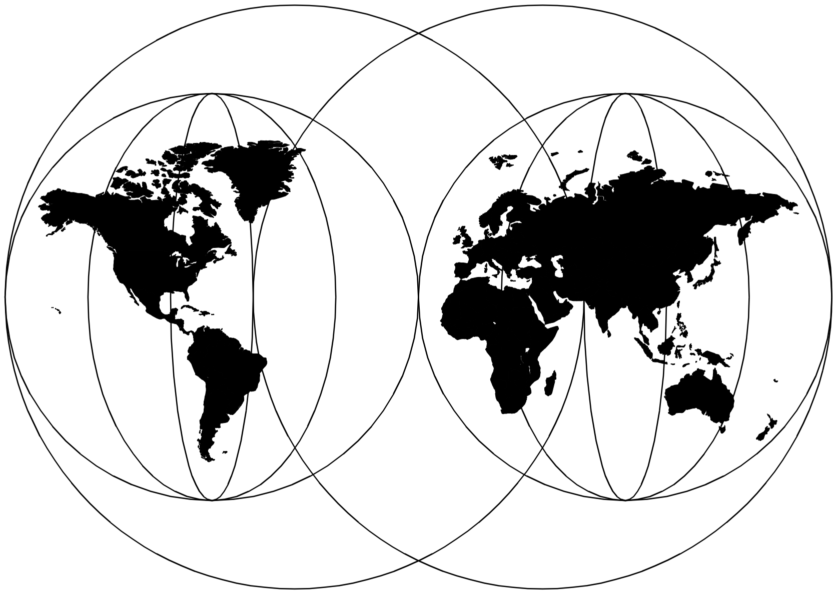
The OS/2 Warp 4 CID Rapid Deployment Tools Migration and Installation Scenarios

Uwe Zimmermann, Ramon Herrera, Jochen Meixner
Norma Angelica Ortiz, Sergio Pasqua



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2012-00

SG24-2012-00

International Technical Support Organization

The OS/2 Warp 4 CID Rapid Deployment Tools Migration and Installation Scenarios

April 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, "Special Notices" on page 433.

First Edition (April 1998)

This edition applies to OS/2 Warp 4 in a software distribution environment and to NetView Distribution Manager/2 (NVDM/2) with Database 2 (DB2) Version 2.11 for OS/2. This edition also applies to the new software distribution tools developed by the Rapid Deployment Team within the Network Computing Software Division (NCSD): Version-to-Version Tools and Enterprise Rescue Tools (Tribble).

Clarification: NCSD is a division with two parts under it: Personal Software Products, also known as PSP, and Network Software.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. DHHB Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook describes the CID (Configuration, Installation and Distribution) enhancements developed by the NCSD (Network Computing Software Division) Rapid Deployment Team (RDT) in Austin, Texas. It is also a handbook that provides step-by-step guidance in all phases of the usage of these new RDT Tools, such as Version-to-Version Tools and Tribble Tools. Tribble Tools are also known as Enterprise Rescue Tools.

The new CID software distribution method, which is a sophisticated cloning technique, allows you to migrate from previous OS/2 versions, such as OS/2 V2.1x and OS/2 Warp 3, to OS/2 Warp 4, as well as to accomplish pristine installation scenarios. We use these tools in an OS/2 LAN environment with NetView DM/2, our software distribution manager of choice.

Contacting the Rapid Deployment Team in Austin, TX, U.S.A.

This redbook ships with a CD-ROM that contains all files discussed in this redbook except the Rapid Deployment Tools. For information on how to obtain the Version-to-Version Tools and Enterprise Rescue Tools, and migration services provided by the Rapid Deployment Team, you need to access the following Web site:

`http://www.software.ibm.com/ncs/services`

Select the Network Computing Operating System Services link. Alternatively, you can contact the Rapid Deployment Team directly at 1-800-932-4777 (toll free within the United States) or through e-mail at:

`ncsdsvcs@us.ibm.com`

This document is intended for workstation specialists and system technical personnel responsible for mass distribution of OS/2 products in an OS/2 LAN. Some knowledge of LAN redirection principles and TCP/IP is assumed. This redbook solely focuses on the new RDT tools. If traditional CID information is needed or information about software distribution techniques other than those provided by the new RDT tools, you need to obtain the redbooks titled *The OS/2 Warp 4 CID Software Distribution Guide*, SG24-2010 and *OS/2 Installation Techniques: The CID Guide*, SG24-4295. The latter book concentrates on previous versions of OS/2 Warp 4.

This redbook provides a broad understanding of new software distribution features introduced with the Rapid Deployment Tools that were not previously available.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Uwe Zimmermann is an Advisory Systems Engineer at the International Technical Support Organization, Austin Center. He has has been working in heterogeneous networks for more than ten years. Before joining the ITSO, he worked in an IBM branch office in Stuttgart as a Networking Systems Engineer and was in charge of in charge of large account customers for more than five years. His areas of expertise include OS/2 LAN/Warp Server, Windows NT Server and Workstation, NetWare, Software Distribution (CID), dynamic TCP/IP, and Network Computing (NC).

Ramon Herrera is an Information Technology Consultant with QBE Insurance Ltd, Sydney, Australia. He is in charge of their operating environment (OS/2 Warp 4 and other applications), mass data distributions, WorkSpace On-Demand issues, and provides other solutions to meet the company's needs for the last three years. Prior to that, Ramon worked in the Health and Textile Manufacturing Industry doing a similar job and was involved with a variety of operating system environments, such as UNIX, VM/ESA, Windows NT, and Novell NetWare 3.x and 4.x. His main project has been related to the automatization of the drying processes of a textile manufacturing company, a Windows NT network deployment for a Sydney Hospital, and the automatization and deployment of an operating environment using OS/2.

Jochen Meixner is a Systems Engineer with IBM Germany, with three years of experience in OS/2, Lotus Notes, and other OS/2-related products. His areas of expertise also include VM, the Windows operating system family, and the REXX and C programming languages. He has extensively written on the migration scenarios.

Norma Angelica Ortiz is a Support Engineer with IBM Mexico. She has two years of experience in OS/2, LAN Server, Warp Server, Novell NetWare and has written extensively on migration scenarios. Her areas of expertise also include teaching classes on the OS/2 base operating system as well as on Novell NetWare.

Sergio Pasqua is a Systems Engineer with Banco do Brasil S.A., Brasilia, Brazil. He has been with Banco do Brasil since 1981, and he has been working with computer systems since 1979. His areas of expertise include Windows NT, Novell NetWare and OS/2 Warp Server. Inside the Banco do Brasil automation project, he is in charge of mass distribution of OS/2 Warp Server and OS/2 Warp 4 for 4000 branches.

Thanks to the following people for their invaluable contributions to this project:

Contributors at IBM sites:

Allen Gilbert, IBM Austin, OS/2 Architect

Kenneth Hubacher, IBM Austin, PSP Swat Team

Achal Khosla, IBM Austin, PSP Rapid Deployment Team

Didier Lafon, IBM France, PS Service Center (Creator of the CUBE utility)

Ken White, IBM Austin, PSP Rapd Deployment Team

International Technical Support Organization, Austin Center:

Marcus Brewer, Staff Information Developer and Senior Editor

Juergen Friedrichs, Advisory Software Engineer

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in Appendix , "ITSO Redbook Evaluation" on page 449 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM intranet users	`http://w3.itso.ibm.com`

- Send us a note at the following Internet address:

  `redbook@us.ibm.com`
  or, if you are an IBM Lotus Notes user, you may use the following address:

  `Redbook Feedback/Cary/IBM@IBMUS`

Redbooks Online

The "Redbooks Online!" page, is **the** source for finding complete redbooks on the World Wide Web. You can view the complete contents of our books. The books are arranged within the structure of the IBM Redbooks CD-ROM libraries. Clearly, we would like to see a CD-ROM or hardcopy book as part of your daily reference material. These online books help you understand what is on our CD-ROMs, what is available on hardcopy, and give you immediate access to the latest books that are not yet distributed on plastic or paper media.

Redbooks Online is on the following Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM intranet users	`http://w3.itso.ibm.com`

Tip of the Day

Have you ever wondered why the Network Applications folder only appears in OS/2 Warp 4's Connection's Network folder after successful logon? If you like to have this folder appear on the user's desktop, this is what you need to do:

1. Place a `PROFILE.CMD` file with the following contents in OS/2 Warp Server's \IBMLAN\DCDB\USERS\*<User ID>* directory:

   ```
   @START /C /MIN \\ITSCFS00\APPS\AdminTools\NWFldr.CMD
   ```

   where:

   `\\ITSCFS00` represents the server where the `APPS` alias is defined. `\AdminTools` is the directory that contains the `NWFldr.CMD` file.

   Note: Make sure that ACLs (access rights) are properly defined for the `APPS` alias.

2. As shown in Figure 1 on page xxv, create the `NWFldr.CMD` file which is executed out of `PROFILE.CMD`. This procedure basically waits for the Network Applications folder to be created in the Connections' Network folder, creates an object shadow, and places it on the user's desktop. The background color of the shadow object is defined as light yellow.

```
/* */
say "Placing your Network Applications folder object shadow on the desktop"

SleepTime=3
NumberOfTries=20

Call RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
Call SysLoadFuncs

Counter=0
do while (SysCreateShadow("<\NETAPPS\NETAPPFLDR>","<WP_DESKTOP>")==0 &
(Counter<NumberOfTries))
  call SysSleep SleepTime
  Counter=Counter+1
end
if Counter<NumberOfTries then do
  rc=SysMoveObject("<\NETAPPS\NETAPPFLDR>","<WP_NETWORK>")
  rc=SysSetObjectData("<\NETAPPS\NETAPPFLDR>","BACKGROUND=(none),,,C,255 255 168")
  rc=SysOpenObject("<\NETAPPS\NETAPPFLDR>","ICON","TRUE")
  rc=SysSetObjectData("<\NETAPPS\NETAPPFLDR>","ALWAYSSORT=YES")
  rc=SysSetObjectData("<\ITSCSV00\EXCEED>","MINWIN=DESKTOP")
  rc=SysSetObjectData("<\ITSCSV00\NOTEPAD>","PROGTYPE=PROG_31_ENHSEAMLESSCOMMON")
  rc=SysSetObjectData("<\ITSCSV00\ACROBAT>","ASSOCFILTER=*.PDF")
end
```

Figure 1. Create Network Applications Folder Object Shadow on the Desktop

Part 1. Common Information

Chapter 1. Illustrating Rapid Deployment Tools

This chapter provides detailed information about the OS/2 Warp 4 CID Rapid Deployment Tools, such as:

Version-to-Version Tools

This set of tools consists of the following components:

SYSCOPY	This image copying utility copies a donor system to a software distribution server / code server.
REPLICAT	This replication utility extracts the files created by SYSCOPY to target systems.
PROBE	This utility extracts system configuration information from either donor systems or target workstations that are being migrated from previous OS/2 versions to OS/2 Warp 4.
CONFIGUR	This configuration utility configures a target system based on input files created by PROBE.

Enterprise Rescue Tools

This set of tools consists of the following components:

CLNDESK	the desktop shuffler, which rearranges the desktop after the target workstation has been installed.
W4TRIM	the trimmer, which deletes files not being used at target workstations.

This chapter documents all these tools in a great substanza detail.

1.1 The SYSCOPY Tool

The SYSCOPY tool extracts partitions from a donor system for subsequent installation by the replicator utility on target systems of the same class. SYSCOPY outputs bundle files containing the contents of those partitions. It also outputs a control file containing information about the partition and bundles. The files are stored on a server for later migration. Use the replicator tool, REPLICAT, to transfer the contents of the donor partitions to various target systems.

Use the SYSCOPY tool as one step in the sequence of migrating OS/2 2.x and OS/2 3.x systems to installing customized OS/2 Warp 4 systems.

SYSCOPY extracts partitions from a donor system for subsequent installation by the replicator utility on target systems of the same class. SYSCOPY outputs bundle files containing the contents of those partitions. It also outputs a control file containing information about the partitions and bundles. The files are stored on a server for later migration. Use REPLICAT, the replicator, to transfer the contents of the donor partitions to various target systems.

You can use SYSCOPY part of a process to copy any partition holding operating system or application data.

SYSCOPY uses the INFOZIP utility which must be in the ZIP Utility at version 2.0 or later.

1.1.1 SYSCOPY Syntax

The SYSCOPY command supports the following parameters:

SYSCOPY Syntax

```
<drive:\path>SYSCOPY
              /C:<class_ID>
              /B:<SYSCOPY_bundle_file_directory>
              /S:<bundle_size_limit>
              /D:<c...z>
              /L:<log_file_directory>
              /X:<logging_level>
              /?
              ?
```

where:

`/?` or `?`	Displays a summary of SYSCOPY command line syntax. Any other parameter is ignored.
`/C:`*<class_ID>*	Required. A name (ID) to identify the group of partitions you want compressed and stored. The *class_ID* must be ASCII-7 characters and contain no blanks. If any path name specified with `/B:` or `/L:` is on a FAT partition, the *class_ID* must be an allowable 8-character name (without extension).
`/D:`*<c...z>*	Required. A list of the logical drives to bundle. Specify the drive letters with no spaces or with

semicolons between them.

For example:

`/D:cdef`

specifies to bundle the local C:, D:, E:, and F: FAT or HPFS drives on the donor system.

If the current system partition is specified, the SYSCOPY command must be issued from the command line, not from the workplace shell, to ensure that files to be bundled are not locked. Locked files cannot be bundled. To access the command line, reboot the system, press **Alt-F1**, then press **F2** to get the command line.

The specified drives must be local; they cannot be network-connected remote drives.

`/B:`*<SYSCOPY_bundle_file_directory>*

Required. The directory to hold the intermediate bundle files produced by SYSCOPY.

`/S:`*<bundle_size_limit>* Optional. The maximum size, in MB, for a bundle. If this parameter is not specified, the default is one bundle per partition.

If this parameter is specified, it must be larger than the size of the largest file to be bundled. The directory path for the *<SYSCOPY_bundle_file_directory>* must have sufficient free space to store both the compressed (zip file) of the partitions' data as well as the individual bundle files created from this file; a good estimate for the size required is twice the size of the compressed size of the partitions' data.

`/L:`*<log_file_directory>* Optional. Pathname of the directory to contain the log file, *client_ID*.LOG, created during processing. If the log file does not exist, SYSCOPY creates it. If it does exist, new log entries are appended to it.

If `/L` is not specified, any setting of `/X` is ignored.

When /L is specified, SYSCOPY creates the log file, *class_ID*.LOG, unless you have specified a different pathname.

/X:*<logging_level>*	Optional. Specifies the level of logging: 1 Log error messages only 2 Log error and warning messages 3 Log error, warning, and all LOG messages created by user directives. This is the default if /X is not specified. 4 Log error, warning, and all LOG messages (equivalent to logging level 3) SYSCOPY does not output any log directives. Logging level 3 is accepted for compatibility with other migration tools. Errors and warnings are always written to the standard output stream. If logging is enabled, they are also written to the log file.

1.1.2 SYSCOPY Return Codes

The SYSCOPY utility can return one of the following codes. The codes are consistent with *CID Enablement Guidelines*, S10H-9666.

0	Successful program termination; no reboot required.
4	Successful program termination; warning messages logged; no reboot required.
8	Successful program termination; error messages logged; no reboot required.
12	Unuccessful program termination; unexpected internal errors detected and logged; no reboot required.

1.1.3 SYSCOPY Output Files

SYSCOPY creates several types of files.

1.1.3.1 SYSCOPY Bundle Files

SYSCOPY creates the bundle files labeled *class_ID*.NNN, where NNN is the number of the file. Later, you can have the replicator tool restart at a specific bundle number after an interruption.

1.1.3.2 SYSCOPY Control File

SYSCOPY also creates the control file, *class_ID*.CTL, with information about each donor partition, and the bundles for each partition.

If you run SYSCOPY against a specific partition and, for any reason, no bundle files are created, then when replicator is run using this donor partition, nothing happens. No formatting, delete list processing, or anything else is done. The donor drive is ignored by replicator.

1.1.3.3 SYSCOPY Log Files

The SYSCOPY log file contains information describing the tool's activity. If the ZIP and UNZIP utilities used have errors, that information is stored in the *client_ID*.ZLG file. See Messages returned by ZIP and UNZIP for a listing of these possible messages.

1.2 The Replicator Tool

The replicator tool represents one step in the sequence of installing customized OS/2 Warp 4 systems on existing OS/2 2.x, OS/2 Warp 3.x, or pristine target systems. The Replicator tool takes the bundle files created by the SYSCOPY utility and installs them on a target system.

You need boot diskettes to boot the target system so that the replicator tool can lay down the bundle files. You cannot overlay or reformat an existing OS/2 system. The replicator uses the INFOZIP utility, which must be in the UNZIP utility in Version 5.31 or later.

1.2.1 Replicator Syntax

The replicator tool, REPLICAT, supports the following parameters:

REPLICAT Syntax

```
<drive:\path>REPLICAT
             /C:<class_ID>
             /B:<SYSCOPY_image_bundle_file_pathname>
             /R:<bundle_number>
             /F:<input_command_file>
             /P
             /L:<log_file_directory>
             /X:<logging_level>
             /?
             ?
```

where:

`/?` or `?`	Display a summary of replicator command line syntax. Any other parameter is ignored.
`/C:`*<class_ID>*	Required. The name (*class_ID*) of the system on which the replicator is to apply the bundle files created with SYSCOPY. The name is used as a stem for the intermediate bundle files SYSCOPY creates.
`/B:`*<SYSCOPY__bundle_file_pathname>*	Required. The directory holding the intermediate bundle files produced by SYSCOPY.
`/R:`*<bundle_number>*	Optional. The restart value. If you restart the replicator, you can specify a starting bundle number. The *bundle_number* must not contain any leading zeroes in the bundle number. The bundle number is the three-digit file extension of the bundle name.
`/F:`*<input_command_file>*	Optional. The pathname of an optional replicator input command file containing directives affecting operation.
`/P:`	Optional. Specifies preprocessing. The replicator performs all the specified actions, including syntax and run-time error checking, but does not copy files to the target system. The target system is not changed. Use `/P:` with the `/L:` and `/X:4` options to build a log file you can examine to see which SYSCOPY files will be copied to which target system partitions. If `/P:` is specified, a CRC check of the SYSCOPY bundle files is done, which makes the process take longer.
`/L:`*<log_file_directory>*	Optional. Pathname of the directory to contain the log file, *class_ID*.LOG, appended to during processing. The directory also contains an `UNZIP` utility log file,

	class_ID.ZLG, which is appended to each time the replicator tool is run. If a power failure or other transmission failure occurs, you can examine the .ZLG file to find out the number of the last bundle successfully transmitted. Then you can reissue `REPLICAT` with the `/R` operand to restart at the next bundle file. You can also examine the .ZLG file to see which files were laid down if you need to do problem diagnosis. See Ziplog messages and errors for other messages that might be placed in *class_ID*.ZLG. If `/L` is not specified, any setting of `/X` is ignored.
`/X`:*<logging_level>*	Optional. Specifies the level of logging: `1` Log error messages only `2` Log error and warning messages `3` Log error, warning, and all LOG messages created by user directives in the `/F` command file. This is the default if `/X` is not specified. `4` Log error, warning, LOG command messages, and all replicator informational messages. Errors and warnings are always written to the standard output stream. If logging is enabled, they are also written to the log file.

Note: All tools used by the replicator must be pointed to by the `PATH` or `LIBPATH` directives.

1.2.2 Replicator Return Codes and Messages

The replicator utility can return one of the following codes. The codes are consistent with *CID Enablement Guidelines*, S10H-9666.

0 Successful program termination; no reboot required.

4 Successful program termination; warning messages logged; no reboot required.

5 Unsuccessful program termination; no reboot required. Error messages logged.

6 Unsuccessful program termination; no reboot required. Unsuccessful program termination with unexpected internal errors detected and logged.

An extraneous message is issued when replicator invokes the FORMAT command to format a target partition.

This message can safely be ignored.

1.2.3 Restarting or Rerunning Replicator

You can restart the replicator tool at a specified bundle file if the command fails due to a system problem while transferring files. Each bundle file is numbered; the bundle number is the file extension: *class_ID*.NNN.

To restart, check the log file to see the last successfully transmitted file. Then rerun replicator with the /R:NNN operand to specify a starting bundle.

If you run the replicator and it gets to the point of formatting the target drives, you must reboot the system before rerunning the replicator.

1.2.4 Replicator Command File

Use the replicator command file for the purposes listed below.

1. If you do not provide a command file, the replicator installs all the captured logical drive images from the SYSCOPY image bundle file package on the corresponding logical drives of the target system. Each target system partition that is to be installed is first reformatted to match the file system type of the corresponding partition on the source system.
2. To install only some of the logical drives contained in SYSCOPY image bundle files.
3. To install a logical drive image on a different logical drive from the one it was captured from.
4. To not reformat a logical drive on the target system before copying files into it. The default action is to reformat before copying files.
5. To set the volume label for a target system logical drive.
6. To check that the minimum size value of a target system's logical drive is adequate before the replicator will copy to it. The default minimum size is the size of the partition captured on the donor system.
7. To indicate that a specified target system will be the system partition. This option is necessary only if you do not want to reformat the system partition before copying files to it. In that case, you must provide an OS/2 DEL.LST

file the replicator uses to perform OS/2 Install-style delete list processing of any old critical system files that are on the partition. The `SYSTEM` command parameter is the pathname of the delete list file.

8. To pass along messages to be stored in the log file.

1.2.4.1 Syntax of the Replicator Command File

At a high level, the command file is a collection of command stanzas. Each starts with a `NAME` and ends with a corresponding `ENDNAME`. A command stanza can contain substanzas, each starting with `SUBNAME` and ending with `ENDSUBNAME`. The descriptions here each first show the high-level structure of a command stanza block, then walk down the structure to the innermost substanzas.

The replicator command file can include blanks, tabs, new lines, vertical tabs, form feeds, and carriage returns to separate command parameters and keywords from other command parameters and keywords. More than one command is allowed on a line, and commands can span lines. EOF characters are ignored in command files.

The `LABEL`, `LOG`, and `SYSTEM` commands support DBCS characters; all others must be single-byte characters.

The syntax of the replicator command files is as follows:

Replicator Command File Syntax

```
DRIVE
    DONORLETTER <drive_letter>
    TARGETLETTER <drive_letter>
    MINSIZE <size in MB>
    NOFORMAT
    LABEL <label>
    SYSTEM <file pathname>
ENDDRIVE
LOG <Message string>
```

where:

`DRIVE` The `DRIVE` command starts a command stanza that specifies directives about a disk partition contained in the `SYSCOPY` bundle file package.

`DRIVE` maps the donor system partition to the target system partition and controls how replicator performs operations on the target system.

You can have multiple DRIVE stanzas within a replicator input command file.

End the DRIVE stanza with the ENDDRIVE command.

The following subcommands are allowed in the DRIVE stanza:

DONORLETTER	Use the DONORLETTER command to specify the drive letter of a partition within the SYSCOPY bundle file package. Do not code the colon (:) in the DONORLETTER command. Only one DONORLETTER can be coded in a DRIVE stanza.
TARGETLETTER	Use the optional TARGETLETTER command to specify that the captured partition image is to be replicated to a different logical drive on the target system. If no TARGETLETTER is supplied, the partition image is installed on the same drive letter it was captured from. Do not code the colon (:) in the TARGETLETTER command. **Note:** Use this command only for logical drive images containing data files. If you use this command for drive images containing programs, you may lay down programs where the PATH and LIBPATH values do not match the actual locations. Only one TARGETLETTER can be coded in a DRIVE stanza.
MINSIZE	Use the optional MINSIZE to change the minimum size value in MB the replicator checks against before laying down files. Code this value as a decimal number in megabytes. Do not include the letters MB. For example: `MINSIZE 100` to specify 100 megabytes.

	The default value is the size in MB of the partition on the donor system this logical drive was captured from. The size a target partition must be in order to contain all the files from a given donor system may differ depending on the file system type, and on the sector size used on the target partition once the target system is formatted, if it is reformatted. Only one `MINSIZE` can be coded in a `DRIVE` stanza.
`NOFORMAT`	Use the optional `NOFORMAT` command to specify that replicator is not to format the target system partition before starting to copy the logical drive image. If `NOFORMAT` is specified, then `SYSTEM` is required. The default is to reformat the target partition to the file system type in effect for the corresponding donor system partition. Only one `NOFORMAT` can be coded in a `DRIVE` stanza.
`LABEL`	Use the optional `LABEL` command to specify a volume label for the specified target system partition. Relabelling occurs only if the partition is being reformatted. The label must be a printable ASCII value, 11 bytes or fewer. If a longer label is specified, it is truncated. Only one `LABEL` can be coded in a `DRIVE` stanza.
`SYSTEM`	Use the `SYSTEM` command to specify that the target system partition will be a system partition. If you have chosen not to reformat the target partition, the replicator must do OS/2-style delete list processing before beginning the logical drive copy operation. `SYSTEM` also specifies the directory in which the OS/2 Warp 4 DEL.LST and REPLICAT.OS2 files reside. Only one `SYSTEM` can be coded in a `DRIVE` stanza.

`LOG`	Use the `LOG` command to place messages in the replicator log file. Log message strings are logged at the top of the log file.

Note: A comment line begins with two slashes (//). A comment line can also begin with one slash (/). Indentation and upper case is for readability only.

1.2.5 Replicator Operation

The replicator does the following actions. (You must have already created the files to be used in this sequence with SYSCOPY).

1. Reads control information from the control file in the SYSCOPY image bundle file package.
2. Checks the file bundles for consistency. If any check fails, the errors are logged and the operation stops.
3. Reads any commands in the replicator input command file.
4. If you are starting from the beginning (you have not specified `/R:` to restart), the replicator:
   - Makes sure each donor drive letter matches a logical drive image in the SYSCOPY bundles and makes sure the target drive letters exist on the target system.
   - Makes sure the corresponding target system logical drive is at least as large as specified with `MINSIZE`, or as large as the corresponding donor partition.
   - If you specified NOFORMAT, make sure the target system is formatted and make sure you are not trying to copy from a donor system HPFS logical drive to a target system FAT drive because you will likely lose long filenames.
5. Formatting and volume labeling are done as specified in the command file.
6. If one of the target system partitions is marked as a system partition, and you have specified NOFORMAT, replicator does OS/2-style delete list processing before beginning the logical drive copy operation.
7. Replicator iterates through each file bundle in the package. (This is where processing starts if you specified restart with the `/R:` parameter.) Each file bundle is unbundled. The files are copied to the correct directories on the target system. If you specified that files are to be laid down on different drive from the one they were extracted from on the donor system, make sure this occurs by interacting with the bundling utility. If you run had SYSCOPY against a specific partition and, for any reason, no bundle files were created, then when replicator is run using this donor partition, nothing happens. No formatting, delete list processing, or anything else is done. The donor drive is ignored by the replicator.

When the replicator operation is complete, reboot the system; then run the configurator on the target system.

1.2.6 Using Replicator in a CID Environment

Typically, you run the replicator in a CID environment. Although you can use the tool in a normal OS/2 environment to copy logical disk images into non-system partitions, the most likely use is to install complete system images. And, you will most likely use the CID environments, LAN CID Utility (LCU), and NetView Distribution Manager/2 (NVDM/2).

The CID boot environments establish a minimal LAN Redirector connection through a SrvIFS LAN redirector to a CID server. Once that connection is made, the replicator and all its data files can reside on the CID server. Or, they can reside on a CD-ROM.

Because the replicator is a small command-line utility, it might fit on LCU or NVDM/2 boot diskettes or on a CD-ROM. If not, it can reside on the CID server. You can use environment variable substitution to allow run-time specification of replicator command-line parameters.

The replicator does not handle target system repartitioning or lay boot system boot files because the CID tool set already does this.

If you are laying down a system partition image and also choose not to reformat the logical drive before copying the system partition image, make sure to run the CID `SYSINSTX.COM` utility, or an equivalent, to make sure the bootstrap record is properly linked into the system partition.

1.3 The PROBE Tool

The system configuration extractor utility, `PROBE`, extracts relevant configuration information from an existing OS/2 target system and uses that information to generate a Feature Install (FI) object and the `PROBE` object. These objects are used for installation on a target system by the configurator utility, `CONFIGUR`.

Some typical information extracted by `PROBE` includes:

- LAN Requester configuration

`Computername`	A name that uniquely identifies the requester name
`Domain`	A name that identifies the domain to which this requester belongs

`NETx`	Describes the protocol interface and resources for each adapter. The `[requester]`, `[workstation]`, and `[peer]` sections in the IBMLAN.INI file are taken into consideration.
`wrknets`	Describes the `NETx` lines that are valid for that section
`services`	Describes the Requester and Server services that are automatically started
`wrkheuristics`	A bitwise configuration string used by LAN Requester/Server
`othdomains`	Describes the other domains to which this requester belongs

- TCP/IP configuration

Hostname	A unique name primarily used by domain name servers
Domain name	A group name primarily used by domain name servers
IP address	A unique address defining your machine in a TCP/IP environment
Subnet Mask	A way of dividing the addresses into subnet masks

- MPTS configuration
  - NetBIOS

TI	Timeout value for inactivity on the link
T1	Timeout value for sending on the link
T2	Timeout for responding on the link
NetBIOS retries	Number of retries NetBIOS attempts if a timeout occurs
NetBIOS timeouts	Timeout value when sending NetBIOS packets between machines
Sessions	Number of NetBIOS sessions defined (NetBIOS resources)
Commands	Number of NetBIOS commands defined (NetBIOS resources)
Names	Number of NetBIOS names defined (NetBIOS resources)

  - Token-ring

Ring speed	Speed which the token-ring driver enters the ring

Netaddress	The network address at which the adapter will be known

- WIN-OS/2 system configuration
  - WIN.INI

`[Desktop]` session settings	Size, position, and color of WIN-OS/2 desktop
`[Colors]` for controls	Colors for all controls within WIN-OS/2
`[windows]` session devices	Printer devices within WIN-OS/2
`[ports]` session settings	LPT and COM port settings within WIN-OS/2

  - SYSTEM.INI

Display settings	Display driver for WIN-OS/2

  - PROGRAM.INI

    Program Manager INI file that includes WIN-OS/2 group settings
  - CONTROL.INI

    Migrate the current color schemes within WIN-OS/2
  - *.GRP

    Copy all group files such as Main, Accessories, Startup, and so forth.
- Printer configuration
  - OS/2 Printer configuration

Default printer	Printer that jobs go to by default
Printer view	Icon or details view of the print objects
Print priority	Value to specify amount of time slices given to print priority
Job properties	Values such as portrait or landscape orientation
Printer driver properties	Values such as top drawer, lower drawer, paper size

  - WIN-OS/2 Printer configuration

WIN-OS/2 default queue	Printer that WIN-OS/2 jobs go to by default
WIN-OS/2 printer view	Icon or details view within WIN-OS/2

WIN-OS/2 print priority	Value to specify amount of time slices given to print priority

Before you use PROBE, read about using the migration tools in an orderly sequence.

The PROBE utility accepts directions from one or more command stream files. Use any ASCII editor to create the command stream files; follow the command stream file directions described here.

You can trigger user exits from within a command stream. The user exits can, in turn, create additional PROBE command stream files to be operated on by PROBE.

1.3.1 Output from PROBE

The output from PROBE is a set of file bundles that hold the contents of the PROBE extracted from the old target system. PROBE may create one or more bundles, depending on the size of the PROBE and on the setting of the parameter /S: on the PROBE command line. The bundle files are named using the ID provided on the PROBE command line.

The first file bundle contains the FI response file, *client_ID*.RSP, and a control file, *client_ID*.CTL, that contains summary information about the PROBE. That file contains, at least, a list of all the files created in the intermediate bundle set including size, date, time, and a CRC value. These bundle files are the input to the configurator utility. You do not directly interact with the bundle files.

1.3.2 Syntax of the PROBE Command

The following parameters are supported with the `PROBE` tool:

PROBE Syntax

```
<drive:\path>PROBE
          /E:<client_ID >
          /B:<intermediate_bundle_file_directory_pathname>
          /V:<{FI variable name}=FI_variable_value>
          /W:<temporary_work_directory_pathname>
          /S:<bundle_size_limit>
          /P:
          /L:<log_file_directory_pathname>
          /X:<logging_level>
          <command_stream-file_1>.prb, <command_stream_file_2>.prb,
          /?
          ?
```

where:

`/?` or `?`	Display a summary of `PROBE` command-line syntax. Any other parameter is ignored.
`/E:`*<client_ID>*	Required. The name (ID) of the system on which the probe is taken. This is the existing OS/2 2.x or 3.x system. The name is used as a stem for the intermediate bundle files `PROBE` creates. Files are named *client_id*.NNN. The numbers NNN start at 000 for the first bundle and increment sequentially. The *client_ID* must be ASCII-7 characters and can contain no blanks. If the intermediate file bundles are to be stored on a FAT partition, the name cannot be greater than eight characters.
`/B:`*<intermediate_bundle_file_pathname>*	The directory to hold the intermediate bundle files produced by `PROBE`.

The bundle files contain the pieces the configurator uses to create an FI install object representing the directives created by PROBE. (You do not have to handle these files; just make sure there is space for them.)

An error is returned if files with matching names already exist in this directory.

/V:*<{FI variable name}= FI_variable_value>*

Optional. Sets the value of the specified variables within the FI install object created by the configurator tool. The tool does not create the variables; they must already be created within the intermediate PROBE bundle files referred to on the configurator command line.

Using this parameter lets you avoid creating a PROBE SETFIVAR command in a .PRB script or running a user exit to set the value of an FI variable. You can provide any number of these FI variable specifications on the command line.

For example:

PROBE /V:*{WORKING_DIR}*=D:\TEMP\FOO /V:*{HOME_DRIVE}*=W:

Do not code any spaces in the parameter specification. If you want spaces in the FI variable value, enclose the entire parameter in double quotation marks. You **cannot include an imbedded quotation mark** in the FI variable value.

Another example:

PROBE "/V:*{WORKING_DIR}*=D:\TEMP\Inventory Control"

/W:*<temporary_work_directory_pathname>*

A directory to serve as the root of the temporary work area. Specify a directory with enough space to contain the full set of encompassed files created by `PROBE`. Run one `PROBE` sequence with the `/P` parameter to help estimate the space requirements.

A subdirectory named *client_id* is created as the root of the temporary work area. An error is returned if this subdirectory already exists.

On successful completion, this space is cleaned up. On error termination, any files created are left for your error analysis.

If you tell `PROBE` to create output files, but omit this parameter, `PROBE` looks for a temporary directory pathname in the `TEMP` environment variable. In that case, an error is returned if `PROBE` cannot find a valid directory pathname in the `TEMP` environment variable.

`/S:`*<bundle_size_limit>*

Optional. The limit, in MB, of uncompressed bundle file size during `PROBE` operations.

When specified, `/S` tells `PROBE` not to create intermediate bundle files larger than the limit.

(Later, during the process, the bundle files are compressed.) If this parameter is not specified, `PROBE` creates one bundle file, a separate control file and an `FI` response file.

`/P:`

Optional. Directs `PROBE` to run in preprocessor mode. `PROBE` does all its actions, including syntax and run-time error checking, but does not use up temporary work space and does not create the intermediate bundle files. Using `/P` is a good way to check for errors in the command stream files.

The log file tells approximately how large the temporary work area needs to be to hold the intermediate bundle files. When you use `/P`, you should also specify `/L` for logging and `/X:4` to create as complete a log file as possible.

/L:*<log_file_directory_pathname>*

Pathname of the directory to contain the log file, *client_ID*.LOG, created during processing. If the log file does not exist, PROBE creates it. If it does exist, new log entries are appended to it.

If /L is not specified, any setting of /X is ignored.

PROBE also appends to or creates a log of zip processing information called *client_ID*.ZLG. That file might contain information about errors encountered during processing. See ZIP and UNZIP messages and errors for a list of messages that might be in the .ZLG file. If /L is not specified, the log of ZIP messages is sent to standard out.

/X:*<logging_level>*	Specifies the level of logging: 1 Log error messages only 2 Log error and warning messages 3 Log error, warning, and all LOG messages created by user directives. This is the default if /X is not specified. 4 Log error, warning, LOG command messages, and a complete list of the actions taken during PROBE processing.

<command_stream_file_n>.PRB

The pathname of one or more command stream files. The file names cannot contain wildcard characters.

The structure of the command stream file is simpler than it appears at first. Start with the high-level overview; then move down to the section you need to create. The file suffix must be .PRB.

You can use and expand on some of the samples of PROBE command files provided for you.

1.3.3 PROBE Return Codes

PROBE can return one of the following codes. The codes are consistent with *CID Enablement Guidelines*,S10H-9666.

0 Successful program termination; no reboot required.

4 Successful program termination; warning messages logged; no reboot required.

5 Unsuccessful program termination; no reboot required. Error messages logged.

6 Unsuccessful program termination; no reboot required. Unsuccessful program termination with unexpected internal errors detected and logged.

PROBE sets return code 4 when it is not able to delete temporary files created for the zipping of files for the bundle files. This can happen on DBCS systems, where certain characters prevent proper deletion of these files.

You can delete the temporary files and directory manually without affecting the integrity of the PROBE bundle files.

1.3.4 Syntax of PROBE Stream Commands

The PROBE stream commands are structured in stanzas. A command stanza can contain substanzas, which, in turn, can contain sub-substanzas. Any level of stanza inherits state from its parent stanza. The commands are not case sensitive; they are shown uppercase here for ease of reading.

Each command is identified by an ASCII command keyword, an ASCII-7 word. String parameters can contain non-ASCII-7 characters. All strings are interpreted based on the code page the PROBE was initialized with.

You can place multiple commands on a line. A command with a parameter can be split over several lines.

Comment strings can separate a command keyword on one line from its parameter on the next line. Blank lines are valid anywhere in the command stream.

PROBE processes commands sequentially. The description of each command documents any sequence dependencies.

1.3.4.1 Parameter Strings

The types of command parameter strings are:

Ordinary Strings	Used to specify pathnames, file names, INI file applications, key names, and parameter values.
Regular Expression Strings	Used for pattern matching within some text file commands and some file copy commands. FI regular expression syntax differs from XPG4 regular expression syntax. If you plan to use regular expressions, read “Regular Expression Strings” on page 25 before you create the text file.

All strings must be enclosed in double quotation marks: "This is a typical string."

1.3.4.2 Ordinary Strings

Within an ordinary string, you can use a pair of double quotation marks ("") to indicate a single double quotation mark that is part of the string.

For example:

"This is a valid ""ordinary"" string with a quoted word."

Use a pair of double quotation marks with no space between them to specify a `NULL` string.

FI Variables within Ordinary Strings

Any ordinary command parameter string can contain one or more references to FI variables. Within the string, the FI variable is specified within braces.

For example:

{variable_1}

Any string within braces, for example, *{variable}* within a parameter is interpreted as an FI variable.

When the `PROBE` command relates to a data object aimed at the target system, the FI variable is resolved by the configurator on the target system. When the `PROBE` command relates to a data object aimed on the source system, the FI variable is resolved during the `PROBE` operation when the command is executed.

The value of an FI variable resolved during `PROBE` operation is based on the most recently executed `FIVAR` defining the variable. You can use multiple `FIVAR` command stanzas to define and then change the value of an FI variable. Only one instance of that FI variable occurs on either the source or the target system.

An FI variable can refer to other FI variables. `PROBE` does only one level of indirection. For example, if FI variable *{variable1}* has a value of *{variable2}*, and FI variable *{variable2}* has a value of *{variable3}*, the resolution of a reference to FI variable *{variable1}* returns the string `"{variable3}"`.

The value of an undefined FI variable is the string formed by the name of the variable surrounded by its braces "*{*" and "*}*". For example, if FI variable *{ABC}* is undefined, the resolution of *{ABC}* returns the string `"{ABC}"`.

1.3.4.3 Regular Expression Strings

`PROBE` uses slightly different regular expression syntaxes, depending on the target of a search. There are also considerations that apply to all regular expression syntaxes: FI regular expression syntax is different from XPG4 regular expression syntax.

FI Regular Expression Syntax Considerations

The FI regular expression syntax is used for text file configuration commands that refer to patterns to be matched on the target system because the FI run-time environment does this matching.

XPG4 Regular Expression Syntax Considerations

The XPG4 Basic regular expression syntax matching is used for text file commands that refer to patterns to be matched on the source system and for file-copy commands specifying which files are to be copied from the source system.

You can specify that some special characters are to be escaped, that is, treated as normal characters, not as control characters. The period (.) is used to match any single character within both XPG4 and FI regular expressions. Use the sequence backslash period (\.) to explicitly place a period within a pattern to be matched.

Regular expression strings on both the source and target sides cannot contain references to FI variables.

1.3.4.4 Relative Path Conventions

Many PROBE commands have a parameter specifying a pathname string. When a relative path is specified, the path begins at the working directory from which the PROBE was started. PROBE does not change the location of its working directory.

1.3.4.5 Comments in a PROBE Command Stream File

Specify a comment in a command stream file, *command_stream-file_1*.PRB, for example, by placing double slashes (//) at the beginning of the comment. All characters following the // are ignored until the end of the line. Comments are not logged in the log file.

The replicator command file can include blanks, tabs, new lines, vertical tabs, form feeds, and carriage returns to separate command parameters and keywords from other command parameters and keywords. More than one command is allowed on a line, and commands can span lines. EOF characters are ignored in command files. A comment can start in the middle of a line:

```
TEXTFILE
SOURCE "<BOOTDRIVE>\config.sys"    // This is a comment in a line.
```

A pair of slashes imbedded in a string is not interpreted as a comment delimiter.

1.3.4.6 Warning and Error Controls

The WARNING and ERROR commands let you put information about PROBE processing in the log. The parameter is a simple string following the command. You can replace the parameter field with an FI variable that is resolved during execution when the command is interpreted. Use the WARNING command to generate warning messages in the log file (if one is specified). WARNING has no effect on execution other than logging.

Use the ERROR command to generate error messages in the log file, if you specified a log file. Upon execution, processing stops. The string is also sent to stdout, even if logging is disabled.

How PROBE Deals with Errors

When PROBE starts, it checks the syntax of all the command stream files, including those defined in INCLUDE commands, but not those generated by user exits. All the syntax errors found are logged. If there are any, PROBE stops execution after logging the error messages. You can examine the log for these error messages.

If `PROBE` finds no static syntax errors in the first pass, it begins normal operation. If it finds an error, it logs the error and then stops.

`PROBE` does a syntax check whenever a command stream file is passed from a user exit. If a syntax error is found in one of these files, `PROBE` logs the error.

The `ERROR` command lets your user exit return error indications to `PROBE` so that information can be logged about a failure. A user exit can also place `WARNING` and `LOG` commands before an `ERROR` command to log more information before a possible error is found.

Controlling Error Determination of Not-found Conditions

In some situations, you may want the failure of a string matching attempt to be an error. In other cases, you want execution to continue. Use the `ISNOTFOUNDERROR` to specify the action `PROBE` should take in these situations.

The parameter field can be replaced with an FI variable that is resolved during execution when the command is interpreted.

`ISNOTFOUNDERROR` specification	**Result**
`YES`	Failure to find a match is an error. Execution stops.
`WARN`	Situation is logged, execution continues.
`NO`	The situation is ignored. Execution continues.

You can include `ISNOTFOUNDERROR` commands anywhere in a command stream file outside of command stanzas. A subsequent `ISNOTFOUNDERROR` command overrides the setting of a previous command. The scope of `ISNOTFOUNDERROR` is within the command stream file it is in. When a new command stream file starts, this command defaults to `NO`. If a command stream interrupts the action of another, the value of `ISNOTFOUNDERROR` is remembered and restored when the first file is resumed later.

