from a Netware server, refer to the topic Running a Script from a Server.

Purpose:

Emit a default beep, a beep number (between 1 and 5) or a .WAV file.

Syntax:

BEEP x file.wav

Where:

x : 1-5

file.wav : The name of the wave file to use. (Windows 95 and Windows NT only)

Example:

BEEP 2

Would emit the level 2 beep.

Notes:

- The use of BEEP without any parameters will cause the default beep to sound.
- If a .WAV file is used, the full path should be specified.

An N_Dist script can be used to build another N_Dist script either on the server or on a workstation. This technique is useful if the secondary script contains variables that relate to a User ID or the Workstation ID as they can be defined using environment or other (eg. Novell script) variables. Use the N_Dist Insert command to build the secondary script.

There is only one line that N_Dist will have difficulty writing, that is, the last line of the script which contains the NXD END command. N_Dist will read it literally and quit. To overcome this behavior write the line as two components as follows:

```
...
insert $Target+'\NDIST\FILENAME.NXD' end 'NXD '+'END'
...
```

☺ Note the space after the NXD and before the single quote. This is to create the line NXD END.

The following example N_Dist script will build, run, then delete a batch file. The batch file is built in the 'Temp' directory on the workstation, so the location of this directory needs to be passed to the script on the command line. The technique used is to pass the DOS environment variable TEMP.

The script would be invoked from a batch with the command:

```
n_dist.exe map.nxd $Temp=%%temp%%\ /q
```

⚠ Note the double '%' signs, DOS will strip the first pair if run from a batch file.

This script builds a batch file that maps a drive on a Windows NT server. It will also display an error message if the workstation operating system is not Windows 95 or, the Temp variable is not present.

MAP.NXD

```
NXD BEGIN

; Determine O/S
 if $System !WIN95
      goto #OSError
 goto #ChkTmp

; Check Temp variable
#ChkTmp
 if $Temp=''
      goto #VarError
 goto #SetVars

; Set script variables
#SetVars
 set $Drv='z:'
 set $Server='\\NT_Server'
 set $Share='Data$'
 set $Batch='map.bat'
 goto #BildBat

; Build Temporary batch file
#BildBat
 insert $Temp+$Batch beginning 'net use '+$Drv+' '+$Server+'\'+$Share+' /y'
 goto #RunBat

; Run then delete temporary batch file
#RunBat
 run $Temp+$Batch
 if exist $Temp+$Batch
      delete $Temp+$Batch
 goto #End

; Display error message if O/S not Win95
#OSError
 display ' '
 display 'ERROR'
 display ' '
 display 'This script must be run on a Windows 95 workstation'
 display ' '
 display 'Press any key to exit...'
 wait
 goto #End

; Display error message if Temp variable missing
#VarError
 display ' '
 display 'ERROR'
 display ' '
 display 'The Temp variable does not exist.   Either it has not'
 display 'been set on the command line or, the Temp environment'
 display 'variable does not exist on this workstation.'
 display ' '
 display 'Command Syntax:'
 display ' '
 display '          N_Dist.Exe Map.Nxd $Temp=%%temp%%\ /q'
 display ' '
 display 'Press any key to exit...'
 wait
 goto #End

#End

NXD END
```

How the script works:

- The workstation operating system is checked at the top of the script.   If it is not Windows 95, processing is transferred to the #OSError label and a message is displayed.
- The #ChkTmp section checks that the $Temp variable has been set.   If not, processing is transferred to the #VarError label where an error message and the syntax is displayed.
- The #SetVars section sets the script variables.
- The #BildBat section uses the INSERT command to create the batch file in the Temp directory.
- The #RunBat section uses the RUN command to run the batch file.   The batch file is then deleted after it has finished running.   The nowait parameter has not been used, so that the batch file will complete before an attempt is made to delete it.
- Processing is then transferred to the #End label and the script quits.

Purpose:

To remove a parameter in the registry (Windows 95 and Windows NT only).

To remove a parameter in the OS/2 user or system profiles.

Syntax:

for Windows 95 and Windows NT

```
CLEARREGISTRY key variable
```

for OS/2

```
CLEARREGISTRY user/system application key
```

Where:

user : defines the user profile (OS2.INI).

system : defines the system profile (OS2SYS.INI).

application : is the name of the application (OS/2).

key : is the registry/profile key to clear.

variable : is the variable associated with the registry key.

Example:

```
CLEARREGISTRY 'HKEY_CURRENT_USER\Environment' 'TEMP'
```

Would remove the TEMP variable stored in the HKEY_CURRENT_USER\Environment key.

Notes:

Use with caution. Clearing entries in the registry/profile may render the system inoperable.

Purpose:

To copy a file from source to target,

Syntax:

```
COPY source target mode (deferred)
```

Where:

source : is the name of the source file.

target : is the name of the target file.

mode : is always or update. See notes for further information.

deferred : is the (optional) command modifier to cause the copy operation to be performed at the next system restart if the target file is locked. (Win32 environments only)

Examples:

```
COPY c:\*.exe d: always
```

Would copy all files that have the extension of .exe from the root of c: drive to drive d: without checking the file date/time if they exist on the target drive.

```
COPY $Source+'\data\*.doc' $Target+'\data update'
```

Would copy all files with the extension of .doc from the the source location to the target location only if they did not exist or were newer than the target files. This example assumes that the variables $Source and $Target have been set within the script.

Notes:

- Wildcards are allowed.
- Always will perform the copy every time N_Dist is executed.
- Update checks the time and date of the source and target files.
- If the target file exists the copy operation is only performed if the source file is newer than the target file.

Purpose

To remove a string from the line stored in $variable.

Syntax:

CUTWORD $variable 'string'

Where:

$variable : is the name of the variable used to store the string.

string : is the string to cut from the variable.

Examples:

CUTWORD $String 'nvcsys'

Remove a string containing 'nvc.sys' from the variable named $String.

CUTWORD $String '   nvcsys'

Remove a string containing '   nvc.sys' from the variable named $String. including up to 2 leading spaces.

Notes:

Any number of spaces can be removed before or after the word or string by including spaces.   In the second example above, up to 2 spaces before the word/sentence containing 'nvcsys' would be removed

The delimiter for the word or string is 'space'.

This command is useful for editing lines like 'LOAD=' in 'win.ini'.

The variable will store the result.

The CUTWORD script command is case insensitive.

The string can be a partial string.   For example, nvcsys would match c:\norman\dos\nvcsys.exe.   This overcomes the problem of the path being variable across different target systems.

Application example:

The CUTWORD command is used in conjunction with other N_Dist commands to edit values in an INI file. The following shows how to use N_Dist to edit the LOAD line in WIN.INI and uses the DOS/Windows implementation of NVC as an example.

A typical WINI.INI with NVC installed might be:

```
[windows]
spooler=yes
load=nwpopup.exe c:\norman\win16\nvcsys.exe c:\norman\win16\claw31.exe
run=
...
```

In order to do an update of NVC on the workstation it is necessary to remove any existing commands on the LOAD line that invoke components of NVC.   In doing so, any other commands must be preserved.

The technique used is: Get the existing value of the LOAD line and store it in a variable, then using CUTWORD, remove any commands that invoke NVC.   The CUTWORD command works on a variable so, after cutting any NVC commands, the remainder is stored in the variable and can be used to write a new value to the LOAD line, together with any new commands required for the update.

A simple script to achieve this would be:

```
NXD BEGIN

 set $Target='c:\norman\win16'
 getini $Line $Windir+'\win.ini' [windows] 'load'
 cutword $Line ' nvcsys'
 cutword $Line ' claw31'
 setini $Windir+'\win.ini' 'windows' 'load' $Line+' '+$Target+'\nvcsys.exe'+' '+$Target+'\claw31.exe'

NXD END
```

How the script works:

The SET command defines the $Target variable, in this case, the location of the NVC files.

GETINI is used to record the value of the LOAD line in the [windows] section of WIN.INI and store it in a variable named $Line.   At this point the variable $Line would contain:

```
nwpopup.exe c:\norman\win16\nvcsys.exe c:\norman\win16\claw31.exe
```

The first CUTWORD command is used to remove any string containing the text ' nvcsys'.   Note that a leading space will also be removed.   The $Line variable would now contain:

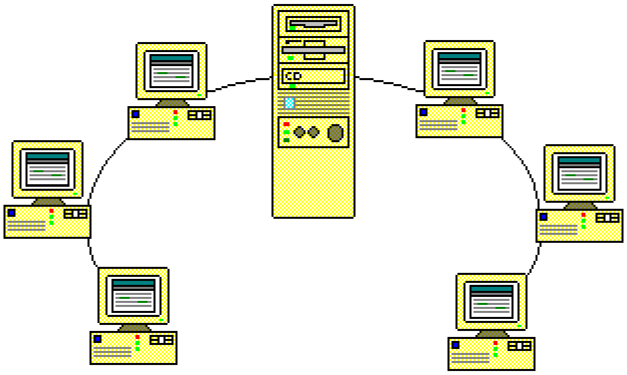
```
nwpopup.exe c:\norman\win16\claw31.exe
```

The second CUTWORD command is used to remove any string containing the text ' claw31'.   A leading space will also be removed.   The $Line variable would now contain:

```
nwpopup.exe
```

This has effectively preserved any other command(s) on the LOAD line.

The SETINI command is used to write back any preserved command(s) and add the new commands required for the NVC update.


CD

This topic describes how to control a workstation update by using an INI file which contains the update level as a value.   The script compares the update level on the server with the update level on the workstation and only proceeds if the values are different.   Using this technique, a System Administrator can control when an update is to take place.

The following INI file (UPDATE.INI) is stored on the server and copied to the workstation during the update.

UPDATE.INI

```
[update]
level=001
```

The script is as follows:

UPDATE.NXD

```
NXD BEGIN

// Set Script variables.
set $Source='F:\PUBLIC\DATA'
set $Target='C:\DATA'
goto #CheckOS

// Check workstation O/S.
#CheckOS
if $System !WIN95
    goto #End
goto #CheckLevel

// Check update level.
#CheckLevel
getini $Server $Source+'\NDIST\UPDATE.INI' UPDATE LEVEL
getini $WStation $Target+'\NDIST\UPDATE.INI' UPDATE LEVEL
if $Server=$WStation
    goto #End
goto #CopyFiles

// Copy files to workstation.
#CopyFiles
copy $Source+'\NDIST\UPDATE.INI' $Target+'\NDIST\UPDATE.INI' update
copy $Source+'\MEMO.DOC' $Target+'\MEMOS\MEMO.DOC' update
copy $Source+'\STAFF.DOC' $Target+'\ADMIN\STAFF.DOC' update
copy $Source+'\PHONE.DOC' $Target+'\ADMIN\PHONE.DOC' update
goto #End

#End

NXD END
```

To initiate an update the System Administrator need only copy the new files to the server source directory and increment the update level in UPDATE.INI by 1.   As each user logs on, the update will take place one time only. The next time a user logs on, the values in UPDATE.INI will be the same so the script will transfer processing to the #End label and quit.

An N_Dist script is a plain text file.   Use a text editor to create a script.

A script can be in any order that the author requires but the following rules and guidelines should be followed.

Script Rules.

- Only the lines between NXD BEGIN and NXD END (must be uppercase) will be interpreted.
- Comments start with any of the following three identifiers:

  ;   //   /*

- Script files have the extension of .NXD
- String arguments that contain spaces must be enclosed in single quotes.
- When directory names are given as arguments, it is not necessary to enclose them in quotes unless they contain spaces.
- Commands, variable names etc are case insensitive unless specified otherwise.
- A variable name can be up to 255 characters.
- Extra spaces between arguments are not interpreted.
- When a "=" is used in a command, there are no requirements on spaces before or after the "=" sign.
- When a ":" is used in a command, there are no requirements on spaces before or after the ":" sign.
- N_Dist will not override the original attribute setting on files or directories, and it will not reset or change access rights.
- The position of arguments following N_Dist commands is significant.