1.3.4.7 Logging Controls

Use the `LOG` command to provide any additional information you want in the log. Specify a logging level of 3 or 4 to have LOG messages placed in the log. `/X:3` is the default log level; `/X:4` stores information about each `PROBE` operation.

The `LOG` command's parameter is a simple string. The parameter field can be replaced with an FI variable that is resolved during execution when the command is interpreted. A LOG command can be coded anyplace in a command stream file outside of command stanzas.

1.3.4.8 Extended Logging

If you specify logging level 4 by coding /X:4, PROBE logs actions including:

- The value of each command line parameter, whether not you specified that parameter in the command line.
- When each command stream file is opened or closed
- Each stream command as it is performed:
  - The complete command listing including any parameter values.
  - The resolved value of the parameter string if source-side FI variable resolution was done.
  - The starting ordinal line number of the command within the command stream file.
- All values read from the source system
- All replacement values to be passed to the target system
- The pathnames of all files that are copied over to the target system

1.3.5 Inclusions of other Command Stream Files

Use the INCLUDE command to specify another command stream file to be processed. The parameter of the command is an ordinary string in double quotation marks specifying the name of the file to process. The string can contain references to FI variables. Any FI variable referred to in an INCLUDE command parameter string must already exist when the command is parsed. A reference to an FI variable that does not exist when the command is parsed causes a syntax error (even if a FIVAR command stanza would create the variable before the INCLUDE command is executed).

You can use the /V: parameter to define FI variables before any INCLUDE commands are parsed.

An example of the INCLUDE command is:

```
...
INCLUDE "PROBEcmd_3.prb"
...
```

The new command stream file begins execution immediately, interrupting the processing of the current command stream file. INCLUDE is a simple way to specify additional command stream files for processing. You can code INCLUDE commands anywhere in a command stream file outside of command stanzas.

An example of PROBE control commands:

```
// This is an example of a comment line.
// This is another.
...
WARNING "This is a warning - execution doesn’t stop"
...
LOG "Use the log command for informational messages"
...
ISNOTFOUNDERROR YES
...
INCLUDE "TCP.PRB" // This comment is on a command line.
...
ERROR "This is an error command. It stops execution."
...
```

Figure 2. Example of PROBE Control Commands

1.3.6 The Text File Commands

At a high level, the text file is a collection of command stanzas. Each starts with a NAME and ends with a corresponding ENDNAME. A command stanza can contain substanzas, each starting with SUBNAME and ending with ENDSUBNAME. In turn that substanza can contain sub-substanzas. The descriptions here each first show the high-level structure of a command stanza block, then walk down the structure to the innermost sub-substanzas.

You can obtain entire lines from a file on the original donor system and then use them as a replacement on the updated target system. Or you can generate new replacement lines.

You can change every occurrence of a specified line, or add lines to either the front or back of the text file on the updated target system, or before or after a specific line of the file.

Search strings used to find lines on the updated target system can contain FI variable references so that the behavior of the search can be modified when the FI PROBE object is installed.

You could also obtain an individual string from lines in a file on the old target system and then use them as replacements or additions on the updated target system. You can provide specific values for those strings instead of extracting them from the old system.

Within the TEXTFILE stanza, you can use the REPLACELINE command to modify lines and the EDITLINE command to modify strings within lines. The following is the high-level syntax of the text file configuration. Click on any command for more detailed information. Some examples of text file commands are also provided.

The TEXTFILE command is the beginning of a stanza block defining a set of modifications to be made to a text file on a target system.

```
// Text file configuration commands
// Indentation and upper case are not required, they are
// used here for ease of reading.
TEXTFILE
   SOURCE "source_system_text_file_pathname"
   TARGET "target_system_text_file_pathname"

      [REPLACELINE command substanza]
   ...
   ENDREPLACELINE

      [EDITLINE command substanza]
   ...
   ENDEDITLINE
ENDTEXTFILE
```

Figure 3. TEXTFILE Stanza

1.3.6.1 The TEXTFILE Command Stanza Block

This section describes the commands valid within the TEXTFILE stanza block:

- SOURCE
- TARGET
- REPLACELINE
- EDITLINE

SOURCE Command <- TEXTFILE Stanza

Use the SOURCE command to specify the pathname of the text file on the source system that is searched when you extract text strings from the source system.

The SOURCE command must come before any REPLACELINE or EDITLINE commands that refer to that source file. The SOURCE command only to any REPLACELINE and EDITLINE blocks following it.

Only one SOURCE command can be in a TEXTFILE stanza; an error is returned if there are multiple SOURCE commands.

Enclose the source file path name in double quotation marks. It is an ordinary string and can refer to FI variables. FI variable references are resolved during execution. Other FI and string considerations are described separately.

TARGET Command <- TEXTFILE Stanza

The TARGET command specifies the pathname of the text file on the target system to be modified by the REPLACELINE and EDITLINE commands within the TEXTLINE stanza.

The TARGET command parameter is an ordinary string within double quotation marks.

For example:

```
TARGET "d:\dir_name\sub_dir_name\target.fil"
```

It can contain references to FI variables which are resolved on the target system during the installation of the FI Install object during the execution of the configurator.

You can specify only one TARGET command within a TEXTFILE stanza.

If SOURCE is specified, but TARGET is not specified within this TEXTFILE block, then the source is considered to also be the target of any modifications. However, in this case, any FI variable references are resolved on the source system, but FI variables on the target system are left unresolved so that they can be subsequently resolved during the installation of the FI install object during the execution of the configurator.)

If TARGET is specified, but SOURCE is not specified within this TEXTFILE block, you must provide specific replacement text strings with the REPLACEMENT command for each REPLACELINE and EDITLINE command in the TEXTFILE block. If not provided, an error is returned.

The TARGET command only to any REPLACELINE and EDITLINE blocks following it.

REPLACELINE Substanza <- TEXTFILE Stanza

Use the REPLACELINE command substanza when you want to find a line and replace it with another. REPLACELINE starts a substanza block that defines how to add, replace, or delete lines in the text file on the target system identified with the TARGET and, optionally, SOURCE, commands in the TEXTFILE stanza block.

You can have zero or multiple REPLACELINE commands within the TEXTFILE block. If there are no REPLACELINE commands within the TEXTFILE block, there must be at least one EDITLINE. REPLACELINE and EDITLINE commands can be mixed in any order. The syntax of the REPLACELINE command substanza is illustrated in Figure 4 on page 33.

```
// Indentation is for ease of reading, it is not required

REPLACELINE
   NOGREP
   CASESENSITIVE
   SOURCELINE
      PATTERN "XPG4 Extended Regular Expression pattern string"
      DELETELINE "string"
      REPLACEMENT "string"
      SETFIVAR "variable_name"
   ENDSOURCELINE
   TARGETLINE
      PATTERN "FI Extended Regular Expression pattern string"
      SCOPE [FIRST | LAST| EVERY]
      LOCATEDACTION [REPLACE | NOTHING]
      NOTLOCATEDACTION [BOF | EOF | OTHER | NOTHING]
      OTHERACTIONS
         ADDBEFOREPATTERN "FI regular expression pattern string"
         ADDBEFORESCOPE [FIRST | LAST]
         ADDAFTERPATTERN "FI regular expression pattern string"
         ADDAFTERSCOPE [FIRST | LAST]
         LOCATEDACTION [ADD | NOTHING]
         NOTLOCATEDACTION [BOF | EOF | NOTHING]
      ENDOTHERACTIONS
   ENDTARGETLINE
ENDREPLACELINE
```

Figure 4. REPLACELINE Textfile Substanza

NOGREP Command <- REPLACELINE Substanza <- TEXTFILE Stanza

The optional NOGREP command specifies all string searches in the REPLACELINE block are simple string comparisons. The default, when NOGREP is omitted, is to do all string searches as XPG4 Extended Regular Expression comparisons for SOURCELINE patterns and as FI regular expression comparisons for TARGETLINE patterns.

The scope is limited to the REPLACELINE block containing the NOGREP. Any REPLACELINE or EDITLINE command following the REPLACELINE block that has NOGREP set starts with the default.

An error is returned if more than one NOGREP is defined within a REPLACELINE block.

CASESENSITIVECommand<-REPLACELINESubstanza<-TEXTFILEStanza

The optional CASESENSITIVE command specifies that all string searches are case sensitive. The sensitivity affects both simple string comparisons and Regular Expression searches.

The default, if CASESENSITIVE is omitted, is to ignore case.

The scope is limited to the REPLACELINE block containing the CASESENSITIVE. Any REPLACELINE or EDITLINE command following the REPLACELINE block with CASESENSITIVE set starts with the default.

An error is returned if more than one CASESENSITIVE is defined within a REPLACELINE block.

SOURCELINE Command <- REPLACELINE Substanza <- TEXTFILE Stanza

The required SOURCELINE command marks the start of a sub-substanza specifying what text line, if any, to use to replace text lines in the target system.

An error is returned if more than one SOURCELINE is defined within a REPLACELINE block.

The SOURCELINE command block can contain one each of the PATTERN, REPLACEMENT, SETFIVAR, DELETELINE, and ENDSOURCELINE commands. An error is returned if SOURCELINE is followed immediately by ENDSOURCELINE.

End the SOURCELINE block with the ENDSOURCELINE command.

PATTERN Command <-SOURCELINE Sub-Substanza <- REPLACELINE Substanza <-TEXTFILE Stanza

The optional PATTERN command specifies a search string. PROBE uses the string to search the source system file and selects the first line containing a matching string. The line selected becomes the replacement line. The SOURCELINE block can contain only one PATTERN command.

The PATTERN is a simple string, or an XPG4 Extended Regular Expression enclosed in double quotation marks. It cannot contain FI variable references.

If you want not finding a PATTERN match to generate an error message and terminate processing, code:

```
ISNOTFOUNDERROR = YES
```

If the PATTERN command’s string search fails to find a line with a matching string in the source file, the search is considered a failure with the following conditions:

Table 1. Results when Pattern Command Text Search Fails

REPLACEMENT specification	ISNOTFOUNDERROR specification	Result
Specified	Not set	The REPLACEMENT command parameter becomes the replacement line.
Not specified	Set to YES	PROBE fails with an error.
Not specified	Set to NO or WARN	The SOURCELINE sub-substanza only sets the SETFIVAR command, FI variable, if any, to NULL.

If PATTERN is omitted, the default value is to do no text search. To avoid an error in this case, you must provide either a REPLACEMENT command or a DELETELINE within the SOURCELINE block. If the source search pattern is found, the DELETELINE is ignored. If PATTERN finds a matching line, the entire line is logged.

DELETELINE Command <-SOURCELINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

Use DELETELINE to specify a that lines found as a result of the TARGETLINE stanza are to be deleted.

Line deletions occur only if the decision is made to replace lines on the target system, which depends on how the TARGETLINE stanza is coded. If the decision is made to add lines on the target system, then this case is a no-op. This case is the same as when the PATTERN search fails to find a matching line within the source system text file.

If the DELETELINE command is omitted, you have provided no directive to delete text lines on the target system.

It is an error to specify both DELETELINE and REPLACEMENT within the same SOURCELINE command sub-substanza block, or to define more than one DELETELINE command within the same SOURCELINE command sub-substanza block.

REPLACEMENT Command <-SOURCELINE Sub-Substanza <-REPLACELINE Substanza <-TEXTFILE Stanza

Use REPLACEMENT to specify a string to be used as a full replacement for the selected lines in the target system.

Specify the REPLACEMENT as an ordinary string enclosed in double quotation marks. The string can contain references to FI variables which are resolved on the target system during the installation of FI Install object by the configurator tool.

If REPLACEMENT is omitted, there is no specific text line to place in the target system. To avoid an error in this case, supply a PATTERN command within the SOURCELINE block.

If PATTERN is also specified in the SOURCELINE sub-substanza, then the REPLACEMENT command specifies the replacement line to choose if the search through the source system text file fails to find a line that contains a matching PATTERN string.

You can specify that the replacement line is to be a blank line by specifying a null string as the REPLACEMENT operand.

If REPLACEMENT changes a replacement line, the new replacement line string is logged. It is an error to specify both DELETELINE and REPLACEMENT within the same SOURCELINE command sub-substanza block or to define more than one REPLACEMENT command within the same SOURCELINE command sub-substanza block.

SETFIVAR Command <-SOURCELINE Sub-Substanza <- REPLACELINE Substanza <-TEXTFILE Stanza

Use the optional SETFIVAR command to specify the name of an FI variable. The variable is set to the value of the replacement line generated by the SOURCELINE command substanza block. This lets you pass source system text file lines and strings to the configurator tool and to FI user exits.

If the DELETELINE command becomes the replacement line, the FI variable value is set to a NULL string. If the FI variable is already defined, the value is changed to any new value. If the FI variable does not yet exist, the variable is created and its value is set.

The SETFIVAR command parameter is an ordinary string enclosed in double quotation marks and may contain references to FI variables. The FI variable name is specified without the pair of enclosing braces {}. Any FI variable references in the SETFIVAR command parameter string are resolved during PROBE execution.

Only one SETFIVAR command can be specified within a SOURCELINE command sub-substanza block.

TARGETLINE Command <- REPLACELINE Substanza <- TEXTFILE Stanza

The optional TARGETLINE command marks the start of a sub-substanza specifying which lines, if any, to replace in the target system. The TARGETLINE stanza ends with the ENDTARGETLINE command.

If you code only TARGETLINE and ENDTARGETLINE, the following is true:

- The target pattern defaults to the source pattern.
- All other TARGETLINE commands are set to their default values.
- PATTERN must be specified within the SOURCELINE sub-substanza block.

An error is returned if more than one TARGETLINE is defined within a REPLACELINE block. End the TARGETLINE block with the ENDTARGETLINE command.

The PATTERN Command <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <-TEXTFILE Stanza

The optional PATTERN command specifies a search string. PROBE uses the string to search the target system text file to select one or more lines for possible modification. The TARGETLINE sub-substanza block can contain only one PATTERN command.

The default value, if PATTERN is omitted, is to use the PATTERN command specified in the SOURCELINE command sub-substanza.

The PATTERN is an FI Regular Expression string enclosed in double quotation marks. It cannot contain FI variable references.

SCOPE Command <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

The optional SCOPE command specifies which target text file lines are to be selected for possible modification.

Table 2. SCOPE Command Keywords

Keyword	Action
FIRST	Select the first line containing the search string.
LAST	Select the last line containing the search string.
EVERY	Select every line containing the search string.

The default value is FIRST if SCOPE is omitted. The TARGETLINE sub-substanza block can contain only one SCOPE command.

LOCATEDACTION Command <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

Use the LOCATEDACTION command to specify what the configurator tool should do if at least one line in the target system text file is selected that satisfies the target line search criteria.

Table 3. LOCATEACTION Command Keywords

REPLACE	Replace each selected line with the replacement line. If the replacement line is DELETE, the result is one or more line deletions.
NOTHING	Do nothing.

If the LOCATEDACTION command is omitted, the default parameter is REPLACE.

You can use LOCATEDACTION only once within an TARGETLINE sub-substanza block.

NOTLOCATEDACTION Command <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

NOTLOCATEDACTION specifies what action to take if a target line search is unsatisfied.

The NOTLOCATEDACTION command is not enclosed in double quotation marks. If specified, it must be set to one of the following values:

Table 4. NOTLOCATEACTION Command Keywords

Value	Action
BOF	Replacement line placed at front of the target file.
EOF	Replacement line placed at end of the target file.
OTHER	Replacement line placed before or after other specified lines in the target file. Specify the other lines with the OTHERACTIONS command.
NOTHING	Do nothing.

NOTLOCATEDACTION, if omitted, is assumed to be NOTHING.
You can use NOTLOCATEDACTION only once within a TARGETLINE block.

OTHERACTIONS Command <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <-TEXTFILE Stanza

OTHERACTIONS provides a set of extended line editing choices within a REPLACELINE stanza.

Use OTHERACTIONS only if NOTLOCATEDACTION specifies OTHER. It is ignored in all other cases.

Only one OTHERACTIONS can be used within a REPLACELINE stanza. OTHERACTIONS can be set to the following values:

Table 5. OTHERACTIONS substanza Values

Value	Usage
ADDBEFOREPATTERN	Specifies a string pattern used to select a line within the target system text file. If a matching line is found, the replacement line is added immediately before the selected line. If omitted, the pattern string is assumed to be null, no string searches are made, and the ADDBEFORESCOPE command is ignored. The command is specified as an FI Regular Expression enclosed in double quotation marks. FI variables are not allowed within the command parameter string. Only one ADDBEFOREPATTERN is allowed within an OTHERACTIONS block. ADDAFTERPATTERN and ADDBEFOREPATTERN are independent. You can supply two patterns and have the same replacement line inserted before one located line and after a different located line.
ADDBEFORESCOPE	Specifies the line within the target system text file that the ADDBEFOREPATTERN selects. The command is a keyword, not enclosed in double quotation marks. Set ADDBEFORESCOPE to either: FIRST Select the first line in the target file that contains a string matching the ADDBEFOREPATTERN search string. LAST Select the last line in the target file that contains a string matching the ADDBEFOREPATTERN search string. If this command is omitted, the implied value is FIRST. Only one ADDBEFORESCOPE is allowed within an OTHERACTIONS block.

Value	Usage
ADDAFTERPATTERN	Specifies a string pattern used to select a line within the target system text file. If a matching line is found, the replacement line is added immediately after the selected line. If omitted, the string pattern is assumed to be NULL, no string searches are made, and the ADDAFTERSCOPE command is ignored. The command is specified as an FI Regular Expression enclosed in double quotation marks. FI variables are not allowed within the command parameter string. Only one ADDAFTERPATTERN is allowed within an OTHERACTIONS block. ADDAFTERPATTERN and ADDBEFOREPATTERN are independent. You can supply two patterns and have the same replacement line inserted before one located line and after a different located line.
ADDAFTERSCOPE	Specifies the line within the target system text file that the ADDAFTERPATTERN selects. The command is a keyword, not enclosed in double quotation marks. Set ADDAFTERSCOPE to either: FIRST Select the first line in the target file that contains a string matching the ADDAFTERPATTERN search string. LAST Select the last line in the target file that contains a string matching the ADDAFTERPATTERN search string. If this command is omitted, the implied value is FIRST. Only one ADDAFTERSCOPE is allowed within an OTHERACTIONS block.

LOCATEDACTION Command <- OTHERACTIONS Sub-Sub-substanza <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

Use the LOCATEDACTION command to specify what PROBE should do if the ADDBEFOREPATTERN or the ADDAFTERPATTERN search strings find target text file lines within the currently selected files.

Table 6. LOCATEDACTION Command Keywords (OTHERACTIONS)

Value	Action
ADD	Add the replacement line to the selected file at the selected before or after position.
NOTHI	Do nothing.

LOCATEDACTION, if omitted, is assumed to be ADD. You can use LOCATEDACTION only once within an OTHERACTIONS sub-sub-substanza.

NOTLOCATEDACTION Command <- OTHERACTIONS Sub-Sub-Substanza <- TARGETLINE Sub-Substanza <- REPLACELINE Substanza <- TEXTFILE Stanza

The optional NOTLOCATEDACTION action specifies what action to take if no matching lines were found by either the ADDBEFOREPATTERN or the ADDAFTERPATTERN searches. It is also the action taken if no ADDBEFOREPATTERN or ADDAFTERPATTERN commands are included within the OTHERACTIONS sub-sub-substanza.

Table 7. NOTLOCATEDACTION Command Keywords (OTHERACTIONS)

Value	Action
BOF	Add the replacement line at the front of the target file.
EOF	Add the replacement string at the end of the target file.
NOTHING	Do nothing.

NOTLOCATEDACTION, if omitted, is assumed to be NOTHING. You can use NOTLOCATEDACTION only once within a OTHERACTIONS sub-sub-substanza block.

EDITLINE Substanza <- Textfile Stanza

Use the EDITLINE substanza to modify text within a target line.

The syntax of the EDITLINE command is as follows. Each level of indentation shows a level of subordination.

```
EDITLINE
   NOGREP
   CASESENSITIVE
   SOURCESTRING
      LINEPATTERN "XPG4 Extended Regular Expression pattern string"
      BEFOREPATTERN "XPG4 Extended Regular Expression pattern string"
      MATCHPATTERN "XPG4 Extended Regular Expression pattern string"
      REPLACEMENT "string"
      SETFIVAR "variable_name"
   ENDSOURCESTRING
  TARGETSTRING
      LINEPATTERN "FI Regular Expression pattern string"
      LINESCOPE [FIRST | LAST| EVERY]
      STRINGPATTERN "FI Regular Expression pattern string"
      STRINGSCOPE [FIRST | LAST| EVERY]
      DELIMITER "delimiter string"
      ADDLASTDELIMETER
      LOCATEDACTION [REPLACE | NOTHING]
      NOTLOCATEDACTION [BOF | EOF | OTHER | NOTHING]
      OTHERACTIONS
         ADDBEFOREPATTERN "FI regular expression pattern string"
         ADDBEFORESCOPE [FIRST | LAST]
         ADDAFTERPATTERN "FI regular expression pattern string"
         ADDAFTERSCOPE [FIRST | LAST]
         LOCATEDACTION [ADD | NOTHING]
         NOTLOCATEDACTION [BOL | EOL | |NOTHING]
      ENDOTHERACTIONS
   ENDTARGETSTRING
ENDEDITLINE
```

Figure 5. EDITLINE Textfile Substanza

EDITLINE starts a substanza block that defines how to edit lines in the text file on the target system identified with the TARGET and, optionally, SOURCE, commands in the TEXTFILE stanza block.

The target system for editing is the TARGET specified in the surrounding TEXTFILE block.

You can have zero or multiple EDITLINE commands within the TEXTFILE block. If there are no EDITLINE commands within the TEXTFILE block, there must be at least one REPLACELINE. REPLACELINE and EDITLINE command stanza blocks can be mixed in any order.

NOGREP Command <- EDITLINE Substanza <- TEXTFILE Stanza

The optional NOGREP command specifies to do all string searches in the EDITLINE block as simple string comparisons. The default, when NOGREP is omitted, is to do all string searches as XPG4 Extended Regular Expression comparisons for SOURCESTRING patterns and as FI regular expression comparisons for TARGETSTRING patterns. Other FI and string considerations are described separately.

The scope is limited to the EDITLINE block containing the NOGREP. Any REPLACELINE or EDITLINE command following the EDITLINE block with NOGREP set starts with the default.

An error is returned if more than one NOGREP is defined within a EDITLINE block.

CASESENSITIVE Command <- EDITLINE Substanza <- TEXTFILE Stanza

The optional CASESENSITIVE command specifies that all string searches are case sensitive. The sensitivity affects both simple string comparisons and Regular Expression searches. The default, if CASESENSITIVE is omitted, is to ignore case.

The scope is limited to the EDITLINE block containing the CASESENSITIVE. Any REPLACELINE or EDITLINE command following the EDITLINE block with CASESENSITIVE set starts with the default.

An error is returned if more than one CASESENSITIVE is defined within a EDITLINE block.

SOURCESTRING Command <- EDITLINE Substanza <- TEXTFILE Stanza

The required SOURCESTRING command marks the start of a sub-substanza specifying what string, if any, to use to replace strings in the target system. The result of the SOURCESTRING command is called the replacement string.

An error is returned if more than one SOURCESTRING is defined within a EDITLINE block.

The SOURCESTRING command block can contain one each of the LINEPATTERN, BEFOREPATTERN, MATCHPATTERN, REPLACEMENT, SETFIVAR, and ENDSOURCESTRING commands. An error is returned if SOURCESTRING is followed immediately by ENDSOURCESTRING.

End the SOURCESTRING block with the ENDSOURCESTRING command.

LINEPATTERN Command <- SOURCESTRING Sub-Substanza <- EDITLINE Substanza <- TEXTFILE Stanza

The optional LINEPATTERN command specifies a search string used by PROBE to search the source system text file and select the first line containing a matching string. The SOURCESTRING block can contain only one LINEPATTERN command.

The LINEPATTERN selects not only the line but also a current position in the line, the position following the last character in the string pattern that was recognized within the line.

If the string search fails to find a line in the source system text file containing a matching string, the search is considered to have failed. In that instance:

If a REPLACEMENT command is also included within this block, the REPLACEMENT command parameter then becomes the replacement string.

If there is no REPLACEMENT command within this block, the EDITLINE substanza terminates, only setting any SETFIVAR FI variables specified to a NULL string value.

The default value, if LINEPATTERN is omitted, is to do no source system text file search. In this case, avoid an error by providing a REPLACEMENT command within the SOURCESTRING command sub-substanza.

If the LINEPATTERN command finds a matching line, it logs both the entire line that was found and the "rest of the line" following the current position within the line that was set by the search. This helps you see where the end of the recognized string is.

The LINEPATTERN is an XPG4 Extended Regular Expression enclosed in double quotation marks. It cannot contain FI variable references.

If you want not finding a LINEPATTERN match to generate an error message and terminate processing, code ISNOTFOUNDERROR = YES.

BEFOREPATTERN Command <- SOURCESTRING Sub-Substanza <-EDITLINE Substanza <-TEXTFILE Stanza

Use the BEFOREPATTERN to set the cursor position for MATCHPATTERN before looking for the replacement string within the line. The BEFOREPATTERN lets you provide a pattern to search with within the source text line selected by LINEPATTERN. The search begins at the start of the selected text line.

The default, if BEFOREPATTERN is omitted, is to omit this string search and leave the current position unchanged.

If the BEFOREPATTERN command finds a match within the selected line, it logs the "rest of the line" following the "current position" within the line that was set by the search. The BEFOREPATTERN is an XPG4 Extended Regular Expression enclosed in double quotation marks. It cannot contain FI variable references.

Only one BEFOREPATTERN command can be coded within a SOURCESTRING command sub-substanza block.

MATCHPATTERN Command <- SOURCESTRING Sub-substanza <- EDITLINE substanza <-TEXTFILE Stanza

Use the MATCHPATTERN command to select the replacement string from the currently selected line within the source text file. The search begins at the current position within the text line. The default, if MATCHPATTERN is omitted, is to do no source system text file search. In this case, avoid an error by providing a REPLACEMENT command within the SOURCESTRING command sub-substanza.

If LINEPATTERN, BEFOREPATTERN, or MATCHPATTERN's string search fails to find a string in the source system text file that contains a matching string, the search is considered to have failed with the following conditions:

Table 8. Results when Matchpattern Command Text Search Fails

REPLACEMENT specification	ISNOTFOUNDERROR specification	Result
Specified	Not set	The REPLACEMENT command parameter becomes the replacement line.
Not specified	YES	PROBE fails with an error.
Not specified	Set to NO or WARN	The EDITLINE substanza only sets the SETFIVAR command, FI variable, if any, to NULL.

If MATCHAPTTERN finds a match within the selected line, it logs the substring that was found within the selected line.

The MATCHPATTERN is an XPG4 Extended Regular Expression enclosed in double quotation marks. It cannot contain FI variable references.

Only one MATCHPATTERN command can be coded within a SOURCESTRING command sub-substanza block. If MATCHPATTERN is coded, and STRINGPATTERN is not coded in the TARGETSTRING sub-substanza, the MATCHPATTERN is used as the string pattern.

REPLACEMENT Command <- SOURCESTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

Use the REPLACEMENT command for one of the following:

- Specify a string to be used as a full replacement for the selected strings in the target system. Specify the REPLACEMENT as an ordinary string enclosed in double quotation marks. The string can contain references to FI variables.

- Specify that selected strings in the target file are to be deleted. Code the line:

```
REPLACEMENT ""
```

The default value, if REPLACEMENT is omitted, is that there is no specific text line to place in the target system. To avoid an error in this case, supply a LINEPATTERN and MATCHPATTERN command within the SOURCESTRING command sub-substanza.

If both LINEPATTERN and MATCHPATTERN are specified in the SOURCESTRING command sub-substanza, the REPLACEMENT value is used as the replacement string if either the LINEPATTERN or the MATCHPATTERN searches fail.

If REPLACEMENT results in a change to the replacement string, the new replacement string is logged.

The REPLACEMENT command parameter is an ordinary string within double quotation marks. It can contain references to FI variables. FI variable references are resolved on the target system during the installation of the FI Install object during the execution of the configurator tool.

SETFIVAR Command <- SOURCESTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

Use the optional SETFIVAR command to specify the name of an FI variable. The variable is set to the value of the replacement string generated by the SOURCESTRING command substanza block. This lets you pass source system text file lines and strings to the configurator tool and to FI user exits.

If the FI variable is already defined, the value is simply changed to any new value. If the FI variable does not yet exist, the variable is created and its value is set.

The SETFIVAR command parameter is an ordinary string enclosed in double quotation marks and may contain references to FI variables. Any FI variable references in the SETFIVAR command parameter string are resolved during PROBE execution. The FI variable name is specified without the pair of enclosing braces {}.

Only one SETFIVAR command can be specified within a SOURCESTRING command sub-substanza block.

TARGETSTRING Command <- EDITLINE Substanza <- TEXTFILE Stanza

The optional TARGETSTRING command starts sub-substanza block specifying what string, if any, will be used to replace strings on the target system.

End the TARGETSTRING sub-substanza with the matching ENDTARGETSTRING command.

LINEPATTERN Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

The optional LINEPATTERN command specifies a search string used by the configurator to search the target system text file to select one or more lines to operate on. The TARGETSTRING block can contain only one LINEPATTERN command.

The default value, if LINEPATTERN is omitted, is to use the LINEPATTERN command specified in the SOURCESTRING command sub-substanza.

The LINEPATTERN is an FI Regular Expression enclosed in double quotation marks. It cannot contain FI variable references. Other FI and string considerations are described separately.

LINESCOPE Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

The optional LINESCOPE command specifies which target system text file lines to select. The LINESCOPE keyword parameter must be one of the following:

Table 9. LINESCOPE Command Keywords

Keyword	Action
FIRST	Select the first line containing the search string.
LAST	Select the last line containing the search string.
EVERY	Select every line containing the search string.

The default value is FIRST if LINESCOPE is omitted. The TARGETSTRING block can contain only one LINEPATTERN command.

STRINGPATTERN Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

After you have specified the target system text lines, you must also select particular strings within those lines. Use the optional

STRINGPATTERN command to specify the search string to be used within those lines.

If STRINGPATTERN is omitted, then MATCHPATTERN must be specified in the SOURCESTRING command sub-substanza block. That MATCHPATTERN is used as the string pattern. You can specify both STRINGPATTERN and MATCHPATTERN, or just one of the two.

The STRINGPATTERN is specified as an FI Regular Expression enclosed in double quotation marks. The STRINGPATTERN command parameter cannot contain FI variable references.

The TARGETSTRING block can contain only one STRINGPATTERN command.

STRINGSCOPE Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

The optional STRINGSCOPE command specifies which matching strings within the currently selected target system text file lines to select. The STRINGSCOPE keyword parameter must be one of the following:

Table 10. LINESCOPE Command Keywords

Keyword	Action
FIRST	Select the first string within the line containing the search string.
LAST	Select the last string within the line containing the search string.
EVERY	Select every string within the line containing the search string.

The default value is FIRST if STRINGSCOPE is omitted. The TARGETSTRING block can contain only one STRINGSCOPE command.

DELIMITER Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

The DELIMITER command helps you edit lines consisting mostly of delimited fields. For example, the PATH and LIBPATH statements in CONFIG.SYS consist mostly of directory names delimited by semicolons. When you specify DELIMITER, the replacement string value is used to replace an entire delimited field, not just the part of the field that the search strings match. For example, assume you want to replace a field in a path like:

```
;X:\FOO;
```

You know to look for FOO, but you do not know what the drive will be on any given system. You can code:

```
DELIMITER ";"
```

to replace X:\FOO; with the replacement string.

The TARGETSTRING block can contain only one DELIMITER command. The default, if DELIMITER is omitted, is to have no specified delimiter string. If DELIMITER is omitted, any ADDLASTDELIMITER string specified is ignored.

ADDLASTDELIMITER Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <- TEXTFILE Stanza

Use the optional ADDLASTDELIMITER command to ensure that, if a replacement string is appended to a line, the specified delimiter string is appended to the end of the replacement string. The TARGETSTRING block can contain only one ADDLASTDELIMITER command.

LOCATEDACTION Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

Use the optional LOCATEDACTION command to tell PROBE what to do if it finds at least one string in the selected target system lines matching the STRINGPATTERN search criteria.

Table 11. LOCATEACTION Command Keywords

REPLACE	Replace each matching string with the replacement string. If the replacement string is NULL, the result is one or more string deletions.
NOTHING	Do nothing.

The default, if LOCATEDACTION is omitted, is REPLACE. The TARGETSTRING block can contain only one LOCATEDACTION command.

NOTLOCATEDACTION Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <- TEXTFILE Stanza

The optional NOTLOCATEDACTION action specifies what action to take if no target system text file lines are selected by the STRINGPATTERN command. The search is considered unsatisfied, and this command is executed.

The NOTLOCATEDACTION keyword is one of the following:

Table 12. NOTLOCATEACTION Command Keywords

Value	Action
BOL	Add the replacement line in front of each selected line.
EOL	Add the replacement string at the end of each selected line.
NOTHING	Do nothing

NOTLOCATEDACTION, if omitted, is assumed to be NOTHING.

You can use NOTLOCATEDACTION only once within a TARGETSTRING sub-substanza block.

OTHERACTIONS Command <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <-TEXTFILE Stanza

OTHERACTIONS provides a set of extended line editing choices within an EDITLINE stanza. Use OTHERACTIONS only if NOTLOCATEDACTION specifies OTHER. It is ignored in all other cases.

To get to this point, the initial target system text file search has either failed to find at least one line within the text file that matches one given search pattern, or has failed to find at least one string within those selected lines that matches a second given search pattern.

This command gives you a chance to search for different strings within the selected text file lines. You can specify that the replacement string be added either before or after the new search string, or be added at the start or the end of the currently selected lines.

Only one OTHERACTIONS can be used within an EDITLINE stanza. If specified, it must be set to one of the following values:

Table 13. NOTLOCATEACTION Command Keywords

Value	Usage
ADDBEFOREPATTERN	Specifies a string pattern used to select one or more strings within each currently selected line. If any strings are found, the LOCATEDACTION command in this block determines whether the replacement string added immediately before the selected strings. If omitted, the pattern string is assumed to be null; no string searches are made, and the ADDBEFORESCOPE command is ignored. The command is specified as an FI Regular Expression enclosed in double quotation marks. FI variables are not allowed within the command parameter string. Only one ADDBEFOREPATTERN is allowed within an OTHERACTIONS block.

Value	Usage
ADDBEFORESCOPE	Specifies whether the replacement string should be added immediately before either the first substring or the last substring selected by the ADDBEFOREPATTERN string search. ADDBEFORESCOPE takes either of the following keywords: FIRST Add the replacement string immediately before the first string within the line that contains the ADDBEFOREPATTERN search string. LAST Add the replacement string immediately before the last string within the line that contains the ADDBEFOREPATTERN search string. If this command is omitted, the implied value is FIRST. ADDAFTERPATTERN and ADDBEFOREPATTERN are independent. You can supply two patterns and have the same replacement line inserted before one located line and after a different located line. Only one ADDBEFORESCOPE is allowed within an OTHERACTIONS block.

Value	Usage
ADDAFTERPATTERN	Selects one or more strings within each currently selected line. If any strings are selected by this command, the LOCATEDACTION command in this block determines whether to add the replacement string immediately after the selected strings. If omitted, the pattern string is assumed to be null, no string searches are made, and the ADDBEFORESCOPE command is ignored. The command is specified as an FI Regular Expression enclosed in double quotation marks. FI variables are not allowed within the command parameter string. Only one ADDAFTERPATTERN is allowed within an OTHERACTIONS block. ADDAFTERPATTERN and ADDBEFOREPATTERN are independent. You can supply two patterns and have the same replacement line inserted before one located line and after a different located line.
ADDAFTERSCOPE	Specifies whether the replacement string should be added immediately after either the first substring or the last substring selected by the ADDAFTERPATTERN string search. ADDAFTERSCOPE takes either of the following keywords: FIRST Add the replacement string immediately after the first string within the line that contains the ADDAFTERPATTERN search string. LAST Add the replacement string immediately after the last string within the line that contains the ADDAFTERPATTERN search string. If this command is omitted, the implied value is FIRST. Only one ADDAFTERSCOPE is allowed within an OTHERACTIONS block.

LOCATEDACTION Command <- OTHERACTIONS Sub-Sub-Substanza <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <- TEXTFILE Stanza

Use the LOCATEDACTION command to specify what PROBE should do if the ADDBEFOREPATTERN or the ADDAFTERPATTERN search strings find target text file strings within the currently selected lines.

The LOCATEDACTION command takes one of the following keywords:

Table 14. LOCATEACTION Command Keywords

ADD	Add the replacement string to the selected lines at all the selected before or after positions.
NOTHING	Do nothing.

LOCATEDACTION, if omitted, is assumed to be ADD. You can use LOCATEDACTION only once within an OTHERACTIONS sub-sub-substanza block.

The **NOTLOCATEDACTION Command <- OTHERACTIONS Sub-Sub-Substanza <- TARGETSTRING Sub-Substanza <- EDITLINE Substanza <- TEXTFILE Stanza**

The optional NOTLOCATEDACTION action specifies what action to take if no matching strings were found within the currently selected lines by either the ADDBEFOREPATTERN or the ADDAFTERPATTERN string searches. It is also the action taken if no ADDBEFOREPATTERN or ADDAFTERPATTERN commands are included within the OTHERACTIONS sub-sub-substanza.

The NOTLOCATEDACTION keyword is one of the following:

Table 15. NOTLOCATEACTION Command Keywords (OTHERACTIONS)

Value	Action
BOL	Add the replacement line in front of each selected line.
EOL	Add the replacement string at the end of each selected line.
NOTHING	Do nothing.

NOTLOCATEDACTION, if omitted, is assumed to be NOTHING.

You can use NOTLOCATEDACTION only once within a OTHERACTIONS sub-sub-substanza block.

1.3.6.2 Text File Examples

The following examples show several sequences of text file commands.

Finding and Transferring a String from CONFIG.SYS

This example finds a line in CONFIG.SYS that contains the HOSTNAME string and transfers it to the updated target system. It also shows the use of FI variables specifying the location of the boot drive on both the source and target systems.

```
TEXTFILE SOURCE "<BootDrive>\CONFIG.SYS"
  REPLACELINE
    SOURCELINE PATTERN "HOSTNAME" ENDSOURCELINE
    TARGETLINE PATTERN "HOSTNAME" ENDTARGETLINE
  ENDREPLACELINE
ENDTEXTFILE
```

Because the target system text file pathname is the same as the source system's text file pathname, you did not need to supply a TARGET command. The *<BootDrive>* FI variable is set by both PROBE and by the FI run-time; so you need not supply a PROBE user exit or FI user exit to set the variable.

This example could be simplified because the SOURCELINEPATTERN and the TARGETLINEPATTERN are the same, as may often happen. You could omit the TARGETLINE sub-substanza. The following example shows this coding efficiency.

Extracting an IP Address

In the following example, the IP address is extracted from the ifconfig line in the MPTS SETUP.CMD command file:

```
TEXTFILE
  SOURCE "<BootDrive>\MPTN\BIN\SETUP.CMD"
    EDITLINE
      SOURCESTRING
        LINEPATTERN "ifconfig"
        MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
    ENDEDITLINE
ENDTEXTFILE
```

The regular expression `"[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"` is used to match a four-part IP address. The `\.` sequence is the escape sequence to match the period character.

You do not need a `TARGETSTRING` sub-substanza because the source and target line and string selectors are identical.

1.3.7 OS/2 INI File Configuration Commands

The OS/2 INI file configuration commands let you tell `PROBE` to read information from an OS/2-style INI file in order to copy information into the INI file of the same name on an updated OS/2 target system. You can also provide specific values to place on the updated OS/2 target system. In that case, values are not read from the old target system. There is also a block of stanza-style INI commands to use with stanza-style files.

You can use the `ISNOTFOUNDERROR` to control some interpretation of search failures. The `ISNOTFOUNDERROR` must precede the `OS2INI` command stanza block where you want to control search failure actions.

If a `TARGET` parameter is not specified, and if the corresponding `SOURCE` parameter contains an FI variable reference, the resulting default `TARGET` parameter—inherited from the corresponding `SOURCE` parameter string—does not have its FI variable reference resolved until FI install object installation on the target system.

The corresponding `SOURCE` string has its FI variable reference resolved during `PROBE` execution.

The syntax of the OS/ 2 INI file configuration command stanza is listed below. Some examples of `OS2INI` command file syntax are also provided.

```
// OS/2 INI file configuration command stanza
// Indentation is for ease of reading, it is not required

OS2INI
   SOURCE "source system OS/2 INI file pathname"
   TARGET "target system OS/2 INI file pathname"
   DELAPPL "target system application name"
   APPL
      SOURCE "source system application name"
      TARGET "target system application name"
      DELKEY "target system key name
      KEY
         SOURCE "source system key name"
         TARGET "target system key name"
         BINARY
         VALUE "user-supplied value for the (application,key) pair"
         SETFIVAR "variable name"
      ENDKEY
   ENDAPPL
   COPYAPPL
      SOURCE "source system application name"
      TARGET "target system application name"
   ENDCOPYAPPL
ENDOS2INI
```

Figure 6. OS/2INI Stanza

1.3.7.1 OS2INI Stanza

The `OS2INI` command marks the beginning of a stanza block defining a list of specifications relative to a specific OS/2-style INI file.

Each `OS2INI` block must end with `ENDOS2INI`.

SOURCE Command <- OS2INI Stanza

Use the `SOURCE` command to specify the pathname of an OS/2-style INI file on the source system. This line is used to search for specific key values.

A `SOURCE` command must be executed before any `KEY` sub-substanza blocks that obtain their values from the source system. A `SOURCE` command must be executed before any `DELAPPL` or `COPYAPPL` commands.

An OS2INI command stanza block can contain only one SOURCE command.

The SOURCE command parameter is an ordinary string within double quotation marks. It may contain FI variables which are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET <- OS2INI Stanza

Use the TARGET command to specify the pathname of an OS/2-style INI file on the target system. This is the file modified by the KEY sub-substanza blocks and by the DELAPPL, COPYAPPL, and DELKEY commands in the OS2INI stanza block.

If you do not specify TARGET, and a SOURCE is specified in the OS2INI command stanza block, the target pathname is assumed to be the same as the source pathname. However, in this case, any FI variable references in the source pathname are not resolved so that they can be resolved later on the target system during the installation of the FI install object during execution of the configurator tool.

If TARGET is specified, but no SOURCE command is specified within the OS2INI command stanza block, then you must provide specific values for all the (Application,Key) pairs subsequently coded in the KEY command sub-substanza blocks. Otherwise, PROBE returns an error.

If the TARGET command is placed after APPL substanza blocks, DELAPPL and COPYAPPL commands, the TARGET pathname does not have an effect on those APPL substanza blocks or DELAPPL commands. It does have an effect on any APPL substanza blocks, COPYAPPL substanza blocks and DELAPPL commands following the TARGET command within the OS2INI command stanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved during the installation of the FI install object during execution of the configurator tool.

An OS2INI command stanza block can contain only one TARGET command.

DELAPPL <- OS2INI Stanza

Use the DELAPPL command to define the name of an entire application section to be deleted along with all its (Key,Value) pairs in the specified target system OS2-style INI file.

The identity of the specified target system OS2-style INI file depends on the preceding SOURCE and TARGET commands within the OS2INI command stanza block.

The DELAPPL command parameter is an ordinary string enclosed in double quotation marks. There can be zero or more DELAPPL commands in an OS2INI command stanza block. Any FI variable references within the DELAPPL command parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

APPL <- OS2INI Stanza

Use the APPL command to mark the start of a substanza block defining the information necessary to identify application sections in OS/2-style INI files.

The identities of the specified source and target system OS2INI-style INI file depend on the preceding SOURCE and TARGET commands within the OS2INI command stanza block.

There can be zero or more APPL commands in an OS2INI command stanza block.

SOURCE Command <- APPL Substanza <- OS2INI Stanza

Use the SOURCE command to specify the name of an application section within an OS2-style INI file on the source system. This application name is used to search for specific key values. The application name cannot be specified as a binary value.

An APPL command substanza block SOURCE command must be executed before any KEY sub-substanzas requiring key values from application sections on the source system.

An OS2INI command stanza block can contain only one TARGET command. FI variables in the source command parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET <- APPL Substanza <- OS2INI Stanza

Use the TARGET command to specify the application section within an OS2-style INI file on the target system.

This is the application section to be modified by the KEY command sub-substanza blocks and the DELKEY commands that are coded in the APPL command substanza block.

If a TARGET command is not specified and a SOURCE command is specified within the APPL command substanza block, the target application name is assumed to be the same as the source application name. However, in this case, any FI variable references in the source application name are left unresolved so that they can be subsequently resolved on the target system during the installation of FI install object during the execution of the configurator tool.

If TARGET is specified, but no SOURCE command is specified within the APPL command substanza block, you must provide specific values for all the (Application,Key) pairs subsequently coded in the KEY command sub-substanza blocks. Otherwise, PROBE returns an error.

If the TARGET command is placed after KEY sub-substanza blocks and DELKEY commands, the TARGET application name does not have an effect on those KEY sub-substanza blocks and DELKEY commands. It has an effect on any following DELKEY commands and KEY sub-substanza blocks within the OS2INI command stanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

An APPL command substanza block can contain only one TARGET command.

DELKEY <- APPL Substanza <- OS2INI Stanza

Use the DELKEY command to define the name of a key to delete, along with its value, in the specified application section of the specified target system OS2-style INI file.

The identity of the specified target system OS2-style INI file depends on the preceding SOURCE and TARGET commands within the OS2INI command stanza block. The identity of the specified application section depends on the preceding SOURCE and TARGET commands within the APPL command sub- stanza block.

There can be zero or more DELKEY commands in an OS2INI command stanza block.

The DELKEY command parameter is an ordinary string enclosed in double quotation marks. FI variables in the COPYAPPL parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

KEY Command <- APPL Substanza <- OS2INI Stanza

Use the KEY command to mark the beginning of a sub-substanza block identifying key values with an application section within OS/2-style INI files. End the KEY sub-substanza block with the ENDKEY command. There can be zero or more KEY commands in an OS2INI command stanza block.

The KEY command parameter is an ordinary string enclosed in double quotation marks. The KEY name cannot be specified as a binary value.

SOURCE <- KEY Sub-Substanza <- APPL Substanza <- OS2INI Stanza

Use the SOURCE command to specify the name of a key within an application section in an OS/2-style INI file on the source system. The key name cannot be specified as a binary value. This key name is used to search for specific key values.

If the OS/2-style INI file search fails to find a matching key, the search has failed. In this case, the following happens:

Table 16. Results when Source Key Command Search Fails

VALUE command	ISNOTFOUNDERROR specification	Result
Specified	Not set	VALUE provides the key value to pass to the target system.
Not specified	YES	PROBE fails with an error.
Not specified	Set to NO or WARN	The KEY only sets the SETFIVAR command, FI variable, if any, to NULL.

If you do not code a SOURCE command within the KEY sub-substanza block, no INI file search is done. In this case, you must provide a VALUE command within this block.

If neither a SOURCE command nor a VALUE command are specified within a KEY command sub-substanza block, an error is returned.

Only one SOURCE command can be defined within a KEY command sub- substanza block.

The SOURCE command parameter is an ordinary string enclosed in double quotation marks. FI variables in the SOURCE parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGETCommand<-KEYSub-Substanza<-APPLSubstanza<-OS2INIStanza

Use the TARGET command to specify the name of a key within an application section in an OS/2-style INI file on the target system. This is the key whose value is modified by the KEY sub-substanza block.

If the TARGET command is not specified and a SOURCE command is specified within the KEY sub-substanza block, the target key name is assumed to be the same as the source key name. However, any FI variable references in the source key name are left unresolved so that they can be subsequently resolved on the target system during the installation of the FI install object during execution of the configurator tool.

If a TARGET command is specified and no SOURCE command is specified within the KEY sub-substanza block, you must provide a specific value with the VALUE command for the (Key,Value) pair on the target system. If not provided, PROBE returns an error.

Only one TARGET command can be defined within a KEY command sub- substanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

BINARYCommand<-KEYSub-Substanza<-APPLSubstanza<-OS2INIStanza

Use the optional BINARY command to specify that the key values on the source and target sides are both binary values instead of string values.

If you have specified a value with the VALUE command, the user-specified value of the (Application,Key) pair string parameter is interpreted as a string representing a hexadecimal number of arbitrary size, that is "127BEF45" or "46FB34906F4A". There must be an even number of hexadecimal digits in the string. The valid digits are "0123456789ABCDEFabcdef."

Only one BINARY command can be defined in a KEY command sub-substanza block.

VALUECommand<-KEYSub-Substanza<-APPLSubstanza<-OS2INIStanza

Use the VALUE command to specify a value to be associated with the corresponding key on the target system.

Use VALUE as follows:

- To provide specific values if no KEY SOURCE name is provided.
- To provide default values if a KEY SOURCE name is provided, but the search for a matching source-side key name fails.
- To specify that a (Key,Value) pair on the target system is to be deleted by specifying the VALUE command as a null string. (This can work as a default or as an explicit setting.)

If you do not code a VALUE in the KEY command sub-substanza block, then PROBE reads the value associated with the corresponding key on the source system.

The (Key,Value) pair, if any, read from the source system and the replacement (Key,Value) pair generated by the KEY sub-substanza are logged.

Only one VALUE command can be defined within a KEY command sub- substanza block.

The VALUE command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

SETFIVAR Command <- KEY Sub-Substanza <- APPL Substanza <- OS2INI Stanza

Use the optional SETFIVAR command to specify the name of an FI variable. The variable is set to the value field of the (Key,Value) pair generated by the KEY command sub-substanza block. This lets you pass source system text file lines and strings to PROBE and FI user exits.

If the FI variable is already defined, the value is simply changed to any new value. If the FI variable does not yet exist, the variable is created and its value is set.

The SETFIVAR command parameter is an ordinary string enclosed in double quotation marks and may contain references to FI variables. Any FI variable references in the SETFIVAR command parameter string are resolved during PROBE execution. The FI variable name is specified without the pair of enclosing braces {}.

Only one SETFIVAR command can be specified within a KEY command sub-substanza block.

COPYAPPL Command <- OS2INI Stanza

Use the COPYAPPL command to start a substanza defining a set of specifications defining how to copy an entire application section from the specified source system INI file to the specified target system INI file.

When an OS/2-style INI file application section is copied, (Key,Value) replacement is done if a source (Key,Value) pair shares its key name with a (Key,Value) pair in the target application section. All other source (Key,Value) pairs are added to the target application section. This behavior matches the behavior of the OS2INI KEY command sub-substanza block.

There can be zero or more COPYAPPL commands in an OS2INI command stanza block. End the COPPYAPPL sub-substanza with the ENDCOPYAPPL command.

SOURCE Command <- COPYAPPL Substanza <- OS2INI Stanza

Use the SOURCE command to specify the name of an application section within an OS/2-style INI file on the source system. This is the application section to be copied to the target system.

An COPYAPPL substanza block can contain only one TARGET command. FI variables in the source command parameter string are resolved during PROBE execution. The resolved parameter string and the set of (Key,Value) pairs copied from the source system are logged if you specified extended logging.

TARGET Command <- COPYAPPL Substanza <- OS2INI Stanza

Use the TARGET command to specify the application section within an OS2-style INI file on the target system. This is the application section to be replaced with or appended to by the set of (Key,Value) entries obtained from the source system.

If a TARGET command is not specified and a SOURCE command is specified within the COPYAPPL command substanza block, the target application name is assumed to be the same as the source application name. However, in this case, any FI variable references in the source application name are left unresolved so that they can be subsequently resolved on the target system during the installation of FI install object during the execution of the configurator tool.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

A COPYAPPL command substanza block can contain only one TARGET command.

1.3.7.2 Examples of OS/2 INI Command File Syntax

In both examples here, the FI variable *{BootDrive}*, which is maintained on the source system side by PROBE and on the target system by the FI run-times, is used to help find the location of the OS2.INI file.

The first example shows how a value from the OS2.INI file on a source system is copied over to the OS2.INI file on the target system.

The second example shows how an OS2.INI (Application,Key) pair on the target is set to a specific value.

```
//Command stream file modifying an OS/2 INI file

OS2INI
SOURCE "{BootDrive}\OS2\OS2.INI"

// Example 1. Copying a value from the source to the target
APPL
SOURCE "PM_DISPLAYDRIVERS"
KEY
SOURCE "DEFAULTSYSTEMRESOLUTION"
BINARY
ENDKEY
ENDAPPL

// Example 2. Placing a specific value on the target system
APPL
TARGET "PM_COM1"
VALUE "9600;8;1;1"
ENDAPPL
ENDOS2INI
```

Figure 7. Examples of OS/2 INI Command File Syntax

1.3.8 Stanza-based INI File Configuration Commands

Use the stanza-based INI file configuration commands to tell PROBE to read information from a stanza-based INI file for eventual copying onto the corresponding INI file on the updated target OS/2 system. Or, you can provide new values to copy onto the updated target OS/2 system. In that case, the values from the old system are not used.

The main types of stanza-based INI files found on OS/2 systems are Windows INI files and OS/2 communications INI files such as PROTOCOL.INI.

The stanza-based INI files are similar to the OS/2 INI configuration commands. The STANZAINI command stanza block ends with the ENDSTANZAINI command.

You can use the ISNOTFOUNDERROR command to control the interpretation of failures to located specified INI files, application names, and key names. The ISNOTFOUNDERROR command must precede the STANZAINI command block where you want it to control interpretation.

Within the STANZAINI block, if a TARGET parameter is not specified, and if the corresponding SOURCE parameter contains an FI variable reference, the resulting default TARGET parameter, (the one inherited from the corresponding SOURCE parameter) will not have its FI variable reference resolved until FI object installation on the target system. The corresponding SOURCE string has its FI variable resolved during PROBE execution.

The syntax of the STANZAINI file configuration command stanza is described below.

```
// Summary of Stanza-based INI file configuration
// command stanza
// The indentation is for ease-of-reading

STANZAINI
   SOURCE "source system Stanza-based INI file pathname"
   TARGET "target system Stanza-based INI file pathname"
   DELAPPL "target system application name"
   APPL
      SOURCE "source system application name"
      TARGET "target system application name"
      DELKEY "target system key name"
      KEY
         SOURCE "source system key name"
         TARGET "target system key name"
         VALUE "user- specified value for the (application,key) pair"
         ADDVALUES
         SETFIVAR "variable name"
         SETLASTFIVAR "variable name"
      ENDKEY
   ENDAPPL
   COPYAPPL
      SOURCE "source system application name"
      TARGET "target system application name"
      ADDVALUES
   ENDCOPYAPPL
ENDSTANZAINI
```

Figure 8. STANZAINI Stanza

1.3.8.1 STANZAINI Command Stanza Block

The following STANZAINI command stanza blocks are available:

SOURCE Command <- STANZAINI Stanza

Use the SOURCE command to specify the pathname of a stanza-style INI file on the source system. This line is used to search for specific key values.

A SOURCE command must be executed before any KEY sub-substanza blocks that obtain their values from the source system.

A STANZAINI command stanza block can contain only one SOURCE command.

The SOURCE command parameter is an ordinary string within double quotation marks. It may contain FI variables which are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET Command <- STANZAINI Stanza

Use the TARGET command to specify the pathname of an stanza-style INI file on the target system. This is the file modified by the KEY sub-substanza blocks and by the DELAPPL and DELKEY commands in the STANZAINI stanza block.

If you do not specify TARGET, and a SOURCE is specified in the STANZAINI command stanza block, the target pathname is assumed to be the same as the source pathname. However, in this case, any FI variable references in the source pathname are not resolved so that they can be resolved later on the target system during the installation of the FI install object during execution of the configurator tool.

If TARGET is specified, but no SOURCE command is specified within the STANZAINI command stanza block, then you must provide specific values for all the (Application,Key) pairs subsequently coded in the KEY command sub-substanza blocks. Otherwise, PROBE returns an error.

If the TARGET command is placed after APPL substanza blocks, and DELAPPL commands, the TARGET pathname does not have an effect on those APPL substanza blocks or DELAPPL commands following the TARGET command within the STANZAINI command stanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved during the installation of the FI install object during execution of the configurator tool.

An STANZAINI command stanza block can contain only one TARGET command.

DELAPPL Command <- STANZAINI Stanza

Use the DELAPPL command to define the name of an entire application section to be deleted, along with all its (Key,Value) pairs, in the specified target system stanza-style INI file.

The identity of the specified target system stanza-style INI file depends on the preceding SOURCE and TARGET commands within the STANZAINI command stanza block.

There can be zero or more DELAPPL commands in an STANZAINI command stanza block. The TARGET command parameter is an ordinary string enclosed in double quotation marks. Any FI variable references within the DELAPPL command parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

APPL Command <- STANZAINI Stanza

Use the APPL command to mark the start of a substanza block defining the information necessary to identify application sections in stanza-style INI files.

The identities of the specified source and target system stanza-style INI file depend on the preceding SOURCE and TARGET commands within the OS2INI command stanza block.

There can be zero or more APPL commands in an STANZAINI command stanza block.

SOURCE Command <- APPL Substanza <- STANZAINI Stanza

Use the SOURCE command to specify the name of an application section within a stanza-style INI file on the source system. This application name is used to search for specific key values. The application section name must be in the ASCII-7 character set and is mapped to lowercase characters when compared to application section names within a stanza-style INI file.

An APPL command substanza block SOURCE command must be executed before any KEY sub-substanzas requiring key values from application sections on the source system.

An APPL command substanza block can contain only one SOURCE command. FI variables in the source command parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET Command <- APPL Substanza <- STANZAINI Stanza

Use the TARGET command to specify the application section within a stanza-style INI file on the target system. This is the application section to be modified by the KEY command sub-substanza blocks and the DELKEY commands that are coded in the APPL command substanza block. The application section name must be in the ASCII-7 character set and is mapped to lowercase characters when compared to application section names within a stanza-style INI file.

If a TARGET command is not specified and a SOURCE command is specified within the APPL command substanza block, the target application name is assumed to be the same as the source application name. However, in this case, any FI variable references in the source application name are left unresolved so that they can be subsequently resolved on the target system during the installation of FI install object during the execution of the configurator tool.

If TARGET is specified, but no SOURCE command is specified within the APPL command substanza block, then you must provide specific values for all the (Application,Key) pairs subsequently coded in the KEY command sub-substanza blocks. Otherwise, PROBE returns an error.

If the TARGET command is placed after KEY sub-substanza blocks and DELKEY commands, the TARGET application name does not have an effect on those KEY sub-substanza blocks and DELKEY commands. It has an effect on any following DELKEY commands and KEY sub-substanza blocks within the STANZAINI command stanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

An APPL command substanza block can contain only one TARGET command.

DELKEY Command <- APPL Substanza <- STANZAINI Stanza

Use the DELKEY command to define the name of a key to delete, along with its value, in the specified application section of the specified target system stanza-style INI file.

The identity of the specified target system stanza-style INI file depends on the preceding SOURCE and TARGET commands within the STANZAINI command stanza block. The identity of the specified application section depends on the preceding SOURCE and TARGET commands within the APPL command sub- stanza block.

There can be zero or more DELKEY commands in an OS2INI command stanza block.

The DELKEY command parameter is an ordinary string enclosed in double quotation marks. FI variables in the COPYAPPL parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

KEY Command <- APPL substanza <- STANZAINI Stanza

Use the KEY command to mark the beginning of a sub-substanza block identifying key values with an application section within stanza-style INI files. End the KEY sub-substanza block with the ENDKEY command.

The identities of the specified source and target system stanza-style INI files depend on the preceding SOURCE and TARGET commands within the STANZAINI command stanza block. The identities of the specified source and target application section names depend on the preceding SOURCE and TARGET commands within the APPL command substanza block.

There can be zero or more KEY commands in an STANZAINI command stanza block.

SOURCE Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the SOURCE command to specify the name of a key within an application section in an stanza-style INI file on the source system. This key name is used to search for specific key values. The application section name must be in the ASCII-7 character set and is mapped to lowercase characters when compared to application section names within a stanza-style INI file.

Multiple (Key,Value) pairs that share the same key name can be found within an application section. PROBE extracts all (Key,Value) pairs within the specified application section with the specified key name.

If the stanza-style INI file search fails to find a matching key, the search has failed. In this case, the following happens:

Table 17. Results when Source Key Command Search Fails

VALUE command	ISNOTFOUNDERROR specification	Result
Specified	Not set	VALUE provides the key value to pass to the target system.
Not specified	YES	PROBE fails with an error.
Not specified	Set to NO or WARN	The KEY only sets the SETFIVAR command, FI variable, if any, to NULL.

If you do not code a SOURCE command within the KEY sub-substanza block, no INI file search is done. In this case, you must provide a VALUE command within this block.

If neither a SOURCE command nor a VALUE command are specified within a KEY command sub-substanza block, an error is returned.

Only one SOURCE command can be defined within a KEY command sub- substanza block.

The SOURCE command parameter is an ordinary string enclosed in double quotation marks. FI variables in the SOURCE parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the TARGET command to specify the name of a key within an application section in a stanza-style INI file on the target system. This is the key whose value is modified by the KEY sub-substanza block. The key name must be in the ASCII-7 character set and is mapped to lowercase characters when compared to application section names within a stanza-style INI file.

If the TARGET command is not specified and a SOURCE command is specified within the KEY sub-substanza block, the target key name is assumed to be the same as the source key name. However, any FI variable references in the source key name are left unresolved so that they can be subsequently resolved on the target system during the installation of the FI install object during execution of the configurator tool.

If a TARGET command is specified and no SOURCE command is specified within the KEY sub-substanza block, you must provide a specific value with the VALUE command for the (Key,Value) pair on the target system. If not provided, PROBE returns an error.

The default action of the configurator tool is to replace all the (Key,Value) pairs within the target application section that share the specified key name with the set of (Key,Value) pairs that were specified on the source system side. However, you can use the optional ADDVALUES command to tell the configurator tool to add those (Key,Value) pairs to the existing (Key,Value) pairs of the application section.

Only one TARGET command can be defined within a KEY command sub- substanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

VALUE Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the VALUE command to specify a value to be associated with the corresponding key on the target system.

Use VALUE as follows:

- To provide specific values if no SOURCE name is provided in the KEY sub-substanza block.
- To provide default values if a KEY SOURCE name is provided, but the search for a matching source-side key name fails.

- To specify that a (Key,Value) pair on the target system is to be deleted by specifying the VALUE command as a null string. (This can work as a default or as an explicit setting).

The (Key,Value) pair, if any, read from the source system and the replacement Key,Value) pair generated by the KEY sub-substanza are logged.

Only one VALUE command can be defined within a KEY command sub-substanza block. If neither a VALUE nor a SOURCE are provided within a KEY sub-substanza block, an error is returned.

The VALUE command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

ADDVALUES Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the ADDVALUES command to specify whether (Key,Value) pairs on the target system are to be added to or replaced. The default is replace.

The scope of the ADDVALUES command is the KEY sub-substanza it is contained in. Only one ADDVALUES can be coded in each KEY sub-substanza.

SETFIVAR Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the optional SETFIVAR command to specify the name of an FI variable. The variable is set to the value generated by the KEY command sub-substanza block. This lets you pass source system text file lines and strings to PROBE and FI user exits.

If there are multiple (Key,Value) pairs with the same key name, SETFIVAR returns the value field of the first (Key,Value) pair in the application section that shares the key name.

If the FI variable is already defined, the value is simply changed to any new value. If the FI variable does not yet exist, the variable is created and its value is set.

The SETFIVAR command parameter is an ordinary string enclosed in double quotation marks and may contain references to FI variables. Any FI variable references in the SETFIVAR command parameter string are resolved during PROBE execution. The FI variable name is specified without the pair of enclosing braces { }.

Only one SETFIVAR command can be specified within a KEY command sub-substanza block.

SETLASTFIVAR Command <- KEY Sub-Substanza <- APPL Substanza <- STANZAINI Stanza

Use the optional SETLASTFIVAR command to specify the name of an FI variable. The variable is set to the value generated by the KEY command sub-substanza block. This lets you pass source system text file lines and strings to PROBE and FI user exits.

If the source application section contains multiple (Key,Value) pairs that share the same key name, SETLASTFIVAR returns the value field of the last (Key,Value) pair in the application section that shares the key name.

If the source application section contains only one (Key,Value) pairs that has the specified key name, SETLASTFIVAR returns the value field of that (Key,Value) pair.

If the FI variable is already defined, the value is simply changed to any new value. If the FI variable does not yet exist, the variable is created and its value is set.

The SETLASTFIVAR command parameter is an ordinary string enclosed in double quotation marks and may contain references to FI variables. Any FI variable references in the SETLASTFIVAR command parameter string are resolved during PROBE execution. The FI variable name is specified without the pair of enclosing braces *{ }*.

Only one SETLASTFIVAR command can be specified within a KEY command sub-substanza block.

COPYAPPL Command <- STANZAINI Stanza

Use the COPYAPPL command to start a substanza defining a set of specifications defining how to copy an entire application section from the specified source system stanza-style INI file to the specified target system stanza-style INI file.

When a stanza-style INI file application section is copied, you can specify whether the source application will replace a target application section, or will be added to the target application section. The default is to replace the application section. This behavior matches the behavior of the `OS2INI KEY` command sub-substanza block.

There can be zero or more `COPYAPPL` commands in a `STANZAINI` command stanza block. End the `COPPYAPPL` substanza with the `ENDCOPYAPPL` command.

SOURCE Command <- COPYAPPL Substanza <- STANZAINI Stanza

Use the `SOURCE` command to specify the name of an application section within a stanza-style INI file on the source system. This is the application section that will be copied to the target system.

A `COPYAPPL` substanza block can contain only one `SOURCE` command.

The `SOURCE` command parameter is an ordinary string enclosed in double quotation marks. FI variables in the source command parameter string are resolved during `PROBE` execution. The resolved parameter string and the set of (Key,Value) pairs copied from the source system are logged if you specified extended logging.

TARGET Command <- COPYAPPL Substanza <- STANZAINI Stanza

Use the `TARGET` command to specify an application section within a stanza-style INI file on the target system. This is the application section to be replaced with or appended to by the set of (Key,Value) entries obtained from the source system.

If a `TARGET` command is not specified and a `SOURCE` command is specified within the `COPYAPPL` command substanza block, the target application name is assumed to be the same as the source application name. However, in this case, any FI variable references in the source application name are left unresolved so that they can be subsequently resolved on the target system during the installation of FI install object during the execution of the configurator tool.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

A COPYAPPL command substanza block can contain only one TARGET command.

ADDVALUES Command <- COPYAPPL Substanza <- STANZAINI Stanza

Use the optional ADDVALUES command to specify to the configurator utility to add, rather than replace, the application section copied from the source system. A replace operation deletes all the (Key,Value) pairs within the target application section. An add does not delete any (Key,Value) pairs within the target application section.

The default, if ADDVALUES is not specified, is replace.

The scope of ADDVALUES is to the COPYAPPL substanza it is in. Any subsequent COPYAPPL command substanzas are reset to their default and perform (Key,Value) replacement.

Only one ADDVALUES command can be coded in a COPYAPPL command substanza block.

1.3.9 File Copy Commands

Use the file copy commands to tell PROBE to copy one set of files or directories into the FI probe objects so that they can subsequently be copied onto the updated OS/2 target system.

You specify the two file search patterns that are used to select which files to copy. The first pattern specifies a set of filename patterns that are marked for inclusion in the copy set.

The second pattern specifies a set of filename patterns that are marked for exclusion from the copy set. The subtraction of the exclusion pattern from the inclusion pattern yields the pattern used to select files from the source directory to be copied over to the target system.

The filename patterns cannot include any directory path specifiers; use them to search only within a single directory.

Specify the COPYFILES commands as:

- A single directory file copy directive. Use this to copy files found within the specified source system directory. You cannot use this to copy directories found within the source system directory.

  If the target system directory path for the copies files contains directory names that do not exist on the target system, the directories are created on the target system.

- As a recursive subdirectory search file copy directive. Specify the SUBDIR command within the COPYFILES command stanza block to copy both files and directories from the source system to the target system.

  PROBE starts at the specified source system directory and recursively searches through that directory and all of its subdirectories looking for files that satisfy the specified file name search patterns.

  The selected files and all the subdirectories will be copied to the target system along with all their system attributes and Extended Attributes.

You can specify inclusion and exclusion patterns as either:

- Typical OS/2 filename patterns that can contain OS/2 wildcard filename search characters such as the asterisk (*).
- XPG4 Extended Regular Expressions. If you use XPG4 regular expressions, you must remember to escape the period with a preceding backslash (\.).

An example of File Copy use is provided. A simple example of the use of inclusion and exclusion patterns follows. Imagine the inclusion pattern is "*.BAT" and the exclusion pattern is "AUTOEXEC.*". The resulting patterns selects all files with an extension of ".BAT" except for "AUTOEXEC.BAT". In this example, both the inclusion and exclusion patterns were specified as simple OS/2 filename patterns.

You can use ISNOTFOUNDERROR to control error handling behavior in the following cases:

- If the specified source directory is not found on the source system, PROBE follows the option specified on ISNOTFOUNDERROR.
- If the resolution of INCULDEFILES and EXCLUDEFILES results in an empty list of files to copy and if no SUBDIR command is specified, ISNOTFOUNDERROR is executed. (If SUBDIR is specified, this case is never an error, and ISNOTFOUNDERROR does not force errors or warnings.)

PROBE file copy commands, COPYFILES, copy over the system attributes and any extended attributes attached to an OS/2 file. They handle both long and 8.3 file names.

PROBE copies all files specified for inclusion, regardless of system attributes such as "hidden" associated with the file. All the system attributes set for a file are set equivalently when the file is copied to the updated target system. The syntax of the COPYFILES command stanza is summarized below.

```
// Summary of the File Copy command stanza
// Indentations are for ease-of-reading, they are
// not required.

COPYFILES
   SOURCE "source system directory name"
   TARGET" target system directory name"
   SUBDIR
   INCLUDEFILES
      PATTERN "OS/2 filename inclusion pattern"
      REGEXPR  "XPG4 Extended Regular Expression filename inclusion
                          pattern"
   ENDINCLUDEFILES
   EXCLUDEFILES
      PATTERN "OS/2 filename exclusion pattern"
      REGEXPR "XPG4 Extended Regular Expression filename exclusion
                        pattern"
   ENDEXCLUDEFILES
ENDCOPYFILES
```

Figure 9. COPYFILE Stanza

1.3.9.1 COPYFILES File Copy Command Stanza

Use COPYFILES to mark the beginning of a stanza block defining a set of files to be copied from the source system to the target system. The valid commands within a COPYFILES command stanza block can be in any order, but the block must end with ENDCOPYFILES.

SOURCE Command <- COPYFILES Stanza

The required SOURCE command specifies the directory on the source system from which files are to be copied.

Only one SOURCE command can be defined within a COPYFILES command block.

The SOURCE command parameter is an ordinary string enclosed in double quotation marks. FI variables in the SOURCE parameter string are resolved during PROBE execution. The resolved parameter string is logged if you specified extended logging.

TARGET Command <- COPYFILES Stanza

Use the optional TARGET command to specify the directory on the target system into which the copied files are to be placed.

If you do not provide a TARGET command, the directory specified in the SOURCE command in the stanza block is used for the value of the TARGET command. However, in this case, any FI variable references in the source directory pathname are left unresolved so that they can be subsequently resolved on the target system during the installation of the FI install object during execution of the configurator tool.

Only one TARGET command can be defined within a COPYFILES command stanza block.

The TARGET command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables which are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

SUBDIR Command <- COPYFILES Stanza

Use the optional SUBDIR command to specify that the COPYFILES command stanza block should recursively search the specified source directory and also all its subdirectories. Separate file name pattern searches are done within each directory. In this case, all the sub-directories with their system attributes and Extended Attributes, even if they are empty, are copied to the target system. Only the files that are selected based on the file name patters are copied to the target system, within their same directory.

Only one SUBDIR command can be defined within a COPYFILES command stanza block.

INCLUDEFILES Command <- COPYFILES Stanza

Use the optional INCLUDEFILES command to mark the beginning of a substanza block listing one or more filename patterns that together are the inclusion pattern for the COPYFILES command stanza block. There must be at least one PATTERN or REGEXPR command defined within each INCLUDEFILES command substanza.

If no INCLUDEFILES command is specified, then the inclusion pattern is assumed to match any files in the specified directory and, optionally, in its subdirectories.

The set of PATTERN and REGEXPR command filename patterns is applied against the set of file names found in the specified source directory. The result of the operation is an inclusion list of filenames in the directory that match at least one specified filename pattern in the list of filename patterns.

If the SUBDIR command was specified in the COPYFILES command stanza block, then an inclusion list is also constructed for all the subdirectories under the specified source directory.

Only one INCLUDEFILES command can be defined within a COPYFILES command stanza block.

PATTERN Command <- INCLUDEFILES Substanza <- COPYFILES Stanza

Use the PATTERN command to specify a single filename inclusion pattern.

The command parameter is a typical OS/2 filename pattern string enclosed in double quotation marks. The filename can include both * and ? characters.

No FI variable references are allowed in the PATTERN command parameter string.

REGEXPR Command <- INCLUDEFILES Substanza <- COPYFILES Stanza

Use the REGEXPR command to specify a single filename inclusion pattern.

The command parameter is a XPG4 Extended Regular Expression string enclosed in double quotation marks.

No FI variable references are allowed in the REGEXPR command parameter string.

EXCLUDEFILES Command <- COPYFILES Stanza

Use the EXCLUDEFILES command to mark the beginning of a substanza block listing one or more filename patterns forming the exclusion pattern for the COPYFILES command stanza block. There must be at least one PATTERN or REGEXPR command defined within each INCLUDEFILES command substanza.

The set of PATTERN and REGEXPR command filename patterns are applied against the set of file names found in the specified source directory. The result of the operation is an exclusion list of filenames in the directory that match at least one specified filename pattern in the list of filename patterns. If no EXCLUDEFILES command is included within the COPYFILES command stanza, the exclusion pattern is assumed to be empty.

Only one EXCLUDEFILES command can be defined within a COPYFILES command stanza block.

If the SUBDIR command was specified in the COPYFILES command stanza block, then an exclusion list is also constructed for all the subdirectories under the specified source directory.

The final list of files to be copied from the source system to the target system includes filenames included within an inclusion list, but not included within its corresponding exclusion list.

PATTERN Command <- EXCLUDEFILES Substanza <- COPYFILES Stanza

Use the PATTERN command to specify a single filename exclusion pattern.

The command parameter is a typical OS/2 filename pattern string enclosed in double quotation marks. The filename can include both * and ? characters.

No FI variable references are allowed in the PATTERN command parameter string.

REGEXPR Command <- EXCLUDEFILES Substanza <- COPYFILE Stanza

Use the REGEXPR command to specify a single filename inclusion pattern.

The command parameter is a XPG4 Extended Regular Expression string enclosed in double quotation marks.

No FI variable references are allowed in the PATTERN command parameter string.

1.3.9.2 Example of a File Copy Command Stanza

The following is an example of copying all the application Windows-style INI files from the Windows \SYSTEM directory. It has a list of system INI files that the OS/2 installation put there. Assume, in this example, that the exclusion list consists of WIN.INI and SYSTEM.INI. The COPY command substanza here copies all the other Windows-style INI files. You could use Windows INI file configuration commands to separately pull key definitions from the WIN.INI and SYSTEM.INI files on the source system.

```
// An example of a File Copy command stanza
// The identations are for readibility, they are not required.

COPYFILES
  SOURCE "{BootDrive}\OS2\MDOS\WINOS2\SYSTEM"
    INCLUDEFILES
      PATTERN "*.INI"
    ENDINCLUDEFILES
      EXCLUDEFILES
        PATTERN "SYSTEM.INI"
        PATTERN "WIN.INI"
      ENDEXCLUDEFILES
ENDCOPYFILES
```

Figure 10. Example of a File Copy Command Stanza

1.3.10 PROBE User Exit Definition Commands

Use the PROBEUSEREXIT command to define user exits called from PROBE when the command is executed. The user exits are .CMD command files that are executable programs. PROBE starts a user exit by starting an instance of CMD.EXE and passing in the name of the user exit to the CMD.EXE command processor.

Each user exit has a PROBE command stream file associated with it. You must specify the pathname of the command stream file in the command line defining the user exit.

The command stream pathname is passed to the user exit as its first parameter. The user exit typically interacts with PROBE by constructing PROBE commands within the file. If the user exit terminates successfully, by returning a successful return code, PROBE begins to process the commands within the file. This temporarily interrupts the command stream that PROBE was processing when the user exit was encountered and launched.

PROBE cannot know what events transpire within a user exit, and, therefore, cannot log any of them. There are no PROBE functions the user exit can call to pass errors and messages back to PROBE before the user exit terminates. However, your user exit can return specific messages and errors back to PROBE if you place ERROR, WARNING, and LOG commands within the PROBE stream file whose name the user exit was passed.

The syntax of the User Exit Definition command stanza is summarized below.

```
// Summary of the User Exit Definition command stanza
// Indentations are for ease-of-reading, they are
// not required.
PROBEUSEREXIT
   PATH "user exit file pathname"
   PRB "PROBE command stream file pathname.prb"
   PARMS
      PARM "optional parameter 1"
      ...
      PARM "optional parameter n"
   ENDPARMS
ENDPROBEUSEREXIT
```

Figure 11. PROBEUSEREXIT Stanza

1.3.10.1 PROBEUSEREXIT

Use the PROBEUSEREXIT command to mark the beginning of a stanza block defining a user exit to be executed immediately by PROBE.

You can use the PATH, PRB, PARMS, commands within the block started with PROBEUSEREXIT and ended with ENDPROBEUSEREXIT.

PATH Command <- USEREXIT Stanza

Use the required PATH command to specify the file path name to the user exit routine on the source system. If specified as a relative path, the path is considered relative to the working directory for PROBE. You must make sure the user exit is installed on the source system in the specified directory. It is easiest if you install the user exit executable file in the same directory as PROBE. Then have the file path name include the standard FI variable

{VTVToolsDir} that points to the directory containing PROBE.

Only one PATH statement can be included in a PROBEUSEREXIT command stanza block.

The PATH command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PATH command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

PRB Command <- USEREXIT Stanza

Use the required PRB command to specify the name of the PROBE command stream file that the user exit will generate. If the specified file does not exist, PROBE does not attempt to create it. Your user exit routine must handle the creation and writing of the PROBE command stream file.

If specified as a relative path, the path is considered relative to the working directory for PROBE.

Only one PRB statement can be included in a PROBEUSEREXIT command stanza block.

The PRB command parameter is an ordinary string specifying the directory pathname to a PROBE command stream file on the source system. It is enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PRB command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

PARMS Command <- USEREXIT Stanza

Use the optional PARMS command to mark the beginning of a substanza block used to define an optional parameter list to be passed to the user exit. The parameter strings are coded as PARM commands. There must be at least one PARM command within the PARMS substanza block. The parameters are passed to the user exit in the order they are coded with PARM commands.

Only one PARMS statement can be included in a PROBEUSEREXIT command stanza block.

PARM Command <- PARMS Substanza <- USEREXIT Stanza

Use the PARM command to specify an optional parameter to be passed to the user exit specified in the PROBEUSEREXIT command stanza block.

The PARM command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PRB command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

You can use the ISNOTFOUNDERROR to control some interpretation of search failures. The ISNOTFOUNDERROR must precede the command stanza block where you want to control search failure actions.

1.3.11 Feature Install User Exit Definition Commands

You can use Feature Install User Exit Definition commands to define user exits to run on the updated version of the target system when the configurator installs the FI PROBE object on the target system. These commands make the target end of the PROBE more flexible.

You can specify the types of user exits, DLL, EXE, or CMD, that are supported by the FI run-time environment. PROBE also supports the same configuration options as FI for the types of user exits.

The FI user exit executable files are deleted when the configurator step completes.

The following is a high-level view of the FI User Exit Definition command stanzas.

```
// Summary of the FIDLLUSEREXIT command stanza
// Indentations are for ease-of-reading, they are
// not required.
FIDLLUSEREXIT
   PATH "DLL file pathname"
   PROC procedure name within DLL
   STACK stack size
   INVOKE When to invoke the user exit during FI installation
ENDFIDLLUSEREXIT
```

Figure 12. FIDLLUSEREXIT Stanza

```
// Summary of the FIDLLUSEREXIT command stanza
// Indentations are for ease-of-reading, they are
// not required.
FIUSEREXIT
   TYPE CMD | EXE
   PATH "CMD or EXE file pathname"
   INVOKE When to invoke the user exit during FI installation
   PARMS
      PARM "optional parameter 1"
      ...
      PARM "optional parameter n"
   ENDPARMS
ENDFIUSEREXIT
```

Figure 13. FIUSEREXIT Stanza

FIDLLUSEREXIT Stanza

Use the `FIDLLUSEREXIT` command to mark the beginning of a stanza block defining a DLL-based FI user exit that is ultimately executed on the target system by the FI run-time. Within the `FIDLLUSEREXIT` command stanza block, the following commands can be placed in ary order, but the block must end with `ENDFIDLLUSEREXIT`

PATH Command <- FIDLLUSEREXIT Stanza

Use the required `PATH` command to specify a file path name to the FI user exit DLL module on the target system. If it is a relative path, the path is considered relative to configurator tool's working directory.

You must make sure that the FI user exit is installed on the target system in the specified directory. It is easiest if you install the FI user exit executable file in the same directory as the configurator tool. Then have the file path name include the standard FI variable *{VTVToolsDir}* that points to the directory containing the configurator tool.

Or, you can use a `COPYFILES` command stanza to include an FI user exit within a `PROBE` bundle file package. In this case, the file path name should include the standard FI variable *{VTVFI_WorkSrcDir}* that points to the root of the temporary file tree passed to the FI run-times on the target system.

The PATH command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PATH command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

Only one PATH statement can be included in a FIDLLUSEREXIT command stanza block.

PROC Command <- FIDLLUSEREXIT Stanza

Use the required PROC command to specify the name of the FI user exit procedure within the DLL routine.

The PROC command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PROC command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

Only one PROC statement can be included in a FIDLLUSEREXIT command stanza block.

STACK Command <- FIDLLUSEREXIT Stanza

Use the optional STACK command to specify the size of the stack that the FI run-times use to call the DLL user exit. If not specified, the default is 8 KB.

The PROC command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PROC command parameter string are resolved on the target system during the installation of the FI install object during execution of the configurator tool.

Only one STACK statement can be included in a FIDLLUSEREXIT command stanza block.

INVOKE Command <- FIDLLUSEREXIT Stanza

Use the required INVOKE command to specify when, during the installation of the PROBE FI install object, the FI user exit should be invoked by the FI run-times.

The `INVOKE` command parameter must be one of the following keywords not enclosed in double quotation marks:

Table 18. INVOKE Command Parameter (FIDLLUSEREXUT Stanza)

`BEGIN`	Before the install process begins
`END`	Just after the installation is completed
`PRE_FILE`	Just before transferring files during installation
`POST_FILE`	Just after transferring files during installation
`PRE_CONFIG`	Just before updating the Text, OS/2 INI, or stanza-based INI files.
`POST_CONFIG`	Just after updating the Text, OS/2 INI, or stanza-based INI files

Only one `INVOKE` statement can be included in a `FIDLLUSEREXIT` command stanza block.

FIUSEREXIT Stanza

Use the `FIUSEREXIT` command to mark the beginning of a stanza block defining either a CMD-based or an EXE-based FI user exit. The user exit is ultimately executed on the target system by the FI run-times.

TYPE Command <- FIUSEREXIT Stanza

Use the required TYPE command to specify the type of FI user exit. TYPE is a keyword parameter whose file extension must be either CMD or EXE.

Only one TYPE statement can be included in a FIUSEREXIT command stanza block.

PATH Command <- FIUSEREXIT Stanza

Use the required PATH command to specify a file path name to the FI user exit CMD or EXE module on the source system. If it is a relative path, the path is considered relative to the configurator tool's working directory.

You must make sure that the FI user exit is installed on the target system in the specified directory. The FI user exit executable file can be installed in the same directory as the configurator tool. Then have the file path name include the standard FI variable *{VTVToolsDir}* that points to the directory containing the configurator tool.

Or, you can use a COPYFILES command stanza to include an FI user exit within a PROBE bundle file package. In this case, the file path name should include the standard FI variable *{VTVFI_WorkSrcDir}* that points to the root of the temporary file tree passed to the FI run-times on the target system.

The PATH command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the PATH command parameter string are resolved during PROBE execution. The resolved parameter string is logged if extended logging is specified.

Only one PATH statement can be included in a FIUSEREXIT command stanza block.

INVOKE Command <- FIUSEREXIT Stanza

Use the required INVOKE command to specify when, during the installation of the PROBE FI install object, the FI user exit should be invoked by the FI run-times.

The `INVOKE` command parameter must be one of the following keywords not enclosed in double quotation marks:

Table 19. INVOKE Command Parameter (FIUSEREXIT Stanza)

`BEGIN`	Before the install process begins
`END`	Just after the installation is completed
`PRE_FILE`	Just before transferring files during installation
`POST_FILE`	Just after transferring files during installation
`PRE_CONFIG`	Just before updating the Text, OS/2 INI, or stanza-based INI files.
`POST_CONFIG`	Just after updating the Text, OS/2 INI, or stanza-based INI files

Only one `INVOKE` statement can be included in a `FIUSEREXIT` command stanza block.

PARMS Command <- FIUSEREXIT Stanza

Use the optional `PARMS` command to mark the beginning of a substanza block used to define an optional parameter list to be passed to the CMD or EXE FI user exit. The parameter strings are coded as `PARM` commands. There must be at least one `PARM` command within the `PARMS` substanza block. The parameters are passed to the user exit in the order they are coded with `PARM` commands.

PARM Command <- PARMS Substanza <- FIUSEREXIT Stanza

Use the `PARM` command to specify an optional parameter to be passed to the FI user exit specified in the `FIUSEREXIT` command stanza block.

The `PARM` command parameter is an ordinary string enclosed in double quotation marks. It can contain references to FI variables. Any FI variable references within the `PARM` command parameter string are resolved on the target system during installation of FI install object during the execution of the configurator tool.

1.3.12 Feature Install Variable Definition Commands

The FI variable definition commands define FI variables and set their default variables (in the sense that FI Toolkit considers the setting of default variables). If a PROBE command stream command encounters an FI variable definition command for a previously defined FI variable, the variable is reset to the new value and, optionally, the description of the variable. It also updates the FI variable description if a new description is specified in the FIVAR command stanza block.

The following is a high-level of the FI Variable Definition command stanzas.

```
// Summary of the FI Variable Definition Command Stanza
// Indendations are for ease-of-reading, they are not required.

FIVAR
   NAME "variable name"
   VALUE "value string"
ENDFIVAR
```

Figure 14. FIVAR Stanza

1.3.12.1 FIVAR FI Variable Definition Stanza

Use the FIVAR command to mark the beginning of a stanza block defining an FI variable. End the FIVAR command stanza block with the ENDFIVAR command.

NAME <- FIVAR Stanza

Use the required NAME command to specify the name of an FI variable.