The following example will copy a group of files from a server to a workstation only if the files located on the server are newer than those on the workstation and the workstation operating system is Windows 95.

DEMO.NXD

```
NXD BEGIN

// Set script variables
set $Source='F:\PUBLIC\DATA'
set $Target='C:\DATA\WINWORD'
goto #CheckOS

// Check O/S
#CheckOS
if $System !WIN95
    goto #End
goto #CopyFiles

// Copy files to workstation
#CopyFiles
copy $Source+'\MEMO.DOC' $Target+'\MEMOS\MEMO.DOC' update
copy $Source+'\STAFF.DOC' $Target+'\ADMIN\STAFF.DOC' update
copy $Source+'\PHONE.DOC' $Target+'\ADMIN\PHONE.DOC' update
goto #End

#End
```

```
NXD END
```

💡 Script comments are not mandatory, but they do enhance the readability and are recommended.

Purpose:

To delete a file, a set of files, a directory, or a group of directories

Syntax:

```
DELETE file or path (directory) (hidden) (recurse)
```

Where:

file	:	is name of the file to delete.
path	:	is the name of the directory to delete.
directory	:	is the (optional) command modifier to cause N_Dist to delete a directory.
hidden	:	is the (optional) command modifier to cause N_Dist to delete hidden files if they exist.
recurse	:	is the (optional) command modifier to cause N_Dist to delete recursively. This is functionally similar to the DOS 'deltree' command.

Examples:

```
DELETE c:\norman\nvc.* hidden
```

Delete all files that conform to the filespec even if they are hidden.

```
DELETE c:\data\*.* recurse
```

Delete the 'data' directory and all subdirectories.including all files regardless of attributes.

Use recursion with caution. Erroneous use may render the system inoperable.

Notes:

A directory must be empty before it can be deleted.

Wildcards can be used when specifying both files and directories.

When specifying a directory as an argument the trailing '\' is not required.

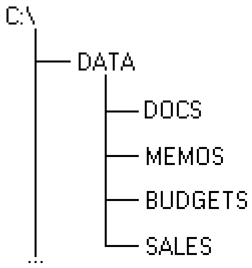
The recurse parameter will delete files regardless of attributes.

Application example:

This example script will remove all files and directories in the tree structure shown in the adjacent diagram, including hidden/read-only files.

By using the recurse parameter together with wildcards as file specifications, only two lines of code are necessary for the deletion task.

This is particularly useful in building a script to perform maintenance/cleanup tasks on a group of network connected workstations.



```
NXD BEGIN

 set $Dir='C:\DATA'

 delete $Dir+'\*.*' recurse
 delete $Dir directory

NXD END
```

How the script works:

The SET command is used to define the variable $Dir.

The DELETE command is used to delete all files (including hidden/read-only files) starting at the path stored in the $Dir variable and to recurse down the tree until all files and subdirectories are removed.

The DELETE command is used again, this time to delete the directory stored in the $Dir variable.

If the workstation is running the Norman Smart Behavior Blocker in a Windows 95 environment, the deletion of directories will fail as there is a time delay between the execution of the delete command and the actual deletion, if the files are executables.   This type of script would normally be run from the login script and consequently the Smart Behavior Blocker would not be running.   However, during any testing, unload the Smart Behavior Blocker or, place a WAIT 2 command between the two delete commands in the script.

Purpose:

To display a string and/or variables. Several strings and variables can be displayed on one line. See notes below for further information.

Syntax:

```
DISPLAY 'string' $variable ... ...
```

Where:

string : is a text string to display. If the string contains spaces enclose it in quotes.

$variable : is the name of a variable that contains a string to display.

Example:

```
DISPLAY 'End of program, log file is: '+$Logdrv+'\LOG.DAT'
```

Would display the text 'End of program, log file is: C:\LOG.DAT' assuming the variable $Logdrv was set to C:.

Notes:

A '+' sign between sub-strings concatenates strings/variables into a continuous string.

Introduction

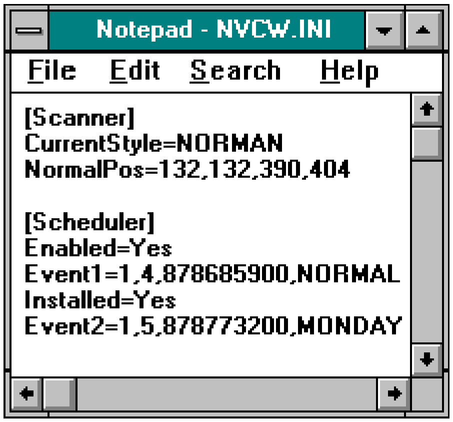
The default NVC scripts provided by Norman (NVCW.NXD, NVC95.NXD, NVCNT.NXD) allow administrators to distribute and install standard scanning and scheduling configurations on each workstation, through the use of distributed REG and INI files.

Distributing Norman Virus Control configuration settings:

When configuring Norman Virus Control, the changes are stored in the NVCW.INI file (Windows 3.1x) , or in the Registry (Windows 95 or NT). To standardize NVC configurations across your network, you must distribute this information to all users using N_Dist.

For Windows 3.1x:

After configuring NVC options, including styles and scheduled scans if appropriate, you will see the changes reflected in the files called <path>NORMAN\WIN16\NVCW.INI, containing information on options and scheduling, and <path>NORMAN\WIN16\NVCW.DAT, containing information on styles.



```
[Scanner]
CurrentStyle=NORMAN
NormalPos=132,132,390,404

[Scheduler]
Enabled=Yes
Event1=1,4,878685900,NORMAL
Installed=Yes
Event2=1,5,878773200,MONDAY
```

Copy these files to the server in the following location:

<path>NVCADMIN\NDIST\WIN16\NVCW.INI

<path>NVCADMIN\NDIST\WIN16\NVCW.DAT

When using the default N_Dist script, N_Dist will look for these files in the source directory and copy them to the workstation. In this way, each workstation will have the same configuration settings.

◆ In the default script, N_Dist copies these files with the 'update' parameter, meaning that each file will be copied to the workstation only if the source file has a newer date than than the target file on the workstation. However, once NVC has been installed on the workstation, users will be able to alter the NVCW.INI and NVCW.DAT files on their workstation. If you wish to 'reset' the configurations each time the user logs on, you should edit the the NVCW.NXD script, changing the copy commands for these two files, found in the #NVC_Win16 section. For example change:

```
copy $Source+'\WIN16\NVCW.INI'   $Target+'\WIN16\NVCW.INI' update
```

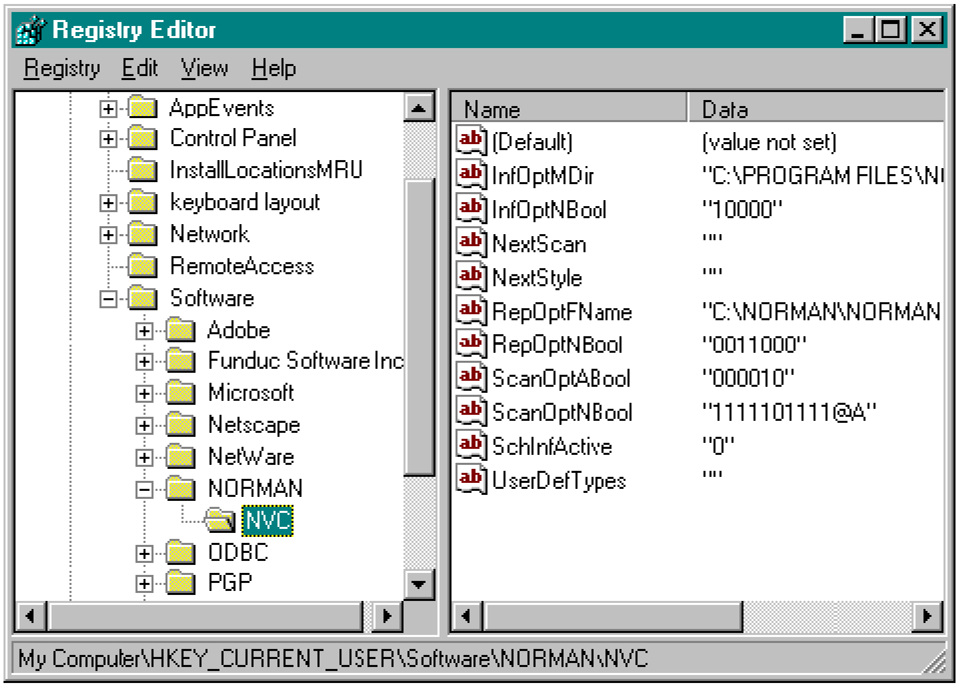
to:

```
copy $Source+'\WIN16\NVCW.INI'   $Target+'\WIN16\NVCW.INI' always
```

and similarly the line copying the NVCW.DAT file.

For Windows 95 and NT:

After configuring NVC's options, including styles and scheduled scans if appropriate, you will see the changes reflected in the Registry, in the following key: HKEY_CURRENT_USER\Software\NORMAN\NVC



Name	Data
(Default)	(value not set)
InfOptMDir	"C:\PROGRAM FILES\N
InfOptNBool	"10000"
NextScan	""
NextStyle	""
RepOptFName	"C:\NORMAN\NORMAN
RepOptNBool	"0011000"
ScanOptABool	"000010"
ScanOptNBool	"1111101111@A"
SchInfActive	"0"
UserDefTypes	""

My Computer\HKEY_CURRENT_USER\Software\NORMAN\NVC

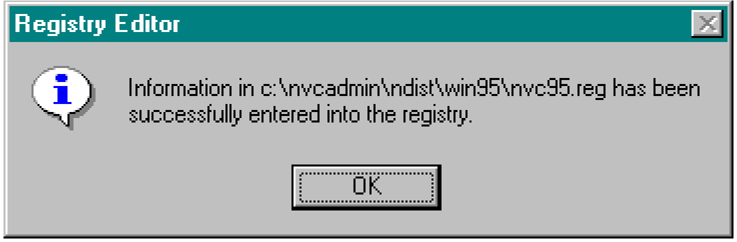
Using the Registry Editor, highlight the NVC key and export the registry settings (Registry | Export Registry Settings) to a file called NVC95.REG (for Windows 95) or NVCNT.REG (for Windows NT). Then copy the files to the server in the following locations:

<path>NVCADMIN\NDIST\WIN95\NVC95.REG

<path>NVCADMIN\NDIST\WIN32\NVCNT.REG

When using the default N_Dist script, N_Dist will look for this NVC95.REG file in the source directory and copy it to the workstation. It will then call REGEDIT with the REG file and import the registry settings on the workstation.

- The user will see a dialogue box requiring them to acknowledge these registry settings;



Information in c:\nvcadmin\ndist\win95\nvc95.reg has been successfully entered into the registry.

- Here again, if you wish to 'reset' the configurations each time the user logs on, you should change the script so that it always copies the REG file. For example in the NVC95.NXD script (in the #Scheduling section), change the line from:

```
copy $Source+'\WIN95\NVC95.REG' $Target+'\WIN95\NVC95.REG' update
```

to:

```
copy $Source+'\WIN95\NVC95.REG' $Target+'\WIN95\NVC95.REG' always
```

Distributing Cat's Claw Configuration Settings:

When Cat's Claw is configured, the following files are created:

1. NVCMACRO.CRT, if you certify macros.
2. All other configuration settings in CLAW31.INI or in CLAW95.REG, for Windows 3.1x and Windows 95, respectively.

- Unlike the NVC95.REG file, the CLAW95.REG file is created automatically.

Make sure that these files are copied to the server in their respective folders:

- NVCMACRO.CRT in \NVCADMIN where the definition files reside
- CLAW31.INI in \NVCADMIN\WIN16
- CLAW95.REG in \NVCADMIN\WIN95

If you're using default N_Dist distribution, Cat's Claw will be updated with new files whenever changes are made to the files on the server.

- In the default installation, users are not provided with the configuration program necessary to change the INI or REG files, and therefore there is no need to 'reset' the configurations at each login as discussed above with NVC.

Purpose:

To extract the drive letter from a path and store the result in the specified variable.

Syntax:

```
GETDRIVE $variable 'path'
```

Where:

$variable : is the name of the variable to store the drive letter.

path : is the path to extract the drive letter from. Enclose the path in quotes.

Examples:

```
GETDRIVE $Drive 'c:\dos\chkdsk.exe'
```

Would return c: in the variable $Drive.

```
GETDRIVE $Drive $Source
```

Would return the drive component of the variable $Source in the variable $Drive.

Notes:

The 'path' can be a string as in the first example or, a variable as in the second example.

❖

Purpose:

To extract a value from an INI-file and store it in a variable.

Syntax:

GETINI $variable ini-file [section] 'keyword'

Where:

$variable : is the name of the variable to store the value.

ini-file : is the name of the INI-file to get the value from.

[section] : is the section name in the INI-file.

keyword : is the keyword in the INI-file.

Example:

GETINI $Result win.ini [windows] 'load'

The variable $Result will contain the value associated with the 'load' keyword in WIN.INI.

Notes:

❖ Enclose the INI-file keyword in quotes.
❖ The section name is as it would appear in the INI-file eg. [windows].

Application example:

For an example of how GETINI is used, refer to the CUTWORD topic.

❖

Purpose:

To extract the directory name from a complete path and store the value in the specified variable.

Syntax:

```
GETPATH $variable 'path'
```

Where:

$variable : is the name of the variable that will store the result.

path : is the path to extract the directory name from.

Examples:

```
GETPATH $Path 'c:\dos\chkdsk.exe'
```

Will return the path c:\dos\ and store it in the variable $Path.

```
GETPATH $Path $Source+'\data\*.doc'
```

Will return the path component of $Source+'\data*.doc' and store it in the variable $Path. Assuming the $Source variable was set to f:\users the $Path variable would contain f:\users\data.

Notes:

❖ If the path specified in the argument does not end with a filename, the value of the variable will be the same as the path, but without the trailing "\".

❖ The path specified in the argument can be a string, variable or, a combination as in the second example.

Purpose:

To extract the value of a parameter in the registry and store the result in the specified variable. (Windows 95 and Windows NT only)

To extract the value of a parameter in the OS/2 user or system profiles and store the result in the specified variable.

Syntax:

for Windows 95 and Windows NT

GETREGISTRY $variable 'key' 'parameter'

for OS/2

GETREGISTRY $variable 'application' 'key'

Where:

$variable : is the name of the variable to store the value.

key : is the registry/profile key to get the value from.

parameter : is the parameter name in the registry.

application : is the name of the application (OS/2).

Example:

GETREGISTRY $Result 'HKEY_CURRENT_USER\Environment' 'TEMP'

The value of the 'TEMP' registry key would be stored in the variable $Result.

Notes:

Users with read-only rights are able to access all parts of the Registry.

Purpose:

To transfer processing to a line in an N_Dist script starting with '#Label'.

Syntax:

```
GOTO #Label
```

Where:

#Label : is the label to go to.

Example:

```
GOTO #End
```

Would cause script processing to be transferred to the #End label.

Notes:

A label is prefixed with #.

How to contact Norman

❖

Purpose:

To (conditionally) execute the next line of an N_Dist script.

Syntax:

```
IF EXIST/!EXIST/ERROR/$variable =/! $variable/'value' (WRITE) (EXECUTE)
```

Where:

EXIST	:	is the parameter to test for the existence of a path or file.
!EXIST	:	is the parameter to test for the non-existence of a path or file.
ERROR	:	is the parameter to test the error condition.
$variable	:	is the name of a variable to test.
value	:	is a value to test.
WRITE	:	is the (optional) parameter to cause N_Dist to check for 'write' rights.
EXECUTE	:	is the (optional) parameter to cause N_Dist to check for 'execute' rights.

Example:

```
IF EXIST 'c:\norman\nvc.exe' execute
```

This checks if the file C:\NORMAN\NVC.EXE exists with execute rights. If 'true' (the file does exist) the next line of the script will be processed, otherwise, it will be skipped.

Notes:

- If the result of the test is true, the next line of the script will be executed.
- If the result of the test is false, the next line of the script will be skipped.
- The EXIST command checks for existence of paths and files.
- Paths and files may also be checked for write and execute rights.

Application example:

The following example script uses the GETINI command to get values located in two INI-files and store them in variables named $Server and $Wstation. The IF command is used to compare the two variables. If the values are the same (IF returned 'true'), processing is transferred to the #End label and the script finishes. If the values are not the same (IF returned 'false'), processing is transferred to the label #CopyFiles and a file update takes place.

```
NXD BEGIN

// Set Script variables.
set $Source='F:\PUBLIC\DATA'
set $Target='C:\DATA'
goto #CheckOS

// Check workstation O/S.
#CheckOS
if $System !WIN95
    goto #End
goto #CheckLevel

// Check update level.
#CheckLevel
getini $Server $Source+'\NDIST\UPDATE.INI' UPDATE LEVEL
getini $WStation $Target+'\NDIST\UPDATE.INI' UPDATE LEVEL
if $Server=$WStation
    goto #End
goto #CopyFiles

// Copy files to workstation.
#CopyFiles
copy $Source+'\NDIST\UPDATE.INI' $Target+'\NDIST\UPDATE.INI' update
copy $Source+'\MEMO.DOC' $Target+'\MEMOS\MEMO.DOC' update
copy $Source+'\STAFF.DOC' $Target+'\ADMIN\STAFF.DOC' update
copy $Source+'\PHONE.DOC' $Target+'\ADMIN\PHONE.DOC' update
goto #End

#End
```

```
   NXD END
```

This technique can be used to control workstation updates. The System Administrator need only increment the value stored in the server based INI-file to initiate an update. For a full explanation of this technique, refer to Controlling an Update.

Purpose:

To (conditionally) execute the next line of an N_Dist script.   The next line will be executed if the the source is newer than the target.

Syntax:

```
IF UPDATE source target
```

Where:

source : is the source path

target : is the target path.

Example:

```
IF UPDATE f:\users\templates c:\data\templates
display 'Your wordprocessing templates are out of date.'
```

This checks the date/timestamps of document templates on a workstation against those stored on a server.   If the workstation copies are older, a message is displayed.

Notes:

Both the source and target parameters may be defined as variables.

❖

Purpose:

To Increment the integer value stored in a variable by one.

Syntax:

```
INCREMENT   $variable
```

Where:

$variable : is the name of the variable that is used to store the integer value.

Example:

```
INCREMENT $Count
```

Would increment the value of the variable $Count by 1.

Notes:

❖ This command can be used to cause N_Dist to execute a procedure multiple times. See the IF command for further information on conditional processing.

Application examples:

Example 1

The first example uses the INCREMENT command to control the number of iterations the code section of the script is executed.

It also employs the technique of setting the value of a variable on the command line by using a DOS batch file replaceable parameter. The batch file looks like this:

COUNT.BAT

```
@echo off
 n_dist.exe count.nxd $Count=%1 /q
```

The script is as follows:

COUNT.NXD

```
NXD BEGIN

; Set script variables
 set $Counter='0'

; Increment counter
#Loop
 increment $Counter
 goto #Code

; Code section of script
#Code
 display $Count+' '+$Counter
 if $Counter=$Count
      goto #End
 goto #Loop

#End

NXD END
```

To run the script, invoke COUNT.BAT with a command line parameter that will become the value of the $Count variable referred to in the script. eg.

```
COUNT.BAT 5
```

In this example, the #Code section of the script will simply display the values of the two variables. In a real-time application, this would be replaced by the commands required to satisfy the purpose of the script.

Example 2

The second example uses the same basic script, except that it has been modified to prompt the user for the number of times to run.

PROMPT.NXD

```
NXD BEGIN
```

```
; Set script variables
 set $Counter='0'

; Prompt user for value
 display 'How many times should this run?'
 display 'Enter a number between 1 and 9.'
 wait
 set $Count=$Key

; Increment counter
#Loop
 increment $Counter
 goto #Code

; Code section of script
#Code
 display $Count+' '+$Counter
 if $Counter=$Count
      goto #End
 goto #Loop

#End

NXD END
```

The DISPLAY command is used to display a user prompt.   Followed by the WAIT command used without any parameters which causes the script to wait indefinitely for a keystroke and store the value of the key in the special variable called $Key.   The $Count variable is then set to the value of $Key.

❖ The WAIT command will accept a single keystroke, consequently the input range is limited to 0-9.

❖

Purpose:

To add a string to the beginning or end of a line containing a keyword in a text-file.

Syntax:

INSERT file beginning or end [section] 'keyword' 'string' (nodup)

Where:

file	:	is the name of the file to insert the string.
beginning	:	causes N_Dist to insert the string at the beginning of the section/file/line. (see notes below)
end	:	causes N_Dist to insert the string at the end of the section/file/line. (see notes below)
[section]	:	is the section name if the file is an INI-file. Enclose the section name in square brackets.
keyword	:	is the name of the keyword if the file is an INI-file.
string	:	is the string to insert.
nodup	:	is the (optional) parameter that causes N_Dist not to execute the insertion if the value in string already exists.

Examples:

INSERT config.sys end [common] 'device=nvc.sys' nodup

The line device=nvc.sys is inserted at the end of the common section in config.sys.   If the line already exists anywhere in config.sys, the nodup arguments instructs N_Dist not to insert the string.

INSERT config.sys end 'device=nvc.sys' nodup

The line device=nvc.sys is inserted at the end of config.sys.   If the line already exists anywhere in config.sys, no insertion is done.

INSERT config.sys beginning 'nvc.sys' 'REM '

A REM entry is inserted at the beginning of all lines containing 'nvc.sys'.

Notes:

❖ nodup skips the command if the entry already exists.

❖ If a file and/or a section does not exist, it is created.

❖ If a section is specified the beginning or end applies to the beginning or end of the section.

❖ If no section is specified the beginning or end applies to the beginning or end of the file.

❖ If a keyword is specified the beginning or end applies to the beginning or end of the line that contains the keyword.

❖ Whenever a file is specified, the full path should be given.   In addition, the position of the arguments is significant, it is recommended that this be taken into consideration when distinguishing between a keyword and a string.

Application examples:

Example 1

The following example script will add the [restrictions] section and, define the Startup group in the [settings] section of PROGMAN.INI in a Windows 3.x environment.

```
NXD BEGIN

; Set script variables.
 set $Windir='C:\Win311'

; Add Restrictions section.
 insert $Windir+'\PROGMAN.INI' [restrictions] beginning 'EditLevel=4' nodup
 insert $Windir+'\PROGMAN.INI' [restrictions] beginning 'NoFilemenu=1' nodup
 insert $Windir+'\PROGMAN.INI' [restrictions] beginning 'NoSaveSettings=1' nodup
 insert $Windir+'\PROGMAN.INI' [restrictions] beginning 'NoClose=1' nodup
 insert $Windir+'\PROGMAN.INI' [restrictions] beginning 'NoRun=1' nodup

; Define Startup group.
 insert $Windir+'\PROGMAN.INI' [settings] end 'Startup=STARTUP' nodup

#End
```

```
NXD END
```

How the script works:

- The variable $Windir is set at the top of the script. Normally, this variable should not be set by the script, however, in a Windows 3.x environment, Windows is not running when a user logs in. If this script was run from the server login script, N_Dist would set the $Windir variable to c:\windows by default. This is not always the location of the Windows directory, as in this example.
- The INSERT command is used to write the items in the [restrictions] section. If the [restrictions] section does not exist, it will be created. If all or any of the items exist, they will not be duplicated, the nodup parameter causes N_Dist to check for existence of the string.
- The INSERT command is used again to define the name of the Startup group in the [settings] section.