Only one NAME statement can be included in a FIVAR command stanza block. The NAME command parameter is an ordinary string enclosed in double quotation marks. It may contain references to FI variables. Any FI variable reference in the NAME command parameter string are resolved during PROBE execution. Make sure that any FI variable references in NAME are already defined; references to undefined FI variable can cause problems.

VALUE Command <- FIVAR Stanza

Use the required VALUE command to specify the value of an FI variable. Only one VALUE statement can be included in a FIVAR command stanza block. It can contain references to FI variables. Any FI variable references within the VALUE command parameter string are resolved during PROBE execution.

The resolved parameter string is logged if extended logging is specified.

1.3.12.2 Standard FI Variables

The following FI variables are automatically defined and maintained as illustrated in Table 20.

Table 20. Standard FI Variables

{BootDrive}	During PROBE execution, this is the boot drive of the old target system. When the configurator is run on the target system, this resolves to the boot drive of that system.
{ClientID}	Supports the development of user exits that make decisions based on the ID of the system. For example, you can set up custom PROBE scripts for each target system. This variable could be passed as an optional parameter to a PROBE user exit to help it decide which PROBE command streams to include.
{VTVFI_WorkSrcDir}	The configurator tool sets this FI variable to the path string pointing to the root of the temporary file tree passed to the FI run-times. The temporary file tree contains all the files, specified in COPYFILES command stanzas, copied over from the source system.
{VTVTargetEnvVar}	The configurator tool sets this FI variable to the path string in the VTVTARGET environment variable passed in to the configurator tool.
{VTVTargetSysWork}	The configurator tool sets this FI variable to the path of the temporary working directory on the target system. This is the path specified with the `/W:` parameter of the configurator tool.
{VTVToolsDir}	Both the configurator tool and PROBE set this variable to the path of the directory in which the tools are located. This variable lets you install PROBE and FI user exit executable files easily. You can place a user exit in the same directory as the tool that calls it.

1.4 The Configurator Tool

The configurator runs on the updated target system to accept the bundle files created by the PROBE utility. It converts the files in the bundle files to an FI install object and then installs the FI object.

The configurator runs on the updated target system to accept the bundle files created by the PROBE utility. It converts the files in the bundle files to an FI install object and then installs the FI install object. Read about the sequence of using the migration tools before you use the replicator.

1.4.1 Configurator Syntax

The CONFIGUR tool supports the following parameters:

CONFIGUR Syntax

```
<drive:\path> CONFIGUR
          /E:<client_ID>
          /B:<intermediate_bundle_file_pathname>
          /V:<{FI_variable_name}=FI_variable_value>
          /W:<temporary_work_directory_pathname>
          /L:<log_file>
          /X:<logging_level>
          <command_stream-file_1>.PRB, <command_stream_file_2>.PRB
          /?
          ?
```

where:

/? or ?	Display a summary of configurator command line syntax. Any other parameter is ignored.
/E:*<client_ID>*	The name (ID) of the system on which the probe is taken. The name is used as a stem for the intermediate bundle files PROBE creates. Files are named *client_ID*.NNN. The numbers NNN start at 000 for the first bundle and increment sequentially. The *<client_ID>* must be ASCII-7 characters and contain no blanks. It the intermediate file bundles are to be stored on a FAT partition, the name cannot be greater than eight characters.

/B:*<intermediate_bundle_file_path>*

The directory holding the intermediate bundle files produced by PROBE.

The bundle files contain the pieces the configurator uses to create an FI install object representing the directives created by PROBE.

/V:*<{FI variable name}= FI_variable_value>*

Optional. Sets the value of the specified variables within the FI install object created by the configurator tool. The tool does not create the variables; they must already be created within the intermediate PROBE bundle files referred to on the configurator command line.

You can provide any number of these FI variable specifications on the command line.

For example:

CONFIGUR /V:*{WORKING_DIR}*=D:\TEMP\FOO /V:*{HOME_DRIVE}*=W:

Do not code any spaces in the parameter specification. If you want spaces in the FI variable value, enclose the entire parameter in double quotation marks. You cannot include an imbedded quotation mark in the FI variable value.

CONFIGUR "/V:*{WORKING_DIR}*=D:\TEMP\Inventory Control"

/W:*<temporary_work_directory_path>*

A directory to serve as the root of the temporary work area. Specify a directory with enough space to contain twice the full set of uncompressed files within the set of intermediate bundle files created by PROBE.

The PROBE control file, *client_id*.CTL, lists the size

	of the full set of uncompressed files. The configurator creates a subdirectory, in this directory, named *client_ID* as the root of the of the temporary work area. It is an error if this subdirectory already exists. On successful completion, this space is cleaned up. On error termination, any files created are left for error analysis. If you tell `PROBE` to create output files, but omit this parameter, the configurator looks for a temporary directory pathname in the `TEMP` environment variable. In that case, an error is returned if the configurator cannot find a valid directory pathname in the `TEMP` environment variable.
`/L:`*<log_file_directory>*	Pathname of the directory to contain the log file, *client_ID*.LOG, created during processing. If the log file does not exist, the configurator creates it. If it does exist, new log entries are appended to it. If `/L` is not specified, any setting of `/X` is ignored. When `/L` is specified, the configurator creates the following files:

<client_ID>.LOG	The configurator log file.
<client_ID>.PHL	Created during the FI package creation step, this is an FI history log.
<client_ID>.PEL	Created during the FI package creation step, this is an FI error log.
<client_ID>.IHL	Created during the FI installation step, this is an FI history log.
<client_ID>.IEL	Created during the FI installation step, this is an FI error log.
<client_ID>.ZLG	The `UNZIP` utility log file.

If the .LOG file does not exist, the configurator creates it. If it exists, the configurator appends to it. The unzip log file and the FI log files are always overwritten.

The .ZLG file entries created when you use replicator can tell you, if a power failure or other incident occurs during file transmission, the file name of the last successfully transmitted bundle file. The *<client_ID>*.ZLG file might also contain information about zip or unzip processing. See `ZIP` and `UNZIP` messages and errors.

`/X:`*<logging_level>* Specifies the level of logging:

1 Log error messages only
2 Log error and warning messages
3 Log error, warning, and all LOG messages created by user directives. This is the default if `/X` is not specified.
4 Log error, warning, LOG command messages, and all configurator informational messages

Errors and warnings are always written to the standard output stream. If logging is enabled, they are also written to the log file.

1.4.2 Configurator Return Codes

The configurator utility can return one of the following codes. The codes are consistent with *CID Enablement Guidelines*, S10H-9666.

0 Successful program termination; no reboot required.

4 Successful program termination; warning messages logged; no reboot required.

5 Unsuccessful program termination; no reboot required. Error messages logged.

6 Unsuccessful program termination; no reboot required. Unsuccessful program termination with unexpected internal errors detected and logged.

1.4.3 Configurator Operation

The configurator does the following actions. (You must have already created the files to be used in this sequence with the `PROBE`).

1. Reads the stored CRC values from the *client_ID*.CTL file in the intermediate `PROBE` bundle file and compares the CRC values against all

the file bundles in the package. If any files fail the CRC check, the CRC failures are logged and processing stops.

2. Checks whether there appears to be enough space on the partition to hold the temporary work area to hold the expanded file bundles and the FI install object that is to be created. If not, the failure is logged and processing stops.
3. Unbundles all the intermediate PROBE package file bundles in the temporary work area.
4. Calls the CLIFI utility, the command line interface to Feature Install, to create an FI install object that represents the PROBE object.
5. Deletes the intermediate PROBE files that are now successfully copied into the FI install object.
6. Calls CLIFI again to install the newly created FI install object. This step also does the following:
   - Invokes all the FI user exits defined within the PROBE package.
   - Runs the TARGET.CMD user exit supplied with the configurator. TARGET reads the file whose pathname you supplied in the VTVTARGET environment variable. This is a simple way to update FI variables at the beginning of the FI installation step.
7. Deletes the files that were copied into the FI install object and deletes all the temporary directories created in the work area.

1.5 The Desktop Shuffler Tool

The Desktop Shuffler, CLNDESK, lets you move, delete, or add objects to an OS/2 Warp 4 desktop in attended and unattended mode. You can issue the CLNDESK command at the actual desktop to be modified, or you can issue the command at one central donor system as part of building a specific desktop to be downloaded to other systems. No programming or REXX utilities are required.

Read about the sequence of using the migration tools before you use the shuffler.

The CLNDESK command reads the list of actions from a control file and from a keywords file. You can create, copy, move, shadow, delete, and modify objects.

1.5.1 CLNDESK Syntax

The syntax of the `CLNDESK` command is as follows:

CLNDESK Syntax

<x:\path\>`CLNDESK` *<control_file> <keywords_file>*

where:

<control_file>	Required. Contains actions to take on the specified objects.
<keywords_file>	Required. Contains blank-delimited keywords of the form: products=`keyword1` [`keyword2 keyword2 keyword2`]

Example:

```
C:\IBMINST\CLNDESK shuffle_ctl.lst shuffle_key.lst
```

1.5.2 Return Codes

The Desktop Shuffler utility can return one of the following codes. The codes are consistent with these in the *CID Enablement Guidelines*, S10H-9666.

0	Successful program termination, no reboot required.
4	Successful program termination, warning messages logged; no reboot required.
5	Unsuccessful program termination, no reboot required. Error messages logged.
6	Unsuccessful program termination, no reboot required. Unsuccessful program termination with unexpected internal errors detected and logged.

1.5.3 Control File Syntax

You can use the Desktop Shuffler with the supplied sample control file to set the desktop objects for an OS/2 Warp 4 system to the same arrangement as an attended-installed system, or you can read the rules and examples below to create your own file.

Follow the general rules below when creating or editing a control file. Each section has any additional rules or options that apply to that section.

The control file is an ASCII file in typical .INI format. All the file sections are required, but each section can be empty.

Lines beginning with a semicolon (;) are comment lines. Blank lines are allowed. Each item on a line can be separated by zero or more spaces. Actions fail if objects referred to (except for create) do not exist. Section titles, for example `[create]`, are not case sensitive.

Sections in the control file are executed in the following order:

1. `[System]`
2. `[Create]`
3. `[Delete]`
4. `[Update]`
5. `[Move]`
6. `[Shadow]`
7. `[Copy]`
8. `[Update]`

A created object is created in the hidden folder, updated, and then moved or shadowed to its appropriate locations.

Use `#if condition`, `#endif condition`, and `#else` condition directives in the keywords file to control execution.

1.5.3.1 Error Reporting from the Control File

If a syntax error is detected in the control file, an error message is issued to standard error.

1.5.3.2 Sample Control File

This annotated sample, as shown in Figure 15 on page 104, is a typical control file. It may or may not work; it is just for illustration.

```
; Shuffler Sample Control File
;
; A line starting with a semicolon is a comment line.
; This control file sets the desktop objects for a
; OS/2 Warp 4 system to the same arrangements as an attended
; installed OS/2 Warp 4 system.
;
; Note: The contents of the keyword files to be used
; with this example are:

Products=WARP OS2PEER TCPIP MPTS LDR
;
```

Figure 15. Shuffler Sample Control File

1.5.3.3 Section Syntax Information

The following options exist for control file keywords:

`[System]` This section can be either empty or all comment lines.

`[Create]` *Object Class*, *<ObjectID>*, *"Title"*, O*ption*, *<Location>*

There can be 0 or more spaces before or after each item.

Object Class must be valid Workplace Shell class.

<ObjectID> must be a valid Workplace Shell *Object ID* name.

You must specify the brackets, for instance < and >.

"Title" is a string surrounded by double quotation marks. Use a carat (^) to insert a new line into the object title displayed to the user.

Option can be:

`FAIL` The object is not created if another object with the same *object_id* exists. `FAIL` must be uppercase.

`REPLACE` The new object replaces an existing object with the same *object_id*. `REPLACE` must be uppercase.

`UPDATE` The new object is not created if another object with the same *object_id* exists. The corresponding attributes are set to those of the new object. `UPDATE` must be uppercase.

One or more WPClass setup-strings (keyname/value pairs) can follow the option. See the *Workplace Shell Programming Reference* for additional information.
Location can be the *object_id* of any folder or the desktop *<WP_DESKTOP>*.

Sample create section:

```
[create]
; this will create a folder object
WPFolder, <MY_Folder>, "This is a title", REPLACE
; this will not create a folder object
WPFolder, <MY_Folder>, "This is a title", FAIL
; this will update the object with a new title
WPFolder, <MY_Folder>, "This is an updated title", UPDATE
```

`[Delete]` ; The next two lines each delete one object

```
<CPPVISBLD>
<CPPIPMD>
```

`[Move]` *<ObjectID>*, *<Destination ObjectID>*

An error is generated if both objects do not exist.

The object is not moved if it already exists in the destination object.

Example:

```
[move]
;The next line moves <CPPVISBLD> to the desktop
<CPPVISBLD> <WP_DESKTOP>
```

`[Shadow]` *<ObjectID>*, <Destination ObjectID>

An error is generated if both objects do not exist.

The object is not shadowed if it already exists in the destination object.

The resulting object’s ID is the original ID with "_SHADOW" appended to it.

When the original object is deleted, all its shadows are removed simultaneously.

If an object ID of the name `<object_id_SHADOW>` already exists in the destination folder, a new shadow is not created.

Example:

```
[shadow]
;The next line shadows <CPPVISBLD> from its current
; location to the desktop
; The shadow's Object ID will be <CPPVISBLD_SHADOW>
<CPPVISBLD> <WP_DESKTOP>
```

`[Copy]` *<ObjectID>*, *<Destination ObjectID>*

An error is generated if both objects do not exist.

The object is not copied if it already exists in the destination object

The resulting object's ID is the original ID with "`_COPY`" appended to it.

When the original object is deleted, all its copies are unaffected.

If an object ID of the name `<object_id_COPY>` already exists in the destination folder, a new copy is not created.

Example:

```
[copy]
; The next line copies <CPPVISBLD> from its current
; location to the desktop
; The copy's Object ID will be <CPPVISBLD_COPY>
<CPPVISBLD> <WP_DESKTOP>
```

`[Update]` 0 or more lines with one of the following formats:

<ObjectID>, `keyword = value`
`FOLDER`*<ObjectID>*, `keyword = value` ; `keyword = value`
`SETUPSTRING<ObjectID>`, *setupString*

Separate multiple `keyword = value` pairs with semicolons (;).

See the *Workplace Shell Programming Reference* for information about *setupString*.

For TITLE=value pairs, do not use quotation marks around the title string.

Example:

```
[Update]
SETUPSTRING <WP_INFO>, TITLE=Utilities; DEFAULTVIEW=TREE
; connections folder
FOLDER <WP_CONNECTIONS> DEFAULTVIEW = TREE
FOLDER <WP_CONNECTIONS> TREEVIEW = MINI,LINES
; rename Productivity Folder to Utilities
<WP_TOOLS> TITLE = Utilities
; set information folder to treeview
<WP_INFO> DEFAULTVIEW = TREE
;
; end of sample control file
```

Keywords for Use in Control File [Create] and [Update] Options

Supported keywords for non-FOLDER entries correspond to the following subset of `wpSetup` override key/values supported by the WPObject class. See the Workplace Object Classes section, WPObject subsection of the *Workplace Shell Programming Reference* for additional information, or see a table of the keyname=value pairs.

```
CCVIEW=DEFAULT | YES | NO
DEFAULTVIEW = SETTINGS | DEFAULT | 0-9
ICONFILE = iconfile.ICO
ICONPOS = x,y  //values between 0 and 100
MINWIN = HIDE | VIEWER | DESKTOP
NOCOPY = YES | NO
NODELETE = YES | NO
NODRAG = YES | NO
NODROP = YES | NO
NOLINK = YES | NO
NOMOVE = YES | NO
NOPRINT = YES | NO
NORENAME = YES | NO
NOSETTINGS = YES | NO
NOSHADOW = YES | NO
NOTVISIBLE = YES | NO
OPEN = SETTINGS DEFAULT
```

```
TEMPLATE = YES | NO
TITLE = Title_string
```

There is a lot of helpful information in the online information for the OS/2 Toolkit in the \TOOLKIT\TOOLKIT\BOOKS directory.

1.5.4 Errors and Error Recovery

The Desktop Shuffler utility ignores most lines in error. If the utility cannot interpret a line or if the line is missing a comma, the action is not done. The following lines produce no action:

Example 1:

```
[create]
WPFolder <R_FOLDER> "Z folder" FAIL, <Other_APP>
```

If the section titles are missing or out of order, the clndesk.log file contains the entry:

```
CCL0011: key not contained
```

Example 2:

```
C:\IBMINST\CLNDESK.EXE
```

If the keywords file is missing, an error is displayed in a dialog box, and the `CLNDESK.EXE` file is not created.

1.5.5 Limitations and Restrictions

The shuffler utility cannot perform actions on the Warp Center desktop object. For instance, no folder or program objects can be copied to the Warp Center. Also, no WPS objects already in the Warp Center can be updated by the shuffler utility. For example, the following shuffler control file line produces no action:

```
[copy]
<WP_EPM>, <WP_WARPCENTER>
```

1.6 The Trimmer Tool

The trimmer utility lets you remove functions from an OS/2 Warp 4 system in unattended mode. The trimmer removes the files, .INI entries, and Workplace Shell objects associated with the component that is trimmed. It also updates ASCII configuration files such as CONFIG.SYS, and Windows .INI files.

Read about the sequence of using the migration tools before you use the trimmer.

You can issue the command at the actual machine to be modified. Or, you can issue the command at one central donor system as part of building a specific desktop to be downloaded to other systems. If you issue the command from a central machine, the trimmer trims the system which is currently booted on that client in case it has several boot partitions or is booted from a diskette. (The latter case results in no trimming action.)

The trimmer utility reads the list of actions from the W4TRIM.CTL file supplied with the utility. No user input is required.

1.6.1 Installing the Trimmer Utility

Perform the following steps to install the trimmer utility:

1.6.1.1 Installing the Correct Feature Install Version

You must have Feature Install Version 1.2.1 or later. Version 1.2 of level 970826.1 fixes many hang and trap problems encountered in FI.

1. Copy the FIRUNPKG.ZIP file to a directory. It automatically looks for one called FISETUP on the boot drive, but you can call it anything.
2. Unzip it.
3. Run the command `FISETUP /NN` to set up FI.
4. Reboot.

1.6.1.2 Installing Trimmer

The trimmer utility is only for OS/2 Warp 4 and not for any other version.

1. The trimmer utility is available on the build share: \\AUSPBS09\VERCURR or on the intranet download page:
   `http://finstall.austin.ibm.com/testv2v.htm`

   You must have been given access to this share.
2. Files needed from the build distribution:

   W4TRIM.CMD
   W4TRIM.CTL
   W4TRIM.MSG
   TRIMIT.CMD
   TRIMEXIT.EXE
   TRIMMER.RSP
   DIS386.DLL

EXCEPTQ.DLL
FISETUP.EXE

3. Create the directory \trimmer from the root directory. You can call this directory another name, if you prefer.
4. Copy the listed files into this directory.
5. Copy the files in DELFILES into the \TRIMMER\DELFILES subdirectory. This directory must be named \DELFILES.
6. Create a subdirectory \MEDIA under \TRIMMER. Leave \TRIMMER\MEDIA empty.
7. Issue the following FI command from the trimmer directory (or from the directory where you stored the trimmer files:

   ```
   CLIFI /A:B /R:TRIMMER.RSP /F:MEDIA
   ```

The trimmer utility is installed.

1.6.1.3 Verifying Trimmer Installation

After the trimmer utility is installed, verify the installation by running the utility without actually trimming any files.

1. Issue the command:

   ```
   W4TRIM
   ```

2. When it completes, make sure the log file *<BootDrive:>*\OS2\INSTALL\W4TRIM.LOG was updated or created with the information about the most recent trimming. The date and time indicated must be current, and the GrandTotalSize must be 0.

Now you can modify the W4TRIM.CTL file to select actual components for trimming and logging.

1.6.2 Trimmer Utility Syntax

The `W4TRIM` command supports the following parameters:

W4TRIM Syntax

<drive:\path> `W4TRIM /T /C:W4TRIM.CTL`

where:

`/T`	Trim the files. If `/T` is not provided, the utility only writes a log file, \OS2\INSTALL\W4TRIM.LOG, listing the files that would have been deleted and warnings for any locked files.

`Press Enter to continue, Ctrl-Break to terminate`, is displayed.

You can set up a CID command file to run the trimmer unattended by coding:

```
W4TRIM W4TRIM.CTL /T /INP.TXT
```

where the content of the file INP.TXT is the hexadecimal character 0x0A.

`/C:W4TRIM.CTL` The control file listing the components that can be trimmed. The supplied file is W4TRIM.CTL, but you can rename it when you edit the file to specify the trim components.

1.6.3 Examples

The following examples show typical trimmer usage.

1. This example reads the control file and creates the log file, but does not trim any files.

   ```
   C:\INSTALL\W4TRIM
   ```

   The command must be run in the directory that `W4TRIM.CMD` resides in.

2. This example reads the control file and trims the specified files.

   ```
   C:\INSTALL\W4TRIM /T
   ```

1.6.4 Running Trimmer

Only one instance of the trimmer can be run at a time. The trimmer cannot be run by more than one client or more than one session on a single client simultaneously.

1.6.5 Trimmer Control File

The trimmer control file, \OS2\INSTALL\W4TRIM.CTL, lists each component that you are allowed to trim. You do not add components to or edit components from this file; you specify the action to be taken for each component. You can add comments to the file; each comment line begins with two slashes (//). Each line of the control file is of the format:

<trim_component_name> = keep

To specify a component for trimming, change its entry to:

<trim_component_name> = `trim`

For example, to trim (remove) the BonusPak from an install image, but keep the Java run-time, edit the TRIM.CTL file to:

```
BonusPakDelete = trim
JavaDelete = keep
```

1.6.5.1 Comments in the Control File

A comment line in the control file begins with two forward slashes (//) as the first non-blank characters. A example of a comment line is:

```
//Keep JavaDelete for southern clients
```

1.6.6 Trimmable Components

The list of trimmable components follows. Before you select components for trimming, carefully review the applications and system operations of the systems you are building. Make sure you do not trim components the target system's users need.

You might follow a sequence of first trimming the largest component your users do not need, then comparing the total operating system and user data requirements at the target system with the newly trimmed donor system. Continue to estimate space needed, space used, and trimming until you have arrived at a system that fits the available space.

The approximate size of each component in the chart below is the byte count of the component. Removal of a component may free more space due to disk block size or cluster size.

Each of these table entries refers to a file with more detail. Click on a component name to see more information about trimming that component.

Name	Approximate size in bytes
Active Registration Tool	428,000
Multimedia	14,567,832
Non-default base bitmaps	4,813,564
TCPIP - PCOM	9,889,636
TCPIP - UMAIL	3,186,295
Language, code pages, and locales	4,287,201
LANReq (IBMLAN)	4,170,824
MAC (LAN) drivers (IBMCOM)	3,281,383

SCSI, CD-ROM, PCMCIA device drivers	989,215
Online books	14,937,186
Speech (VoiceType)	21,319,569
OS/2 Warp 4 tutorial	4,346,726
Coaches	3,015,109
BonusPak	27,257,221
Java Run-time and Toolkit	2,490,763
Miscellaneous TCPIP files	5,420,062
Miscellaneous files	3,298,015

1.6.7 The Trimmer Utility Log File

The trimmer utility's log file is *<BootDrive>*:\OS2\INSTALL\W4TRIM.LOG and is appended to, not re-created, each time the utility is run.

The file contains a stanza for each component, naming the component, its individual files, and the size in bytes of each file.

A sample log file follows. To create the sample, ART was the component specified for trimming and `/T` was not specified. If `/T` had been specified, lines listing files would have been followed with `successfully deleted`.

```
TRIMMING STARTED ON 15 Aug 1997 AT 17:44:44 LOGGING ONLY
Configurator object for ART_Delete not found
Andthisistheinvalidline in the control file ignored
Configurator object for VT_Delete not found
Could not obtain dependee object Babushka handle
Invalid line Andthisistheinvalidline in the control file ignored
Total of 0 top-level snoopers have been run

TRIMUNIT ART_Delete LOGGING ONLY
PROCESSING REMOVAL OF FEATURE ART
8/06/96  12:10p     141824  A----  D:\os2\art\art.dll
8/06/96  12:10p      68123  A----  D:\os2\art\artr.dl
7/22/96   5:29p      28699  A----  D:\os2\art\artregr.dll
7/15/96   3:21p      11980  A----  D:\os2\art\art.hlp
7/15/96   3:21p      33657  A----  D:\os2\art\artfi.exe
7/15/96   3:22p      40448  A----  D:\os2\art\artadmin.exe
7/22/96   5:29p      23178  A----  D:\os2\art\artreg.exe
7/22/96   5:29p       4045  A----  D:\os2\art\prodreg.dat
7/15/96   3:21p      17438  A----  D:\os2\art\artfir.dll
8/28/96   2:51a      32636  A----  D:\os2\art\artchron.exe locked.
Trimming will require a reboot to follow
```

```
8/01/96   1:10p       25600  A----  D:\os2\art\artinet.dll
END TRIMUNIT ART_Delete
TOTALSIZE TRIMUNIT ART_Delete 427628
TRIMMING ENDED 15 Aug 1997 17:42:59

GRANDTOTALSIZE 0
```

The log file indicates when a file is locked. The log file also indicates when a file is in use and will be deleted at the next reboot. A system administrator could manually change the configuration entries that access this component, if those entries are available. However, this component is not one that the trimmer automatically updates configuration entries for.

If you specify a component not in the trim control file, the following is placed in the control file:

```
TRIMUNIT cannot be trimmed - required TRIMUNIT
component_name not specified in control file
```

1.6.8 Trimmer Dependency File

The DEP.REC file has lines for each component with dependencies. A dependency is another component that must also be trimmed whenever the named component is trimmed.

A typical line is:

```
component_name: dependent_component1, dependent_component2,
```

If you select `component_name` for trimming, you must also select `dependent_component1` and `dependent_component2`.

For example, if you want to trim Multimedia, you must also trim Basketball. The DEP.REC file would contain:

Multimedia: Basketball

1.6.9 Messages, Errors and Error Recovery

Examine the log to find files that were trimmed to find out the space saved and to have a record of the trimmed system.

1.6.9.1 Trimmer Messages

The trimmer utility can return the following messages:

Table 21. Trimmer Messages

Message	Explanation
Incorrect OS/2 version.	Trimmer requires OS/2 Warp 4
FI version is incorrect.	Trimmer requires Feature Install Version 1.2.1 or higher.
A control file *<control_file>* cannot be found or accessed.	
Could not obtain top object handle.	
Could not select the top object.	FI object handle for the top object of an FI-installed component
Could not obtain object handle for trimmer object *<trimmer_object>*.	
Could not obtain dependent object *<dependent_object>* handle.	
Could not obtain dependee object *<dependee_object>* handle.	
Could not obtain top inventory object handle.	
Delete file list is not specified.	
Could not resolve the boot drive.	
Could not resolve *<DeleteFlag>*.	
Could not resolve *<Feature ID>*.	
Could not resolve the *<GrandTotalSize>*.	Component trimmers update the FI variable maintained for storing the *<GrandTotalSize>*. If a component trimmer cannot resolve its value through the FI API, this error is returned.

Message	Explanation
Could not obtain object handle for an inventory *<object>* object.	
Cannot find INSTDATA.INI for the feature.	
Could not select the top inventory object.	
Action not specified.	
Action *<action>* failed.	
Invalid action specified.	
Feature must be specified via handle or ID.	
If a feature is specified by ID, a path to look for the feature must be specified.	
Unable to locate the feature.	
Variable name to resolve not specified.	
Cannot open file *<file_name>* specified to store the variable.	
Cannot write to file *<file_name>* specified to store the variable.	
A file name to store the variable must be supplied.	
A file name to get attributes for must be specified.	
Cannot resolve variables in the filename *<file_name>*.	
Could not resolve variable *<variable>*.	

1.6.9.2 Return Codes

The trimmer utility can return one of the following codes. The codes are consistent with *CID Enablement Guidelines*, S10H-9666.

0	Successful program termination; no reboot required.
4	Successful program termination; warning messages logged; no reboot required.
5	Unsuccessful program termination; no reboot required. Error messages logged.
6	Unsuccessful program termination; no reboot required. Unsuccessful program termination with unexpected internal errors detected and logged.

Although "no reboot required" is returned, you must reboot to put the configuration changes into effect and to trim the locked files.

Chapter 2. Traditional CID vs. Rapid Deployment Tools

This chapter discusses the differences of traditional CID and the new set of tools called Rapid Deployment Tools, which consist of Version-to-Version Tools and Enterprise Rescue tools.

In both cases, traditional software distribution through CID and software distribution through Rapid Deployment Tools meet the same scenario: The number of LAN-attached workstations in your company continues to grow and so do the resources required to install and maintain software on those workstations.

As of with these tools, the system administrator in charge of software distribution can take advantage of the following:

- Simplify the installation of software on pristine OS/2 workstations.
- Simplify the migration process from previous OS/2 versions (such as OS/2 V2.11 or the OS/2 Warp 3 family) to OS/2 Warp 4 workstations.
- Reduce the time required to perform software installations.
- Centralize the configuration and remote installation of software.
- Reduce software installation costs.
- Eliminate human intervention at the target workstation when preparing and executing the configuration, installation, migration, and maintenance processes that are necessary to operate the workstation and maintain the software on it.
- Enable the code executed at the target workstation to perform all required configuration and installation tasks, including the integration of previous customization.
- Centralize human intervention at a central preparation site.

2.1 Traditional CID Software Distribution

The most common method of installing workstation software is to install from diskettes or a CD-ROM. This method has the following critical factors:

- Human intervention is required to customize the product by passing configuration information to the installation program via its dialog interface. This process must be performed by a person who is familiar with this dialog interface, with the product features to be installed to meet the end user's needs, and with the system environment where the product is installed.

- Since most of today's products are shipped on large numbers of diskettes or CD-ROMs, media exchange is required during the installation process.
- Information about the progress of the installation process as well as information about whether the process completed successfully must be checked to guarantee a fully operational system.
- Some software requires the workstation to be rebooted in order to activate configuration changes.

The last three tasks also require human intervention. Although they do not require as much system-specific knowledge as the first task, they may require some basic knowledge about how to install workstation software. In order to achieve the goal of unattended installation, all of these tasks must be executed automatically.

With the standard installation method, an installation program is shipped with the product and is tailored to the specific installation needs of that product. With this installation program, one critical factor is automated: The installation program contains logic to check underlying hardware and software to determine which code modules need to be installed on the workstation and which files (such as `CONFIG.SYS` or `*.INI`) need to be updated.

The standard installation method requires the end user to provide installation and configuration information at the time of product installation. Thus, product configuration and product installation are not separable tasks. Installation and configuration information must be provided at the same time by the same person at the workstation itself. In addition, skilled people must be present at the workstation to perform this standard installation process. In other words, this installation method is not feasible if the installation process must be managed remotely.

Table 22 briefly summarizes the basic characteristics of the standard installation method:

Table 22. Basic Characteristics of the Standard Installation Method

Ability to exploit the software product's tailored installation program at the target workstation	X
Ability to remotely manage the process of software configuration, installation, migration and maintenance. No human intervention at the target workstation is required	–
Ability to migrate previous customization	X

2.2 Installation by Replication Using Rapid Deployment Tools

To overcome the drawback of not being able to manage a standard installation remotely, a technique known as replication is used. A new set of tools were written to ease the replication process: Version-to-Version Tools and Enterprise Rescue Tools, summarized in Rapid Deployment Tools (RDT). The RDT tools are built by performing the following steps:

1. Install the product at the preparation site in the same way in which it would be customized and installed at the target workstation using the installation program delivered with the product. This machine will act as a donor system.
2. At the donor system, all files must be packed into one macro file and be copied to a code server that is accessed by a software distribution server.
3. The `PROBE` tool extracts user-specific and machine-specific configuration information that later will be used by the configurator tool.
4. Use of a software distribution manager to enable the distribution of clone images to other workstations.
5. Invoke the software distribution agent at the target workstation to enable the machine being installed by replication with the clone image. This can be done either by a local administrator or by a program. By using a program, the process does not require human intervention since the customization dialog is eliminated.
6. Machine-specific and user-specific configuration is done by the configurator tool. This tool inputs files created by the `PROBE` tool. The system administrator has modified those input files to configure the clone machine properly.

The replication method may not be suited to installing software on target workstations that have a different hardware or software configuration than the preparation site workstation. For each different machine, we recommend setting up individual donor systems.

Although replication achieves the goal of minimizing human intervention at the target workstation by centralizing installation tasks at the administration site, it requires reverse engineering the installation process and distinguishing and copying with the configuration differences between the administrator and the target workstation. In addition, replication does not migrate software from previous customization unless the new Rapid Deployment Tools are used. This set of Rapid Deployment Tools makes use of replication and configuration techniques to allow both installation and migration of OS/2 workstations.

Table 23 briefly summarizes the basic characteristics of the replication method to date:

Table 23. Basic Characteristics of the Rapid Deployment Tools Method

Ability to exploit the software product’s tailored installation program at the target workstation	X
Ability to remotely manage the process of software configuration, installation, migration and maintenance. Human intervention at the target workstation is commonly not required although some operating systems still have human intervention dependencies	X
Ability to migrate previous customization	X

Chapter 3. Setting Up a Software Distribution Server

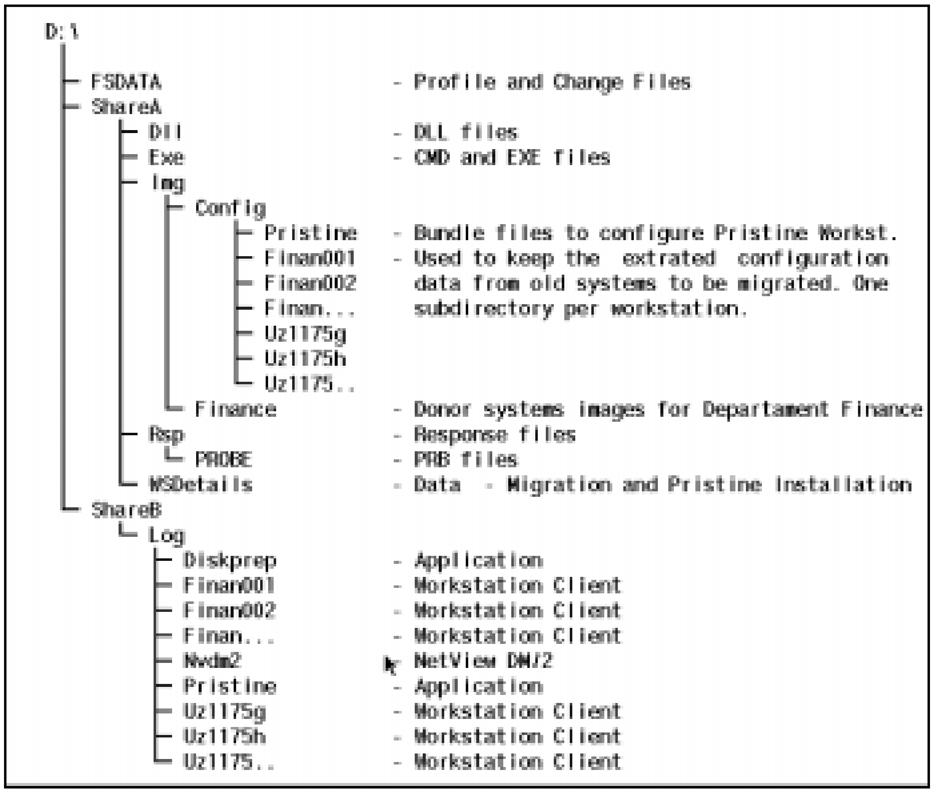
This chapter illustrates configuring the Netview Distribution Manager/2 as our software distribution server. As for installation and requirements of NetView DM/2, refer to the redbook titled *The OS/2 Warp 4 CID Software Distribution Guide,* SG24-2010.

3.1 NetView DM/2 Server Directory Structure Considerations

The NetView DM/2 Server provides three shareable directories (SharedDirA SharedDirB and FSDATA) that are accessible by CC clients during an install. The parameters for these shares are defined in the IBMNVDM2.INI file during the NetView DM/2 installation. They can be changed later.

Figure 16 on page 124 shows the directory structure used in our scenarios. The NetView DM/2 Server was installed on the C: drive. The shareable directories were defined on the D: drive. You can specify a different directory structure, but be sure that you have available disk space for all requirements of your site. These parameters are defined in the IBMNVDM2.INI file. The following items describes the shareable directories:

File Services Dir (FSDATA)	Used by NetView DM/2 Server for storing change file profiles (ASCII files) and change files.
ShareDirA (ShareA)	Shared by NetView DM/2. Used for storing response files, product images, and CMD, EXE and DLL files. You can create subdirectories if needed.
ShareDirB (ShareB)	Shared by NetView DM/2. Used for writing log files. The subdirectory LOG is under this directory. We recommend creating a subdirectory for each product under the LOG directory.



```
D:\
 │
 ├─ FSDATA                     - Profile and Change Files
 ├─ ShareA
 │    ├─ Dll                   - DLL files
 │    ├─ Exe                   - CMD and EXE files
 │    ├─ Img
 │    │   ├─ Config
 │    │   │     ├─ Pristine    - Bundle files to configure Pristine Workst.
 │    │   │     ├─ Finan001    - Used to keep the  extrated  configuration
 │    │   │     ├─ Finan002      data from old systems to be migrated. One
 │    │   │     ├─ Finan...      subdirectory per workstation.
 │    │   │     ├─ Uz1175g
 │    │   │     ├─ Uz1175h
 │    │   │     └─ Uz1175..
 │    │   └─ Finance           - Donor systems images for Departament Finance
 │    ├─ Rsp                   - Response files
 │    │   └─ PROBE             - PRB files
 │    └─ WSDetails             - Data  - Migration and Pristine Installation
 └─ ShareB
      └─ Log
          ├─ Diskprep          - Application
          ├─ Finan001          - Workstation Client
          ├─ Finan002          - Workstation Client
          ├─ Finan...          - Workstation Client
          ├─ Nvdm2               NetView DM/2
          ├─ Pristine          - Application
          ├─ Uz1175g           - Workstation Client
          ├─ Uz1175h           - Workstation Client
          └─ Uz1175..          - Workstation Client
```

Figure 16. Directory Structure of NetView DM/2 Server Used in Our Scenarios

3.1.1 The NetView DM/2 Server IBMNVDM2.INI File

The IBMNVDM2.INI file of NetView DM/2 Server is created during the installation of the server. If needed, you can change the parameters editing this file. It is located under the \IBMNVDM2 directory.

The server also sets SA (ShareA directory) and SB (ShareB directory) variables to be used in change files. See a sample of the SA variable in Figure 19 on page 128.

By default, NetView DM/2 sets the shareable directories ShareA and ShareB with Read an Write access. You can modify the access by adding the `R` or `W` parameters in the definition of the shareable directories. See the example below.

```
SharedDirA = D:\ShareA,R
```

```
//******************************************************************
//*                                                                *
//*  IBM NetView DM/2 Ver.2  INI File                              *
//*                                                                *
//******************************************************************
ServerName         = NVDM1175
FileserviceDir     = D:\FSDATA
SharedDirA         = D:\SHAREA
SharedDirB         = D:\SHAREB
MaxRequests        = 8
MaxClients         = 10
MaxShrFiles        = 500
AdapterNum         = 0
AgentTimeOut       = -1
MessageLogFile     = C:\IBMNVDM2\MESSAGE.DAT
ErrorLogFile       = C:\IBMNVDM2\ERROR.DAT
LogOption          = NVDM
RebootDelay        = 2
AutomaticPurgeReport    = YES
```

Figure 17. NetView DM/2 Server's IBMNVDM2.INI File Used in Our Scenarios

3.1.2 The CC Client IBMNVDM2.INI File

There are some differences between the IBMNVDM2.INI of the boot diskettes with a minimal NetView DM/2 CC Client and the client installed on the maintenance partition (see “Setting Up the Maintenance Partition” on page 175). One of them is how the NetView DM/2 Client attaches the drives ShareA, ShareB and FSDATA. See the table below.

Table 24. Attached Drives Using Boot Diskettes and Maintenance Partitions

	Boot Diskettes	**Maintenance Partition**
FSDATA	X:	Y:
ShareA	W:	X:
ShareB	V:	W:

The `DriveLetters` parameter in the IBMNVDM2.INI file of the maintenance partition is the parameter that you have to modify to attaches the same drives than the boot diskettes. The default parameter is:

```
DriveLetters = ZYXWVUTSRQPONMLKJIH
```

Modify to:

```
DriveLetters = ZXWVYUTSRQPONMLKJIHGF
```

Note

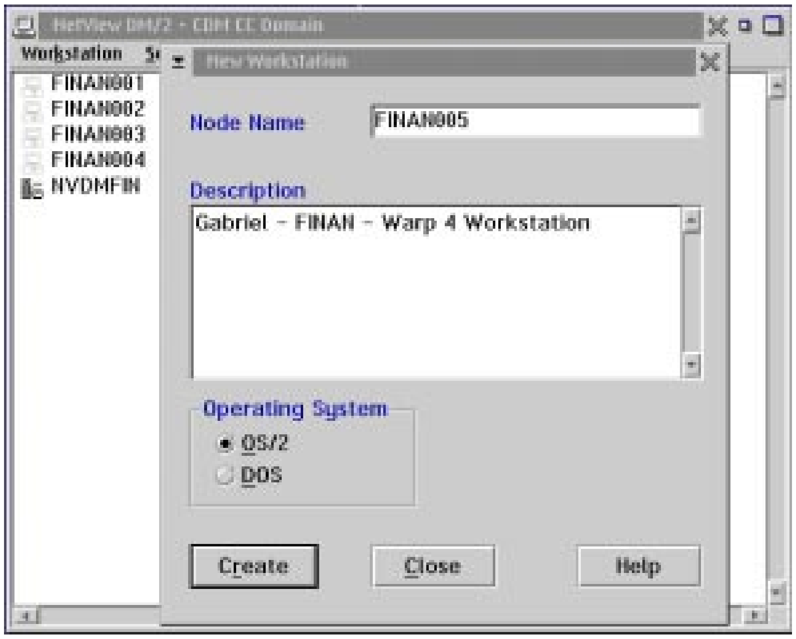
Adding the `DriveLetters` parameter into the IBMNVDM2.INI file of the boot diskettes will not enable the client to work properly because this parameter is not supported by the boot diskette's NVDM/2 Client.

3.2 Creating a New Client Workstation

There are two ways to create a new workstation client. One method is to use a dialog box, and the other is to use a command line `CDM ADD_WS` command.

3.2.1 Creating a New Client Workstation Using the PM Interface

To create a new Client Workstation, use the NetView DM/2 - CDM CC Domain window. Select **Workstation > New**. See Figure 18 on page 126.



Node Name: FINAN005

Description: Gabriel - FINAN - Warp 4 Workstation

Operating System: [✓ OS/2] DOS

Figure 18. Including a New Client Workstation in NetView DM/2

In the New Workstation window, configure the following items:

Name In the Node Name box, enter the Client Workstation name, for example, FINAN005. Names are not case

	sensitive, can be up to eight characters long, and the first digit **must** be an alpha character.
Description	In this box, enter your Client Workstation description. This field is optional and can be up to 242 characters long.
Operating System	Select the Operational System of your Client Workstation.

Select **Create** to include the Client Workstation.

3.2.2 Creating New Workstation Client Using the CDM ADD_WS Command

NetView DM/2 allows you to create a new workstation client using the CDM ADD_WS command. See the syntax below.

CDM ADD_WS Syntax

`CDM ADD_WS` *<workstation client> <"description">*

<workstation client>	Enter a valid name for the new workstation client.
<"description">	Enter a description of you new client.

You should receive a message telling you the workstation was successfully defined.