Example 2

The following example script will add an item to the load line of WIN.INI in a Windows 3.x environment.

A typical WINI.INI might look like this:

```
[windows]
spooler=yes
load=nwpopup.exe
run=
Beep=yes
NullPort=None
...
```

The N_Dist script to do the editing task is:

```
NXD BEGIN

; Set script variables
 set $Windir='C:\Win311'

; Add items to load line in WIN.INI
 insert $Windir+'\WIN.INI' end 'load' ' c:\mouse\pointer.exe'

#End

NXD END
```

How the script works

- The variable $Windir is set at the top of the script. See the note above on setting the $Windir variable.
- The INSERT command is used to append the string ' c:\mouse\pointer.exe' to the load line. Note the leading space.
- WIN.INI would now look like this:

```
[windows]
spooler=yes
load=nwpopup.exe c:\mouse\pointer.exe
run=
Beep=yes
NullPort=None
...
```

Module: Norman Distribution Tool
Filename: NDist.Hlp
Build Date: 16th April, 98
Help Author: Peter Maher

Purpose:

To create a directory or complete directory path.

Syntax:

MAKEDIR path

Where:

path : is the directory or complete path to create.

Examples:

MAKEDIR c:\norman\dos

The complete path c:\norman\dos would be created.

MAKEDIR $Target+'\dos'

The complete path c:\norman\dos would be created assuming that the $Target variable had been set to c:\norman.

Notes:

All non-existent directories in the path will be created at one time.

Purpose:

To control the behavior of N_Dist during execution of a script.

Syntax:

OPTION: option

Where option is one of the following:

Option	Description	Notes
Log	Log to default log file	By default, the log will be written to the current directory, and the filename will be the name of the script with a .log extension. This option can also be specified from the command line with the parameter /LF(filename). Unlike Option: Log, however, the filename can be specified from the command line.
Append	Append to the log file	By default, N_Dist overwrites the log file. This option can also be specified from the command line with the parameter /LG(filename).
Verbose	Verbose log file	The /V command line parameter may also be used.
Single	Single-step script interpretation	The /S command line parameter may also be used.
Nosingle	Turn off single-step	There is no associated command line parameter for this function.
Nosound	Turn off sound	There is no associated command line parameter for this function.
Quiet	Turn off sound and screen output (except for the intro logo, system messages, syntax errors and user messages)	The /Q command line parameter may also be used.
Veryquiet	No output at all	The /Q! command line parameter may also be used.
Normal	Resets "Nosound" "Quiet", "Veryquiet", and "Verbose" options	There is no associated command line parameter for this function.
		The command line parameter /DEBUG will generate debug numbers used for support

Example:

Option: Verbose

Notes:

In the example above, N_Dist will give verbose screen output from the 'Option' statement down.

-

The **Norman Distribution Tool** (N_Dist) can be used to install or update software from server to server or, from a server to multiple workstations. The environments supported by N_Dist are:

- DOS
- Windows 3.x
- Windows 95
- Windows NT
- OS/2

The Corporate versions of **Norman Virus Control** products include N_Dist and support an 'Administrator' server installation. During the 'Administrator' installation the N_Dist script is generated. This is then used to install/update **Norman Virus Control**.

N_Dist can be used for the following on network connected workstations:

- Install software.
- Update software.
- Rollout updated wordprocessing templates.
- Update mailing lists.
- Update data files.

💡 This is only a suggested list of uses. N_Dist can be used for any server to server or, server to workstation distribution task.

❖

Purpose:

To display a popup message to the user during script execution.   (Window 95 and Window NTonly)

Syntax:

```
POPMESSAGE text
```

Where:

text : is the text to be displayed in the message.

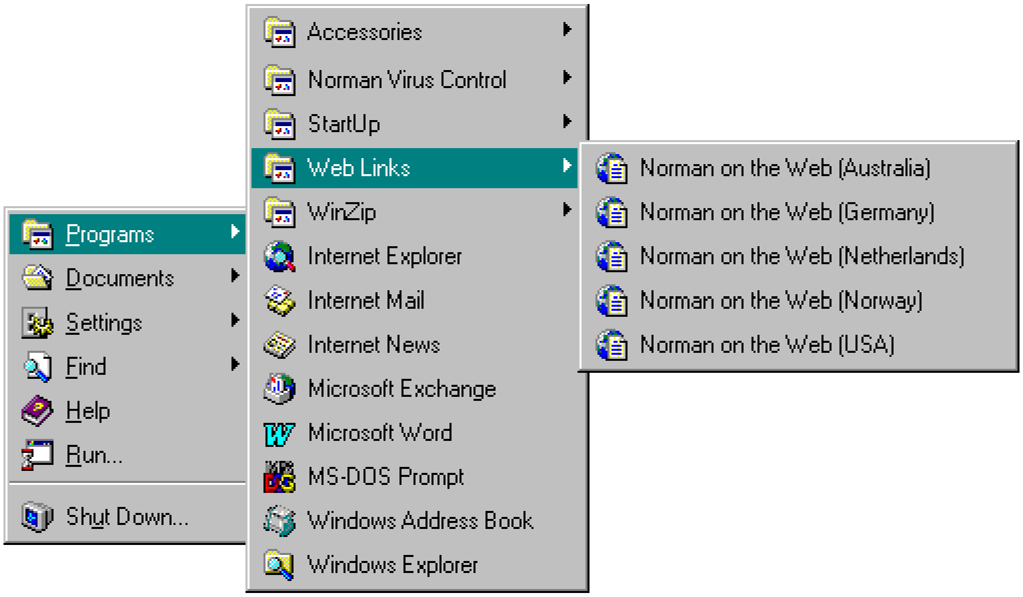
Example:

```
POPMESSAGE 'Error: the file does not exist'
```

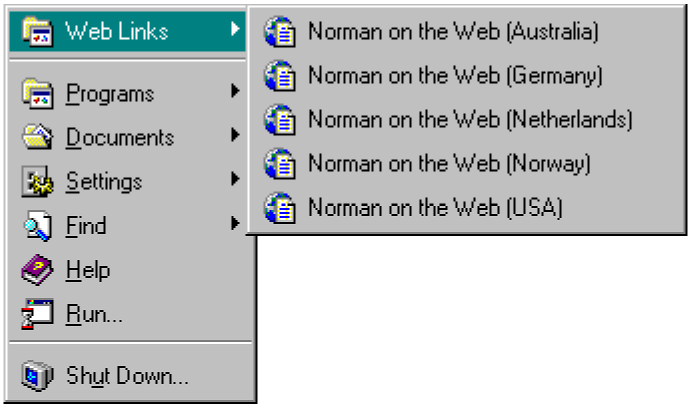
Notes:

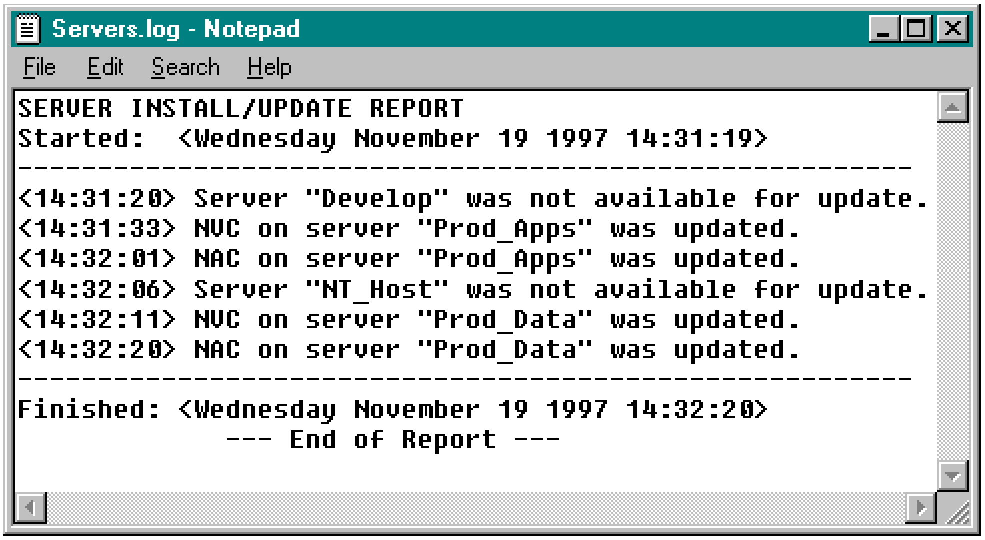
❖ The user must click 'OK' for script execution to continue.

❖ POPMESSAGE is ignored if N_Dist is running in 'very quiet' mode (Q!)


Programs
Documents
Settings
Find
Help
Run...
Shut Down...
Accessories
Norman Virus Control
StartUp
Web Links
WinZip
Internet Explorer
Internet Mail
Internet News
Microsoft Exchange
Microsoft Word
MS-DOS Prompt
Windows Address Book
Windows Explorer
Norman on the Web (Australia)
Norman on the Web (Germany)
Norman on the Web (Netherlands)
Norman on the Web (Norway)
Norman on the Web (USA)

This release of the on-line documentation is compatible with N_Dist version 1.36 and above.   Some of the sample scripts that appear in 'Script Techniques' and the 'Application Examples' will not work with earlier versions.


Web Links
Programs
Documents
Settings
Find
Help
Run...
Shut Down...
Norman on the Web (Australia)
Norman on the Web (Germany)
Norman on the Web (Netherlands)
Norman on the Web (Norway)
Norman on the Web (USA)



```
SERVER INSTALL/UPDATE REPORT
Started:  <Wednesday November 19 1997 14:31:19>
--------------------------------------------------------
<14:31:20> Server "Develop" was not available for update.
<14:31:33> NVC on server "Prod_Apps" was updated.
<14:32:01> NAC on server "Prod_Apps" was updated.
<14:32:06> Server "NT_Host" was not available for update.
<14:32:11> NVC on server "Prod_Data" was updated.
<14:32:20> NAC on server "Prod_Data" was updated.
--------------------------------------------------------
Finished: <Wednesday November 19 1997 14:32:20>
             --- End of Report ---
```

❖

Purpose:

To create a folder/group and/or program icons on workstation desktop.

Syntax:

```
REGISTER 'folder or group name' path 'icon name' (common)
```

Where:

folder : is the name of the folder to create. (Windows 95/NT)

group : is the name of the group to create. (Windows 3.x)

path : is the path to the executable item in the new folder or group.

icon name : is the name of the new icon.

common : is the (optional) parameter that causes N_Dist to create the new link in the common area. (Windows 95 and Windows NT only)

Example:

```
REGISTER 'Norman' c:\norman\nvcnt\nvcnt.exe 'NvcNT'
```

This will create an icon for nvcnt.exe named NvcNT in the Norman folder.

```
REGISTER 'Norman' c:\norman\nvcnt\nvcnt.exe 'NvcNT' common.
```

This will create an icon for nvcnt.exe named NvcNT in the Norman folder in the common area.

Notes:

❖ If the folder does not exist, it is created.

❖ In Windows NT and Windows 95, the folder 'startup' is automatically resolved to the correct startup folder name for the current language.

❖ The profile 'common' can be specified on Windows 95 and Windows NT platforms to register links in the 'common' programs area.

❖

Purpose:

To remove a complete line from an ASCII file.

Syntax:

```
REMOVE file string
```

Where:

file	:	the name of the ASCII file.
string	:	any string within the line to be removed.

Example:

```
REMOVE config.sys nvc.sys
```

Would remove the line device=c:\norman\dos\nvc.sys /n /t from config.sys, if that was how it appeared. It would also have removed the complete line if, for example, the device had been loaded high, was located in a different directory or, had different command line parameters.

Notes:

❖ The above example will remove only the first line in config.sys which has the string nvc.sys, all remaining lines with the string will be left untouched.

❖ The specified string can be a partial string. This overcomes problems associated with paths and command line parameters as shown in the example.

❖ The string defined in the REMOVE command is case insensitive.

❖

Purpose:

To rename a directory or file.

Syntax:

RENAME source target

Where:

source : is souce directory/filename.

target : is the target directory/filename.

Example:

RENAME c:\config.sys c:\config.bak

This would rename config.sys to config.bak located in the root of drive c:.

RENAME c:\data\docs c:\data\sales

This would rename the data\docs directory to data\sales.

Notes:

- Both the source and target parameters may be defined as variables.
- Wildcards are not allowed, consequently groups of files cannot be renamed.

Purpose:

To replace one or all instances of string1 with string2 in a file.

Syntax:

```
REPLACE 'string1' 'string2' file (nocase) (all)
```

Where:

string1 : is the string to be replaced.

string2 : is the replacement string.

file : is the name of the file

nocase : is the (optional) parameter to make the search case insensitive.

all : is the (optional) parameter to cause N_Dist to replace all occurrences of string1 in the specified file.

Example:

```
REPLACE 'NVC 4.20' 'NVC 4.30' c:\norman\nvc.ini all nocase
```

This would replace all occurrences of the string 'NVC 4.20' with the string 'NVC 4.30' in the file nvc.ini, regardless of case.

Notes:

When specifying the file, the full path should always be given.

❖

Purpose:

To run an application and return to script when finished.

Syntax:

RUN application 'parameters' (nowait)

Where:

application : is the name of the application to run.

parameters : are any command-line parameters to pass to the application.

nowait : is the (optional) parameter to cause N_Dist not to wait for the spawned application to finish before returning to the script. (32-bit environments only)

Example:

RUN $Windir+'\Notepad.exe' 'c:\norman\norman.rpt' nowait

This would run Notepad and display norman.rpt, then continue script execution without waiting for Notepad to terminate.

RUN net 'use w: \\NT_Server\Data$'

This would run net.exe to map drive W: to the share Data$ on the server named NT_Server.

Notes:

❖ The application return code is stored in the special variable $Returnvalue.

❖ The 'Run' command detects 16 bit Windows applications and executes them via 'NRMWINST' even when N_Dist is run in a DOS session prior to Windows.

This topic refers to the DOS/Windows 3.x environment.

This technique is part of a software deinstallation, the first part of which is relatively simple as all path/file locations should be known. An N_Dist script can be built to carry out this task. The only part of the deinstallation that is unknown, is the group number in PROGMAN.INI. In a group of similar workstations this will vary according to how many applications have been installed and in what order. A typical PROGMAN.INI might be:

```
[Settings]
Window=20 13 629 428 1
display.drv=vga.drv
Order= 1 3 2 4 5 6

[Groups]
Group1=C:\WINDOWS\MAIN.GRP
Group2=C:\WINDOWS\ACCESSOR.GRP
Group3=C:\WINDOWS\NETWORK.GRP
Group4=C:\WINDOWS\GAMES.GRP
Group5=C:\WINDOWS\STARTUP.GRP
Group6=C:\NORMAN\NVCLOCAL.GRP
```

The following example N_Dist script will check if the NVCLOCAL.GRP group is present in PROGMAN.INI and if so, remove it.

```
NXD BEGIN

; Set script variables
 set $Count='1'

; Get group number from PROGMAN.INI
#GetGrp
 getini $Group $Windir+'\progman.ini' [groups] 'Group'+$Count
 if $Group=''
      goto #End
 if $Group='C:\NORMAN\NVCLOCAL.GRP'
      remove $Windir+'\progman.ini' NVCLOCAL.GRP
 increment $Count
 goto #GetGrp

 #End

 NXD END
```

How the script works

- The variable $Count is set to '1' at the top of the script. This variable is a counter that is incremented each iteration of the script.
- The GETINI command is used to determine the group name in PROGMAN.INI and store it in a variable named $Group. During the first iteration of the script the variable $Count would contain '1' so the GETINI command would return the name of the first group (MAIN.GRP) in the variable $Group.
- The next line checks the content of the variable $Group. If it is a nul string, processing is transferred to the #End label and the script quits. This is a completion test.
- The IF command compares the value of the variable $Group with the string 'C:\NORMAN\NVCLOCAL.GRP' and if it matches, the next line is executed.
- The REMOVE command removes the line containing the string 'NVCLOCAL.GRP'
- The INCREMENT command increments the variable $Count by 1.
- The GOTO command transfers processing to the #GetGrp label.
- This process continues until all groups defined in PROGMAN.INI have been checked at which time the value of the variable $Group will be a nul string and the script will quit.

❖

This technique stores the names of scripts to run in an INI-file. All scripts defined in the INI-file will be run sequentially by a master script. This means that only one line needs to be added to a server login script.

Each iteration of the master script will get the name of the script to run and, whether or not is to run, from the INI-file. This allows a System Administrator to control execution of multiple scripts from within the INI-file. For example, a maintenance script that does a cleanup task might be run once a month. The System Administrator would simply change the value in the INI-file from 'No' to 'Yes' for a period of 24 hours, then change it back to 'No'.

The INI-file is in the following format.

MAIN.INI

```
[scripts]

1=scriptname1.nxd
1Run=Yes

2=scriptname2.nxd
2Run=Yes

3=scriptname3.nxd $Variable=value
3Run=Yes
```

Each script is controlled by a pair of entries. The keywords for these entries are a sequential number starting at 1 and the same number with the word Run appended to it (no spaces). The first entry defines the name of the N_Dist script. The second controls if the script is to be run or skipped. The valid entries for this value are Yes or No.

It is also possible to set the value of variables to pass to script at run-time. See item 3 in the above example.

A practical example of this would be a site that has a mixed workstation environment and, has **Norman Virus Control** implemented on them all. The INI-file would look like this:

MAIN.INI

```
[scripts]

1=win31.nxd
1Run=Yes

2=win95.nxd
2Run=Yes

3=winnt.nxd
3Run=Yes
```

The master N_Dist script is as follows:

```
NXD BEGIN

; Set script variables
 set $Total='1'

; Get name of script from INI-file
#GetName
 getini $Run $Startpath+'\main.ini' [scripts] $Total
 if $Run=''
      goto #End

; Check if script is to run
 getini $Exec $Startpath+'\main.ini' [scripts] $Total+'Run'
 if $Exec='Yes'
      run $Startpath+'\n_dist.exe' $Startpath+'\'+$Run
 increment $Total
 goto #GetName

 #End

 NXD END
```

How the script works

❖ The variable $Total is set to '1' at the top of the script. This variable is a counter that is incremented each iteration of the script.

❖ In the #GetName section, GETINI is used to obtain the name of the script to run from the INI-file and store it in a variable named $Run.

- The next line checks the content of the variable $Run. If it is a nul string, processing is transferred to the #End label and the script quits. This is a completion test.
- GETINI is used again to determine if the script is to be run by obtaining the second value from the INI-file and storing it in a variable named $Exec. If the value of $Exec is 'Yes', the script is run, otherwise it is skipped.
- The counter stored in the variable named $Total is incremented by 1 and processing loops back to the #Getname label.
- This process continues until all scripts defined in the INI-file have been run, or not run as the case may be, at which time the value of the variable $Run will be a nul string and the script will quit.

❖

This topic details how to initiate a script from a NetWare or Windows NT server.

NetWare:

Add a line to the NetWare login script conforming to the following:

```
#server_name/volume:\path\n_dist.exe \\server_name\volume\path\script_name.nxd /q
```

The first part of the command line uses NetWare script conventions so that NetWare can find and execute N_Dist.Exe. The second part uses UNC path naming conventions as this is passed to N_Dist.Exe and must be in this format for N_Dist to interpret. An example follows:

```
#prod_apps/sys:\public\nvcadmin\ndist\n_dist.exe \\prod_apps\sys\public\nvcadmin\ndist\nvc95.nxd /q
```

Windows NT:

Add a line to the Windows NT login script conforming to the following:

```
\\server_name\path\n_dist.exe \\server_name\path\script_name.nxd /q
```

Both parts of the command line use UNC path naming conventions as both Windows NT and N_Dist expect this format. The path can be a share name as in the following example:

```
\\prod_apps\nvcadm$\ndist\n_dist.exe \\prod_apps\nvcadm$\ndist\nvc95.nxd /q
```

Notes:

- In both of the above examples you may use drive letters, however the same drive must be mapped by all users for this approach to work.
- UNC path naming conventions are recommended as this provides greater flexibility than hard coded paths.
- If UNC path names are used to initiate N_Dist, use the same convention for setting the $Source variable in the N_Dist script. The syntax is: (followed by an example)

  NetWare:

  ```
  set $Source = '\\server_name\volume\path'
  ```

  ```
  set $Source = '\\prod_apps\sys\public\nvcadmin'
  ```

  Windows NT:

  ```
  set $Source = '\\server_name\path'
  ```

  ```
  set $Source = '\\prod_apps\nvcadm$'
  ```

  In this example, a share name has been used to define the path.
- If UNC path naming conventions are used in both the server and N_Dist scripts, no drive mapping is necessary to run a server based install/update using N_Dist. The exception to this is a DOS/Windows workstation which requires a drive to be mapped.

Purpose:

To search the environment for the specified value and return the complete path (without the "=") in the specified variable.

Syntax:

```
SEARCH $variable ENVIRONMENT 'string'
```

Where:

$variable : is the name of the variable used to store the result.

ENVIRONMENT : is the command to cause N_Dist to search the environment for the 'string'

'string' : is the string to search for.

Example:

```
SEARCH $Result ENVIRONMENT 'buffers'
```

N_Dist would search for the value associated with the DOS environment variable 'buffers' and store the value in the variable $Result.

Notes:

When using this command, the word ENVIRONMENT must be used.

Purpose:

To search for a file in a directory tree and store the complete path in the specified variable.

Syntax:

```
SEARCH $variable startpath filename (case)
```

Where:

$variable : is the name of the variable used to store the result.

startpath : is the path to start the search from.

filename : is the name of the file to search for.

case : is the (optional) parameter to make the search case sensitive.

Example:

```
SEARCH $Result c:\*.* win.ini
```

This would search for the file win.ini in all of c:, and the complete path for win.ini would be returned in the variable $Result.

Notes:

By default the search is case insensitive.

Purpose:

To search for a line within a file that contains up to three phrases. Returns the complete line in the specified variable.

Syntax:

```
SEARCH $variable file 'phrase1' 'phrase2' 'phrase3'
```

Where:

$variable : is the name of the variable to store the result.

file : is the name of the file to search.

phrase1 to 3 : is the phrase(s) to search for. Phrase2 and 3 are optional.

Example:

```
SEARCH $Result config.sys 'device' '=' 'nvc.sys'
```

Stores device=c:\norman\dos\nvc.sys in the variable named $Result.

Notes:

- The search returns the complete line and stores it in the defined variable.
- If a phrase to search for contains spaces, enclose the phrase in single-quotation marks (').
- The phrase is case insensitive.

Purpose:

To assign a value to a specified variable.

Syntax:

SET $variable value

Where:

$variable	:	is the name of the variable.
value	:	is the value to assign to the variable.

Example:

SET $InstallPath='c:\norman'

Stores the value c:\norman in the variable named $InstallPath.

Notes:

The variable may also be one of the special variables.

Purpose:

To insert or change a parameter in an INI-file.

Syntax:

```
SETINI ini-file section keyword value
```

Where:

ini-file : is the name of the INI-file.

section : is the section name in the INI-file.

keyword : is the keyword in the INI-file.

value : is the value associated with the keyword in the INI-file.

Example:

```
SETINI win.ini 'windows' 'load' 'c:\norman\nvcsys.exe'
```

This would replace everything in the 'load' line in the [windows] section of WIN.INI with c:\norman\nvcsys.exe.