3.3 Building a Change File from an Existing Change File Profile

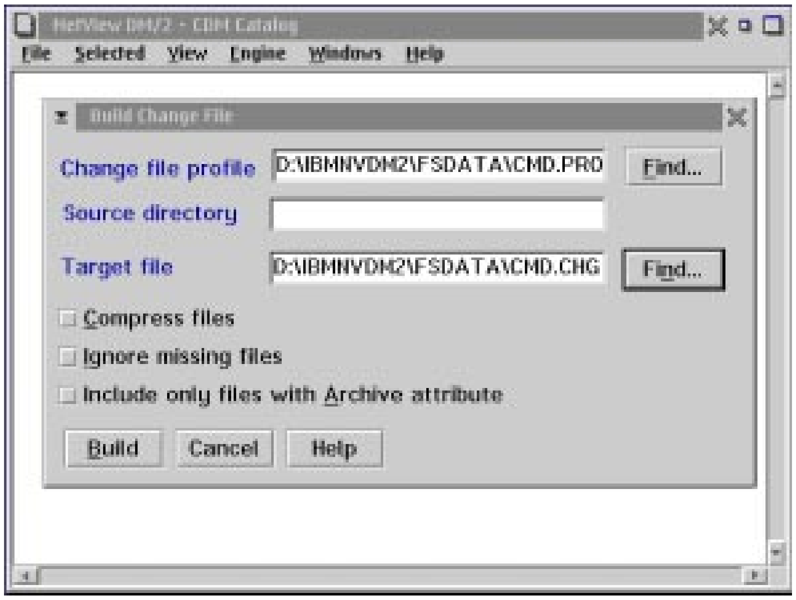
There are two methods of creating change files. One way is to use the dialog box, and the other is with the `CDM BUILD` command. In this section, you will create the change file from an existing change file profile (CMD.PRO was our ASCII file profile - see the example below) to open a command line in a NetView DM/2 Client using both methods.

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.WARP4.CMD.REF.1
     Description="OS/2 Warp 4 Command Prompt"
End

Section Install
Begin
      Program =  SA:\EXE\CMD.EXE
      Parms = /k
End
```

Figure 19. Example of an ASCII Change File Profile

3.3.1 Using the PM Interface

To create a change file from an existing change file profile, use the NetView DM/2 - CDM Catalog window as shown in Figure 20. Select **File > Build from Profile.**



Change file profile: D:\IBMNVDM2\FSDATA\CMD.PRO

Source directory:

Target file: D:\IBMNVDM2\FSDATA\CMD.CHG

☐ Compress files

☐ Ignore missing files

☐ Include only files with Archive attribute

Figure 20. NetView DM/2 Build Change File Window

Fill in the **Change file profile** field with your ASCII change file profile (in our scenarios, we used the .PRO extension) and the **Target file** with the name of the change file (.CHG extension), as shown in Figure 19.

Select the **Build** button. Verify that the change file successfully built and cataloged.

3.3.2 Using the CDM BUILD Command

NetView DM/2 allows you to build a change file from an existing change file profile using the `CDM BUILD` command. See the syntax below.

CDM BUILD Syntax

`CDM BUILD` *<change profile file> <change file>*

<change profile file>	The name of ASCII change file profile.
<change file>	The name of the target change file (.CHG extension)

You should receive a message informing you that the change file was successfully built and cataloged.

3.4 Creating Boot Diskettes with NetView DM/2 Client

In a pristine installation or in a version-to-version migration scenario, you need to create a set of boot diskettes that connect the client to the NetView DM/2 Server. To create this set of boot diskettes, use the `SEDISK.EXE`, `THINLAPS.EXE` and `NVDMBDSK.EXE` tools.

3.4.1 Creating the Set of Three Boot Diskettes

`SEINST` creates the "Installation Diskette" (Disk_0), Disk_1 and Disk_2. It requires three formatted diskettes. The general concepts of `SEDISK` are covered in *The OS/2 Warp 4 CID Software Distribution Guide*, SG24-2010.

SEDISK.EXE Syntax

`SEDISK /S:`*<source path>* `/T:`*<target drive>* `/P:`*<pcmcia_id#>*

`/S:`	Local hard drive or redirected drive that contains the OS/2 diskette images.
`/T:`	Target drive.

`/P:`	Optional parameter for PCMCIA support.

3.4.2 Installing LAN Support

`THINLAPS` creates a seed LAN transport system. It copies necessary LAN transport files from the source path to the target seed system. The CONFIG.SYS file on the "seed" system is updated with `DEVICE` and `RUN` statements. The general concepts of `THINLAPS.EXE` are covered in *The OS/2 Warp 4 CID Software Distribution Guide,* SG24-2010.

THINLAPS.EXE Syntax

`THINLAPS` *<source path> <target drive> <NIF file name>* **|** *<protocol>*

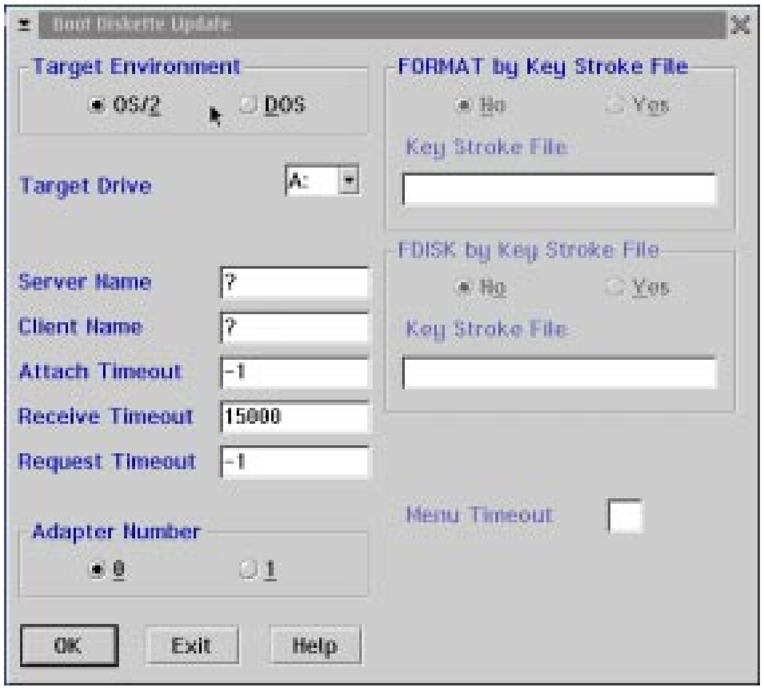
<source path>	The MPTS product image tree.
<target drive>	Target drive for seed system.
<NIF file name>	The name of the .NIF file that corresponds to the network adapter driver that will be stored on the target.
<protocol>	If not specified, `THINLAPS` uses the default NetBIOS. In our scenarios, we used: </NetBIOS>.

3.4.3 Installing the NVDM/2 Redirector

`NNDVBDSK.EXE`, located at your NetView DM/2 Server in the \IBMNVDM2\BIN directory, installs a minimal NetView DM/2 CC Client on the boot diskettes. Before using, copy the CONFIG.SYS file from Disk_1 to Disk_2. Leave the Disk_2 in drive A: and enter:

```
<drive:>\IBMNVDM2\BIN\NVDMBDSK
```

The NetView DM/2 Boot Diskette Update window appears. Enter the values as shown in Figure 21 on page 131.


Boot Diskette Update

Target Environment: [✓ OS/2] DOS

Target Drive: A:

Server Name: ?

Client Name: ?

Attach Timeout: -1

Receive Timeout: 15000

Request Timeout: -1

Adapter Number: [✓ 0] 1

FORMAT by Key Stroke File: [✓ No] Yes

Key Stroke File:

FDISK by Key Stroke File: [✓ No] Yes

Key Stroke File:

Menu Timeout:

Figure 21. NetView DM/2 Boot Diskette Update Window

Note

When a "**?**" is specified in Server Name and Client Name, you will be prompted to enter the both names on a NetView DM/2 Client start-up. This allows you to use the same set of diskettes for more than one workstation.

Click on **Ok** and observe the message `The diskette has been updated correctly!`. Select **Exit** and **Yes** to confirm.

To complete the install you **must** copy the **CONFIG.SYS** file from Disk_2 to the Disk_1.

3.5 Installing an Object on a Defined Workstation

There are two methods of installing an object on a defined workstation. One way is to use the dialog box, and the other is with the `CDM INSTALL` command.

3.5.1 Initiating a Remote Install Using the PM Interface

To install, use the NetView DM/2 - CDM Catalog window. You may select more than one object to install on a client if needed. Select the objects with the mouse, and from the menu bar, select the options **Selected > Install > No Force**.

The Install window contains two lists. The first list contains the objects that you had selected for installation, and the other contains the list of workstations on which the selected objects may be installed.

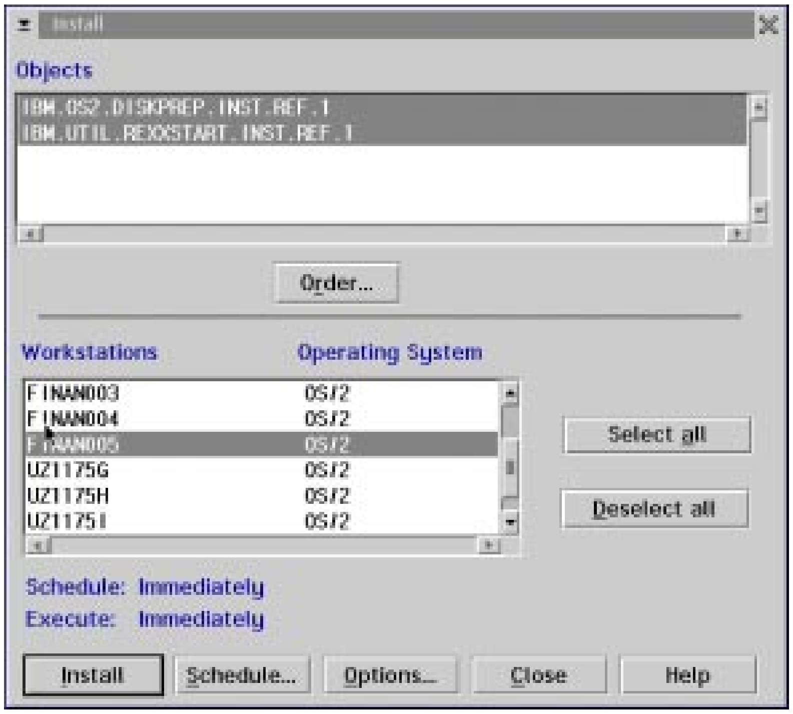
Figure 22 on page 133 shows the Install window. If more than one object are to be selected, you may need to change the order of installation. Select the **Order** button and in the **Install Order** window change the installation order by selecting an object and selecting **Up** or **Down**.

In the workstations list, select the workstations on which the selected objects will be installed.

To specify when the installation should be submitted to clients, select the **Schedule** button.

Select the **Options** button to set Disk Area, Removability, Space checking Automatic Acceptance and Corequisite group installations. In our scenarios, we used the default settings for all options except for the Corequisite group installations, which needs to be selected.

To proceed with the installation, select the **Install** button.



IBM.OS2.DISKPREP.INST.REF.1
IBM.UTIL.REXXSTART.INST.REF.1

Workstations	Operating System
FINAN003	OS/2
FINAN004	OS/2
FINAN005	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2

Schedule: Immediately
Execute: Immediately

Figure 22. Initiating Software Distribution in NetView DM/2

3.5.2 Initiating a Remote Install Using the CDM INSTALL Command

NetView DM/2 allows you to install an object in a workstation client using the `CDM INSTALL` command. See the syntax below.

CDM INSTALL Syntax

`CDM INSTALL` *<global name object>* `/WS:`*<workstation client>*

<global name object>	The object name as shown in the NetView DM/2 CDM Catalog window.
<workstation client>	The name of workstation client.

Chapter 4. Feature Installer FI / CLIFI

This chapter discusses the Feature Installer (FI) and its command line interface, called `CLIFI`.

4.1 Feature Installer (FI)

The Feature Installer offers a set of installation services available to software developers which frees the software developer from writing customized installation code to install software. In addition, for every install service used, Feature Install automatically supplies uninstall services.

The set of installation services includes file copies, updates to any ASCII file, and updates to any .INI file. A software product can be broken down into manageable parts that can be selectively installed by the software consumer. Also, if there are important relationships between these various parts, there are services that run condition checks and perform specific actions. Finally, if there is a needed service that is not readily provided, there is the ability to run programs supplied by the software developer.

By using Feature Install to install your software, the software developer can standardize the installation process for their customers. This is particularly useful if you produce more than one product. Feature Install also provides the software developer the ability to customize the final product by providing a totally authorable view of your software product and override capabilities for important help panels.

4.1.1 Using the Command Line Interface (CLIFI)

The `CLIFI` command enables you to perform various Feature Install tasks using a command line interface. To use `CLIFI`, type `CLIFI` at an OS/2 command prompt.

The options are defined as follows:

`/A`: Action code
The following action codes are selectable, and the following requirements apply when using these actions:

`B` Build

Requires `/R` (`/L1`, `/L2` are optional).

C CID Install

Requires /R, /R2, and /S (/L1 and /L2 are optional).

D Delete Feature

I Install

Requires /O (/F, /L1, /L2 are optional) or /R (/L1, /L2 are optional).

O Open Feature ID view

Requires /O (/F, /L1, /L2 are optional).

/O should always be set to /O:"INV_xxx" (where xxx is the feature ID of the original install object) because inventory objects have an "INV_" prefix.

P Package

Requires /O (/F, /L1, /L2 are optional) or /R (/L1, /L2 are optional). Note that packaging can only be executed on the top install object.

T Template object

W Write response file

requires /O and /R (/F, /L1, /L2 are optional).

U Uninstall

Requires /O (/F, /L1, /L2 are optional).

If you use the /F option, specify /F:"<WP_INSTALLED>" because all inventory objects are located in that folder.

/B: Boot drive with drive letter and colon (:)

/R: Drive, path, and file name of response file. This option must specified with actions B, C, and W.

It is optional with actions I and P.

If a response file is not specified with actions I and P, an object ID (/O) must be specified to perform I or P actions.

If a response file (/R) and an object ID (/O) are specified using the same CLIFI command for actions I and P, then /O will be ignored.

`/R2`: Drive, path, and file name of partial response file.

This option is valid only with action `C` (CID install).

`/S`: Fully qualified path to installation media.

This option is valid only with action `C` (CID install).

`/O`: Feature ID to perform action on existing object.

This parameter must be specified with actions `D`, `O`, `W`, and `U`.

It is optional with actions `I` and `P`.

`/F`: Folder containing Feature Install object.

The default is `<WP_DESKTOP>`.

This parameter can be used with every action parameter.

With actions `B` and `T`, `/F` specifies the destination folder for newly created objects. If `/F` is not specified, an object is created on the desktop `<WP_DESKTOP>`.

For the following actions (together with `/O`): `D`, `O`, `W`, `U`, `I`, and `P`, `/F` specifies the location to start the search for the object ID.

If `/F` is not specified, `CLIFI` will set the search start location to `<WP_DESKTOP>`.

If the `/F` parameter is an object, such as `<WP_DESKTOP>`, or a path name containing spaces or unusual characters, enclose the object or directory name in double-quotation marks.

`/REG`: Specifies DLL to register. This parameter must be specified with `/CLASSNAME` (`/L1` and `/L2` are optional).

`/CLASSNAME`:
Specifies object class name to register. This parameter must be specified with `/REG` (`/L1` and `/L2` are optional).

`/SET`: Sets a variable (or response file keyword) for a given object. This parameter must be specified with `/O` (`/F`, `/L1`, `/L1` are optional).

`/PARMS`:
Drive, path, and file name of a parameters file, which is a file that contains `CLIFI` parameters. Each `CLIFI` parameter must be on a

separate line.

For example, the `CLIFI` parameter file for the
`CLIFI /A:I /R:C:\CLIRSP.FIL`
command would contain the following:

```
/A:I
/R:C:\CLIRSP.FIL
```

`/L1`: Drive, path, and file name of error log file.

`/L2`: Drive, path, and file name of history log file.

The following examples demonstrate how to use the single-line `CLIFI` command:

```
CLIFI /A:I /R:\APPS\WIGET.RSP /L1:D:\ERR.LOG /L2:D:\HIST.LOG
CLIFI /A:P /O:NewViewInc /F:"<WP_NOWHERE>"
CLIFI /A:W /O:NewViewInc /F:C:\OS2\INSTALL /R:RESPNEW.RSP
CLIFI /O:NewViewInc /SET:MediaSet[0].Media[0].MediaPath=C:\
CLIFI /REG:C:\OS2\DLL\INSTALL.DLL /CLASSNAME:WPInstall
CLIFI /A:C /S:Z:\OS2IMAGE /B:C: /R:C:\OS2\INSTALL\FIBASE.RSP
      /R2:C:\OS2\INSTALL\SAMPLE.RSP /F:C:\OS2\INSTALL
      /L1:Z:\LOGS\CLERR.LOG /L2:Z:\LOGS\CLHIST.LOG
```

Note: If packaging in an unattended environment using `CLIFI`, the media directory must be empty. If it is not empty, Feature Install displays a confirmation dialog asking permission to erase the directory.

4.2 New Keywords in OS/2 Warp 4

These keywords are valid keywords, but are not used for the installation of OS/2 through `SEINST`/`RSPINST`. They are related to applications loaded by the command line interface of Feature Installer (`CLIFI`).

- **ART.Selection**

  Specifies whether or not to install the application registration (also known as the dancing elephant) option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **BASKPSP** (BonusPak AskPSP Knowledge Database)

  Specifies whether or not to install BonusPak AskPSP support during the installation. In total, there are two keywords to be specified:

  - **BASKPSP.Selection**

```
0 = do NOT install AskPSP support
1 = install AskPSP support (DEFAULT)
```

  - **BASKPSP.TarDrv**

    Specifies the target drive in which AskPSP is to be installed. The key value is a valid drive letter.

    ```
    C:
    ```

- **BPCIM** (BonusPak Compuserve Information Manager)

  Specifies whether or not to install Compuserve files during the installation. In total, there are two keywords to be specified:

  - **BPCIM.Selection**

    ```
    0 = do NOT install Compuserve files
    1 = install Compuserve files (DEFAULT)
    ```

  - **BPCIM.TarDrv**

    Specifies the target drive in which Compuserve is to be installed. The key value is a valid drive letter.

    ```
    C:
    ```

- **BPFAXWORKS**

  Specifies whether or not to install FaxWorks files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPFAXWORKS.Selection**

    ```
    0 = do NOT install FaxWorks files
    1 = install FaxWorks files (DEFAULT)
    ```

  - **BPFAXWORKS.TarDrv**

    The key value is a valid drive letter.

- **BPHAL**

  Specifies whether or not to install HyperAccess Lite files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPHAL.Selection**

    ```
    0 = do NOT install HyperAccess Lite files
    1 = install HyperAccess Lite files (DEFAULT)
    ```

  - **BPHAL.TarDrv**

The key value is a valid drive letter.

- **BPHPJETCLIENT**

  Specifies whether or not to install Jet Admin Client files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPHPJETCLIENT.Selection**

    ```
    0 = do NOT install Jet Admin Client files (DEFAULT)
    1 = install Jet Admin Client files
    ```

  - **BPHPJETCLIENT.TarDrv**

    The key value is a valid drive letter.

- **BPHPJETSERVER**

  Specifies whether or not to install Jet Admin Server files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPHPJETSERVER.Selection**

    ```
    0 = do NOT install Jet Admin Server files (DEFAULT)
    1 = install Jet Admin Server files
    ```

  - **BPHPJETSERVER.TarDrv**

    The key value is a valid drive letter.

- **BPIBMWORKS**

  Specifies whether or not to install IBM Works files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPIBMWORKS.Selection**

    ```
    0 = do NOT install IBM Works files
    1 = install IBM Works files (DEFAULT)
    ```

  - **BPIBMWORKS.TarDrv**

    The key value is a valid drive letter.

- **BPMARKNET**

  Specifies whether or not to install MarkNet Port Driver files during the installation.

It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

- **BPMARKNET.Selection**

  ```
  0 = do NOT install MarkNet Port Driver files (DEFAULT)
  1 = install MarkNet Port Driver files
  ```

- **BPMARKNET.TarDrv**

  The key value is a valid drive letter.

- **BPMARKVIS**

  Specifies whether or not to install MarkVision files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPMARKVIS.Selection**

    ```
    0 = do NOT install MarkVision files (DEFAULT)
    1 = install MarkVision files
    ```

  - **BPMARKVIS.TarDrv**

    The key value is a valid drive letter.

- **BPRS2**

  Specifies whether or not to install Remote Support files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPRS2.Selection**

    ```
    0 = do NOT install Remote Support files
    1 = install Remote Support files (DEFAULT)
    ```

  - **BPRS2.TarDrv**

    The key value is a valid drive letter.

- **BPVIDEOIN**

  Specifies whether or not to install VideoIn files during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **BPVIDEOIN.Selection**

    ```
    0 = do NOT install VideoIn files
    1 = install VideoIn files (DEFAULT)
    ```

  - **BPVIDEOIN.TarDrv**

  The key value is a valid drive letter.

- **COACHES.Selection**

  Specifies whether or not to install the Warp guide option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **DAXCOMP1**

  Specifies whether or not to install the Developer API Extensions during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **DAXCOMP1.Selection**

    ```
    0 = do NOT install
    1 = install (DEFAULT)
    ```

  - **DAXCOMP1.TarDrv**

    The key value is a valid drive letter.

- **HOTPLUG.Selection**

  Specifies whether or not to install support for an external floppy drive for a laptop.

  ```
  0=Don’t install (DEFAULT)
  1=Install
  ```

- **JAVASMPLS.Selection**

  Specifies whether or not to install the Java sample files.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **JAVATLKT.Selection**

  Specifies whether or not to install the Java toolkit option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **ODBASE**

  Specifies whether or not to install OpenDoc during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **ODBASE.Selection**

    ```
    0 = do NOT install
    ```

```
1 = install (DEFAULT)
```

- **ODBASE.TarDrv**

  The key value is a valid drive letter.

- **ODSECBASE**

  Specifies whether or not to install the Security Extensions Services during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **ODSECBASE.Selection**

    ```
    0 = do NOT install
    1 = install (DEFAULT)
    ```

  - **ODSECBASE.TarDrv**

    The key value is a valid drive letter.

- **SRVDIAG.Selection**

  Specifies whether or not to install the serviceability and diagnostic aids option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **SRVDOC.Selection**

  Specifies whether or not to install the serviceability documentation option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **SYSMGT.Selection**

  Specifies whether or not to install the system management option.

  ```
  0 = Don’t install
  1 = Install (DEFAULT)
  ```

- **VT**

  Specifies whether or not to install the VoiceType option during the installation.

  It has two subkeywords to indicate if it is or is not to install. The second keyword provides a valid drive where to install:

  - **VT.Selection**

    ```
    0 = do NOT install
    1 = install (DEFAULT)
    ```

  - **VT.TarDrv**

    The key value is a valid drive letter.

- **WARMDOCK.Selection**

  Specifies whether or not to install support for Dock II docking station for an IBM Thinkpad.

  ```
  0=Don’t install (DEFAULT)
  1=Install
  ```

- **WARMSWAP.Selection**

  Specifies whether or not to install support for floppy and CD-ROM swapping for UltraBay devices on an IBM Thinkpad.

  ```
  0=Don’t install (DEFAULT)
  1=Install
  ```

Keyword WARMPLUG.Selection Not Supported!

The SAMPLE.RSP file documents a keyword that actually is not supported:

```
WARMPLUG.Selection
```

This keyword has been replaced by:

```
WARMSWAP.Selection
```

4.3 Feature Install (CLIFI) Response File

The following section introduces Feature Installer techniques with the focus on installing objects through response files. As mentioned in “New Keywords in OS/2 Warp 4” on page 138, Feature Installer is used to install additional OS/2 Warp 4 components, such as Java, Developer Extensions and so forth, and BonusPak features, such as IBM Works, VoiceType and so forth. Find a more common approach of how Feature Installer uses a particular response file which might help you to get your software installed through Feature Installer rather than a product software installer.

4.3.1 Creating and Reading a Response File

The install object response file is an ASCII file that contains an entire install object hierarchy. This file includes all of the instance data from each install object in the hierarchy as well as information that defines the hierarchy structure.

A response file is created by selecting **Response file** and then **Save** from the install object's pop-up menu. A Save Response File dialog is displayed where you can specify the drive, path and file name of the response file. The response file will contain the install object hierarchy beginning from the install object where the Response File Save option was selected.

For example, if **Response File Save** is selected from the top install object, then the entire object hierarchy will be saved to the response file. If **Response File Save** is selected from one of the subfeature install objects, then only that subfeature's hierarchy will be saved.

A response file can be used to create an entire install object hierarchy. The `CLIFI` utility provided with the Feature Install Developer's Toolkit will create an install object hierarchy from a response file. Also, an existing install object's settings and underlying hierarchy can be loaded by selecting **Response File** and then **Read** from the object's pop-up menu. The Read Response File dialog is displayed where you can specify the drive, path and file name of the response file.

4.3.2 Response File Example

Following is an example of a response file. Comments can be placed anywhere in the response file. If `#` is the first character in the line, the rest of the line is treated as a comment. If you edit the response file, be sure to use an editor that preserves tabs.

```
# Sample response file
TopObjectID=MYFID
MYFID=(
    SubfeatureID=MYFID Child
        ObjectTitle[0]=FI Object
        Mode=Development
        PackageTitle=MPT
        CompanyName=IBM
        VersionNumber=1.45
        VersionDate=03-19-97

        Variable=(
            Name=variable1
            Description=Dsc. of var1
            Value=Default_value
            ValidationExit=(
                CallTime=19
                ExitType=2
                FilePath=DLLNAME.DLL
                ExitData=UserValidate
```

```
                        SessionVisible=1
                    )

                    ResolutionExit=(
                        CallTime=18
                        ExitType=2
                        FilePath=CMDRES.CMD
                        ExitData=-pcmd1 -pcmd2
                        SessionVisible=1
                    )
                )

        Dependency=(
                    FeatureID=ConditionID
                    ActionFeatureID=ResultID
                    WhenFeatureID=CauseID
                    ResolveTime=1
                    Condition=1
                    Action=1
                    ActionText[0]=Text Text Text
                    Setting=SettingKey
                    Value=SettingValue
                    ResolveNow=1
                    CaseSensitive=0
                )

        UserExits=(
                    ExitType=1
                    FilePath=DLLNAME.DLL
                    ExitData=-pcmd1 -pcmd
                    SessionVisible=1
                    CallTime=1
                ObjectCreation=(
                    ClassName=WP_CLASS
                    ObjectTitle[0]=My Program
                    SetupString=EXENAME=exefile.exe^;PARAMS=param
                    Location=<WP_FOLDERNAME>
                    Flags=0
                )

                ClassRegistration=(
                    ClassName=WP_CLASS
                    DLLName=regclass.dll
                    ClassReplace=WP_OLD
                )

                MediaSet=(
                    SetName=MediaSetName
                    Media=(
                        Type=2
                        Capacity=48
                        ClusterSize=34
                        Description=CD
                        MediaPath=D:\
                        ReservedSpace=1000
                    )

        MediaFinishedExit=(
                        CallTime=20
                        ExitType=1
                        FilePath=DLLNAME.DLL
                        ExitData=-pcmd1 -pcmd2
                        SessionVisible=1
                    )
                )

        AsciiFile=(
                    CaseSensitive=1
                    UseGrep=0
                    FileName=config.sys
                    EditLine=Line To Edit
                    SearchLine=set path
                    NewLine=rem set path
                    SearchFound=2
                    SearchNotFound=2
                    AddBeforeLine=beforeline
                    AddAfterLine=afterline
                    Delimiter=*
                    SubstrChk=1
                    DelimChk=1
                    Action=1
                    EditLineOcc=2
                    SrcLineOcc=1
                    AddBLineOcc=1
                    AddALineOcc=1
                    MatchPosition=2
                    NoMatchPosition=2
                )

                Os2PrfIni=(
                    FileName=d:\path\os2.ini
                    Application=ApplicationName
                    Key=KeyName
                    Value=KeyValue
                    IniType=1
                )

                WinOs2Ini=(
                    FileName=d:\winpath\win.ini
                    Application=WinAppName
                    Key=WinKeyName
                    Value=WinKeyValue
                    IniType=2
                )

        File=(
                    EAMediaIndex=0
                    Source=S:\srcpath\srcname
                    SourcePath=S:\srcpath
                    SourceFileName=srcname
                    SourceChecksum=0
                    SourceEASize=0
                    CBFile=0
                    CBFileAlloc=0
                    AttrFile=0
                    TargetPath=T:\targetpath
                    TargetFileName=srcname
                    MediaPath=mediapath
                    MediaFileName=srcname
                    MediaNumber=0
                    Flags=0
                    TargetAttrib=3

                 Restriction=(
                        MediaSet=MyMediaSet
                        MediaNumber=1
                    )
                )

                Prerequisite=(
                     FeatureID=PrereqFeature
               )
        )
```

4.3.3 FI Response File Keywords

The following list describes each of the keywords that can be used in a Feature Installer response file. The keywords listed include sample values as shown in “Response File Example” on page 145.

`TopObjectID=MYFID`	Feature ID that identifies the top install object. *Valid Values*: Any unique string. The string can contain spaces but must not contain any periods.
`MYFID=(`	Current feature ID marking the beginning of the install object description. This type of block can appear multiple times. *Valid Values*: Any previously defined feature ID.
`SubfeatureID=MYFID Child`	Subfeature ID. *Valid Values*: Any defined feature ID (can appear multiple times if more than subfeature exists, or can be omitted if there are no subfeatures).
`ObjectTitle[0]=FI Object`	Object title[line# - 1]. *Valid Values*: Any ordered set of strings.
`Mode=Development`	State of the object. *Valid Values*: Development, User, or Inventory.
`PackageTitle=MPT`	Package title. *Valid Values*: Any string.
`CompanyName=IBM`	Company name. *Valid Values*: Any string.
`VersionNumber=1.45`	Version number. *Valid Values*: Any string.
`VersionDate=03-19-97`	Version date. *Valid Values*: Any string.
`Variable=(`	Marks the beginning of a variable description. This type of block can appear multiple times. *Valid Values*: No value associated with this keyword.
`Name=var1`	Variable name. *Valid Values*: Any string (the string can contain spaces).

`Description=Description of var1`	Description of the variable. *Valid Values*: Any string.
`Value=Default_value`	Default value for the variable. *Valid Values*: Any string.
`ValidationExit=(`	Marks the beginning of a variable validation user exit description, *Valid Values*: No value associated with this keyword.
`CallTime=19`	Determines when the user exit is called (can be omitted but can not be specified with an invalid value). *Valid Values*: `19` Variable validation user exit `18` Variable resolution user exit
`ExitType=2`	Type of user exit. *Valid Values*: `0` DLL (default; can be omitted) `1` EXE `2` CMD
`FilePath=DLLNAME.DLL`	User exit location. *Valid Values*: Fully qualified path name.
`ExitData=UserValidate`	Details for the user exit. *Valid Values*: For exit type `0`, this is the name of the procedure in the DLL that is being called. For exit type `1`, this is the EXE file parameters. For exit type `2`, this is the CMD parameters
`SessionVisible=1`	If the session is visible or minimized (for ExitType=1 or 2 only). *Valid Values*: `0` minimized `1` visible to user
`ResolutionExit=(`	Marks the beginning of a variable resolution user exit description. *Valid Values*: No value associated with this keyword.

`CallTime=18`	Determines when the user exit is called (can be omitted). *Valid Values*: `19` Variable validation user exit `18` Variable resolution user exit
`ExitType=2`	Type of user exit. *Valid Values*: `0` DLL (default) `1` EXE `2` CMD
`FilePath=CMDRES.CMD`	User exit location. *Valid Values*: Fully qualified path.
`ExitData=-pcmd1 -pcmd2`	User exit details. *Valid Values*: For exit type 0, the name of the procedure in DLL that is being called. For exit type 1, the EXE file parameters For exit type 2, the CMD file parameters
`SessionVisible=1`	If the session is visible or minimized (for ExitType=`1` or `2` only). *Valid Values*: `0` minimized `1` visible to user
`Dependency=(`	Marks the beginning of a dependency description. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`FeatureID=ConditionID`	ID of the feature for which the state is examined to perform a dependency action. If the FeatureID in a dependency is blank, the current object's feature ID will be used. *Valid Values*: Any previously defined feature ID.
`ActionFeatureID=ResultID`	ID of the feature for which the state is examined to perform a dependency action. *Valid Values*: Any previously defined feature ID.

`WhenFeatureID=CauseID`	ID of the feature of which the state is examined to evaluate the need to check the existence of the dependency condition. *Valid Values*: Any previously defined feature ID.
`ResolveTime=1`	Determines when the condition of the feature defined by When FeatureID is evaluated. *Valid Values*: `0` selected `1` deselected `2` installed `3` uninstalled
`Condition=1`	Determines the condition of the feature defined by `FeatureID` that has to be met for the dependency action to be performed. *Valid Values*: `0` exists (default) `1` does not exist `2` selected `3` is not selected `4` equal `5` not equal `6` less than `7` less or equal `8` greater than `9` greater or equal `10` contains `11` is part of
`Action=1`	Determines what dependency action is to be performed if all conditions are met. *Valid Values*: `0` select feature `1` deselect feature `2` uninstall feature `3` display warning `4` display error `5` set variable
`ActionText[0]=Text Text Text`	Text to be displayed for the warning (`Action=3`) or for the error (`Action=4`) [line#-1] (for `Action=3` or `Action=4` only).

	Valid Values: Any ordered set of strings.
`Setting=SettingKey`	Key to be compared to the value defined by Value key in the feature defined by `ActionFeatureID` (the true comparison is defined by `Condition` key). This is valid for `Condition=4`, `Condition=5`, `Condition=6`, `Condition=7`, `Condition=8`, and `Condition=9` only. *Valid Values*: Any key valid for this feature.
`Value=SettingValue`	Value to be compared to the value of the key. The key is defined by the `Setting` key in the feature, which is defined by `ActionFeatureID` (the true comparison is defined by `Condition` key). This is valid for `Condition=4`, `5`, `6`, `7`, `8`, and `9` only. *Valid Values*: Any value compatible to the Setting key type.
`ResolveNow=1`	If a Value key setting contains variables, this key determines if they are resolved before the Value is assigned to the Setting. Otherwise, the Setting is assigned the value with variable references in it (for `Action=5` only). *Valid Values*: `0` variables resolved at some later point (default) `1` variables resolved before the value is assigned
`CaseSensitive=0`	Determines if the comparison defined by `Condition` is case sensitive (for `Condition=4`, `5`, `6`, `7`, `8`, and `9` only). *Valid Values*: 0 case insensitive comparison `1` case-sensitive comparison (default)
`UserExits=(`	Marks the beginning of a User Exit Description. This type of block can appear multiple times. *Valid Values*: No value associated with this keyword.

`ExitType=1`	User exit type. *Valid Values*: `0` DLL (default) `1` EXE `2` CMD
`FilePath=DLLNAME.DLL`	User exit location. *Valid Values*: Fully qualified path name.
`ExitData=-pcmd1 -pcmd2`	User exit details. *Valid Values*: For exit type `0`, name of the procedure in DLL that is being called. For exit type `1`, EXE file parameters. For exit type `2`, CMD file parameters.
`SessionVisible=1`	Specifies if the session is visible or minimized (for ExitType=1 or 2 only). *Valid Values*: 0 minimized (default) 1 visible to user
`CallTime=1`	Determines when the user exit is called. *Valid Values*: `0` after awakening (default) `1` before sleeping `2` before install `3` before file transfer `4` after file transfer `5` before configuration updates `6` after configuration updates `7` before object creation `8` after object creation `9` after install `10` before uninstall `11` before file deletion `12` after file deletion `13` before configuration entry removal `14` after configuration entry removal `15` before object deletion `16` after object deletion `17` after uninstall
`ObjectCreation=(`	Marks the beginning of the description of an object to be created. This type of block can

	appear multiple times. *Valid Values*: No values associated with this keyword.
`ClassName=WP_CLASS`	Class name for the object. *Valid Values*: Any registered class name.
`ObjectTitle[0]=My Program`	Object title[line#-1]. *Valid Values*: Any ordered set of strings.
`SetupString=EXENAME=exefile.exe^;PARAMS=param`	Setup string for the object. *Valid Values*: Setup string without any spaces consisting of pairs Key=Value, delimited by ^; (caret-semicolon combination). For a list of valid keys and appropriate values, refer to the *Workplace Shell Programming Guide*.
`Location=<WP_FOLDERNAME>`	Location of the object on the Desktop. *Valid Values*: Object ID for a folder. If folder does not exist, the object is placed on the Desktop (`<WP_DESKTOP>`).
`Flags=0`	Object creation flags that determine action performed if the object already exists. *Valid Values*: `0` CO_FAILIFEXISTS `1` CO_REPLACEIFEXISTS (default) `2` CO_UPDATEIFEXISTS `3` CO_PRESERVEOLD (This flag can be used to create only Workplace Shell objects on OS/2 Warp 4 systems.)
`ClassRegistration=(`	Marks the beginning of the description of a class to be registered. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`ClassName=WP_CLASS`	Class name *Valid Values*: Class name defined in DLL determined by DLLName key. For a full description of this

	key, refer to the description of *WinRegisterObjectClass* in the *Workplace Shell Programming Guide*.
`DLLName=regclass.dll`	Name of the DLL containing the definition of the class. *Valid Values*: Fully qualified path name. For a description of valid keys and their appropriate values, refer to the description of *WinRegisterObjectClass* in the *Workplace Shell Programming Guide*.
`ClassReplace=WP_OLD`	Class to replace with the class determined by ClassName key (can be omitted if no class has to be replaced). *Valid Values*: Registered class name. For a description of valid keys and their appropriate values, refer to the description of *WinRegisterObjectClass* in the Wo*rkplace Shell Programming Guide*.
`MediaSet=(`	Marks the beginning of a media set description. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`SetName=MediaSetName`	Media set name. *Valid Values*: A string without any spaces.
`Media=(`	Marks the beginning of a media description. *Valid Values*: No value associated with this keyword.
`Type=2`	Media type. *Valid Values*: `1` diskette (default) `2` CD `4` hard drive
`Capacity=48`	Media capacity (for Type=2 only). *Valid Values*: Any number.
`ClusterSize=34`	Cluster size (for Type=2 only). *Valid Values*: Any number.

`Description=CD`	Media description. *Valid Values*: Any string (can be omitted for `Type=1`)
`MediaPath=D:\`	Media path. *Valid Values*: Qualified path (can be omitted for `Type=1`).
`ReservedSpace=1000`	Reserved space on the media in bytes. *Valid Values*: Any string (can be omitted if `0`).
`MediaFinishedExit=(`	Marks the beginning of the description of the user exit to be run if there is no more space on the media. *Valid Values*: No value associated with this keyword.
`CallTime=20`	Indicates that the user exit processes the finished media case. *Valid Values*: `20`
`ExitType=1`	User exit type. *Valid Values*: `0` DLL (default) `1` EXE `2` CMD
`FilePath=DLLNAME.DLL`	User exit location. *Valid Values*: Fully qualified path name.
`ExitData=-pcmd1 -pcmd2`	User exit details. *Valid Values*: For exit type `0`, name of the procedure in DLL that is being called. For exit type `1`, EXE file parameters. For exit type `2`, CMD file parameters.
`SessionVisible=1`	Specifies if the session is visible or minimized (for `ExitType=1` or `2` only). *Valid Values*: 0 minimized (default) 1 visible to user
`AsciiFile=(`	Marks the beginning of an ASCII configuration file change description. This type of block can

	appear multiple times. *Valid Values*: No values associated with this keyword.
`CaseSensitive=1`	Determines if search for the entity to be updated is case sensitive. *Valid Values*: `0` case insensitive (default; can be omitted) `1` case-sensitive
`UseGrep=0`	If `grep` is to be used to search for the entity to be updated. *Valid Values*: `0` do not use `grep` `1` use `grep` (default; can be omitted)
`FileName=config.sys`	Configuration file name. *Valid Values*: Qualified file name.
`EditLine=Line To Edit`	Line to edit (for `Action=1` only). *Valid Values*: Any string.
`SearchLine=set path`	Line to search for `Action=0` and a substring in searched lines to scan for `Action=1`. *Valid Values*: Any string.
`NewLine=rem set path`	Line to replace string with. *Valid Values*: Any string.
`SearchFound=2`	Determines action to be performed if the string defined by `SearchLine` is found. *Valid Values*: `1` replace with the string defined by `NewLine` (default; can be omitted) `2` take no action
`SearchNotFound=2`	Determines action to be performed if the string defined by `SearchLine` is not found. *Valid Values*: `1` add new line to beginning of file `2` add new line to end of file `4` add before or after other lines `8` take no action

`AddBeforeLine=beforeline`	Line before which the line defined by SearchLine is to be inserted (for `SearchNotFound=4` only). *Valid Values*: Any string.
`AddAfterLine=afterline`	Line after which the line defined by SearchLine is to be inserted (for `SearchNotFound=4` only). *Valid Values*: Any string.
`Delimiter=*`	If the value is treated as a group of delimited substrings, the delimiter for these substrings (for `Action=1` only). *Valid Values*: Any string without spaces.
`SubstrChk=1`	Determines if the value is treated as a group of delimited substrings (for `Action=1` only). *Valid Values*: `0` no `1` yes
`DelimChk=1`	Determines if there is a delimiter after last entry (for `Action=1` only). *Valid Values*: `0` no `1` yes
`Action=1`	Configuration file editing action to perform. *Valid Values*: `0` replace a line with another line (default; can be omitted) `1` edit a selected line
`EditLineOcc=2`	Determines which occurrence of the string defined by `EditLine` in the configuration file is looked for (for `Action=1` only). *Valid Values*: `0` first (default; can be omitted) `1` last `2` every
`SrcLineOcc=1`	Determines which occurrence of the string defined by `SearchLine` in the configuration file is looked for.

	Valid Values: `0` first (default; can be omitted) `1` last `2` every
`AddBLineOcc=1`	Determines which occurrence of the string defined by `AddBeforeLine` in the configuration file is looked for (for SearchNotFound=4 only). *Valid Values*: `0` first (default; can be omitted) `1` last `2` every
`AddALineOcc=1`	Determines which occurrence of the string defined by `AddAfterLine` in the configuration file is looked for (for `SearchNotFound=4` only). *Valid Values*: `0` first (default; can be omitted) `1` last `2` every
`MatchPosition=2`	Determines action to be performed if the position to insert the string defined by `SearchLine` is found (for SearchNotFound=4 only). *Valid Values*: `1` replace with the string defined by `NewLine` (default; can be omitted) `2` take no action
`NoMatchPosition=2`	Determines action to be performed if the position to insert the string defined by `SearchLine` is not found (for `SearchNotFound=4` only). *Valid Values*: `1` add new line to beginning of file `2` add new line to end of file `8` take no action
`Os2PrfIni=(`	Marks the beginning of an OS/2 .INI configuration file change description. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.

`FileName=d:\path\os2.ini`	Configuration file name. *Valid Values*: Qualified file name.
`Application=ApplicationName`	Application whose OS/2 .INI file is to be changed. *Valid Values*: Any string.
`Key=KeyName`	Name of the key of which the value is to be changed for the application determined by the Application keyword. *Valid Values*: Any string without any spaces.
`Value=KeyValue`	Value to be set for the key determined by the Key keyword. *Valid Values*: Any appropriate value.
`IniType=1`	Set to `1` for the OS/2 .INI configuration files. *Valid Values*: `1` is the only valid value.
`WinOs2Ini=(`	Marks the beginning of a WIN-OS/2 .INI configuration file change description. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`FileName=d:\path\os2.ini`	Configuration file name. *Valid Values*: Qualified file name.
`Application=WinAppName`	Application whose WIN-OS/2 .INI file is to be changed. *Valid Values*: Any string.
`Key=WinKeyName`	Name of the key of which the value is to be changed for the application determined by the Application keyword. *Valid Values*: Any string without any spaces.
`Value=KeyValue`	Value to be set for the key determined by the Key keyword.

	Valid Values: Any appropriate value.
`IniType=2`	Set to `2` for the WIN-OS/2 .INI configuration files. *Valid Values*: `2` is the only valid value.
`File=(`	Marks the beginning of the description of the file to be transferred during installation. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`EAMediaIndex=0`	Indicates that the media supports extended attributes. *Valid Values*: Filled in automatically during packaging.
`Source=S:\srcpath\srcname`	File path of the source. *Valid Values*: Fully qualified file name.
`SourcePath=S:\srcpath`	Location of the source path. *Valid Values*: Qualified path name.
`SourceFileName=srcname`	File name of the source. *Valid Values*: Qualified file name.
`SourceChecksum=0`	Checksum for the file on the source. *Valid Values*: Set automatically.
`SourceEASize=0`	EA (Extended Attributes) size for the file on the source. *Valid Values:* Set automatically.
`CBFile=0`	Current number of bytes used for the file on the source. *Valid Values*: Set automatically.
`CBFileAlloc=0`	Current number of bytes allocated for the file on the source. *Valid Values*: Set automatically.

`AttrFile=0`	Not used.
`TargetPath=S:\targetpath`	File path of the target. *Valid Values*: Qualified path name.
`TargetFileName=targetname`	File name of the target. *Valid Values*: Qualified file name.
`MediaPath=mediapath`	File path on the media. *Valid Values*: Qualified path name.
`MediaFileName=srcname`	File name on the media. *Valid Values*: Qualified file name.
`MediaNumber=0`	Media number in the set. *Valid Values*: Number between 0 and n-1 where n is the number of media defined in the media set.
`Flags=0`	Not used.
`TargetAttrib=3`	Attributes on the file on the target. *Valid Values*: `0` - `7` in binary representation where the first digit stands for system file (`0` - off, `1` - on), second digit stands for hidden file (`0` - off, `1` - on), and third digit stands for read-only file (`0` - off, `1` - on). The values are: `0` regular file `1` read-only file `2` hidden file `3` hidden read-only file `4` system file `5` read-only system file `6` hidden system file `7` hidden read-only system file
`Restriction=(`	Marks the beginning of a media restriction for the file section. This type of block can appear multiple times. *Valid Values*: No value associated with this keyword.

`MediaSet=MyMediaSet`	The name of the media set for which the file is restricted to be placed. *Valid Values*: Any previously defined media set name.
`MediaNumber=1`	The ordinal number of the media in the media set determined by the `MediaSet` keyword. *Valid Values*: An ordinal number (starting with 1) of any media defined in the media set determined by the `MediaSet` keyword.
`Prerequisite=(`	Marks the beginning of the prerequisite description. This type of block can appear multiple times. *Valid Values*: No values associated with this keyword.
`FeatureID=PrereqFeature`	Feature ID of a prerequisite object. *Valid Values*: Any previously defined feature ID.

Part 2. Working with Rapid Deployment Tools

Chapter 5. Introducing Migration and Installation Scenarios

This chapter provides a quick overview over the different scenarios illustrated in this book. In all scenarios, identical system configuration images provided by donor systems are distributed to the target workstations. Only user-specific data is different in the migration scenarios. The set up and the configuration of the donor system is described in detail in Chapter 5.4, “Configuring the Donor System” on page 168.

The last part of this chapter deals with some things that appear in all migration scenarios. For example, the migration of Communications Manager/2 (CM/2) to eNetwork Personal Communications for OS/2 Warp (PCOMOS2) is illustrated here. The migration chapters, 8 and 9, refer to this chapter for these types of issues.

The Version-to-Version Tools are basically designed to help customers migrate their OS/2 workstations to the most actual OS/2 version: OS/2 Warp 4 with the latest FixPak (at the time this redbook was written, FixPak 5 was available). The book covers two migration scenarios:

- OS/2 V2.11 to OS/2 Warp 4
- OS/2 Warp 3 (OS/2 Warp Connect) to OS/2 Warp 4

The Version-to-Version Tools are OS/2-version independent which means that all kinds of migration scenarios are possible. The only prerequisite is that REXX is available on the workstations that are supposed to be migrated. Above mentioned migration scenarios were chosen because OS/2 V2.11 and OS/2 Warp 3.0 (all kinds of flavors) are the most common OS/2 versions installed at the customers’ site.

In addition, the Version-to-Version Tools can also be used for pristine installations. This scenario is illustrated in the our third one called "Pristine Installations".

5.1 Installing OS/2 Warp 4 without Migration

An image of the donor system is distributed to target workstations. After installing the image, several configuration steps with different tools are made in order to take care of unique IP addresses, workstation names, and so on. Using this scenario, a quicker deployment of workstations with identical hard- and software is possible rather than using the standard CID process. This scenario is illustrated in Chapter 7, “Pristine Installation of OS/2 Warp 4” on page 213.

5.2 Migrating from OS/2 Warp 3 to OS/2 Warp 4

The workstation-specific configuration data is extracted with the Version-to-Version tool, PROBE. This tool creates a Feature Installer object out of the retrieved data and transfers it to the software distribution server. Data that was not retrieved with the Probe tool, such as NET.ACC, is backed up with other tools and stored on the software distribution server, too.

After saving the configuration data, the donor system image is distributed in order to migrate to OS/2 Warp 4.

After the donor system image has been successfully distributed, the configuration data is applied to the new workstation. When this procedure has ended, the new workstation has the same configuration values as the old one and the user can continue working with it in same way as they did before with the old workstation.

This scenario is illustrated in Chapter 8, “Migrating from OS/2 Warp Connect (OS/2 Warp 3)” on page 243.

5.3 Migrating from OS/2 V2.11 to OS/2 Warp 4

In most aspects, this scenario is equal to the migration scenario from OS/2 Warp Connect (OS/2 Warp 3) to OS/2 Warp 4. Only some OS/2 V2.1-specific things may be different. This scenario is illustrated in Chapter 9, “Migrating from OS/2 V2.11” on page 289.

5.4 Configuring the Donor System

This section details the hardware and software configuration of the donor system we used in our scenarios. In addition, this section provides step-by-step guidance in setting up a maintenance partition on target workstations.

5.4.1 Hardware Configuration of the Donor System

The following hardware configuration was used in our scenarios:

- IBM Personal Computer 750
  - Pentium 166 MHz
  - 64 MB RAM
  - 512 KB Cache

    - 1,44 MB FDD
    - 1,6 GB HDD
    - 6x CD-ROM Drive
    - S3 Trio 64V+ Video
- Monitor IBM 15V

5.4.2 Software Configuration of the Donor System

The following software configuration was used in our scenarios:

- Operating System
    - OS/2 Warp Version 4
- Installed OS/2 Warp 4 components
    - All components available with OS/2 Warp 4, except IBM VoiceType
    - All components of the BonusPak
- Installed networking components
    - File & Print Client (includes LAN Requester)
    - TCP/IP services
    - NetWare Requester

Note on NetWare

We included the NetWare requester in OS/2 Warp Connect to cover the migration of the NetWare configuration in our scenarios.

When you want to install the NetWare requester in any workstation, you must set up a NetWare server first in order to be able to install a requester properly.

- System Upgrades
    - OS/2 Warp 4 Fixpak 5
    - Feature Installer Version 1.2.1 (prerequisite for RDT tools)
    - MPTS FixPak (product refresh) WR08415
    - OS/2 Warp 4 File and Print Services CSD IP08402
    - TCP/IP 4.1

Note on Local Administered Addresses

When using the process described in this book, no local administered address (LAA) should be provided with the MPTS configuration. The migration scenarios support including LAA addresses when one is found in the old system's configuration. When using the pristine installation scenario, you can provide the LAAs along with the other values, and it will be taken care of, too.

- Installed protocols on one physical token-ring adapter
  - IBM NetBIOS on logical adapter 0
  - TCP/IP on logical adapter 0
  - IEEE 802.2 on logical adapter 0
  - IBM NetWare Requester Support on logical adapter 0
  - IBM TCPBEUI on logical adapter 1
- Additional Software
  - Netscape Navigator Version 2.02 for OS/2
  - Lotus Notes 4.52
  - Lotus SmartSuite 96 for OS/2
  - eNetwork Personal Communications for OS/2 Warp (Full version, which includes SNA support)

TCP/IP Personal Communications

You may want to uninstall the lite version of Personal Communications shipped with the TCP/IP component in OS/2 Warp 4. To do so, refer to 5.4.7, “Deleting Unneeded System Components” on page 181.

  - Java Run-time Environment 1.1.4
  - NetView Distribution Manager/2 Client

Lotus Notes, Java 1.1.4 and Lotus SmartSuite are installed on drive D: instead of C:. Personal Communications is installed in C:\PCOMOS2 instead of C:\TCPIP\PCOMOS2. All other products are installed in their default paths.

For further information about how to install the products, especially the NVDM/2 Client, please refer to the product installation guides that come with the product.

Order of the NVDM/2 Redirected Drive Letters

NVDM/2 assigns redirected drive letters to the shared drives and directories in reversed alphabetically order (starts with Z and goes down to A). To be consistent with the drive letter assignment on the NVDM/2 Boot Diskettes, you need to change the order in the installation process. Select **Configure...** in the first window of the install process and change the **Free Drive Letters** from **ZYXWU**. . . to **ZYWXU**. . . (change the positions of X and W).

Then the SHAREA directory on the codeserver is always the redirected drive W. This is required to successfully work with our programs and REXX

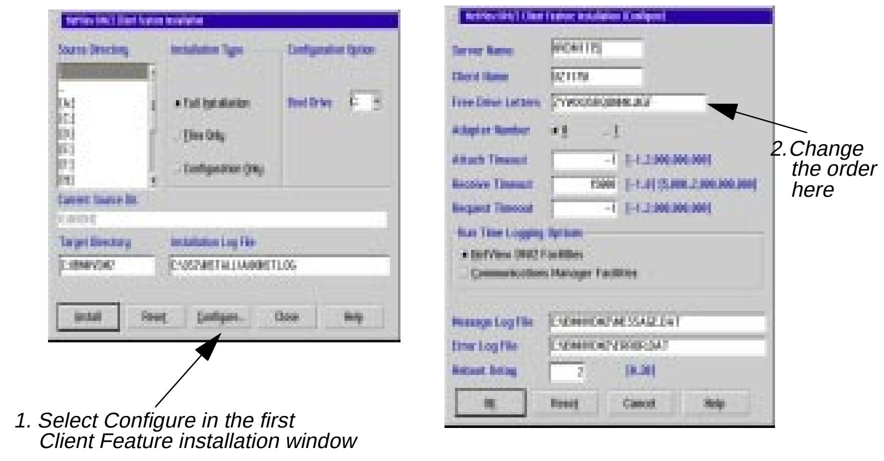


Figure 23. NVDM/2 Client Feature Installation Window

5.4.3 Setting Up the Donor System's Hard Drive

As shown in Figure 24 on page 172, we created the following partitions on the donor system:

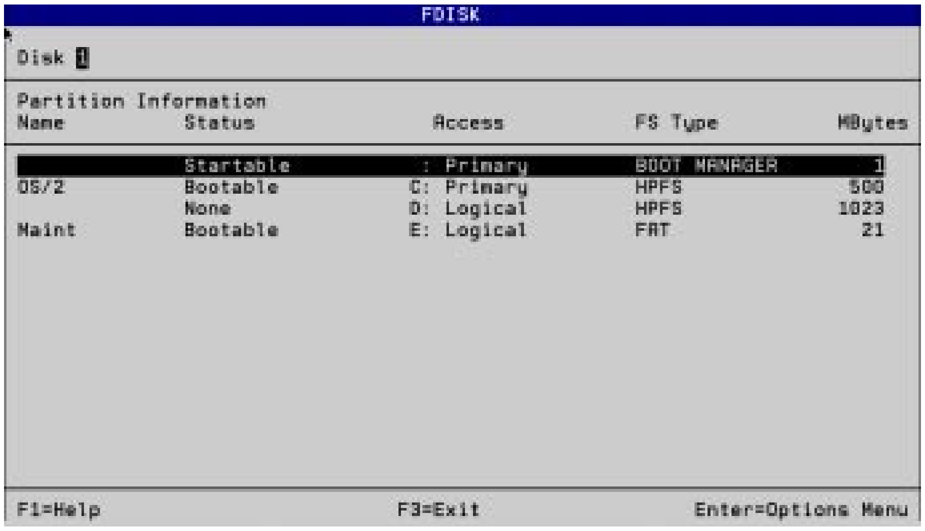
- Boot Manager as a primary partition
- C: primary partition with a size of 500 MB for OS/2 and the system components, HPFS-formatted.
- D: logical partition with a size of 1000 MB for applications and data, HPFS-formatted.
- E: logical partition with a size of 20 MB as a maintenance partition, FAT-formatted.

You should not make the C: partition smaller than 350 to 400 Megabytes. The OS/2 Warp 4 system and the different networking components need plenty of disk space. At run time, the swapfile and several log files are written so that additional disk space is needed assuming the swapper file resides on the C: partition.

The size for the maintenance partition needs to be at least four MB for a command line-based OS/2 system. Since additional network components are installed on this partition, we recommend creating a maintenance partition of 20 MB in order to avoid problems that deal with disk space limitations.

The size of the D: partition depends on applications you want to install and the amount left of the hard drive once the two primary partitions plus maintenance partition have been created.

In our scenarios, the C: partition was 500 MB in size. The E: partition was 20 MB of size. The rest of the available disk space is assigned as the D: partition.