Notes:

If the section and/or file does not exist, it is created.

❖

Purpose:

To set or replace a parameter in the registry (Windows 95 and Windows NT only).

To set or replace a parameter in the OS/2 system or user profiles.

Syntax:

for Windows 95 and Windows NT

```
SETREGISTRY 'key' 'parameter' 'value'
```

for OS/2

```
SETREGISTRY user/system application key value
```

Where:

user : defines the user profile (OS2.INI).

system : defines the system profile (OS2SYS.INI).

application : is the name of the application. (OS/2)

key : is the registry/profile key to set or replace.

parameter : is the registry parameter.

value : is the value to set or replace.

Example:

```
SETREGISTRY 'HKEY_CURRENT_USER\Environment' 'TEMP' '%SystemDrive%\TEMP'
```

This would set the value of 'TEMP' to '%SystemDrive%\TEMP'.

Notes:

❖ Only variables of type 'string' may be created or changed.

❖ Use with caution. Invalid entries may render the system inoperable.

This version of Server to Server Install/Update is capable of running multiple sub-scripts to achieve the Install/Update of many applications.

The Install/Update routine is performed from the System Administrators workstation. This can be initiated manually or automatically via a scheduling program.

This process assumes that the files to be installed on target servers have already been installed on the central server from where they will be accessed.

The technique used is to store all details of the servers in an INI-file and have the N_Dist script iterate as many times as there are servers defined. During each iteration, the INI-file is checked for how many scripts to run.

The INI-file is in the following format:

SERVERS.INI

```
[server]
nName=<Server_Name>
nPath=<Path>
nUpdate=<Yes/No>

[scripts]
nName=<Scriptname.Nxd>
nSource=<Path>
nRoot=<Root>
nRun=<Yes/No>
```

Each server is defined in the [server] section by a set of four keywords with associated values, they are:

Keyword	**Value**
nName	The name of the target server.
nPath	The path on the target server. This may be a share name on a Windows NT server. When defining this entry for a NetWare server, use the volume name and path eg. Sys\Public. The nName and nPath entries are concatenated in the script to create a UNC path, for example, \\Prod_Apps\Sys\Public. For a Windows NT server, where a share has been created for the NvcAdmin directory, it might look like this: \\Prod_Data\Nvc$
nUpdate	Defines whether or not the target server is to be updated. Valid entries are Yes or No. This allows the System Administrator to selectively update servers if necessary.

Where 'n' is a sequential number starting at 1 for each target server set.

The scripts to be run on each server are defined in the [scripts] section by a set of four keywords with associated values, they are:

Keyword	**Value**
nName	The name of the sub-script to run.
nSource	The path to the application files on the central server.
nRoot	The root directory of the application on the target server. This is concatenated to the path defined in the [server] section at run-time. This allows flexibility in defining the target location on each server. Each server may store applications in a different location in relation to the root of the server drive.
nRun	Defines if the script is to be run.

Where 'n' is a sequential number starting at 1 for each script set.

A typical SERVERS.INI file might look like this:

SERVERS.INI

```
[server]
1Name=Prod_Apps
1Path=Sys\Public
1Update=Yes

2Name=Prod_Data
2Path=NvcAdm$
2Update=Yes

3Name=Develop
```

```
3Path=Programs
3Update=Yes

[scripts]
1Name=Nvc.Nxd
1Source=\\NT_Server\NvcAdm$
1Root=NVCADMIN
1Run=Yes

2Name=Nac.Nxd
2Source=\\NT_Server\NacAdm$
2Root=NACADMIN
2Run=Yes
```

The following example N_Dist master and sub-scripts will install/update **Norman Virus Control** and **Norman Access Control** on servers that are defined in SERVERS.INI and have the nUpdate and nRun values set to Yes.

It is assumed that the System Administrator has installed **Norman Virus Control** and **Norman Access Control** on the central server and that the files SERVERS.INI, SERVERS.NXD and SERVERS.BAT are in the NRMADMIN directory on the central server together with the N_Dist files.   The installation referred to here is an Administrator Install (setup.exe /a)

The DOS batch file SERVERS.BAT, is used to initiate the process.

SERVERS.BAT

```
@echo off
   n_dist.exe servers.nxd /q
```

SERVERS.NXD

```
NXD BEGIN

; Set script variables

 set $Count='1'
 set $Total='1'

; Initilize Log file.
 if exist $Startpath+'\SERVERS.LOG'
      delete $Startpath+'\SERVERS.LOG'
 insert $Startpath+'\SERVERS.LOG' beginning 'SERVER INSTALL/UPDATE REPORT'
 insert $Startpath+'\SERVERS.LOG' end 'Started:   '+%S
 insert $Startpath+'\SERVERS.LOG' end '--------------------------------------------------------'

; Check if server is to be updated.
#CheckUpd
 getini $Update $Startpath+'\servers.ini' [server] $Count+'Update'
 if $Update=''
      goto #End
 if $Update='Yes'
      goto #GetName
 increment $Count
 goto #CheckUpd

; Get name of server to update.
#GetName
 getini $Server $Startpath+'\servers.ini' [server] $Count+'Name'
 goto #GetPath

; Get path on server.
#GetPath
 getini $Path $Startpath+'\servers.ini' [server] $Count+'Path'
 goto #Paths

; Setup paths for target server.
#Paths
 set $Target='\\'+$Server+'\'+$Path+'\NVCADMIN'
 if !exist '\\'+$Server+'\'+$Path
      goto #Fail
 goto #GetScript

; Record entry in log file if server not available.
#Fail
 insert $Startpath+'\SERVERS.LOG' end %T+' Server "'+$Server+'" was not available for update.'
 increment $Count
 goto #CheckUpd

; Get name of script to run.
#GetScript
 getini $Script $Startpath+'\servers.ini' [scripts] $Total+'Name'
 if $Script=''
      goto #NextServer
 goto #ChkRun

; Check if script is to run.
#ChkRun
 getini $Run $Startpath+'\servers.ini' [scripts] $Total+'Run'
 if $Run='Yes'
      goto #RunScript
 goto #GetNext

; Run the script.
#RunScript
 run $Startpath+'\N_DIST.EXE' $Startpath+'\'+$Script+' $Target='+$Target+' $Server='+$Server+' $Total='+
$Total
 goto #GetNext

; Get the next script to run.
#GetNext
 increment $Total
 goto #GetScript

; Loop back and get next server.
#NextServer
 set $Total='1'
 increment $Count
 goto #CheckUpd

; Complete and display log file.
#End
 insert $Startpath+'\SERVERS.LOG' end '-------------------------------------------------------'
 insert $Startpath+'\SERVERS.LOG' end 'Finished: '+%S
 insert $Startpath+'\SERVERS.LOG' end '                    --- End of Report ---'
 run $Windir+'\NOTEPAD.EXE' $Startpath+'\SERVERS.LOG' nowait

NXD END
```

NVC.NXD

```
NXD BEGIN

 ; Set script variables
  getini $Source $Startpath+'\servers.ini' [scripts] $Total+'Source'
  getini $Root $Startpath+'\servers.ini' [scripts] $Total+'Root'
  set $Target=$Target+'\'+$Root
  goto #Update

 ; Update install/files on server.
 #Update
  display 'Updating Norman Virus Control on Server: '+$Server

  if !exist $Target+'\DOS'
       makedir $Target+'\DOS'
  if !exist $Target+'\WIN95'
       makedir $Target+'\WIN95'
  if !exist $Target+'\NDIST'
       makedir $Target+'\NDIST'

  copy $Source+'\*.*'           $Target+'\*.*'           update
  copy $Source+'\DOS\*.*'     $Target+'\DOS\*.*'     update
  copy $Source+'\WIN95\*.*' $Target+'\WIN95\*.*' update
  copy $Source+'\NDIST\*.*' $Target+'\NDIST\*.*' update
  insert $Startpath+'\SERVERS.LOG' end %T+' NVC on server "'+$Server+'" was updated.'
  goto #End

 #End

 NXD END
```

NAC.NXD

```
NXD BEGIN

 ; Set script variables
  getini $Source $Startpath+'\servers.ini' [scripts] $Total+'Source'
  getini $Root $Startpath+'\servers.ini' [scripts] $Total+'Root'
  set $Target=$Target+'\'+$Root
  goto #Update

 ; Update install/files on server.
 #Update
  display 'Updating Norman Access Control on Server: '+$Server

  if !exist $Target+'\CONTROL'
       makedir $Target+'\CONTROL'
  if !exist $Target+'\HELP'
       makedir $Target+'\HELP'
  if !exist $Target+'\NDIST'
       makedir $Target+'\NDIST'
  if !exist $Target+'\SCRIPT'
       makedir $Target+'\SCRIPT'
  if !exist $Target+'\SETUP'
       makedir $Target+'\SETUP'

  copy $Source+'\CONTROL\*.*' $Target+'\CONTROL\*.*' update
  copy $Source+'\HELP\*.*'      $Target+'\HELP\*.*'      update
  copy $Source+'\NDIST\*.*'     $Target+'\NDIST\*.*'     update
  copy $Source+'\SCRIPT\*.*'   $Target+'\SCRIPT\*.*'   update
  copy $Source+'\SETUP\*.*'     $Target+'\SETUP\*.*'     update
  insert $Startpath+'\SERVERS.LOG' end %T+' NAC on server "'+$Server+'" was updated.'
  goto #End

 #End

 NXD END
```

How the scripts work

SERVERS.NXD

- The variables $Count and $Total are set to '1'. These variables are counters that are incremented each iteration of the script.
- The log file (SERVERS.LOG) is deleted if it exists, and a new version is created by inserting three line as a header which includes the date and time the install/update started by using the special variable %S.
- In the #CheckUpd section, the GETINI command is used to get the associated value of the nUpdate keyword in SERVERS.INI and store it in a variable named $Update. This determines whether or not the target server is to be updated.
- The next line checks the content of the variable $Update. If it is a nul string, processing is transferred to the #End label and the script quits. This is a completion test.
- The value of the $Update variable is tested again, this time to determine if it is set to Yes. If it is Yes, then processing is transferred to the #GetName label. If it is set to No, the variable $Count is incremented by 1 and processing loops back to the #CheckUpd label where the next server is processed.
- The #GetName section uses the GETINI command to get the name of the target server and store it in a variable named $Server.
- The #GetPath section uses the GETINI command to get the path on the target server and store it in a variable named $Path.
- The last line in the #GetPath section transfers processing to the $Paths label.
- The #Paths section uses the SET command to set the $Target variable to the location of the target server.
- Following this is a test to determine if the target server is available. If the test fails, processing is transferred to the #Fail label. If the test succeeds, processing is transferred to the $GetScript label.
- The #Fail section uses the INSERT command to append a line to the log file recording the fact that the server was not available.
- The #GetScript section uses the GETINI command to get the name of the script to run from SERVERS.INI and store it in a variable named $Script.
- The next line checks the content of the variable $Script. If it is a nul string, there are no more scripts to run and processing is transferred to the #NextServer label. If it contains the name of a script, processing is transferred to the #ChkRun label.
- The #ChkRun section uses the GETINI command to determine if the script is to be run. The value returned is stored in the variable named $Run.
- The next line checks the content of the variable $Run. If it is Yes, processing is transferred to the label #RunScript. If it is No, processing is transferred to the label #GetNext.
- The #RunScript section uses the RUN command to run N_Dist with the sub-script stored in the variable $Script and defines the variables $Target, $Server and $Total on the command line. This technique passes the variables to the sub-script.

- The #GetNext section increments the counter $Total, then loops back to the label #GetScript where the next script to process is determined.
- The #NextServer section resets the counter stored in the variable $Total back to 1, increments the counter $Count. Processing is then transferred to the label #CheckUpd where the next server to process is determined.
- The #End section appends three lines to the log file including the date and time the install/update finished by using the special variable %S.
- Finally, the RUN command is used to load Windows Notepad and display the log file. The script will terminate as the nowait parameter has been used with the run command.
- The script will continue until all servers and scripts defined in SERVERS.INI have been processed at which time the value of the variable $Update will be a nul string and the script will quit.

NVC.NXD and NAC.NXD

- The GETINI command is used to get the name of the source directory on the central server and store it in a variable named $Source.
- The GETINI command is used again to get the name of the application root directory on the target server and store it in a variable named $Root.
- The SET command concatenates the $Target variable (passed by SERVERS.NXD) to the $Root variable. This creates the full path on the target server.
- The #Update section displays a message showing the name of the current server, then creates the directories on the target server using the MAKEDIR command, if they do not exist.
- The COPY command is used to copy the application files from the central server to the target server only if they do not exist or are older than those on the central server.
- The INSERT command is used to append a line to the log file recording that the server was updated.

To implement the Install/Update

- Perform an Administrator install (setup.exe /a) of **Norman Virus Control** and **Norman Access Control** on the central server.
- Configure the INI-file (SERVERS.INI) to reflect the target servers and the scripts to run.
- Initiate the Install/Update from the System Administrators workstation by running SERVERS.BAT.

Administrator tasks required for a subsequent update

- Perform an Administrator install (setup.exe /a) of **Norman Virus Control** or **Norman Access Control** (or both) on the central server.
- Initiate the Install/Update from the System Administrators workstation by running SERVERS.BAT.

Notes:

- It is assumed that each target server will have the necessary line(s) present in the login script to initiate the workstation install/update.
- After a Server to Server Install/Update, the target servers will update each workstation as the user logs on.
- The System Administrator must have access to all target servers that are to be updated.
- As of this writing (Jan, 98) the production version of the Administrator install of **Norman Access Control** is not available. It will be supplied in a future release or service pack.

For true centralized management to be a reality, a Server to Server Install/Update routine must be implemented. This topic covers installing/updating software over multiple servers running Windows NT or Novell NetWare.

The Install/Update routine is performed from the System Administrators workstation. This can be initiated manually or automatically via a scheduling program.

This process assumes that the files to be installed on target servers have already been installed on the central server from where they will be accessed.

The technique used is to store all details of the servers in an INI-file and have the N_Dist script iterate as many times as there are servers defined.

The INI-file is in the following format:

SERVERS.INI

```
[server]
nName=<Server_Name>
nPath=<Path>
nUpdate=<Yes/No>
```

Each server is defined in the [server] section by a set of four keywords with associated values, they are:

Keyword	Value
nName	The name of the target server.
nPath	The path on the target server. This may be a share name on a Windows NT server. When defining this entry for a NetWare server, use the volume name and path eg. Sys\Public. The nName and nPath entries are concatenated in the script to create a UNC path, for example, \\Prod_Apps\Sys\Public\NvcAdmin. For a Windows NT server, where a share (NvcAdm$) has been created for the NvcAdmin directory, it might look like this: \\Prod_Data\NvcAdm$
nUpdate	Defines whether or not the target server is to be updated. Valid entries are Yes or No. This allows the System Administrator to selectively update servers if necessary.

Where 'n' is a sequential number starting at 1 for each target server set.

A typical SERVER.INI file might look like this:

SERVERS.INI

```
[server]
1Name=Prod_Apps
1Path=Sys\Public
1Update=Yes

2Name=Prod_Data
2Path=NvcAdm$
2Update=Yes

3Name=Develop
3Path=Programs
3Update=Yes
```

The following example N_Dist script will install/update **Norman Virus Control** on any servers that are defined in SERVERS.INI and have the nUpdate value set to Yes.

It is assumed that an Administrator install (setup.exe /a) of **Norman Virus Control** has been performed on the central server and that the files SERVERS.INI, SERVERS.NXD and SERVERS.BAT are in the NVCADMIN\NDIST directory on the central server.

The DOS batch file SERVERS.BAT, is used to initiate the process.

SERVERS.BAT

```
@echo off
  n_dist.exe servers.nxd /q
```

SERVERS.NXD

```
 NXD BEGIN

; Set script variables
 set $Quote='"'
 set $Count='1'
 set $Source='\\<Server_Name\<Path>'

; Initilize Log file.
 if exist $Startpath+'\SERVERS.LOG'
      delete $Startpath+'\SERVERS.LOG'
 insert $Startpath+'\SERVERS.LOG' beginning 'SERVER INSTALL/UPDATE REPORT'
 insert $Startpath+'\SERVERS.LOG' end 'Started:   '+%S
 insert $Startpath+'\SERVERS.LOG' end '--------------------------------------------------------'

; Check if server is to be updated.
#CheckUpd
 getini $Update $Startpath+'\servers.ini' [server] $Count+'Update'
 if $Update=''
      goto #End
 if $Update='Yes'
      goto #GetName
 increment $Count
 goto #CheckUpd

; Get name of server to update.
#GetName
 getini $Server $Startpath+'\servers.ini' [server] $Count+'Name'
 goto #GetPath

; Get path on server.
#GetPath
 getini $Path $Startpath+'\servers.ini' [server] $Count+'Path'
 goto #Paths

; Setup paths for target server.
#Paths
 set $Target='\\'+$Server+'\'+$Path+'\NVCADMIN'
 if !exist '\\'+$Server+'\'+$Path
      goto #Fail
 goto #Update

; Record entry in log file if server not available.
#Fail
 insert $Startpath+'\SERVERS.LOG' end %T+' Server "'+$Server+'" was not available for update.'
 increment $Count
 goto #CheckUpd

; Update/install files on server.
#Update
 display 'Updating Server: '+$Server
 if !exist $Target+'\DOS'
      makedir $Target+'\DOS'
 if !exist $Target+'\WIN95'
      makedir $Target+'\WIN95'
 if !exist $Target+'\NDIST'
      makedir $Target+'\NDIST'
 copy $Source+'\*.*' $Target+'\*.*' update
 copy $Source+'\DOS\*.*' $Target+'\DOS\*.*' update
 copy $Source+'\WIN95\*.*' $Target+'\WIN95\*.*' update
 copy $Source+'\NDIST\*.*' $Target+'\NDIST\*.*' update
 insert $Target+'\NDIST\NVC95.NXD' beginning [N_Dist] nodup
 setini $Target+'\NDIST\NVC95.NXD' 'n_dist' 'Set $Source' $Quote+$Target+$Quote
 insert $Startpath+'\SERVERS.LOG' end %T+' NVC on server "'+$Server+'" was updated.'
 increment $Count
 goto #CheckUpd

;Complete and display log file.
#End
 insert $Startpath+'\SERVERS.LOG' end '--------------------------------------------------------'
 insert $Startpath+'\SERVERS.LOG' end 'Finished: '+%S
 insert $Startpath+'\SERVERS.LOG' end '                    --- End of Report ---'
 run $Windir+'\NOTEPAD.EXE' $Startpath+'\SERVERS.LOG' nowait

NXD END
```

How the script works

- The variable $Quote is set at the top of the script.   This variable is used to place a literal quote (') in a command line later in the script.

- The variable $Count is set to '1'   This variable is a counter that is incremented each iteration of the script.
- The variable $Source is set to point to the location of the files on the central server.
- The log file (SERVERS.LOG) is deleted if it exists, and a new version is created by inserting three line as a header which includes the date and time the install/update started by using the special variable %S.
- In the #CheckUpd section, the GETINI command is used to get the associated value of the nUpdate keyword in SERVERS.INI and store it in a variable named $Update.   This determines whether or not the target server is to be updated.
- The next line checks the content of the variable $Update.   If it is a nul string, processing is transferred to the #End label and the script quits.   This is a completion test.
- The value of the $Update variable is tested again, this time to determine if it is set to Yes.   If it is Yes, then processing is transferred to the #GetName label.   If it is set to No, the variable $Count is incremented by 1 and processing loops back to the #CheckUpd label where the next server is processed.
- The #GetName section uses the GETINI command to get the name of the target server and store it in a variable named $Server.
- The #GetPath section uses the GETINI command to get the path on the target server and store it in a variable named $Path.
- The last line in the #GetPath section transfers processing to the $Paths label.
- The #Paths section uses the SET command to set the $Target variable to the location of the target server.
- Following this is a test to determine if the target server is available.   If the test fails, processing is transferred to the #Fail label.   If the test succeeds, processing is transferred to the $Update label.
- The #Fail section uses the INSERT command to append a line to the log file recording the fact that the server was not available.
- The $Count variable is incremented by 1, and processing loops back to the #CheckUpd label where the next server is processed.
- The #Update section displays a message showing the name of the current server, then creates the directories on the target server using the MAKEDIR command, if they do not exist.
- The COPY command is used to copy all of the NVC files from the central server to the target server only if they do not exist or are older than those on the central server.
- The INSERT command is used to add a section header [N_Dist] to the top of NVC95.NXD on the target server.   This effectively creates an INI-file which allows the file to edited using the SETINI command.   N_Dist will ignore anything prior to the NXD BEGIN line so it has no effect.
- The SETINI command is used to set the $Source variable in NVC95.NXD on the target server to point to the target server.   The original version copied would have the central server defined as the source.
- The INSERT command is used to append a line to the log file recording that the server was updated.
- The variable count is incremented by 1 and processing loops back to the #CheckUpd label where the next server is processed.
- The #End section appends three lines to the log file including the date and time the install/update finished by using the special variable %S.
- Finally, the RUN command is used to invoke Windows Notepad and display the log file.   The script will terminate as the nowait parameter has been used with the run command.
- The script will continue until all servers defined in SERVERS.INI have been processed at which time the value of the variable $Update will be a nul string and the script will quit.

To implement the Install/Update

- Perform an Administrator install (setup.exe /a) of **Norman Virus Control** on the central server.
- Configure the INI-file (SERVERS.INI) to reflect the target servers.
- Set the $Source variable at the top of SERVERS.NXD to point to the location of the NVC files on the central server.
- Initiate the Install/Update from the System Administrators workstation by running SERVERS.BAT.

Administrator tasks required for a subsequent update

- Perform an Administrator install (setup.exe /a) of **Norman Virus Control** on the central server.
- Initiate the Install/Update from the System Administrators workstation by running SERVERS.BAT.

Notes:

- It is assumed that each target server will have the necessary line(s) present in the login script to initiate the workstation install/update.
- After a Server to Server Install/Update, the target servers will update each workstation as the user logs on.
- The System Administrator must have access to all target servers that are to be updated.

❖

Script variables can be set from within the script or on the command line.   To set variables from within the script define the values to be assigned at the top of the script as in the following example.

```
NXD BEGIN

set $Source=F:\PUBLIC\NVCADMIN
set $Target=C:\NORMAN
...
```

Variables can also be set/reset anywhere within the script.

To set variables on the command line add the variables as command-line parameters as in the following example:

```
<Path>\N_Dist.exe <Path>\Filename.Nxd $Variable1='value' $Variable2='value' ... ...
```

The values assigned on the command line may include DOS batch file replaceable parameters if the script is initiated by a batch file.   The following example shows how a batch file can pass its command line parameters to an N_Dist script.

DEMO.BAT

```
@echo off
cls
N_Dist.exe Demo.Nxd $Var1=%1 $Var2=%2 $Var3=%3 $Var4=%4
```

DEMO.NXD

```
NXD BEGIN

display $Var1+' '+$Var2+' '+$Var3+' '+$Var4

NXD END
```

If DEMO.BAT is run with the following command line parameters:

```
DEMO.BAT Norman Data Defense Systems
```

The N_Dist script (DEMO.NXD) would display Norman Data Defense Systems.

This is a simple example, but could be extended to include environment variables or any of the Novell® variables if the script was run from the login script of a NetWare® server.

❖

The following variables are used by N_Dist to store various values. They should not be reset by the script unless indicated.