```
                                  FDISK
Disk 1

Partition Information
Name         Status               Access         FS Type         MBytes

             Startable           : Primary       BOOT MANAGER         1
OS/2         Bootable          C: Primary        HPFS               500
             None              D: Logical        HPFS              1023
Maint        Bootable          E: Logical        FAT                 21

F1=Help                          F3=Exit            Enter=Options Menu
```

Figure 24. Donor System FDISK Partition Information

5.4.4 Installation Order

We recommend the following installation order to avoid problems with dependencies of different products.

1. OS/2 Warp 4 with all components desired
2. Fixpak 5 or 6 for OS/2 Warp 4

3. File and Print Services CSD IP08402
4. MPTS WR08415
5. Netscape Version 2.02 for OS/2
6. Feature Installer 1.2.1
7. Java Run-time Environment 1.1.4
8. TCP/IP 4.1
9. NVDM/2 Client
10. Any other products

Installing in this order makes sure you have all prerequisites for the program you want to install. For instance, Feature Installer 1.2.1 requires Netscape. Java 1.1.4 and TCP/IP 4.1 require Feature Installer 1.2.1.

Support of Different Machines in One Machine Class

If you plan to support different PC models in one machine class by using one donor system image, we recommend setting the donor system's video resolution to standard VGA. Otherwise, you might experience mal-functioning screen outputs at the target machines after the image has been distributed.

5.4.5 Manual Configuration Changes to the Donor System

The following changes have to be made manually in order to meet some requirements of the later deployment process:

1. Add the following lines to CONFIG.SYS

   ```
   PAUSEONERROR=NO
   SET RESTARTOBJECTS=STARTUPFOLDERSONLY
   ```

   When these values are set, a configuration error does nor require a user action at boot time, which usually occurs when target machine reboots with the donor system image because two or more machines would have the same addresses. This assures the configuration program, `CONFIGUR`, to run unattendedly after reboot.

2. Update the `PATH`, `LIBPATH`, and `DPATH` variables in CONFIG.SYS

To have the NVDM/2 Client properly running from the C: partition, you need to update the LIBPATH, DPATH and PATH statements.

Table 25. Updating the PATH Statements

Variable	Append values
LIBPATH	W:\DLL;
PATH	W:\EXE;
DPATH	W:\EXE;

If these values are not appended, most of our programs and REXX procedures cannot run on the NVDM/2 Client because needed files cannot be found on the redirected W: drive.

3. Create a Personal Communications host session named "SNA"

   If you plan to distribute eNetwork Personal Communications with host sessions over SNA rather than TCP/IP, we recommend creating a host session with the name of SNA.

   This is important when running the configuration program on the target machines after the donor system image has been distributed. This file will be replaced by a file of the same name that contains the CM/2 configuration of the old machine from which you migrate (see “Migrating from CM/2 to Personal Communications” on page 187).

   It doesn’t matter what this file contains. There are only three requirements:

   1. It must be a Personal Communications configuration file named SNA.WS.
   2. It must be stored in the default path (C:\PCOMOS2\PRIVATE).
   3. It must create an icon for this configuration file when you are asked by Personal Communications to do this.

   When these requirements are met, the configuration program has only to replace the SNA.WS file on the target machine, and the new configuration can be used.

4. Create or modify STARTUP.CMD

   The STARTUP.CMD file is used to start applications, such as LAN Requester or Personal Communications host sessions, automatically at boot time while the Workplace Shell is being loaded.

   In our scenarios, we used the one shown in Figure 25 on page 175.

```
NET START REQ
CDM START
START C:\PCOMOS2\PCSBAT.EXE "C:\PCOMOS2\PRIVATE\sessions.BCH" /R
LOGON /VD
EXIT
```

Figure 25. STARTUP.CMD

The STARTUP.CMD file shown before starts the LAN Requester, the NVDM/2 Client, and the configured Personal Communications host sessions. At the end, the LAN log on window is displayed.

5.4.6 Setting Up the Maintenance Partition

The maintenance partition is useful to run maintenance operations, such as FixPaks, ServicePaks, or Corrective Service Diskettes (CSDs), against the production operating system (in most cases installed on the C: partition) or to back up the operating system.

Normally, the maintenance partition has no Presentation Manger (PM) available, assuring no problems with possible locked files. It holds all files needed to boot the machine, to load LAN support, and to connect to a software distribution server, in our case, to the NVDM/2 Server.

We recommend setting up the maintenance partition after the production environment has been installed with the NVDM/2 Client up and running properly. The NVDM/2 Client on the production partition is needed to complete the installation of the maintenance partition.

The commands shown below assume the maintenance partition to be located on the E: drive.

To set up the maintenance partition, execute the following steps:

1. Run SEMAINT

SEMAINT Syntax

```
SEMAINT /S:<source_path>
        /T:<target_path>
        /B:<boot_drive>
         /L1:<log_file_name>
```

where:

`/S`:	Fully qualified source path to the OS/2 diskette images where X: is the drive letter of your CD-ROM or redirected drive
`/T`:	Fully qualified target path. The maintenance OS/2 system will be installed under this directory. The specified drive must be a local drive.
`/B`:	Target boot drive. The drive from which the OS/2 maintenance system will boot. This drive must be a local drive, too.
`/L1`:	Fully qualified path and name of the log file. This directory must already exist.

For example:

```
SEMAINT /S:X:\OS2IMAGE /T:E:\OS2 /B:E:\ /L1:C:\LOG
```

2. Run `THINLAPS`

   The `THINLAPS` utility is located in the IBMCOM subdirectory.

THINLAPS Syntax

`THINLAPS` *<source_path> <target> <nif_file>*

where:

<source_path>	Fully qualified name of the source path, where the MPTS install images are located.
<target>	Target drive.
<nif_file>	Network Adapter Driver NIF file name. The name of the NIF file that corresponds with the installed network adapter.

For example:

```
THINLAPS X:\CID\IMG\MPTS E:\ IBMTOK.NIF
```

3. Run `COPYNVDM.CMD`

   The `COPYNVDM` utility is located on the CD-ROM shipped with this redbook (see “COPYNVDM.CMD” on page 177).

COPYNVDM Syntax

`COPYNVDM` *<source_drive> <target_drive> <OS/2_version>*

where:

<source_drive>	Drive where the installed NVDM/2 Client is located (usually the C: partition).

<target_drive>	Drive where the NVDM/2 Client code is copied to (maintenance partition).
<OS/2_version>	The version number of OS/2 you are using on the target partition. Valid values are: 211, 3 and 4. This parameter is necessary, because the correct DLLs need to be copied for REXX support.

All parameters are required to run the program successfully.

5.4.6.1 COPYNVDM.CMD

The purpose of the COPYNVDM procedure is to copy the NVDM/2 Client code from a source partition to the target partition and to adjust CONFIG.SYS and IBMNVDM2.INI on the target drive. Normally, when you install the NetView DM/2 client on the maintenance partition (target drive) with a NetView DM/2 client installed on your production partition (source drive), the files of the NVDM/2 Client on the production partition will be moved to the maintenance partition. Another function of the procedure is to copy all necessary REXX DLLs to the maintenance partition (target drive).

This program interacts with the CUBE utility shipped with this book. For information about the CUBE utility, please refer to Appendix D, “CUBE Source Code and Documentation” on page 407. The CUBE profiles (see below) are built by COPYNVDM.CMD at execution time.

The changes to the CONFIG.SYS and IBMNVDM2.INI are made with the two profiles shown below.

```
AddLine "DEVICE=E:\IBMNVDM2\BIN\anxifpid.sys" ( IFNEW AFTER
AddLine "DEVICE=E:\IBMNVDM2\BIN\anxifcom.sys" ( IFNEW AFTER
AddLine "IFS=E:\IBMNVDM2\BIN\anxifcom.ifs" ( IFNEW AFTER
AddLine "DEVICE=E:\IBMNVDM2\BIN\ANXACAIP.SYS" ( IFNEW AFTER
AddLine "RUN=E:\OS2\BOS2REXX.EXE" ( IFNEW AFTER
RepString "E:\OS2\CMD.EXE" WITH "E:\IBMNVDM2\BIN\ANXCMCLC.EXE" IN
                                                    "SET OS2_SHELL"
AddString ";E:\IBMNVDM2\DLL;W:\DLL;"  IN "LIBPATH" ( IFNEW AFTER
AddString ";E:\IBMNVDM2\BIN;W:\EXE;W:\CMD;"  IN "SET PATH"
                                                 IFNEW AFTER
AddString ";E:\IBMNVDM2;W:\EXE;W:\CMD;"  IN "SET DPATH" ( IFNEW AFTER
RepString ";;" WITH ";" ( ALL
```

Figure 26. CUBE Profile CONFIG.PRO

Note: All lines are single-line statements.

```
RepString "C:" WITH "E:" ( ALL
RepString "ZYXWVUTSRQPONMLKJIHGF" WITH "ZYWXVUTSRQPONMLKJIHGF" ( ALL
```

Figure 27. CUBE Profile IBMNVDM2.PRO

The CONFIG.PRO CUBE profile updates the PATH, DPATH and LIBPATH statements in the CONFIG.SYS file of the maintenance partition. In addition, additional path extensions are made for the new NVDM/2 Client and several CMD and DLL files that reside on the NVDM/2 Server. The PROTSHELL statement is modified to start the NVDM/2 Cient at boot time and additional DEVICE and IFS statements are added for the NVDM/2 Client. Finally the statement to run BOS2REXX.EXE is added. This program assures that REXX is available.

The IBMNVDM2.PRO CUBE profile changes the order of drive letters in the IBMNVDM2.INI file so that the NVDM/2 Client's drive letter assignments are consistent with the drive letters assigned when the target workstation is booted up using the set of boot diskettes (see "Creating Boot Diskettes with NetView DM/2 Client" on page 129).

COPYNVDM.CMD is a plain REXX command file. You can easily modify the file to meet your own purposes. The following figure displays the COPYNVDM REXX command file.

```
/* REXX cmd file that copies the NVDM/2 software distribution          */
/* agent to a maintenance partition and creates a IBMNVDM2.INI file    */
/* with the correct workstation name and path statements.              */
/* In addition, the program copies REXX support to the maintenance     */
/* partition and updates the CONFIG.SYS of the maintenance partition   */
/* and other necessary files.                                          */
/*                                                                     */
/* BOS2REXX EXE                                                        */
/* REXX     211, REXX     3, REXX     4                                */
/* REXXAPI  DLL, REXXAPI  3, REXXAPI  4                                */
/* REXXINIT 211, REXXINIT 3, REXXINIT 4                                */
/* CUBE     CMD                                                        */
/* CONFIG   PRO                                                        */
/* IBMNVDM2 PRO                                                        */
/* XCOPY    EXE                                                        */

PARSE UPPER ARG source target version .

RCode = 0
'ECHO OFF'
'ECHO COPYNVDM >> output.txt'

IF (source == '') | (target == '') | (version == '') then do
   say 'INCORRECT PARAMETERS !'
   say 'Syntax : COPYNVDM <sourcedrive> <targetdrive> <OS/2version>'
   say ''
   say 'where:'
   say '<sourcedrive>  is the drive where the installed NVDM/2 Client'
   say '               is located.'
   say '<targetdrive>  is the drive where the maintenance NVDM/2'
   say '               client is being copied to.'
   say '<OS/2version>  is the Version of OS/2 you use on the target'
   say '               machine.'
   say '               Valid values are 211, 3 and 4.'
   say ''
   say 'Example: COPYNVDM C: E: 4'
   RCode = 4
end

if length(source) == 1 then source = source||':'
if length(target) == 1 then target = target||':'

if RCode == 0 then do
   if (version == '4') | (version == '3') | (version == '211') then NOP
   else do
      say 'OS/2 Version is not valid !'
      say 'Allowed values are 4, 3 and 211'
      RCode = 4
   end /* do */
end /* do */
if RCode == 0 then do
   say 'COPYNVDM is working. One Moment please...'
   'CD 'target||'\OS2'
   RCode = RC
   if RCode <> 0 then do
      say 'OS2 subdirectory not found on <targetdrive> 'target
      RCode = 8
   end /* do */
end
```

Figure 28. COPYVNVDM.CMD (Part 1 of 3)

```
CALL buildprofile
if RCode == 0 then do
   'XCOPY 'source||'\IBMNVDM2\*.* 'target||'\IBMNVDM2\*.* /s /e >> output.txt'
   if RCode <> 0 then do
      say 'NVDM/2 Client not found on the C: partition !'
      RCode = 8
   end
end

if RCode == 0 then do
   'XCOPY BOS2REXX.EXE 'target||'\OS2 >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXX.'||version' 'target||'\OS2\REXX.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXXINIT.'||version' 'target||'\OS2\REXXINIT.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXXAPI.'||version' 'target||'\OS2\REXXAPI.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   if RCode <> 0 then do
      say 'Copy of the REXX Support File failed!'
      RCode = 8
   end
end

if RCode == 0 then do
   RC = STREAM(target||'\IBMNVDM2\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
   if RC == '' then RCode = 1
   RC = STREAM(target||'\CONFIG.SYS', 'C', 'QUERY EXISTS')
   if RC == '' then RCode = 1
   if RCode <> 0 then do
      say target||'\CONFIG.SYS or 'target||'\IBMNVDM2\IBMNVDM2.INI'
      say 'were not found!'
      RCode = 8
   end
   if RCode == 0 then do
      cubearg = 'ibmnvdm2.pro 'target||'\IBMNVDM2\IBMNVDM2.INI'
      RC = 'cube.cmd'(cubearg)
      if RC <> 0 then RCode = RC
      cubearg = 'config.pro 'target||'\CONFIG.SYS'
      RC = 'cube.cmd'(cubearg)
      if RC <> 0 then RCode = RC
      if RCode <> 0 then do
         say 'One or more errors were found while making the changes'
         say 'to E:\CONFIG.SYS and E:\IBMNVDM2\IBMNVDM2.INI'
         RCode = 8
      end
   end
end
if RCode == 0 then do
   say ''
   say 'COPYNVDM ended successfully'
end /* do */
else do
   say ''
   say 'COPYNVDM ended unsucessfully with return code = 'RCode
end /* do */
'DEL OUTPUT.TXT'
'ECHO ON'
return RCode
```

Figure 29. COPYNVDM.CMD (Part 2 of 3)

```
buildprofile:
RC = STREAM('CONFIG.PRO', 'C', 'QUERY EXISTS')
if RC <> '' then 'DEL CONFIG.PRO'
RC = STREAM('IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
if RC <> '' then 'DEL IBMNVDM2.PRO'

cubeline = 'RepString "'source'" WITH "'target'" ( ALL'
'ECHO 'cubeline' >>IBMNVDM2.PRO'
cubeline = 'RepString "ZYXWVUTSRQPONMLKJIHGF" WITH "ZYWXVUTSRQPONMLKJIHGF" ( ALL'
'ECHO' cubeline' >>IBMNVDM2.PRO'
Call Stream 'IBMNVDM2.PRO','c','close'

cubeline = 'AddLine "DEVICE='target||'\IBMNVDM2\BIN\anxifpid.sys" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddLine "DEVICE='target||'\IBMNVDM2\BIN\anxifcom.sys" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddLine "IFS='target||'\IBMNVDM2\BIN\anxifcom.ifs" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddLine "DEVICE='target||'\IBMNVDM2\BIN\ANXACAIP.SYS" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddLine "RUN='target||'\OS2\BOS2REXX.EXE" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'RepString "'target||'\OS2\CMD.EXE" WITH "'
cubeline2 = target||'\IBMNVDM2\BIN\ANXCMCLC.EXE" IN "SET OS2_SHELL"'
'ECHO' cubeline||cubeline2' >> CONFIG.PRO'
cubeline = 'AddString ";'target||'\IBMNVDM2\DLL;W:\DLL;"  IN "LIBPATH" ( IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddString ";'target||'\IBMNVDM2\BIN;W:\EXE;W:\CMD;"  IN "SET PATH" (
IFNEW AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'AddString ";'target||'\IBMNVDM2;W:\EXE;W:\CMD;"  IN "SET DPATH" ( IFNEW
AFTER'
'ECHO' cubeline' >> CONFIG.PRO'
cubeline = 'RepString ";;" WITH ";" ( ALL'
'ECHO' cubeline' >> CONFIG.PRO'
Call Stream 'CONFIG.PRO','c','close'
RETURN
```

Figure 30. COPYNVDM.CMD (Part 3 of 3)

COPYNVDM.CMD Return Codes:

0 Successful program execution.

4 One or more arguments given by the user are not valid, or one or more arguments are missing.

8 A target directory was not found or a copy operation was unsuccessful.

5.4.7 Deleting Unneeded System Components

By default, the OS/2 installation process installs several components the user often does not or need, such as ART (Active Registration Tool) and the Ultimail component of TCP/IP.

These features can be selectively deleted with the trimmer utility. The following figure shows a complete list of the components you can delete with trimmer . For information about the trimmer utility, refer to “The Trimmer Tool” on page 108.

Name	Approximate size in bytes
Axtive Registration Tool	428,000
Multimedia	14,567,832
Non-default base bitmaps	4,813,564
TCPIP - PCOM	9,889,636
TCPIP - UMAIL	3,186,295
Language, codepages, and locales	4,287,201
LANReq (IBMLAN)	4,170,824
MAC (LAN) drivers (IBMCOM)	3,281,383
SCSI, CD-ROM, PCMCIA device drivers	989,215
Online books	14,937,186
Speech (VoiceType)	21,319,569
OS/2 Warp 4 tutorial	4,346,726
Coaches	3,015,109
BonusPak	27,257,221
Java Runtime and Toolkit	2,490,763
Miscellaneous TCPIP files	5,420,062
Miscellaneous files	3,298,015

Figure 31. Trimmable Components

The trimmer does a complete uninstall. It deletes the files, corresponding CONFIG.SYS entries, and corresponding Workplace Shell objects.

Trimmer Utility, W4TRIM, Syntax

```
W4TRIM [/T] [/C:W4TRIM.CTL]
```

where:

`/T` Specifies to trim the files. If `/T` is not provided, the utility only writes a log file, \OS2\INSTALL\W4TRIM.LOG, listing the files that would have been deleted, and warnings for any locked files. The following message is displayed:

```
Press Enter to continue, Ctrl-Break to terminate
```

`/C:W4TRIM.CTL` Specifies the control file listing the components that can be trimmed. The supplied file is W4TRIM.CTL, but you can rename it when you edit the file to specify the trim components.

For example:

```
W4TRIM /T /C:DONORTRIM.CTL
```

A log file named W4TRIM.LOG is automatically generated in the <bootdrive>:\OS2\INSTALL directory. This file contains a bundle of entries for each trimmed component, listing the name of the component, the affected files and the size of the files.

For example, TCP/IP-Ultimail is to be trimmed from the donor system by using the trimmer utility. Follow the steps below:

1. Change to the directory C:\TRIMMER (or where you have the trimmer utility installed).
2. Modify the W4TRIM.CTL. Change the following entry:

   ```
   UltimailDelete = keep
   ```

   to

   ```
   UltimailDelete = trim
   ```

3. Issue the following command:

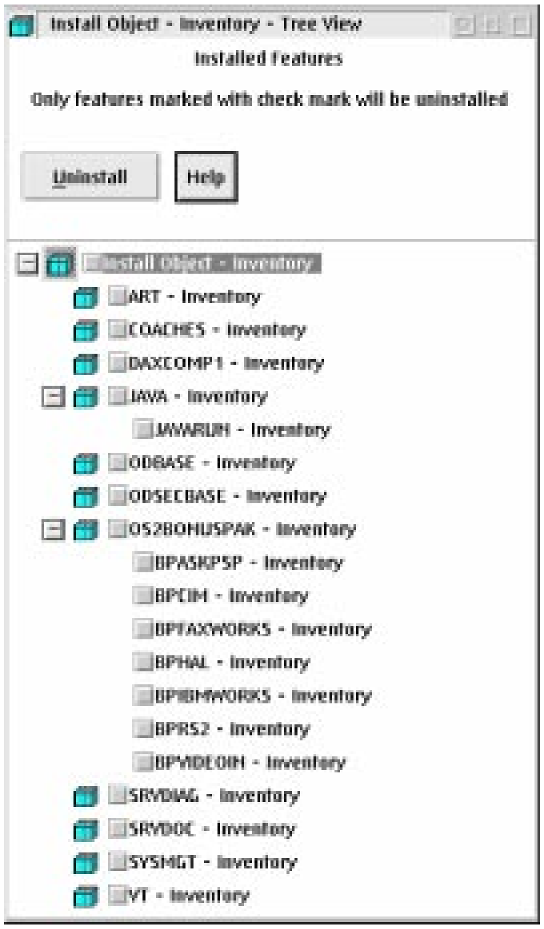
   ```
   W4TRIM /T /C:W4TRIM.CTL
   ```

4. Follow the directions on the screen.

After this process, Ultimail and all its corresponding components, including Workplace Shell objects, shadowed objects, and so on, are uninstalled.

5.4.8 Deleting Unwanted Components Using Feature Installer

Feature Install objects are stored in the \OS2\INSTALL\Installed Features directory. Once you click on the **Installed Features** folder and then **Install Object - Inventory**, the InstallObject - Inventory window is displayed as shown in Figure 32 on page 184.



Installed Features

Only features marked with check mark will be uninstalled

- Install Object - Inventory
  - ART - Inventory
  - COACHES - Inventory
  - DAXCOMP1 - Inventory
  - JAVA - Inventory
    - JAVARUN - Inventory
  - ODBASE - Inventory
  - ODSECBASE - Inventory
  - OS2BONUSPAK - Inventory
    - BPASKPSP - Inventory
    - BPCIM - Inventory
    - BPFAXWORKS - Inventory
    - BPHAL - Inventory
    - BPIBMWORKS - Inventory
    - BPRS2 - Inventory
    - BPVIDEOIN - Inventory
  - SRVDIAG - Inventory
  - SRVDOC - Inventory
  - SYSMGT - Inventory
  - VT - Inventory

Figure 32. Install Object - Inventory Window

All features installed through Alternate Software Distribution (ASD) will be placed here and represented as an object. Uninstalling a feature is quite simple. Just mark the object's checkbox and click on the **Uninstall** button. For example, if you want to remove the OS/2 Warp 4 registration tool called ART, mark the representing object's checkbox and press the **Uninstall** button.

5.4.9 Creating an Image of the Donor System

To distribute the donor system to target workstations, you need to create an image of the donor system and store the image on the software distribution server. The SYSCOPY utility out of the Version-to Version tools creates an image of the donor system. For more information about the SYSCOPY tool, refer to Chapter 1.1, “The SYSCOPY Tool” on page 3.

You need to boot the donor system from the maintenance partition to avoid problems with locked files in the production environment where Presentation Manager is up and running.

After booting the maintenance partition the NVDM/2 Client (which is automatically started) is waiting for requests. Distribute an OS/2 command prompt from the NVDM/2 Server to the donor system. To do so, use the IBM.WARP4.CMD.REF.4 package on the CD-ROM shipped with this redbook. This change file installs the OS/2 Warp 4 command line interpreter, meaning the NetView DM/2 Client will open a command prompt session at the donor system.

Note on OS/2 V2.11 Systems

On systems with OS/2 V2.11 installed, you cannot "install" an OS/2 Warp 4 command line session. In this case, you need to "install" the IBM.OS2211.CMD.REF.1 package located on the CD-ROM shipped with this redbook to get an OS/2 command prompt session.

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.WARP4.CMD.REF.4
     Description="OS/2 Warp 4 Command Prompt"
End

Section Install
Begin
      Program =  SA:\IMG\WARP4\CMD.EXE
End
```

Figure 33. IBM.WARP4.CMD.REF.4 Profile for NetView DM/2

The tools are located in the path W:\EXE (D:\SHAREA\EXE on the NVDM/2 Server). Change to this directory and type the following command:

SYSCOPY Syntax

```
SYSCOPY /C:<class_ID>
        /D:<drives_to_bundle>
         /B:<bundle_file_dir>
```

where:

`/C:` Specifies the *<class_ID>* which represents a name to identify the bundle files after they are created and stored on the code server.

`/D:` Specifies the partitions to bundle. You need to specify the drive letters without spaces or semicolons.

`/B:` Specifies the code server directory to store the created bundle files. You need to create this directory prior to execution.

For example:

```
SYSCOPY /C:IBMPC750 /D:CDE /B:W:\IMG\FINANCE
```

INFOZIP

The INFOZIP-Utility Version 2.1 or later must be located in the same directory as the SYSCOPY utility or in any other path specified in the PATH statement!

The SYSCOPY utility creates a bundle file for each partition specified in the /D: parameter and stores it in the directory specified in the /B: parameter.

After the procedure has finished the donor system image is available on the NVDM/2 Server for distributing using the REPLICAT tool.

Chapter 6. Special Migration Steps Not Covered by RDT

This chapter informs about migration issues that are not yet handled by the Version-to-Version Tools and Enterprise Rescue Tools. All illustrations here are based on our experiences that were gained during the project.

6.1 Migrating from CM/2 to Personal Communications

In actual installations, eNetwork Personal Communications for OS/2 (PCOMOS2) substitutes the Communications Manager/2 (CM/2) in many cases. However, the user wants to keep his/her host session definitions. Therefore, a migration from the CM/2 configuration to a PCOMOS2 configuration is necessary.

CM/2 stores its configuration data in a binary file, with the extension of .CFG. The first migration step is to extract the binary data into an ASCII file in order to access the needed data.

This can be done with the CMRECORD utility. This program is shipped with CM/2 and is located in the \CMLIB directory. It extracts all configuration data from a specified .CFG file into a ASCII-coded response file. Now the needed data needs to be extracted from this response file and written to a PCOMOS2 configuration file.

CMRECORD Syntax

```
CMRECORD <config_file>
          [/O:<output_file>]
          [/M:<model_response_file>]
          [/K:<keyword1> <keyword2> <...>]
           [/I:<network_ini_file>]
```

where:

`/O:`	Optional. Specifies output file.
`/M:`	Optional. Specifies to record keywords in model file.
`/K:`	Optional. Specifies to only record keywords given.
`/I:`	Optional. Specifies ISDN or SNA phone connect file.

CMRECORD /D records the default configuration file. If /O: is not used, CMRECORD creates an output file with the same name as the configuration file and a

.RSP extension. `/M:` and `/K:` are mutually exclusive. By default, all features are recorded.

For example:

```
CMRECORD myconfig /O output.rsp
```

On the CD-ROM we included a REXX program, `CM2PCOMM.CMD`, that extracts the data from a given CM/2 configuration file and creates a PCOMOS2 configuration file.

In addition, multiple PCOMOS2 host sessions file are created, which starts the same number of 3270 sessions as there were configured in the CM/2 configuration file.

These two files are stored in the W:\IMG\CONFIG\ *<workstationname>* directory. The W: drive is the redirected SHAREA directory of the NVDM/2 Server. In this directory, all information about the workstation will be stored (see the chapters about migration from OS/2 2.11 and OS/2 Warp 3 to OS/2 Warp 4). <workstationname> is the client name you defined in NVDM/2 for this specific workstation.

To use `CM2PCOMM`, type the following command:

CM2PCOMM Syntax

`CM2PCOMM` *<CM/2 install path>* [*<Name of the Configuration file>*]

where:

<CM/2 install path>	Specifies the fully qualified path to the CM/2 install directory.
<Name of the Configuration file>	Specifies the file name of the configuration you want to extract. It is not necessary to specify the extension. When this parameter is not omitted, the default CM/2 configuration is extracted.

CM/2 Values can be Incompatible With PCOMOS2

Personal Communications expects hexadecimal values for the identifier (XID in CM/2), and the gateway address (Destination Address in CM/2), CM/2 doesn't.

It is possible that the values are not compatible between the two programms and cannot be migrated. In this case, `CM2PCOMM.CMD` would notify you. Another possibility is that the old definitions may not work with PCOMOS2. You should test the extracted information with one test machine before you are going to migrate all workstations.

When one of these cases becomes true, your host administrator needs to create new definitions that are compatible with PCOMOS2. In this case, use the `MAKECONF.CMD` program (see 6.1.2, "MAKECONF.CMD" on page 196).

6.1.1 CM2PCOMM.CMD

`CM2PCOMM` is a plain REXX program that you can easily modify to meet your own purposes. The following figure displays the REXX command file:

```
/****************************************************************************/
/*                                                                          */
/* REXX Procedure that migrates CM/2 configuration to a PCOMM configuration */
/*                                                                          */
/****************************************************************************/
parse arg cmpath workstationname cmconfigfile
/****************************************************************************/
/* Variables Listing                                                        */
/*                                                                          */
/* RCode              Return code CM2PCOMM gives back to the invoking program */
/* RC                 Return code CM2PCOMM receives from programs it invokes */
/* workstationname    NVDM/2 name of the workstation                        */
/* cmconfigfile       Name of the CM/2 configuration file                   */
/* cmpath             Fully qualified path in which CM/2 is installed       */
/* drive              Driveletter + colon extracted from cmpath             */
/* path               Path extracted from cmpath without leading backslash  */
/* slashpos           Position of the first backslash in cmpath             */
/* point              Position of the point in cmconfigfile                 */
/* configname         Name of the CM/2 configurationfile without the point  */
/*                    and the extension                                     */
/* namelength         Length of cmconfigfile                                */
/* rsp_file           Specifies the cmpath and the configname with          */
/*                    extension .RSP                                        */
/* session3270        Counter for the difines 3270 Sessions in the CM/2     */
/*                    response file                                         */
/* session5250        Counter for the difines 5250 Sessions in the CM/2     */
/*                    response file                                         */
/* type1              Counter for type1 sessions (Terminal Sessions)        */
/* type2              Counter for type2 sessions (Printer Sessions)         */
/* localcp            Counter for the LOCAL_CP section in the CM/2 response */
/*                    file                                                  */
/* logicallink        Counter for the LOGICAL_LINK Section in the CM/2      */
/*                    response file                                         */
/* argument           Specifies the part of the line read from the RSP file */
/*                    before the '=' Sign                                   */
/* wert               Specifies the part of the line read from the response */
/*                    file after the '=' sign                               */
/* PU                 Specifies the PU value                                */
/* nodeid             Specifies the NODE_ID value                           */
/* address            Specifies the DESTINATION_ADDRESS value               */
/* tmpPU              Specifies a temporary PU value                        */
/* trash              Variable to store unneeded information from the parse */
/*                    command                                               */
/* numberOfSessions   Sstores the number of the defined 3270 sessions in the */
/*                    CM/2 response file                                    */
/* destpath           Drive and path on the code server where the PCOMOS2   */
/*                    files                                                 */
/*                    are copied                                            */
/* i                  Loop counter                                          */
/****************************************************************************/
'ECHO OFF'
RCode = 0
/* destpath = 'W:\IMG\CONFIG\'||workstationname */
destpath = 'D:\TOOLS\CM2PCOMM\'workstationname
logpath = 'X:\LOG\'||workstationname||'\'

say 'Checking the Arguments .... '
```

Figure 34. CM2PCOMM.CMD (Part 1 of 7)

```
if cmpath <> '' then do
    slashpos=pos('\',cmpath)
    drive=substr(cmpath,1,slashpos-1)
    path=substr(cmpath,slashpos+1,length(cmpath))
    RC = STREAM(cmpath||'\CMRECORD.EXE', 'C', 'QUERY EXISTS')
    if RC == '' then do
       say ''
       say 'The specified directory does not exist'
       say 'or CM/2 not found in the specified directory...'
       call argerror
    end /* do */
end /* do */
else do
   say ''
   say 'CM/2 install path was not specified ...'
   call argerror
end /* do */

if RCode == 0 then do
    if (cmconfigfile <> '')then do
       cmconfigfile=strip(cmconfigfile)
       point=lastpos(".", cmconfigfile)
       if point <> 0 then do
          configname=substr(cmconfigfile,1,point-1)
       end /* do */
       else do
          configname = cmconfigfile
          cmconfigfile = cmconfigfile||'.cfg'
       end /* do */
       namelength=length(configname)
       if namelength > 8 then do
          say ''
          say 'The specified CM/2 configuration file is not valid ...'
          call argerror
       end /* do */
    end /* do */
    else do
       say ''
       say 'CM/2 configuration file was not specified'
       say 'the default configuration file will be extracted...'
    end /* do */
    if cmconfigfile <> '' then do
       'DIR 'cmpath||'\'||cmconfigfile' >> output.txt'
       if RC <> 0 then do
          say ''
          say 'The specified CM/2 configuration file does not exist ...'
          call argerror
       end /* do */
    end /* do */
end /* do */

if RCode == 0 then do
   say 'Arguments OK ....'
   say ''
end /* do */
```

Figure 35. CM2PCOMM.CMD (Part 2 of 7)

```
if RCode == 0 then do
   say 'Extracting CM/2 Configuration ....'
   if cmconfigfile <> '' then do
      ' 'cmpath||'\CMRECORD 'cmpath||'\'||configname' >> output.txt'
      if RC <> 0 then do   end /* do */
         say 'Extraction of the configuration failed !!!!!!!!!!!!!!!'
         RCode = 8
      end /* do */
      else do
         say 'Configuration is being extracted ....'
         say ''
      end
   end /* do */
   else do
      ' 'cmpath||'\CMRECORD /D >> output.txt'
      if RC <> 0 then do   end /* do */
         say 'Extraction of the configuration failed !!!!!!!!!!!!!!!'
         RCode = 8
      end /* do */
      else do
         say 'Configuration is being extracted ....'
         say ''
         found = 0
         do while found == 0
            line.linenum = Linein(output.txt)
            if substr(line.linenum,1, 26) == 'Recording of configuration' then do
               cmconfigfile = word(line.linenum, 4)
               slashpos = lastpos("\", cmconfigfile)
               configname = substr(cmconfigfile, slashpos+1, length(cmconfigfile))
               configname = strip(configname)
               found = 1
               CALL Lineout(output.txt)
            end /* do */
         end
      end
   end /* do */
end /* do */

if RCode == 0 then do
   say 'Retrieving configuration values ....'
   rsp_file = cmpath||'\'||configname||'.rsp'
   session3270 = 0
   session5250 = 0
   type1 = 0
   type2 = 0
   localcp = 0
   logicallink = 0
   argument = ''
   wert = ''
   PU = ''
   nodeid = ''
   address = ''
```

Figure 36. CM2PCOMM.CMD (Part 3 of 7)

```
  do while lines(rsp_file)
       line.linenum = Linein(rsp_file)
       if session5250 == 0 then do
          select
            when line.linenum == '3270_SESSION = (' then session3270 = session3270 + 1
             when line.linenum == '     SESSION_TYPE = 1' then do
                type1 = type1 + 1
             end /* do */
            when line.linenum == '5250_SESSION = (' then session5250 = session5250 + 1
             when line.linenum == '     SESSION_TYPE = 2' then type2 = type2 + 1
          otherwise NOP
          end  /* select */
       end /* do */
       if (logicallink < 2) | (localcp < 2) then do
          select
             when line.linenum == 'LOCAL_CP = (' then localcp = localcp + 1
            when line.linenum == 'LOGICAL_LINK = (' then logicallink = logicallink + 1
          otherwise NOP
          end  /* select */
          parse VALUE line.linenum with argument '=' wert
          select
             when strip(argument) == 'NAME' then do
                PARSE VALUE wert with trash '.' tmpPU
                if tmpPU <> '' then PU = tmpPU
                PU = strip(PU)
             end /* do */
             when strip(argument) == 'NODE_ID' then do
                nodeid = strip(substr(wert,5,length(wert)))
                nodeid = strip(nodeid)
             end /* do */
             when strip(argument) == 'DESTINATION_ADDRESS' then do
                address = strip(wert)
             end /* do */
          otherwise NOP
          end  /* select */
       end /* do */
    end /* do */
    type1 = type1 - type2
    numberOfSessions = type1
 end /* do */

 if RCode == 0 then do
    do i = 1 to (length(nodeid))
       rc = verify(substr(nodeid,i,1),'0123456789ABCDEF')
       if RC <> 0 then do
          RCode = 12
          say ''
          say '*******************************************************************'
          say '*                                                                 *'
          say '* The CM/2 values are not compatible with Personal Communications ! *'
          say '*                                                                 *'
          say '* PCOMOS2 requires hexadecimal values. Please contact your         *'
          say '* administrator to receive valid definitions.                      *'
          say '*                                                                 *'
          say '* For further information refer to the CM/2 and PCOMOS2 mauals.    *'
          say '*                                                                 *'
          say '*******************************************************************'
          say ''
```

Figure 37. CM2PCOMM.CMD (Part 4 of 7)

```
 end /* do */
   end /* do */
end /* do */

if RCode == 0 then do
   say 'Configuration values are being retrieved ...'
   say ''
end /* do */

if RCode == 0 then do
   curdir = DIRECTORY()
   RC = DIRECTORY(destpath)
   CALL DIRECTORY curdir
   if RC == '' then 'MD 'destpath

   say 'Building PComm Files ....'
   RC = STREAM(destpath||'\SNA.WS', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'destpath||'\SNA.WS'
   RC = STREAM(destpath||'\SNA.BCH', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'destpath||'\SNA.BCH'

   'ECHO [Profile] >> 'destpath||'\SNA.WS'
   'ECHO ID=WS >> 'destpath||'\SNA.WS'
   'ECHO Description= >> 'destpath||'\SNA.WS'
   'ECHO [Translation] >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultView=Y >> 'destpath||'\SNA.WS'
   'ECHO DefaultView= >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultDBCS=Y >> 'destpath||'\SNA.WS'
   'ECHO DefaultDBCS= >> 'destpath||'\SNA.WS'
   'ECHO [Communication] >> 'destpath||'\SNA.WS'
   'ECHO AutoConnect=Y >> 'destpath||'\SNA.WS'
   'ECHO Link=slan >> 'destpath||'\SNA.WS'
   'ECHO Session=3270 >> 'destpath||'\SNA.WS'
   'ECHO [SLAN] >> 'destpath||'\SNA.WS'
   'ECHO Adapter=0 >> 'destpath||'\SNA.WS'
   'ECHO DestinationSAP=04 >> 'destpath||'\SNA.WS'
   'ECHO GatewayAddress='||address' >> 'destpath||'\SNA.WS'
   'ECHO Identifier=061'||nodeid' >> 'destpath||'\SNA.WS'
   'ECHO LinkStations=1 >> 'destpath||'\SNA.WS'
   'ECHO ReceiveBufferCount=12 >> 'destpath||'\SNA.WS'
   'ECHO ReceiveBufferSize=2012 >> 'destpath||'\SNA.WS'
   'ECHO SourceSAP=04 >> 'destpath||'\SNA.WS'
   'ECHO TraceName=snalantrace0 >> 'destpath||'\SNA.WS'
   'ECHO TransmitFrameSize=2012 >> 'destpath||'\SNA.WS'
   'ECHO CPName= >> 'destpath||'\SNA.WS'
   'ECHO ExchangeID=N >> 'destpath||'\SNA.WS'
   'ECHO [LU] >> 'destpath||'\SNA.WS'
   'ECHO CompEnabled=N >> 'destpath||'\SNA.WS'
   'ECHO CompBufSize=4096 >> 'destpath||'\SNA.WS'
   'ECHO SLE=N >> 'destpath||'\SNA.WS'
   'ECHO [3270] >> 'destpath||'\SNA.WS'
   'ECHO ScreenSize=24x80 >> 'destpath||'\SNA.WS'
   'ECHO SessionType=Display >> 'destpath||'\SNA.WS'
   'ECHO HostGraphics=Y >> 'destpath||'\SNA.WS'
   'ECHO NativeGraphics=N >> 'destpath||'\SNA.WS'
   'ECHO LoadPSS=N >> 'destpath||'\SNA.WS'
   'ECHO QueryReplyMode=Auto >> 'destpath||'\SNA.WS'
   'ECHO CharacterCellSize=9x16 >> 'destpath||'\SNA.WS'
```

Figure 38. CM2PCOMM.CMD (Part 5 of 7)

```
 'ECHO HostCodePage=037-U >> 'destpath||'\SNA.WS'
   'ECHO LtNumber=FF >> 'destpath||'\SNA.WS'
   'ECHO LuNumber=FF >> 'destpath||'\SNA.WS'
   'ECHO [Transfer] >> 'destpath||'\SNA.WS'
   'ECHO HostSystem=1 >> 'destpath||'\SNA.WS'
   'ECHO DefaultVMDisk=a >> 'destpath||'\SNA.WS'
   'ECHO DefaultDataSet= >> 'destpath||'\SNA.WS'
   'ECHO DefaultLibrary=qgpl >> 'destpath||'\SNA.WS'
   'ECHO DefaultDirectory=C:\PCOMOS2 >> 'destpath||'\SNA.WS'
   'ECHO ResumeSendFileName= >> 'destpath||'\SNA.WS'
   'ECHO ResumeRecvFileName= >> 'destpath||'\SNA.WS'
   'ECHO [Keyboard] >> 'destpath||'\SNA.WS'
   'ECHO TypeAHead=Y >> 'destpath||'\SNA.WS'
   'ECHO CuaKeyboard=3 >> 'destpath||'\SNA.WS'
   'ECHO ResetInsertbyAttn=N >> 'destpath||'\SNA.WS'
   'ECHO SpotConversion=Y >> 'destpath||'\SNA.WS'
   'ECHO CaretSizeControl=Y >> 'destpath||'\SNA.WS'
   'ECHO CheckCodepage=Y >> 'destpath||'\SNA.WS'
   'ECHO Language=United-States >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultKeyboard=N >> 'destpath||'\SNA.WS'
   'ECHO DefaultKeyboard=C:\pcomos2\PRIVATE\CM101_3.KMP >> 'destpath||'\SNA.WS'
   'ECHO [Profile] >> 'destpath||'\SNA.BCH'
   'ECHO id=BCH >> 'destpath||'\SNA.BCH'
   'ECHO Description= >> 'destpath||'\SNA.BCH'
   'ECHO [Batch] >> 'destpath||'\SNA.BCH'
   'ECHO Run1=;If you want to modify an existing batch file, >> 'destpath||'\SNA.BCH'
   'ECHO Run2=;please choose [File] - [Open]. >> 'destpath||'\SNA.BCH'
   'ECHO Run3=;  >> 'destpath||'\SNA.BCH'
   'ECHO Run4=;If you want WorkStation Profiles or other >> 'destpath||'\SNA.BCH'
   'ECHO Run5=;programs to be included in a batch file, >> 'destpath||'\SNA.BCH'
   'ECHO Run6=;you must enter them in the edit area below. >> 'destpath||'\SNA.BCH'
   'ECHO Run7=;To do this, double-click the filename in >> 'destpath||'\SNA.BCH'
   'ECHO Run8=;the listbox or select it and choose Add. >> 'destpath||'\SNA.BCH'
   'ECHO Run9=;  >> 'destpath||'\SNA.BCH'
   'ECHO Run10=;If you want to suppress the IBM logo, add >> 'destpath||'\SNA.BCH'
   'ECHO Run11=;/Q to the command for the first session. >> 'destpath||'\SNA.BCH'
   'ECHO Run12=; >> 'destpath||'\SNA.BCH'
   'ECHO Run13=;Do NOT delete the remarks above - they will >> 'destpath||'\SNA.BCH'
   'ECHO Run14=;be ignored when the batch file is run. >> 'destpath||'\SNA.BCH'
   'ECHO Run15=;  >> 'destpath||'\SNA.BCH'
   linenumber = 16
   do i=1 to numberOfSessions
   echoline = 'Run'||linenumber||'=c:\pcomos2\pcsws.exe "c:\pcomos2\private\sna.ws"'
   'ECHO 'echoline' >> 'destpath||'\SNA.BCH'
   linenumber = linenumber + 1
   end /* do */
   say 'PComm Files built ....'
   say ''
   Call Stream destpath||'\SNA.WS','c','close'
   Call Stream destpath||'\SNA.BCH','c','close'
end /* do */

Call Stream rsp_file,'c','close'
 'DEL' rsp_file
 'DEL output.txt'
 if RCode == 0 then say 'CM2PCOMM ended successfully.'
 else say 'CM2PCOMM ended unsuccessfully with return code 'RCode
 exit RCode
```

Figure 39. CM2PCOMM.CMD (Part 6 of 7)

```
argerror:
say ''
say '************************************************************************'
say '*                                                                      *'
say '* The provided arguments are not valid !!                              *'
say '*                                                                      *'
say '* Syntax:                                                              *'
say '* CM2PCOMM <CM/2 install path> <NVDM/2 wkst name> <active CFG name>     *'
say '*                                                                      *'
say '* NVDM/2 workstation name is the name you defined for this workstation  *'
say '* at the NVDM/2 Server.                                                *'
say '* The Name of the configuration file has to be a valid 8.3 file name    *'
say '* and all directories specified may be up to 8 characters only          *'
say '*                                                                      *'
say '* Example:                                                             *'
say '* CM2PCOMM C:\CMLIB myconfig                                           *'
say '*                                                                      *'
say '************************************************************************'
say ''

RCode = 4
return RCode
```

Figure 40. CM2PCOMM.CMD (Part 7 of 7)

6.1.1.1 CM2PCOMM.CMD Return Codes

0	Successful program execution.
4	An argument is not valid or is missing.
8	The extraction of the configuration failed.
12	The CM/2 Values are incompatible with PCOMM.

6.1.2 MAKECONF.CMD

In case the CM/2 configuration values are not compatible with PCOMOS2, the corresponding configuration files for CM2PCOMM.CMD will not be available, which results in not having PCOMOS2 host sessions available at the target machines in the migration process.

However, after contacting your mainframe administrator and receiving new definitions to use with PCOMOS2, you can use MAKECONF.CMD to generate a definition file for a PCOMOS2 host session (.WS) and a file to start a defined number of PCOMOS2 host sessions (.BCH). These files are put in the correct subdirectory on the NVDM/2 Server and can be used in the migration process.

There are two ways to use this program:

1. Invocation of the program without passing parameters.

In this case, the program prompts you for needed values for one workstation to create the configuration for this workstation.

2. Invocation of the program providing the name of a data file (see “The MAKECONF Data File” on page 197) as an argument. In this case, the program reads the values from this file and creates as many configurations as specified in this data file.

In both cases, the program checks for the existence of the target directory. In our scenarios the target directory at the NetView DM/2 server is \SHAREA\IMG\CONFIG\<*workstation name*>, where <workstation name> is the name defined at the NVDM/2 Server. If the specific directory does not exist, the program creates it.

6.1.2.1 The MAKECONF Data File

The data file is a plain ASCII file containing all PCOMOS2 configuration values for the workstations and the workstation name:

```
;Put the values in the order shown in line 4
;with (max) the number of digits:
;workstationname destinationaddress nodeid numberOfSessions
;12345678 123456789012 12345 1

 UZ1175G  400021041062 5A0A3 4
 UZ1175H  400021041064 5A0A6 3
 JM230674 404741036497 40589 4
 CF1374J  400004036497 500C4 4
```

Figure 41. Example of an MAKECONF Data File

All lines that begin with a semi-colon (;) are comment lines and are ignored at execution time. You may add empty lines to the file in order to increase the readability of the data file.

6.1.2.2 Using MAKECONF.CMD

We recommend using this program on the NVDM/2 Server. You need to check the value of the variable destination path (`destpath`) in line 45 of the program source code. The path defined here contains the subdirectories with the configuration data for all workstations. In our scenarios, it is "D:\SHAREA\IMG\CONFIG". Check the value and adjust it when you are using a different path at your server. You can use this program on the client machine when the `destpath` variable points to the redirected drive on the NVDM/2 Server. However, it makes more sense to use it directly on the server, especially when there are many configuration values in the data file.

To use the program, perform the following steps:

- Create or modify the data file to your needs. Make sure that the data file is in the same directory as `MAKECONF.CMD`.
- Issue the following command with the data file. Make sure to specify the name and the extension of the data file. When you don't want to use a data file, invoke the program without passing any arguments.

MAKECONF Syntax

`MAKECONF` *<Data File Name>*

where:

<Data File Name>	Specifies the file name of the data file. For example: `MAKECONF CONFDATA.DAT`

- Follow the directions on the screen.

The program checks the provided values and for the existence of all target directories. The `destpath` directory must exists before you execute `MAKECONF`. All subdirectories are created by the program when necessary. When all values are OK and the directories exist, the two configuration files for PCOMOS2 are created and stored in the subdirectory for the specific workstation.

Customize the PATH Statements

If you use a different directory structure on the NVDM/2 Server than illustrated here in this book or if PCOMOS2 is installed in a different directory than illustrated in this book, you need to customize the `PATH` statements in the source code.

You can do this by editing these three statements (lines 45 to 47):

```
destpath = 'W:\IMG\CONFIG'   /* without "\" at the end !! */
pcomdir = 'C:\PCOMOS2'       /* without "\" at the end !! */
logdir = 'X:\LOG'            /* without "\" at the end !! */
```

All other commands in the program references to this three variables and need not to be changed.

If you do not use a data file, the following screen appears:

```
You are about to create PCOMOS2 configuration files to use in the
migration process.
The following values are needed:

Destination Address, NodeID, number of 3270 terminal sessions to
configure and the NVDM/2 workstation name of the client you defined
at the NVDM/2 Server. The configuration files will be stored in
D:\SHAREA\IMG\CONFIG\<workstationname> on your NVDM/2 Server.

Enter Destination Address :
123456789012 (12 Digits)
404741036497
Enter NodeID              :
12345 (5 Digits)
40589
Enter Number of Sessions  :
1 (1 Digit)
4
Enter Workstationname     :
12345678 (up to 8 Digits)
JM230674
```

Note: the highlighted values represent typed entries.

The reasons for the example numbers is to help the user to give the correct numbers of digits to every value. Type the values when prompted and press **Enter** to pass the value to the program.

6.1.2.3 MAKECONF Log File

A log file named MAKECONF.LOG is automatically created in the directory X:\LOG (D:\SHAREB\LOG on the NVDM/2 Server). It contains informational lines for each workstation processed.

6.1.2.4 MAKECONF.CMD

`MAKECONF.CMD` is a plain ASCII file written in REXX. You can make changes to it to meet your own purposes.

The following figure displays the contents of the `MAKECONF.CMD` file.

```
/*****************************************************************************/
/*                                                                           */
/* REXX Procedure to create a PCOMMOS2 configuration files with user-given   */
/* values                                                                    */
/*                                                                           */
/*****************************************************************************/

parse upper arg inputfile .

/*****************************************************************************/
/* Variables Dokumentation                                                   */
/*                                                                           */
/* RCode             Return code MAKECONF returns to the invoking program    */
/* RC                Return code MAKECONF receives from programs it invokes  */
/* workstationname   NVDM/2 name of the workstation                          */
/* nodeid            Specifies the value of the NODE_ID                      */
/* address           Specifies the value of the DESTINATION_ADDRESS          */
/* numberOfSessions  Stores the number of the defines 3270 sessions in the   */
/*                   CM/2 response file                                      */
/* destpath          Drive and path on the code server where the PCOMOS2     */
/*                   files are copied to without the <workstationname>       */
/*                   subdirectory                                            */
/* targetpath        Drive and path on the code cerver where the PCOMOS2     */
/*                   files are copied to with the <workstationname>          */
/*                   subdirectory                                            */
/* i                 Loop counter                                            */
/* inputfile         Name of the file that  contains the configuration data  */
/*                   for many machines                                       */
/* curdir            Name of the directory in which this program is invoked  */
/* Error             Memory variable for unsuccessful execution for single   */
/*                   workstations.                                           */
/* answer            The value of this variable is read from the keyboard,   */
/* success           Counter for number of configurations successful built   */
/* unsuccess         Counter for number of configurations unsuccessful built */
/* echoline          Contains the line to write to the a file. It is used to */
/*                   avoid lines that are too long for viewing               */
/* pcomdir           PCOMOS2 install directory                               */
/*****************************************************************************/


ECHO OFF
RCode = 0
Error = 0
success = 0
unsuccess = 0
destpath = 'W:\IMG\CONFIG'   /* without "\" at the end !! */
pcomdir = 'C:\PCOMOS2'       /* without "\" at the end !! */
logdir = 'X:\LOG'            /* without "\" at the end !! */

RC = STREAM(logdir||'\MAKECONF.LOG', 'C', 'QUERY EXISTS')
if RC <> '' then 'DEL 'logdir||'\MAKECONF.LOG'

if inputfile == '' then do
   'CLS'
   say 'You are about to create PCOMOS2 configuration files to use in the'
   say 'migration process.'
   say 'The following values are needed:'
   say ''
```

Figure 42. MAKECONF.CMD (Part 1 of 6)

```
 say 'Destination Address, NodeID, Number of 3270 terminal sessions to'
   say 'configure and the NVDM/2 workstation name of the client you defined'
   say 'at the NVDM/2 Server. The configuration files will be stored in'
   say destpath||'\<workstationname> on your NVDM/2 Server.'
   say ''
   say 'Enter Destination Address :'
   say '123456789012 (12 Digits)'
   pull address
   say 'Enter NodeID              :'
   say '12345 (5 Digits)'
   pull nodeid
   say 'Enter Number of Sessions  :'
   say '1 (1 Digit)'
   pull numberOfSessions
   say 'Enter Workstationname     :'
   say '12345678 (up to 8 Digits)'
   parse upper pull workstationname
   CALL checkvalues

   if RCode == 0 then do
      call buildprofile
   end /* do */
   if RCode == 0 then do
      'ECHO    configuration for 'workstationname' built >> 'logdir||'\MAKECONF.LOG'
      echoline = '*********************************************************'
      'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
      success = 1
   end /* do */
   else do
      echoline = '   configuration for 'workstationname' not built'
      'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
      echoline = '*********************************************************'
      'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
      unsuccess = 1
      Error = 1
   end /* do */
end /* do */

else do
   RC = STREAM(inputfile, 'C', 'QUERY EXISTS')
   if RC == '' then do
      'ECHO The specified input file does not exist... >> 'logdir||'\MAKECONF.LOG'
      RCode = 8
      'ECHO RCode = 'RCode' >> 'logdir||'\MAKECONF.LOG'
   end /* do */
   if RCode == 0 then do
      do while lines(inputfile)
         line.linenum = Linein(inputfile)
         if (substr(line.linenum, 1, 1)==';') | (strip(line.linenum)=='') then
iterate
         else do
            workstationname = strip(word(line.linenum, 1))
            address = strip(word(line.linenum, 2))
            nodeid = strip(word(line.linenum, 3))
            numberOfSessions = strip(word(line.linenum, 4))
            CALL checkvalues
            if RCode == 0 then do
               call buildprofile
            end /* do */
```

Figure 43. MAKECONF.CMD (Part 2 of 6)

```
 if RCode == 0 then do
              echoline '   configuration for 'workstationname' built'
              'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
              echoline = '*********************************************************'
              'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
           end /* do */
           else do
              echoline = '   configuration for 'workstationname' not built'
              'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
              echoline = '*********************************************************'
              'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
           end /* do */
        end /* do */
        if RCode <> 0 then do
           Error = Error + 1
           unsuccess = unsuccess + 1
        end /* do */
        else success = success + 1
        RCode = 0
     end /* do */
  end /* do */
end /* do */
CALL LINEOUT inputfile
if success == 0 then RCode = 4
if RCode == 0 then do
   'ECHO MAKECONF ended successfully. >> 'logdir||'\MAKECONF.LOG'
   'ECHO 'success' configuration(s) successfully built >> 'logdir||'\MAKECONF.LOG.'
   'ECHO 'unsuccess' configuration(s) failed. >> 'logdir||'\MAKECONF.LOG'
   say ''
   say success' configuration(s) successfully built.'
   say unsuccess' configuration(s) failed.'
   if Error <> 0 then do
      RCode = 2
      say ''
      say '**********************************************************'
      say '* MAKECONF ended successfully                            *'
      say '* But error messages were logged for several workstations! *'
      say '* Check the Logfile...                                   *'
      say '* MAKECONF ended with return code 'RCode'                 *'
      say '**********************************************************'
      pull answer
   end /* do */
end /* do */
else do
   echoline = 'MAKECONF ended unsuccessfully with return code 'RCode
   'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
   say 'MAKECONF ended unsuccessfully with return code 'RCode'!!'
end /* do */
ECHO ON
exit RCode
buildprofile:
if RCode == 0 then do
   curdir = DIRECTORY()
   RC = DIRECTORY(destpath)
   CALL DIRECTORY curdir
   if RC == '' then do
      RCode = 8
      'ECHO    The directory 'destpath' does not exist! >> 'logdir||'\MAKECONF.LOG'
      'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
```

Figure 44. MAKECONF.CMD (Part 3 of 6)

```
end /* do */
  if RCode == 0 then do
   curdir = DIRECTORY()
   targetpath = destpath||'\'||workstationname
   RC = DIRECTORY(targetpath)
   CALL DIRECTORY curdir
   if RC == '' then 'MD 'targetpath
   RC = STREAM(targetpath||'\SNA.WS', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'targetpath||'\SNA.WS'
   RC = STREAM(targetpath||'\SNA.BCH', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'targetpath'\SNA.BCH'

   'ECHO [Profile] >> 'targetpath||'\SNA.WS'
   'ECHO ID=WS >> 'targetpath||'\SNA.WS'
   'ECHO Description= >> 'targetpath||'\SNA.WS'
   'ECHO [Translation] >> 'targetpath||'\SNA.WS'
   'ECHO IBMDefaultView=Y >> 'targetpath||'\SNA.WS'
   'ECHO DefaultView= >> 'targetpath||'\SNA.WS'
   'ECHO IBMDefaultDBCS=Y >> 'targetpath||'\SNA.WS'
   'ECHO DefaultDBCS= >> 'targetpath||'\SNA.WS'
   'ECHO [Communication] >> 'targetpath||'\SNA.WS'
   'ECHO AutoConnect=Y >> 'targetpath||'\SNA.WS'
   'ECHO Link=slan >> 'targetpath||'\SNA.WS'
   'ECHO Session=3270 >> 'targetpath||'\SNA.WS'
   'ECHO [SLAN] >> 'targetpath||'\SNA.WS'
   'ECHO Adapter=0 >> 'targetpath||'\SNA.WS'
   'ECHO DestinationSAP=04 >> 'targetpath||'\SNA.WS'
   'ECHO GatewayAddress='||address' >> 'targetpath||'\SNA.WS'
   'ECHO Identifier=061'||nodeid' >> 'targetpath||'\SNA.WS'
   'ECHO LinkStations=1 >> 'targetpath||'\SNA.WS'
   'ECHO ReceiveBufferCount=12 >> 'targetpath||'\SNA.WS'
   'ECHO ReceiveBufferSize=2012 >> 'targetpath||'\SNA.WS'
   'ECHO SourceSAP=04 >> 'targetpath||'\SNA.WS'
   'ECHO TraceName=snalantrace0 >> 'targetpath||'\SNA.WS'
   'ECHO TransmitFrameSize=2012 >> 'targetpath||'\SNA.WS'
   'ECHO CPName= >> 'targetpath||'\SNA.WS'
   'ECHO ExchangeID=N >> 'targetpath||'\SNA.WS'
   'ECHO [LU] >> 'targetpath||'\SNA.WS'
   'ECHO CompEnabled=N >> 'targetpath||'\SNA.WS'
   'ECHO CompBufSize=4096 >> 'targetpath||'\SNA.WS'
   'ECHO SLE=N >> 'targetpath||'\SNA.WS'
   'ECHO [3270] >> 'targetpath||'\SNA.WS'
   'ECHO ScreenSize=24x80 >> 'targetpath||'\SNA.WS'
   'ECHO SessionType=Display >> 'targetpath||'\SNA.WS'
   'ECHO HostGraphics=Y >> 'targetpath||'\SNA.WS'
   'ECHO NativeGraphics=N >> 'targetpath||'\SNA.WS'
   'ECHO LoadPSS=N >> 'targetpath||'\SNA.WS'
   'ECHO QueryReplyMode=Auto >> 'targetpath||'\SNA.WS'
   'ECHO CharacterCellSize=9x16 >> 'targetpath||'\SNA.WS'
   'ECHO HostCodePage=037-U >> 'targetpath||'\SNA.WS'
   'ECHO LtNumber=FF >> 'targetpath||'\SNA.WS'
   'ECHO LuNumber=FF >> 'targetpath||'\SNA.WS'
   'ECHO [Transfer] >> 'targetpath||'\SNA.WS'
   'ECHO HostSystem=1 >> 'targetpath||'\SNA.WS'
   'ECHO DefaultVMDisk=a >> 'targetpath||'\SNA.WS'
   'ECHO DefaultDataSet= >> 'targetpath||'\SNA.WS'
   'ECHO DefaultLibrary=qgpl >> 'targetpath||'\SNA.WS'
   'ECHO DefaultDirectory='||pcomdir||' >> 'targetpath||'\SNA.WS'
   'ECHO ResumeSendFileName= >> 'targetpath||'\SNA.WS'
```

Figure 45. MAKECONF.CMD (Part 4 of 6)

```
 'ECHO ResumeRecvFileName= >> 'targetpath||'\SNA.WS'
    'ECHO [Keyboard] >> 'targetpath||'\SNA.WS'
    'ECHO TypeAHead=Y >> 'targetpath||'\SNA.WS'
    'ECHO CuaKeyboard=3 >> 'targetpath||'\SNA.WS'
    'ECHO ResetInsertbyAttn=N >> 'targetpath||'\SNA.WS'
    'ECHO SpotConversion=Y >> 'targetpath||'\SNA.WS'
    'ECHO CaretSizeControl=Y >> 'targetpath||'\SNA.WS'
    'ECHO CheckCodepage=Y >> 'targetpath||'\SNA.WS'
    'ECHO Language=United-States >> 'targetpath||'\SNA.WS'
    'ECHO IBMDefaultKeyboard=N >> 'targetpath||'\SNA.WS'
    'ECHO DefaultKeyboard='||pcomdir||'\CM101_3.KMP >> 'targetpath||'\SNA.WS'


    'ECHO [Profile] >> 'targetpath||'\SNA.BCH'
    'ECHO id=BCH >> 'targetpath||'\SNA.BCH'
    'ECHO Description= >> 'targetpath||'\SNA.BCH'
    'ECHO [Batch] >> 'targetpath||'\SNA.BCH'
    echoline = 'Run1=;If you want to modify an existing batch file,'
    'ECHO echoline  >> 'targetpath||'\SNA.BCH'
    'ECHO Run2=;please choose [File] - [Open]. >> 'targetpath||'\SNA.BCH'
    'ECHO Run3=;  >> 'targetpath||'\SNA.BCH'
    'ECHO Run4=;If you want WorkStation Profiles or other >> 'targetpath||'\SNA.BCH'
    'ECHO Run5=;programs to be included in a batch file, >> 'targetpath||'\SNA.BCH'
    'ECHO Run6=;you must enter them in the edit area below. >>
'targetpath||'\SNA.BCH'
    'ECHO Run7=;To do this, double-click the filename in >> 'targetpath||'\SNA.BCH'
    'ECHO Run8=;the listbox or select it and choose Add. >> 'targetpath||'\SNA.BCH'
    'ECHO Run9=;  >> 'targetpath||'\SNA.BCH'
    'ECHO Run10=;If you want to suppress the IBM logo, add >> 'targetpath||'\SNA.BCH'
    'ECHO Run11=;/Q to the command for the first session. >> 'targetpath||'\SNA.BCH'
    'ECHO Run12=; >> 'targetpath||'\SNA.BCH'
    'ECHO Run13=;Do NOT delete the remarks above - they will >>
'targetpath||'\SNA.BCH'
    'ECHO Run14=;be ignored when the batch file is run. >> 'targetpath||'\SNA.BCH'
    'ECHO Run15=;  >> 'targetpath||'\SNA.BCH'
    linenumber = 16
    do i=1 to numberOfSessions
     echoline1 = 'Run'||linenumber||'='||pcomdir||'\pcsws.exe '
     echoline2 = '"'||pcomdir||'\private\sna.ws"'
     'ECHO 'echoline1||echoline2' >> 'targetpath||'\SNA.BCH'
     linenumber = linenumber + 1
    end /* do */
   end /* do */
end /* do */
CALL LINEOUT targetpath||'\SNA.WS'
CALL LINEOUT targetpath||'\SNA.BCH'
return RCode

checkvalues:
'ECHO workstation 'workstationname' >> 'logdir||'\MAKECONF.LOG'
do i = 1 to (length(nodeid))
   rc = verify(substr(nodeid,i,1),'0123456789ABCDEF')
   if RC <> 0 then do
      RCode = 12
      'ECHO    The nodeid is not compatible with PComm! >> 'logdir||'\MAKECONF.LOG'
      'ECHO    Digit number 'i' is not a hex-value! >> 'logdir||'\MAKECONF.LOG'
      'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
   end /* do */
end /* do */
```

Figure 46. MAKECONF.CMD (Part 5 of 6)

```
if RCode == 0 then do
   do i = 1 to (length(address))
      rc = verify(substr(address,i,1),'0123456789ABCDEF')
      if RC <> 0 then do
         RCode = 12
         'ECHO    The address is not compatible with PComm! >>
'logdir||'\MAKECONF.LOG'
         'ECHO    Digit number 'i' is not a hex-value! >> 'logdir||'\MAKECONF.LOG'
         'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
      end /* do */
   end /* do */
end /* do */
address = strip(address)
if length(address) <> 12 then do
   RCode = 8
   'ECHO    The address has to be a 12 digit value! >> 'logdir||'\MAKECONF.LOG'
   'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
end /* do */
numberOfSessions = strip(numberOfSessions)
rc = verify(numberOfSessions, '123456789')
if RC <> 0 then do
   RCode = 8
   echoline = '   numberOfSessions has to be a number between 1 and 9!'
   'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
   'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
end /* do */
return RCode
```

Figure 47. MAKECONF.CMD (Part 6 of 6)

6.1.2.5 MAKECONF.CMD Return Codes

`MAKECONF.CMD` generates the following return codes. They are also logged in the log file:

0 Successful program execution, no error messages logged.

2 Successful program execution for at least one workstation, but error messages for one ore more workstations are logged.

4 No configuration in the data file was correct, no configuration built.

The following return codes appear only in the log file. They indicate an failure in one or more workstation configurations. But at least one configuration was successfully built:

8 An argument or a directory is not valid or missing.

12 The configuration values are not compatible with PCOMOS2.

6.1.3 COPYCONF.CMD

Once you have generated the configuration files for PCOMOS2, you can use the `COPYCONF.CMD` program in the migration process to copy the configuration files to the right directories on the target machines. For the usage in the migration process within NVDM/2, please see chapters 8 and 9.

The program copies the generated PCOMOS2 configuration files from the provided source directory at the NVDM/2 Server to the provided directory on the target machine. With this copy operation, the sample configuration files of the donor system are replaced with the valid files that contain the actual configuration values of the "old" machine.

The following figure illustrates the COPYCONF.CMD REXX command file:

```
/*************************************************************/
/* Cmd File to copy the generated PCOMOS2 configuration files */
/* to the correct directory at the target system              */
/*************************************************************/

parse upper arg workstationname

/****************************************************************************/
/* Variables Documentation                                                  */
/*                                                                          */
/* RCode            Return code COPYCONF returns to the invoking program    */
/* RC               Return code COPYCONF receives from programs it invokes  */
/* workstationname  NVDM/2 name of the workstation                          */
/* sourcepath       Drive and path on the code server where the PCOMOS2 files */
/*                  are copied to without the <workstationname> subdirectory  */
/* i                Loop counter                                            */
/* curdir           Name of the directory in which this program is invoked  */
/* echoline         Contains the lines to write to the file. It is used to  */
/*                  avoid lines which are too long for viewing              */
/* pcompath         PCOMOS2 install directory                               */
/* logpath          Path to the logfiles                                    */
/****************************************************************************/

ECHO OFF
RCode = 0

pcommpath = 'C:\PCOMOS2\PRIVATE'
sourcepath = 'W:\IMG\CONFIG\'||workstationname
logpath = 'X:\LOG\'||workstationname


curdir = DIRECTORY()
RC = DIRECTORY(sourcepath)
CALL DIRECTORY curdir
if RC == '' then do
   RCode = 8
   'ECHO    The directory 'sourcepath' does not exist! >> logpath||.\COPYCONF.LOG'
end /* do */

if RCode == 0 then do
   RC = STREAM(sourcepath||'\SNA.WS', 'C', 'QUERY EXISTS')
   if RC == '' then do
      RCode = 8
      'ECHO The source file SNA.WS is missing ! >> 'logpath||'\COPYCONF.LOG'
   end /* do */
   RC = STREAM(sourcepath||'\SNA.BCH', 'C', 'QUERY EXISTS')
   if RC == '' then do
      RCode = 8
      'ECHO The source file SNA.WS is missing ! >> 'logpath||'\COPYCONF.LOG'
   end /* do */
if RCode == 0 then do
      'COPY 'sourcepath||'\SNA.WS 'pcommpath
      if RC == 0 then do
         'COPY 'sourcepath||'\SNA.BCH 'pcommpath
         if RC <> 0 then do
            RCode = 4
            'ECHO Copy of source file SNA.WS failed ! >> 'logpath||'\COPYCONF.LOG'
         end /* do */
      end /* do */
```

Figure 48. COPYCONF.CMD (Part 1 of 2)

```
 else do
         RCode = 4
         'ECHO Copy of source file SNA.BCH failed ! >> 'logpath||'\COPYCONF.LOG'
      end /* do */
   end /* do */
end /* do */

if RCode == 0 then do
   echoline = 'COPYCONF.CMD ended successfully for workstation 'workstationname
   'ECHO 'echoline' >> 'logpath||'\COPYCONF.LOG'
   'ECHO RCode = 'RCode' >> 'logpath||'\COPYCONF.LOG'
end /* do */
else do
   'ECHO COPYCONF.CMD ended unsuccessfully >> 'logpath||'\COPYCONF.LOG'
   'ECHO RCode = 'RCode' >> 'logpath||'\COPYCONF.LOG'
end /* do */
ECHO ON
exit RCode
```

Figure 49. COPYCONF.CMD (Part 1 of 2)

6.1.3.1 COPYCONF.CMD Return Codes

The following return codes are generated by COPYCONF.CMD:

0 Successful program execution.

4 A copy operation to the target system failed.

8 One of the source files (SNA.WS and SNA.BCH) is missing.

6.2 Migrating the NET.ACC File

In some of your machines, OS/2 Peer services may be installed. If this is the case, you may want to migrate the NET.ACC file. This file stores information about local and remote users defined at the workstation along with their privileges about resources (shares) they are allowed to access at the workstation. It is located in the C:\IBMLAN\ACCOUNTS directory. All values are stored binary and cannot be directly accessed by a programming language. You need to have API extension that actually allow that.

We tried a couple of tools, such as LSMT (LAN Server Management Utilities; available through the redbook titled *How to Manage PC Server Environments*, SG24-4879), BACKACC, RESTACC, and NETUTIL.DLL, to accomplish this task. However, all available tools are written to administer LAN/Warp Server and not Peer Services.

The NETUTIL.DLL, which extends the NET commands, is also not the right way. You cannot retrieve password information, and most of the commands run with LAN Server only.

The BACKACC and RESTACC tools included with LAN Server/Requester cannot be used to do migrations. They are only usable when you want to use a NET.ACC in two machines running the same OS/2 version.

To sum up, none of the tools were able to retrieve all information needed to successfully migrate a NET.ACC file.

There are two possible ways (except to create the NET.ACC manually on the new systems) to migrate the NET.ACC file:

- Coding a native API that would allow accessing peer server information (sorry, we could not do that during the project), however, there is a nice tool out there called LSMT which was mentioned before in this section)
- The administrator is fully informed about defined user accounts, passwords, shares, and ACLs of each workstation being migrated.

We will briefly discuss both ways.

6.2.1 Using Information Provided by the Workstation Owner

In this scenario, the owner of the workstation has to provide information about the user IDs, passwords, shares, and ACLs to the administrator who does the migration.

All information is written to a file and a program reads the needed information from this file. Using NET commands (NET USER, NET SHARE, and so forth.) all information is written against the NET.ACC file at the new system.

With this process, the user can work with his workstation as usual after the migration. He can use his/her normal user IDs, passwords, and all of the formerly defined shares.

It is a fast way to migrate the NET.ACC. Since only standard NET commands can accomplish the migration process, the final program would only need to run the NET commands while passing information read out of an ASCII file.

6.2.2 Using LAN Server REXX APIs - LSRXUTIL.DLL

This scenario requires a lot more knowledge about the structure of the NET.ACC file. However, since native APIs can be used to retrieve information, you can basically do anything with a NET.ACC file.

You can download this DLL from the following Web site:

`http://www.software.ibm.com/os/warp/pspinfo/wsapplets.html`

From this page, select the **LAN Server/Warp Server REXX API extension** item. Download the file WSLSRXUT.ZIP to a temporary directory and extract the file using the PKUNZIP2 or UNZIP tool. You will get following files:

- LSRXUT3.DLL

  This file is used for LAN Server/Requester 3 installations and must be copied/renamed to LSRXUT.DLL. Works with LAN Requester Peer Services, too.

- LSRXUT4.DLL

  This file is used for LAN Server/Requester 4, Warp Server, and OS/2 Warp 4 installations and copied/renamed to LSRXUT.DLL. Works with OS/2 Warp 4 Peer Services, too.

- LSRXUTIL.INF

  This file represents the online copy of the LAN Server REXX API manual.

LSRXUT.DLL must reside in a directory that is included in the `LIBPATH` variable of the CONFIG.SYS file. Type `VIEW LSRXUTIL.INF` to browse through the documentation.

Using `LSRXUT`, you can include the new functions provided by the DLLs in your normal REXX programs and use the interface to the LAN Server APIs which also work with Peer Services of LAN Requester 3 and 4, and OS/2 Warp 4 Peer Services.