$Error Is either 'TRUE' or 'FALSE'. The value is cleared by the next instruction executed. The value is set to 'FALSE' if a line in the script is executed unsuccessfully.

$Key Contains the key value after a 'Wait' command without a time-delay.

$Returnvalue Contains the return value from a routine called by the 'Run' command.

$Startpath Contains the path to the location where N_Dist was started.

$System Is either 'WIN16', 'WIN95', 'WINNT', 'OS2' or 'DOS', depending on platform/program environment.

$Updated The initial value is 'NO'. The $Updated variable will be set to 'YES' if a 'copy/update' actually performs an update of a file. The $Updated variable must be reset manually by the 'SET' command during the execution of the script.

$Windir Contains the windows root directory. Example: 'C:\WINDOWS\'

$Wininidir This variable is a working copy of $Windir and may be reset by the script.

$Winsysdir Contains the path to the Windows system directory. Example: 'C:\WINDOWS\SYSTEM\'

❖ The following must be entered in upper case:

%T Time variable that resolves to <hh:mm:ss>. eg. 15:19.32

%D Date variable that resolves to <day month date year>. eg. Monday July 21 1997

%S This variable resolves to <day month date year hh:mm:ss>. eg. Monday July 21 1997 15:19.32

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems Pty Ltd - Australia

Phone:	+61 3 9558 9011
Fax:	+61 3 9558 9144
E-mail:	support@norman.com.au

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems (UK) Ltd - England

Phone:	+44 1908 847 410
Fax:	+44 1908 847 552
E-mail:	norman@normanuk.com

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Ibas Oy - Finland

Phone:	+358 9 47746 11
Fax:	+358 9 47746 120
E-mail:	jyri.pohja@norman-ibas

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems GmbH - Deutschland

Phone:	+49 212 267 180
Fax:	+49 212 267 1815
E-mail:	norman@norman.de

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems BV - The Netherlands

Phone:	+31 23 563 39 60
Fax:	+31 23 561 31 65
E-mail:	info@norman.nl

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems AS - Norway

Phone: +47 67 58 99 30
Fax: +47 67 58 99 40
E-mail: norman@norman.no

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems AG - Sweden

Phone: +46 8 728 33 50
Fax: +46 8 728 33 52
E-mail: anders.schyllert@verisecure.se

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems AG - Switzerland

Phone:	+41 61 487 2500
Fax:	+41 61 487 2501
E-mail:	norman@norman.ch

Other Locations

Web Sites

To obtain product support please contact your nearest **Norman** office.

Norman Data Defense Systems Inc. - USA

Phone:	+1 703 573 8802
Fax:	+1 703 573 3919
E-mail:	norman@norman.com

Other Locations

Web Sites

Purpose:

To remove a folder/group and/or program icons from a workstation's desktop.

Syntax:

In a Windows 3.x environment:

```
UNREGISTER 'group name' path 'icon name'
```

In a Windows 95/NT environment:

```
UNREGISTER 'folder name' 'icon name' (common)
```

Where:

folder : is the name of the folder. (Windows 95/NT)

group : is the name of the group. (Windows 3.x)

path : is the path to the executable item in the new folder or group. (Windows 3.x)

icon name : is the name icon to remove. (Windows 3.x/95/NT)

common : is the (optional) parameter to cause N_Dist to remove the link from the common area. (Windows 95/NT)

Examples:

In a Windows 3.x environment:

```
UNREGISTER 'Norman' c:\norman\win16\nvcw.exe 'Nvc'
```

In a Windows 95/NT environment:

```
UNREGISTER 'Norman' 'Nvc'
```

In both examples, the icon 'Nvc' will be removed from the 'Norman' group.

Notes:

If no icon name is specified, then the folder will be deleted only if no icons are present. (Windows 95/NT only)

In a Windows 3.x environment, the UNREGISTER command will remove icons from a group, but will not delete and empty group.

In Windows 95/NT environments, the folder 'startup' is automatically resolved to the correct startup folder name for the current langauge.

This topic covers using environment variables in a Windows NT Server login script as well as passing environment variables to an N_Dist script.

Windows NT sets a range of environment variables automatically.   Some of these relate to the number of processors, the processor type etc.   The environment variables referred to here are those that would of most use to a System Administrator in building a script, they are:

Variable	Typical value
HomeDrive	H:
HomePath	\User_Name
HomeShare	\\Server_Name\Users
LogonServer	\\Server_Name
OS	Windows_NT
SystemDrive	C:
SystemRoot	C:\WINNT
Temp	C:\TEMP
Tmp	C:\TEMP
UserDomain	Domain_Name
UserName	User_Name
UserProfile	C:\WINNT\Profiles\User_Name
windir	C:\WINNT

The first example uses environment variables to control processing in a Windows NT Server login script.

LOGIN.BAT

```
@echo off
 cls
 if "%os%" == "Windows_NT" goto WINNT
 if not "%windir%" == "" goto WIN95
 if "%windir%" == "" goto WIN3X

:WINNT
 rem ----------------------------------------------------------
 rem NVC Update for Win NT workstations
 rem ----------------------------------------------------------
 \\NTHOST\NvcUpd$\NDIST\N_Dist.exe \\NTHOST\NvcUpd$\NDIST\NVCNT.NXD /q
 goto END

:WIN95
 rem ----------------------------------------------------------
 rem NVC Update for Win 95 workstations
 rem ----------------------------------------------------------
 \\NTHOST\NvcUpd$\NDIST\N_Dist.exe \\NTHOST\NvcUpd$\NDIST\NVC95.NXD /q
 goto END

:WIN3X
 rem ----------------------------------------------------------
 rem NVC Update for Win 31 workstations
 rem ----------------------------------------------------------
 net use o: \\NTHost\NT_Server
 O:\NVCADMIN\NDIST\N_Dist.exe NVCW.NXD /q
 net use o: /d
 goto END

:END
```

How the script works:

- The first if statement tests if the variable os contains the string Windows_NT. If true, goto the WINNT label where any commands to be run for a Windows NT workstation are processed.
- The second if statement tests if the variable windir is not empty. If true, goto the WIN95 label where any commands to be run for a Windows 95 workstation are processed.
- The third if statement tests if the variable windir is empty. If true, goto the WIN3X label where any commands to be run for a Windows 3.x workstation are processed.

Notes:

- The second and third if statements take advantage of the fact that the windir variable is present in both Windows 3.x and Windows 95 but, is only available for use by other than Windows in Windows 95.
- In a Windows 3.x environment this value would be empty if accessed from a batch file as in this example.

The next example passes the path to the currently logged in user's profile to the N_Dist script. To do this, set a variable on the N_Dist command line as follows:

```
\\NTHOST\NvcUpd$\NDIST\N_Dist.exe \\NTHOST\NvcUpd$\NDIST\NVCLNK.NXD $Profile=%USERPROFILE% /q
```

This can be used by the script to check for the existence of files in the user's profile path. The following example script checks if the shortcut to load the NVC scanner is present. If not create the shortcut.

```
NXD BEGIN

  set $Folder='Norman Virus Control'
  set $Icon_Nvc='Norman Virus Control'

 #CheckOS
  if $System ! 'WINNT'
      goto #End

  #CheckLink
  if !exist $Profile+'\Start Menu\Programs\Norman Virus Control\Norman Virus Control.lnk'
      register $Folder $Target+'\WIN32\NVCNT.EXE' $Icon_Nvc

  #End

NXD END
```

How the script works:

- The set commands define two variables ($Folder and $Icon_NVC) for use by the script.
- The #CheckOS section checks the operating system on the workstation. If it is not Windows NT, goto the #End label and quit processing.
- The #CheckLink section checks for the existence of the file Norman Virus Control.lnk in the user's profile. The variable $Profile (set on the command line) would evaluate to:

```
C:\WINNT\Profiles\JSmith
```

if the logged on user had the User Name of JSmith.

Purpose:

To wait for a keypress or a defined number of seconds.

Syntax:

WAIT n

Where:

n : is the number of seconds to wait.

Example:

WAIT 3

Will cause script execution to wait for 3 seconds.

Notes:

If a number is given, N_Dist will wait for that number of seconds.

The wait time cannot be overridden.

If no number is given, N_Dist will wait for a keypress and store the value of that key in the special variable called $Key.

Norman Data Defense Systems Pty Ltd - Australia
http://www.norman.com.au

Norman Ibas OY - Finland
http://www.norman-ibas.fi

Norman Data Defense Systems GmbH - Germany
http://www.norman.de

Norman Data Defense Systems BV - Netherlands
http://www.norman.nl

Norman Data Defense Systems AG - Sweden
http://www.verisecure.se

Norman Data Defense Systems AG - Switzerland
http://www.norman.ch

Norman Data Defense Systems AS - Norway
http://www.norman.no

Norman Data Defense Systems Inc - USA
http://www.norman.com

Click on any of the above links to open a **Norman** Web site.

[16-04-98] N_Dist version 1.36 released with version 4.50 of Norman Virus Control.

- New N_Dist command in version 1.36:

  POPMESSAGE

- The COPY command has been enhanced to support the 'deferred' option.
- The SEARCH (Phrase) command has been enhanced to support up to three phrases on a line in an ASCII file.
- The SETREGISTRY, GETREGISTRY and CLEARREGISTRY commands are now implemented in OS/2 where they operate on the user and system profiles (OS2.INI and OS2SYS.INI).
- The contact information has been updated and expanded.
- More Web sites have been added.

[08-03-98]

- The topic Adding Web Links to the Start Menu has been enhanced to include adding links to a workstation running Windows NT.
- Note added to the 'Running a Script from a Server' topic relating to the use of UNC paths in setting the $Source variable in an N_Dist script.
- The topic Using Environment variables has been added to the Script Techniques section.

[30-01-98] N_Dist version 1.32 released with version 4.35 of Norman Virus Control.

- New N_Dist commands in version 1.32:

  IF UPDATE

  RENAME

- The application example, that got around case sensitivity, has been deleted from the REMOVE topic as the command is now case insensitive.
- The /DEBUG command line option has been added to the OPTION topic.
- The following topics have been added to the Script Techniques section:

  Adding Web Links to the Start Menu

  Running a Script from a Server

- The Server to Server Install/Update topic has been updated. The script now use UNC paths for both NetWare and Windows NT servers resulting in a simpler solution. The copy of NVC95.NXD on the target server is now modified to point to the target server as the source of an update for the client workstations.
- The Server to Server (Multiple) Install/Update topic has been updated. The script now use UNC paths for both NetWare and Windows NT servers resulting in a simpler solution.

[09-12-97]

- The e-mail addresses in the 'How to contact Norman' topics are now dynamic links. Clicking on an e-mail address loads the configured e-mail client software with the address entered in the 'To' editbox.
- A new topic 'World Wide Web Sites' has been added. Like the e-mail addresses, these are dynamic links that invoke the default Browser and load the selected Norman web page.
- The REMOVE topic has had an Application example added which overcomes case sensitivity.
- The UNREGISTER command has been rewritten to reflect the differences between the Windows 3.x and Windows 95/NT environments.
- The DELETE topic has had an Application example added that explains how recursion works.

[19-11-97]

- The 'Command List' button has been added to the main window.
- More 'Related Topics' have been implemented.
- All topics have been checked for syntactical accuracy and corrected where necessary.
- Some topics have had extra examples and explanations added.
- The new variable $Startpath has been added to the Special Variables topic.
- The following topics have had Application examples added. This process is ongoing, more will be added over time.

  CUTWORD

  GETINI

  IF

  INCREMENT

  INSERT
- The following topics have been added to the Script Techniques section:

  Running Multiple Scripts

  Removing a group from PROGMAN.INI

  Server to Server Install/Update

  Server to Server (Multiple) Install/Update

  Building and Running a Batch File
- A new section Technical Information has been created with the following being added to it:

  Distributing NVC Configuration Settings