The next figure illustrates an example of how to list defined shares of a machine.

```
/* List names of shares defined on a peer server */

call RxFuncAdd 'LoadLsRxutFuncs', 'LSRXUT', 'LoadLsRxutFuncs'
call LoadLsRxutFuncs

NETSHARE = 190
SrvName  = '\\SRV1175'

myRc = NetEnumerate(NETSHARE, 'shareInfo', SrvName)

if myRc <> '0' then do
 say 'Got error from NetEnumerate() ' myRc
 call DropLsRxutFuncs
 exit 9
end

if shareInfo.1 = 0 then do
 say 'Server does not share a resource'
 call DropLsRxutFuncs
 exit 0
end

say 'Number of shared netnames: ' shareInfo.0
say

do i=1 to shareInfo.0
 say shareInfo.i
end

call DropLsRxutFuncs
call RxFuncDrop 'LoadLsRxutFuncs'

exit 0
```

Figure 50. Sample REXX Program that lists Share Names

The next figure displays an output screen when the previously shown REXX program ended:

```
Number of shared netnames: 7

IPC$
ADMIN$
CD-ROM
CDRIVE
DDRIVE
SHAREA
SHAREB
```

To successfully migrate the NET.ACC file, we thought of the following idea:

1. Run a procedure that reads out all data stored in the "old" workstation’s NET.ACC file and stores the results in a simple ASCII file.

   You may use the procedure illustrated before, or use LSMT. Note that in most cases you need to be logged on to the workstation with administrator rights to read out information. In particular, if you want to read out passwords (in hex though!).

2. Another procedure inputs the ASCII file created by the first procedure and creates an NET.ACC file on the new system accordingly.

   To create an NET.ACC file, you need to be logged on with administrator rights.

The advantage of this process is that the end-user is not involved. However, besides the fact that this process is very sophisticated, you need to have access to an admin ID at the remote workstation to create the NET.ACC file. Do end-users like this idea or don’t they? - Well, we cannot answer this question for you, dear Mr./Ms. Administrator.

Which approach do you like best? - We think it depends on the number of NET.ACC files you have to migrate. For a small number of NET.ACC migrations, the second process might be a little too complicated and requires too much work. Then, the first process might be used instead. Alternately, you may want to create the NET.ACC manually on the new systems.

Fortunately, only a low percentage of the workstations are usually configured with Peer Services, meaning this migration step is an exception and not needed for most cases.

Chapter 7. Pristine Installation of OS/2 Warp 4

One of the many strengths of the Rapid Deployment Team (RDT) tools is that they can be used in almost any scenario. This chapter details how to use the Rapid Deployment Tools in a pristine installation scenario for OS/2 Warp 4.

This scenario illustrates the deployment of a OS/2 Warp 4 workstation as described in Chapter 5, “Introducing Migration and Installation Scenarios” on page 167.

7.1 Requirements

In addition to the RDT tools, the following are prerequisites for this scenario:

- NetView DM/2 boot diskettes (NetView DM/2 is used as our software distribution server. Any other kind of software distribution may be used instead. This scenario will use NetView DM/2 to illustrate the process).

  Since—as the word indicates—a pristine workstation does not have anything to boot from, the NetView DM/2 boot diskettes allow the workstation to connect to a software distribution server where all images, code and data files are stored. To create these diskettes, please refer to “Creating Boot Diskettes with NetView DM/2 Client” on page 129.

- `DISKPREP.CMD` program (see “DISKPREP.CMD” on page 390).

  This procedure is written in REXX, and it automates the partitioning and formatting processes. Depending on response files, the procedure can delete and create partitions. Before the procedure is run through the end, it takes care of the system reboot and the formatting of all partitions.

- `CUBE.CMD` program (see “CUBE.CMD Source Code” on page 418)

  This utility is a REXX program that modifies ASCII configuration files, such as CONFIG.SYS, IBMLAN.INI, PROTOCOL,INI, WIN.INI, and so on, based on a set of `CUBE` commands.

  `CUBE` commands provide editing functions through strings. The set of commands includes ADD, REPLACE, and DELETE functions. In addition, `CUBE` commands can be further customized at execution time with the use of variables. A `CUBE` command may contain variables that defined at the command line or in the OS/2 environment. Full documentation is included with the tool and illustrated in Appendix D, “CUBE Source Code and Documentation” on page 407.

7.2 Process Details

The entire process for the pristine installation of OS/2 Warp 4 has four major steps:

1. Creating donor configuration bundle files (one per workstation class)
2. Enabling the REXX support at the target workstation
3. Partitioning and formatting the hard disks (`DISKPREP.CMD`)
4. Downloading the donor image to the target system's hard disk (`REPLICAT.EXE`)
5. Customizing the workstation details, such as IP address, computer name, and so on (`WSCONFIG.CMD`)

The following sections describe the above steps in detail.

7.2.1 Creating the Donor Configuration Bundle Files

The very first thing that needs to be done is to create donor configuration stream files, which, at the end of the distribution process, can be modified dynamically on pristine workstations. For this purpose, a series of `PROBE` command stream files (Figure 51 through Figure 63) must be created.

These stream command files define a series of FI variables to set up the workstation configuration values which allows the use of only one bundle file for all pristine workstations since the FI variable used for each workstation are different. Using these `PROBE` command stream files, run `PROBE.EXE` with the following parameters:

PROBE Syntax

```
<drive:\path>PROBE
             /E:<client_ID >
             /B:<intermediate_bundle_file_directory_pathname>
             /W:<temporary_work_directory_pathname>
             /L:<log_file_directory_pathname>
             /X:<logging_level>
               <command_stream-file_1>.prb
```

where:

`/E:PRISTINE`	Will create a bundle file called PRISTINE.
`/B:W:\IMG\CONFIG\PRISTINE`	Indicates the where the bundle file will be located.

`/W:C:\`	Temporary work directory pathname. This directory can be in the local hard disk drive or on the network. For better performance, the local hard disk is chosen. A different directory may be specified, but if the directory does not exist, `PROBE.EXE` will fail.
`/L:X:\LOG\PRISTINE`	Specifies the path for the log files generated in this process. These log files will give details of any problems, if any, found in the generation of the bundle files. To facilitate any troubleshooting, if need be, it is recommended to use a LAN drive, like the SHAREB drive in NetView DM/2.
`/X:4`	This parameter specifies the level of detail to be logged in the log files. Since this bundle file will be used in many workstations, the maximum level of detail is recommended.
`W:\RSP\PROBE\PRISTINE.PRB`	`PROBE` command stream file used by this process.

Since the pristine configuration bundle files need to be created only once, this process does not need a NetView DM/2 change profile file. The actual `PROBE.EXE` single-line command execution in this example is as follows:

```
PROBE.EXE /E:PRISTINE /B:W:\IMG\CONFIG\PRISTINE /W:C:\
                        /L:X:\LOG\PRISTINE /X:4 W:\RSP\PROBE\PRISTINE.PRB
```

The `PROBE` command stream file PRISTINE.PRB contains the command to define a series of FI variables that will be used by `CONFIGUR.EXE` at run time.

The following FI variables are defined with PRISTINE.PRB:

{COMPUTERNAME}	The *{COMPUTERNAME}* FI variable is used to define the LAN Requester computer name.
{DOMAIN}	The *{DOMAIN}* FI variable is used to define the LAN Requester domain.
{SRVCOMMENT}	The *{SRVCOMMENT}* FI variable is used to set the LAN Requester workstation description.
{PCOMMDESTSAP}	The *{PCOMMDESTSAP}* FI variable is used to define the mainframe MAC address for PCOMOS2.

{PCOMMID}	The *{PCOMMID}* FI variable is used to define the PCOMOS2 unique connection ID.
{HOSTNAME}	The *{HOSTNAME}* FI variable is used to set the IP host name.
{IPDOMAIN}	The *{IPDOMAIN}* FI variable is used to set the IP domain name.
{IP_DNS}	The *{IP_DNS} FI variable* is used to define the domain name server address.
{IPADDRESS}	The *{IPADDRESS}* FI variable is used to set the workstation's IP address.
{IPMASK}	The *{IP_MASK}* FI variable is used to set the workstation's IP address mask.
{IPROUTER}	The *{IPROUTER}* FI variable is used to set the default IP router.
{IPNETADDRESS}	The *{IPNETADDRESS}* FI variable is used to define the IP net address used by the IP router.
{IPNETMASK}	The *{IPNETMASK}* FI variable is used to set the IP net address mask.
{IPHOSTSOCKSNAMESERVER}	The *{IPHOSTSOCKSNAMESERVER}* FI variable is used to define the socks host name server.
{IPSOCKSNAMESERVER}	The {IPSOCKSNAMESERVER} FI variable is used to set the socks name server address.

FI Variable Definitions

All the FI variables must be defined in the PROBE process to be used by the configurator utility, CONFIGUR.EXE.

After all FI variables are defined, PRISTINE.PRB calls four additional PROBE command stream files:

- W:\RSP\PROBE\PRIS-IBMLAN.PRB, which is used to set the following LAN Requester values: Computername, Domain and workstation description (Srvcomment).

- W:\RSP\PROBE\PRIS-PCOM.PRB, used to set the following PCOMOS2 values: `Destination SAP` (Mainframe MAC Address) and `PU ID`.
- W:\RSP\PROBE\PRIS-CONFIG.PRB changes the CONFIG.SYS to define the IP host name and include the `PAUSEONERROR=NO` statement.
- W:\RSP\PROBE\PRIS-TCPIP.PRB makes modifications to several TCP/IP related files to configure the TCP/IP environment. The TCP/IP values are: `IP Address`, `IP Subnet Mask`, `IP Default Router`, `IP Default Router Net`, `IP Default Router Net Mask`, `Socks Name Server Address`, and `Socks Host Name Server`.

Figure 51 through Figure 63 illustrate `PROBE` stream command files used in this scenario.

```
// Define FI Variables to be use in this Workstation
// As the actual FI Variable will be defined at
// CONFIGUR.EXE run time, the FI Variable values are
// being set to themselves. The main purpose is to
// create them, so they exist when CONFIGUR.EXE is run.

FIVAR
   NAME "COMPUTERNAME"
   VALUE "{COMPUTERNAME}"
ENDFIVAR

FIVAR
   NAME "DOMAIN"
   VALUE "{DOMAIN}"
ENDFIVAR

FIVAR
   NAME "SRVCOMMENT"
   VALUE "{SRVCOMMENT}"
ENDFIVAR

FIVAR
   NAME "PCommDestSAP"
   VALUE "{PCommDestSAP}"
ENDFIVAR

FIVAR
   NAME "PCommID"
   VALUE "{PCommID}"
ENDFIVAR

FIVAR
   NAME "HOSTNAME"
   VALUE "{HOSTNAME}"
ENDFIVAR

FIVAR
   NAME "IPDOMAIN"
   VALUE "{IPDOMAIN}"
ENDFIVAR
```

Figure 51. PRISTINE.PRB Probe Stream Command File (Part 1 of 3)

```
FIVAR
   NAME "IP_DNS"
   VALUE "{IP_DNS}"
ENDFIVAR

FIVAR
   NAME "IP_DNS"
   VALUE "{IP_DNS}"
ENDFIVAR

FIVAR
   NAME "IPMask"
   VALUE "{IPMask}"
ENDFIVAR

FIVAR
   NAME "IPRouter"
   VALUE "{IPRouter}"
ENDFIVAR

FIVAR
   NAME "IPNetAddress"
   VALUE "{IPNetAddress}"
ENDFIVAR

FIVAR
   NAME "IPNetMask"
   VALUE "{IPNetMask}"
ENDFIVAR

FIVAR
   NAME "IPHostSocksNameServer"
   VALUE "{IPHostSocksNameServer}"
ENDFIVAR

FIVAR
   NAME "IPSocksNameServer"
   VALUE "{IPSocksNameServer}"
ENDFIVAR
```

Figure 52. PRISTINE.PRB Probe Stream Command File (Part 2 of 3)

```
// Now include all the control files needed
// to configure this workstation

// Call the Probe Stream File to configure the LAN Requester
include "w:\rsp\probe\pris-IBMLAN.prb"

// Call the Probe Stream File to configure PCom
include "w:\rsp\probe\pris-PCom.prb"

// Call the Probe Stream File to make the changes in C:\CONFIG.SYS
include "w:\rsp\probe\pris-config.prb"

// Call the Probe Stream File to configure TCP/IP
// (This includes all values, except HOSTNAME, which
// it's done in C:\CONFIG.SYS
include "w:\rsp\probe\pris-tcpip.prb"
```

Figure 53. PRISTINE.PRB Probe Stream Command File (Part 3 of 3)

```
// Details:     This will set the value for COMPUTERNAME, DOMAIN on the
//              target system C:\IBMLAN\IBMLAN.INI in the LAN Requester
//              section. Also it will also set the value for SRVCOMMENT
//              in the C:\IBMLAN\IBMLAN.INI PEER section.

TEXTFILE
   SOURCE  "C:\IBMLAN\IBMLAN.INI"
   REPLACELINE
      CASESENSITIVE
      SOURCELINE
         REPLACEMENT "  Computername = {COMPUTERNAME}"
      ENDSOURCELINE
      TARGETLINE
         PATTERN "Computername = "
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\IBMLAN\IBMLAN.INI"
   REPLACELINE
      CASESENSITIVE
```

Figure 54. PRIS-IBMLAN.PRB Probe Stream Command File (Part 1 of 2)

```
 SOURCELINE
         REPLACEMENT "  Domain = {DOMAIN}"
      ENDSOURCELINE
      TARGETLINE
         PATTERN "Domain = "
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\IBMLAN\IBMLAN.INI"
   REPLACELINE
      CASESENSITIVE
      SOURCELINE
         REPLACEMENT "  srvcomment = {SRVCOMMENT}"
      ENDSOURCELINE
      TARGETLINE
         PATTERN "  srvcomment = "
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 55. PRIS-IBMLAN.PRB Probe Stream Command File (Part 2 of 2)

```
// Details:     This will take the value specified by REPLACEMENT
//              and replace the matching lines DestinationSAP and
Identifier
//             on C:\TCPIP\PCOMOS2\PRIVATE\SNA.WS of the target system.


TEXTFILE
   SOURCE  "C:\PCOMOS2\PRIVATE\SNA.WS"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         REPLACEMENT "DestinationSAP={PCommDestSAP}"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "DestinationSAP=400021041062"
      ENDTARGETSTRING
```

Figure 56. PRIS-PCOM.PRB Probe Stream Command File (Part 1 of 2)

```
 ENDEDITLINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\PCOMOS2\PRIVATE\SNA.WS"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         REPLACEMENT "Identifier={PCommID}"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "Identifier=05D5A0A3"
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 57. PRIS-PCOM.PRB Probe Stream Command File (Part 2 of 2)

```
// Details:     This will take the value specified by REPLACEMENT
//              and replace the matching line HOSTNAME on c:\config.sys
//              of the target system.

TEXTFILE
   SOURCE  "C:\CONFIG.SYS"      // Target file will be the same as
                                        // source
   REPLACELINE
      SOURCELINE
         REPLACEMENT "SET HOSTNAME={HOSTNAME}"  // This will cause a
                                        // replacement of
                                      // SET HOSTNAME={HOSTNAME} on the
                                        // target system specified by
                                        // TARGETLINE. Where {HOSTNAME}
                                        // will be replace with its
                                        // value at execution time.
      ENDSOURCELINE
      TARGETLINE
         PATTERN "HOSTNAME"
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 58. PRIS-CONFIG.PRB Probe Stream Command File (Part 1 of 2)

```
TEXTFILE
   SOURCE  "C:\CONFIG.SYS"      // Target file will be the same as
                                // source
   REPLACELINE
      SOURCELINE
         DELETELINE            // This will delete this line
                               // on the target system.
      ENDSOURCELINE
      TARGETLINE
         PATTERN "PAUSEONERROR"
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 59. PRIS-CONFIG.PRB Probe Stream Command File (Part 2 of 2)

```
// Details: This will take the value specified by REPLACEMENT
//          and replace the respective matching lines with the
//          relevant values in the files "C:\MPTN\ETC\RESOLV2",
//          "c:\tcpip\dos\etc\resolv", "C:\MPTN\BIN\SETUP.CMD",
//          "C:\MPTN\ETC\SOCKS.CFG" and "C:\MPTN\ETC\SOCKS.ENV"
//          to set up the TCP/IP Values: Domain, IP Address,
//          IP Address Mask, Default Router, Default Router Net
//          Address, Default Router Net Address Mask, Domain Name
//          Server Address, Socks Name Server and Sock Name Server
//          Address.

TEXTFILE
   SOURCE  "C:\MPTN\ETC\RESOLV2"        // Target file will be the same
                                        // as the source
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "domain {IPDOMAIN}"  // This will cause a
                                            // replacement of the line
                                            // identified by "domain"
                                            // on the target system
                                            // specified by TARGETLINE.
                                            // Where {IPDOMAIN} will be
                                           // replaced with its value at
                                            // execution time.
```

Figure 60. PRIS-TCPIP.PRB Probe Stream Command File (Part 1 of 4)

```
 ENDSOURCELINE
        TARGETLINE
           PATTERN "domain"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\ETC\RESOLV2"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "nameserver {IP_DNS}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "nameserver"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "c:\tcpip\dos\etc\resolv"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "domain {IPDOMAIN}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "domain"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "c:\tcpip\dos\etc\resolv"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "nameserver {IP_DNS}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "nameserver"
        ENDTARGETLINE
```

Figure 61. PRIS-TCPIP.PRB Probe Stream Command File (Part 2 of 4)

```
 ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "ifconfig lan0 {IPAddress} netmask {IPMask}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "lan0"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "route add default {IPRouter} -hopcount 1"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "route add default"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "route add -net {IPNetAddress} {IPRouter}
-netmask {IPNetMask} -hopcount 1"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "route add -net 9 9.3.1.74"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 62. PRIS-TCPIP.PRB Probe Stream Command File (Part 3 of 4)

```
TEXTFILE
   SOURCE  "C:\MPTN\ETC\SOCKS.CFG"
   EDITLINE
        CASESENSITIVE
        SOURCESTRING
           REPLACEMENT "{IPHostSocksNameServer}"
        ENDSOURCESTRING
        TARGETSTRING
           LINEPATTERN "sockd @=socks.austin.ibm.com"
           STRINGPATTERN "socks.austin.ibm.com"
        ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\ETC\SOCKS.ENV"
   EDITLINE
        CASESENSITIVE
        SOURCESTRING
           REPLACEMENT "{IPSocksNameServer}"
        ENDSOURCESTRING
        TARGETSTRING
           LINEPATTERN "9.14.1.30"
           STRINGPATTERN "SOCKS_NS"
        ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 63. PRIS-TCPIP.PRB Probe Stream Command File (Part 4 of 4)

7.2.2 Enabling REXX Support

Since most processes uses REXX, we need to make sure that REXX support is enabled at the target workstation. The NetView DM/2 boot diskettes do not have enough disk space to load REXX DLLs from diskettes. To enable the REXX environment, a NetView DM/2 change file is used accessing the necessary code from the NetView DM/2 Server.

To enable REXX, first copy the following files into a directory on the code server that is listed in the `LIBPATH` statement in the CONFIG.SYS file on the NetView DM/2 boot diskettes:

- REXX.DLL
- REXXAPI.DLL

- REXXINI.DLL
- REXXUTIL.DLL
- SHPIINST.DLL
- UHPFS.DLL

In addition, copy the following files into a directory on the code server that is listed in the PATH and DPATH statements in the CONFIG.SYS file on the NetView DM/2 boot diskettes:

- REX.MSG
- REXH.MSG
- REXXTRY.CMD
- RXQUEUE.EXE
- SRVREXX.EXE
- RXSUBCOM.EXE (for OS/2 Warp 3 only)

Once all the above files have been copied to their respective directories, create a NetView DM/2 package as illustrated in Figure 64.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = "REXX Support for Pristine Clients"
End

Section Install
Begin
      Program = SA:\EXE\CMD.EXE
      Parms =  /C $(SA)\EXE\DETREXX.CMD $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 64. LOADREXX.PRO NetView DM/2 Change Profile

The REXX support package executes a batch file, DETREXX.CMD, as shown in Figure 65 on page 228.

```
@echo off

SET PATH=%PATH%;%2;
set helper=%1\SRVREXX.EXE

if not exist %helper% goto error
detach %helper%
call %2\casckrex
goto end

:error
@echo  .
@echo  %helper% not found. NDM-Server updated for REXX-Support  ?
```

Figure 65. DETREXX.CMD Batch File

Because there is a controlled reboot after repartitioning, REXX support is needed after the reboo. The REXX support package must be installed as a corequisite with all the other packages.

For detailed information about how to enable REXX support, please refer to *The OS/2 Warp 4 CID Software Distribution Guide*, SG24-2010.

7.2.3 Partitioning and Formatting the Hard Drives

The first thing to be at pristine workstations is partitioning and formatting the hard disks. You need to have a basic idea how you want to partition the hard disks. You already know that partitioning requires a reboot before you continue to format the drives.

The good thing is that these two tasks can be automated. `DISKPREP.CMD` is a REXX command file that accomplishes these tasks. It allows to input response files for each hard disk, and it partitions the hard disks accordingly, reboots the system, and then formats all partitions. You can even specify to use the long format option rather than the standard quick format.

The following list illustrates the tasks `DISKPREP.CMD` performs:

1. Checks for an existing partition with the name of "OS/2"
   - If a partition with the name of OS/2 exists, the procedure continues with the formatting process.
   - If the partition with the name of OS/2 does not exist, `DISKPREP.CMD` deletes all partitions, if there are any, and then creates partitions according to the response file.

The response files must be named FDSKHDn.RSP, where "n" is the hard disk number, starting with number "1".

2. After all the partitions have been created properly, `DISKPREP.CMD` reboots the workstation.

   Because `DISKPREP.CMD` did not give a return code back to NetView DM/2, NetView DM/2 assumes that the workstation performed a controlled reboot and will restart the `DISKPREP.CMD` process.

3. In the second execution of `DISKPREP.CMD`, an existing partition with the name of "OS/2" will be detected. It will then continue with formatting all partitions.

   `DISKPREP.CMD` by default formats all partitions using long format. If this is not desired, `DISKPREP.CMD` may be edited and adjusted to your needs.

4. After all partitions have been formatted, `DISKPREP.CMD` will give the appropriate return code to NetView DM/2 to continue with the next package.

DISKPREP.CMD Reboot Issues

Since `DISKPREP.CMD` is a REXX command file, the REXX environment must be enabled at all times. To avoid not having the REXX environment enabled after reboot performed by `DISKPREP.CMD`, the NetView DM/2 change file that enables the REXX environment, must be set as corequisite group with `DISKPREP.CMD` and any other REXX programs used in the first phase.

The DISKPREP.PRO change profile file used to build the NetView DM/2 change file is shown in Figure 66 on page 230.

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir)
/L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 66. DISKPREP.PRO NetView DM/2 Change Profile File

The `DISKPREP.CMD` REXX command file is shown in Appendix C.6, “DISKPREP.CMD” on page 390.

7.2.4 Replicating the Donor System Image

Once the hard disk(s) have been partitioned and formatted, the next step is to replicate the donor system image to the target systems using the replicator tool, `REPLICAT.EXE`. This process installs a predefined, cloned OS/2 system to the target system (for more details about the donor system, please refer to Chapter 5.4, “Configuring the Donor System” on page 168). Afterwards, a two- phase process configures the target system (LAN Requester computer name and domain name, IP address, and so on).

To install the donor system image, a NetView DM/2 change file is used. This change file, FIIBM750.PRO, contains commands and relevant parameters to successfully run `REPLICAT.EXE`. In our scenario, the parameters are:

`/C:IBM750`	The class ID of the donor system is IBM750.
`/B:$(SA)\IMG\FINANCE`	Location where the class IBM750 bundle files are located. In this example, its the NetView DM/2 Server’s SHAREA drive and from there the \IMG\FINANCE directory. The

	\IMG\FINANCE directory contains all class IDs for the finance department.
`/F:$(SA)\RSP\FiIBM750.RSP`	Command file with drive stanza directives. `REPLICAT.EXE`, by default, only restores images to a drive with the same partitiion size or larger. In our scenario, we are replicating the donor system image to a smaller partition size. This response file is used to overwrite the minimum size value to enable the restoration process overriding the default partition size.
`/L:$(SB)\LOG\$(WorkStatName)`	Fully qualified path to the log file directory. The `$(WorkStatName)` parameter indicates that the name of the log file directory is identical to the name of the class ID. The log file is stored at NetView DM/2 Server's SHAREB's \LOG\<*workstation name*> directory.

Log File Location

Care should be taken on the log file location. Since `REPLICAT.EXE` always uses the class name for the log file, a unique location should be provided to each workstation. Otherwise, `REPLICAT.EXE` encounters problems accessing the log file. In our scenario, NetView DM/2 provides several variables> One of them is the `$WorkstatName` variable which is used to identify the unique directory name where the log file will be stored.

`REPLICAT.EXE` does not create the directory specified in the /L: parameter. The administrator needs to create the directory before running `REPLICAT.EXE`.

The NetView DM/2 profile mentioned above, FIIBM750.PRO, is shown in Figure 67 on page 232. The response file used by `REPLICAT.EXE`, FIIBM750.RSP, is show in Figure 68 on page 232.

```
TargetDir = "C:\IBMNVDM2"
CompName Len = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
      Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
/L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 67. FIIBM750.PRO NetView DM/2 Profile

```
DRIVE
        DONORLETTER C:
        TARGETLETTER C:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER D:
        TARGETLETTER D:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER E:
        TARGETLETTER E:
        MINSIZE 18
ENDDRIVE
```

Figure 68. FIIBM750.RSP Response File

7.2.5 Configuring the Workstation

All previous steps in this process are pretty much straight forward. They apply to all workstations. However, each workstation has unique values that identifies it to the network. Depending on several factors, these values can be the workstation name, MAC address, IP address, IP mask, IP router address, and so on.

In the scenario illustrated in this chapter, the following values are being configured at the target workstation:

- NetView DM/2 workstation name
- NetView DM/2 Server name
- LAN Requester computer name
- LAN Requester domain name
- LAN Requester workstation description
- MAC address
- PCOMOS2 destination address (Mainframe MAC address)
- PCOMOS2 physical unit ID (PU ID)
- IP host name
- IP address
- IP subnet mask
- IP default router address
- IP net address for the default router address
- IP net address mask
- IP domain name
- IP domain name server address
- IP socks name server
- IP socks name server address

To simplify the naming conventions, the NetView DM/2 workstation name, LAN Requester computer name and the IP host name will be set identical.

Normally, a single process would change all values mentioned above. However, there are a few considerations.

1. `CONFIGUR.EXE`, the main tool to accomplish this task, requires OS/2 Presentation Managers to be active.

2. There are a few values that must be configured after REPLICAT.EXE was run and before the system was rebooted.
3. REPLICAT.EXE is run from boot diskette to avoid having OS/2 Presentation Manager be active.

Considering these limitations and requirements, all workstation values have to be set in at least two phases. To avoid having a process for each workstation, this scenario will use two phases:

1. The first phase will extract all the relevant workstation configuration values from an ASCII file based on the NetView DM/2 workstation name. It must be assured that the minimum values are set successfully reboot without human intervention. The minimum set of values that need to be set before rebooting the system are:
   - NetView DM/2 workstation name
   - NetView DM/2 Server name
   - Locally Administered MAC address

   MAC Address Considerations with PROBE/CONFIGUR

   If you explicitly define MAC addresses in the PROTOCOL.INI file, this value should also be set. However, at the present time, PROBE.EXE and CONFIGUR.EXE do not accept double quotes (") as part of a FI variable.

   Therefore, CUBE.CMD is used to add the locally administered MAC address (LAA), since the donor system image was not configured with a predefined MAC address. CUBE.CMD can also be used to change the an existing MAC Address if it was set in the donor system image.

   To accomplish the above mentioned parameter value changes, WSCONFIG.CMD is used. WSCONFIG.CMD inputs workstation values from the ASCII file, W:\WSDETAILS\LOCALWS.DAT, where W: is the redirected server's SHAREA drive. LOCALWS.DAT uses fixed length fields. After verifying all values, WSCONFIG.CMD calls CUBE.CMD to set the minimum values on the target workstation, including the Locally Administered Address (LAA).

   WSCONFIG.CMD requires two parameters:

   - Computer name

     The computer name is the one defined for the NetView DM/2 Client name, LAN Requester computer name, and IP host name. This name identifies workstation-specific data stored in the W:\WSDETAILS\LOCALWS.DAT data file.

- Process mode

  Since pristine and migration processes differ from each other, the process mode determines which process is used.

  In pristine scenarios, `WSCONFIG.CMD` inputs the data file and then sets the values for NetView DM/2 workstation name, NetView DM/2 Server name, and the locally administered MAC address (LAA). A REXX script is created during the process and stored in the NetView DM/2 Server's SHAREB area where log files after kept for each workstation. The REXX command file name is `WSDETAILS.CMD`.

2. The previously created `WSDETAILS.CMD` REXX command file executes `CONFIGUR.EXE` after the system has been rebooted in phase one.

Note: When `WSCONFIG.CMD` is used in the migration process, it only sets the values for NetView DM/2 workstation name, NetView DM/2 Server name, and the locally administered MAC address. For full details, refer to the migration chapters.

The REXX code for `WSCONFIG.CMD` is illustrated the next figure. `CUBE.CMD` is illustrated in Appendix D, "CUBE Source Code and Documentation" on page 407. The NetView DM/2 change profile files are shown in Figure 74 on page 241 and Figure 75 on page 241.

```
/*****************************************************************/
/* WSCONFIG.CMD:                                                 */
/* This procedure read configuration details from an ASCII       */
/* file, with fixed length field, then pass these values to      */
/* CONFIGUR.EXE to process. The following item are configured by */
/* this program:                                                 */
/*                                                               */
/*      1.- LAN Requester Computername                           */
/*      2.- LAN Requester default Domain                         */
/*      3.- LAN Requester Computername Description               */
/*      4.- IP Hostname (same as LAN Requester Computername)     */
/*      5.- IP Address                                           */
/*      6.- IP Mask                                              */
/*      7.- IP Default Router                                    */
/*      8.- IP Net Address for Router                            */
/*      9.- IP Net Mast                                          */
/*     10.- IP Domain name                                       */
/*     11.- IP Domain Name Server address                        */
/*     12.- IP Hostname Socks Name Server                        */
/*     13.- IP Socks Name Server address                         */
/*     14.- NIC MAC Address                                      */
/*     15.- PCOM Destination MAC Address (Mainframe MAC address) */
/*     16.- PCOM Physical Unit ID                                */
/*                                                               */
/*****************************************************************/

/* Initialize local variables... */

WSDataBase = "W:\WSDetails\LocalWS.DAT"  /* File with Workstation details */
CurrentLine = " "                        /* Holds the line being read from */
                                         /* WSDataBase */
FoundRecord = 0                          /* Boolean variable regarding */
                                         /* record found in WSDataBase */
ConfigurCtrlFile = ""                    /* Location for the REXX script */
                                         /* that sets FI Variable values
                                         /* and runs CONFIGUR.EXE */
ConfigurFileRec = ""                     /* Holds record to be written */
                                         /* into ConfigurCtrlFile */
ComputerName = ""                        /* LAN Requester Computername, */
                                         /* NvDM/2 Workstation Name and */
                                         /* Hostname Name */
NVDMServerName = ""                      /* NVDM/2 Server Name */
ComputerClass = ""                       /* Computer Class */
ClassPath = ""                           /* Path for the Computer Class */
                                         /* Bundle files */
ShareB = ""                              /* NetView DM/2 ShareB drive */
LRDomain = " "                           /* LAN Requester default Domain */
LRDescription = " "                      /* LAN Requester Computername */
                                         /* Description */
IPAddress = " "                          /* IP Address */
IPMask = " "                             /* IP Mask */
IPRouter = " "                           /* IP default Router */
IPNetAddress = " "                       /* IP Net Address */
IPNetMask = " "                          /* IP Router Net Mask */
IPDomain = " "                           /* IP Domain Name */
IP_DNS = " "                             /* IP Domain Name Server */
IPHostSocksNameServer = " "              /* IP Sostname Socks Name Server */
                                         /* address */
```

Figure 69. WSCONFIG.CMD REXX Command File (Part 1 of 5)

```
IPSocksNameServer = " "                  /* IP Socks Name Server address */
LAMacAddress = " "                       /* Localy Administ. MAC Address */
PCommDestinationSAP = " "                /* Destination Address for PComm */
                                         /* SNA Session */
PCommIdentifier = " "                    /* PU Id for PComm SNA Session */
WSConfig_rc = 0                          /* Return Code */
tmp_rc = 0                               /* Temp to hold return codes */
tmp_various = ""                         /* Temp to hold various values */

/* Get NVDM/2 WorkStation Name */

parse UPPER arg ComputerName ProcessMode dummy    /* dummy will separate */
                                                  /* any unwanted        */
                                                  /* parameters from the */
                                                  /* ComputerName */
If ComputerName = ""
then do
   WSConfig_rc = 2
   say "Required Computer Name was not provided."
   say "The systax is WSCONFIG computername process-mode"
   say "Example: WSCONFIG UZ1175E PRISTINE"
   say "Contact LAN Administrator to obtain a Computer Name."
end /* then do */

If (ProcessMode <> "PRISTINE") & (ProcessMode <> "MIGRATION")
then do
   WSConfig_rc = 2
   say "Required Process Mode was not provided."
   say "The systax is WSCONFIG computername process-mode"
   say "Example: WSCONFIG UZ1175E PRISTINE"
end /* then do */

/* Define the location of the REXX script to run CONFIGUR.EXE */
ConfigurCtrlFile = "X:\LOG\"ComputerName"\WSDetails.cmd"

/* Find WorkStation details */

if WSConfig_rc = 0
then do
   say "Searching for "ComputerName" details in data file "WSDataBase"..."
   Do While Lines(WSDataBase) & FoundRecord = 0
      CurrentLine = LineIn(WSDataBase)
      if Strip(SubStr(CurrentLine,1,15),B," ") = ComputerName
      then do
         LRDomain = Strip(SubStr(CurrentLine,16,15),B," ")
         LRDescription = Strip(SubStr(CurrentLine,31,48),B," ")
         IPAddress = Strip(SubStr(CurrentLine,79,15),B," ")
         IPMask = Strip(SubStr(CurrentLine,93,15),B," ")
         IPRouter = Strip(SubStr(CurrentLine,109,15),B," ")
         IPNetAddress = Strip(SubStr(CurrentLine,124,15),B," ")
         IPNetMask = Strip(SubStr(CurrentLine,139,15),B," ")
         IPDomain = Strip(SubStr(CurrentLine,154,64),B," ")
         IP_DNS = Strip(SubStr(CurrentLine,218,15),B," ")
         IPHostSocksNameServer = Strip(SubStr(CurrentLine,233,64),B," ")
         IPSocksNameServer = Strip(SubStr(CurrentLine,297,15),B," ")
         LAMacAddress = SubStr(CurrentLine,312,12)
         NVDMServerName = Strip(SubStr(CurrentLine,324,15),B," ")
         PCommDestinationSAP = SubStr(CurrentLine,339,12)
```

Figure 70. WSCONFIG.CMD REXX Command File (Part 2 of 5)

```
         PCommIdentifier = Strip(SubStr(CurrentLine,351,8),B," ")
         FoundRecord = 1
         say "Details for "ComputerName" have been found."
      end /* do */
   end /* do */
   if FoundRecord = 0
   then do
      WSConfig_rc = 2
      say "No data for Computer Name '"ComputerName"' was found in '",
           WSDataBase"'."
      say "Contact LAN Administrator to obtain a Computer Name."
   end /* do */
end /* do */


/*************************************************************************/
/* Modify C:\IBMNVDM2\IBMNVDM2.INI to configure the NVDM/2 WorkStation */
/* Name and NVDM/2 Server Name. Also do the same to                    */
/* E:\IBMNVDM2\IBMNVDM2.INI.                                           */

if BaseConf_rc = 0
then do
   if Stream(TmpCubeCmdsFile,'c','query exists') <> ""
   then do
      Call Stream TmpCubeCmdsFile,'c','close' /* Just in case the file */
                                              /* is open (this is the  */
                                              /* first time the file is */
                                              /* used.                  */
      'DEL 'TmpCubeCmdsFile               /* To have a new clean file */
   end /* then do */
   TmpCubeFileRec = 'RepLine "ServerName" WITH "ServerName     = '||,
                    NVDMServerName'" ( *ID '
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   TmpCubeFileRec = 'RepLine "ClientName" WITH "ClientName     = '||,
                    ComputerName'" ( *ID '
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   Call Stream TmpCubeCmdsFile,'c','close'
   CubeParamenters = TmpCubeCmdsFile' C:\IBMNVDM2\IBMNVDM2.INI '||,
                     'C:\IBMNVDM2\IBMNVDM2.CUB'

   /* Make sure file exists... */
   if Stream('C:\IBMNVDM2\IBMNVDM2.INI','c','query exists') <> ""
   then BaseConf_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else BaseConf_rc = '4'x

   CubeParamenters = TmpCubeCmdsFile' E:\IBMNVDM2\IBMNVDM2.INI '||,
                                    ' E:\IBMNVDM2\IBMNVDM2.CUB'

   /* Make sure file exists... */
   tmp_various = Stream('E:\IBMNVDM2\IBMNVDM2.INI','c','query exists')
   if (BaseConf_rc = 0) & (tmp_various <> "")
   then BaseConf_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else BaseConf_rc = '4'x
end /* do */

if BaseConf_rc = 0
then do
   if Stream(TmpCubeCmdsFile,'c','query exists') <> ""
```

Figure 71. WSCONFIG.CMD REXX Command FIle (Part 3 of 5)

```
      then do
      'DEL 'TmpCubeCmdsFile             /* To have a new clean file  */
   end /* then do */
   TmpCubeFileRec = 'AddLine ~   NETADDRESS = "'LAMacAddress||,
                    '"~ ( After ~[IBMTOK_nif]~ IFNEW'
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   Call Stream TmpCubeCmdsFile,'c','close'
   CubeParamenters = TmpCubeCmdsFile' C:\IBMCOM\PROTOCOL.INI '||,
                     'C:\IBMCOM\PROTOCOL.CUB ( DLM ~ )'

   /* Make sure file exists... */
   if Stream('C:\IBMCOM\PROTOCOL.INI','c','query exists') <> ""
   then BaseConf_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else BaseConf_rc = '4'x
end /* do */


/***********************************************************/
/* Last, but not least, delete the temp file being use to */
/* hold the CUBE.CMD commands.                            */

if Stream(TmpCubeCmdsFile,'c','query exists') <> ""
then 'DEL 'TmpCubeCmdsFile


/*************************************************************/
/* Once the Workstation details have been found and minimun */
/* values have been set, write a REXX script file to run    */
/* CONFIGUR.EXE with all the Workstation values.            */

if (WSConfig_rc = 0) & (ProcessMode = "PRISTINE")
then do
   if Stream(ConfigurCtrlFile,'c','query exists') <> ""
   then do
      Call Stream ConfigurCtrlFile,'c','close' /* Just in case the   */
                                               /* file is open (this */
                                               /* is the first time  */
                                               /* the file is used. */
      'DEL 'ConfigurCtrlFile              /* To have a new clean file */
   end /* do */

   ConfigurFileRec = "/* Sets FI Variable values and runs CONFIGUR.EXE */"
   WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = " "
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "'CONFIGUR.EXE "||,
                     'E:'ComputerName||,
                     '/B:W:\IMG\CONFIG\'ComputerName||,
                     '/W:C:\'||,
                     '"/V:{COMPUTERNAME}='ComputerName'"'||,
                     '"/V:{DOMAIN}='LRDomain'"'||,
                     '"/V:{SRVCOMMENT}='LRDescription'"'||,
                     '"/V:{PCOMMDESTSAP}='LAMacAddress'"'||,
                     '"/V:{PCOMMID}='PCommIdentifier'"'||,
                     '"/V:{HOSTNAME}='ComputerName'"'||,
                     '"/V:{IPDOMAIN}='IPDomain'"'||,
                     '"/V:{IP_DNS}='IP_DNS'"'||,
```

Figure 72. WSCONFIG.CMD REXX Command File (Part 4 of 5)

```
                     '"/V:{IPADDRESS}='IPAddress'"'||,
                     '"/V:{IPMASK}='IPMask'"'||,
                     '"/V:{IPROUTER}'IPRouter'"'||,
                     '"/V:{IPNETADDRESS}='IPNetAddress'"'||,
                     '"/V:{IPNETMASK}='IPNetMask'"'||,
                     '"/V:{IPHOSTSOCKSNAMESERVER}='IPSocksNameServer'"'||,
                     '"/V:{IPSOCKSNAMESERVER}='IPSocksNameServer'"'||,
                     '/L:X:\LOG\'ComputerName'"'||,
                     '/X:4'
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "tmp_rc = rc"
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "exit tmp_rc"
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)

   /* All lines have been written to the REXX script... */
   /* So, close the file.                               */

   if WSConfig_rc = 0
   then WSConfig_rc = Stream(ConfigurCtrlFile,'c','close')
end /* do */


/**********************************************************/
/* If there were no errors, return a CID rc for succesful */
/* completition with reboot request.                      */

if WSConfig_rc = 0
then do
   WSConfig_rc = 'fe00'x
   say 'WSCONFIG.CMD has exited with return code 0, CID rc = fe00.'
end
else do
   say 'WSCONFIG.CMD failed with rc = 'WSConfig_rc'.'
end
exit WSConfig_rc
```

Figure 73. WSCONFIG.CMD REXX Command File (Part 5 of 5)

The process illustrated in above is executed from the NetView DM/2 change profile files, PRISTINE1.PRO (Figure 74) and PRISTINE2.PRO (Figure 75).

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = FLATDATA
      GlobalName = IBM.CONFIGUR.PRISTINE1.REF.1
      Description = Configure a Pristine WS (Phase 1)
End

Section Install
Begin
      Program = $(SA)\EXE\WSCONFIG.CMD
      Parms = "$(WorkStatName) PRISTINE"
      WorkingDir = C:\
End
```

Figure 74. PRISTINE1.PRO NetView DM/2 Change Profile File

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = FLATDATA
      GlobalName = IBM.CONFIGUR.PRISTINE2.REF.1
      Description = Configure a Pristine WS (Phase 2)
End

Section Install
Begin
      Program = $(SB)\LOG\$(WorkStatName)\WSDETAILS.CMD
      WorkingDir = C:\
End
```

Figure 75. PRISTINE2.PRO NetView DM/2 Change Profile File

The first NetView DM/2 change file, IBM.CONFIGUR.PRISTINE1.REF.1, executes first, followed by a reboot. After the reboot, the NetView DM/2 change file IBM.CONFIGUR.PRISTINE2.REF.1 follows to complete the pristine installation process.

Chapter 8. Migrating from OS/2 Warp Connect (OS/2 Warp 3)

This chapter illustrates the scenario migrating from OS/2 Warp Connect (OS/2 Warp 3) to OS/2 Warp 4, using the RDT tools. Probably, the most common scenario we have deal with are migrations from OS/2 Warp Connect (OS/2 Warp 3) to OS/2 Warp 4.0.

In our migration scenario illustrated in this chapter, the OS/2 Warp Connect system comes with most of the networking features shipped with the product, including MPTS, TCP/IP, LAN Requester, Peer Services, and the NetWare client. Additional software products installed on the machine are mentioned later.

Relevant configuration information from our "old system" is retrieved and stored on our software distribution server. This data can be used later to configure the new system.

Figure 76 on page 244 shows two migration scenarios:

- Migration to the same system
- Migration to a new system

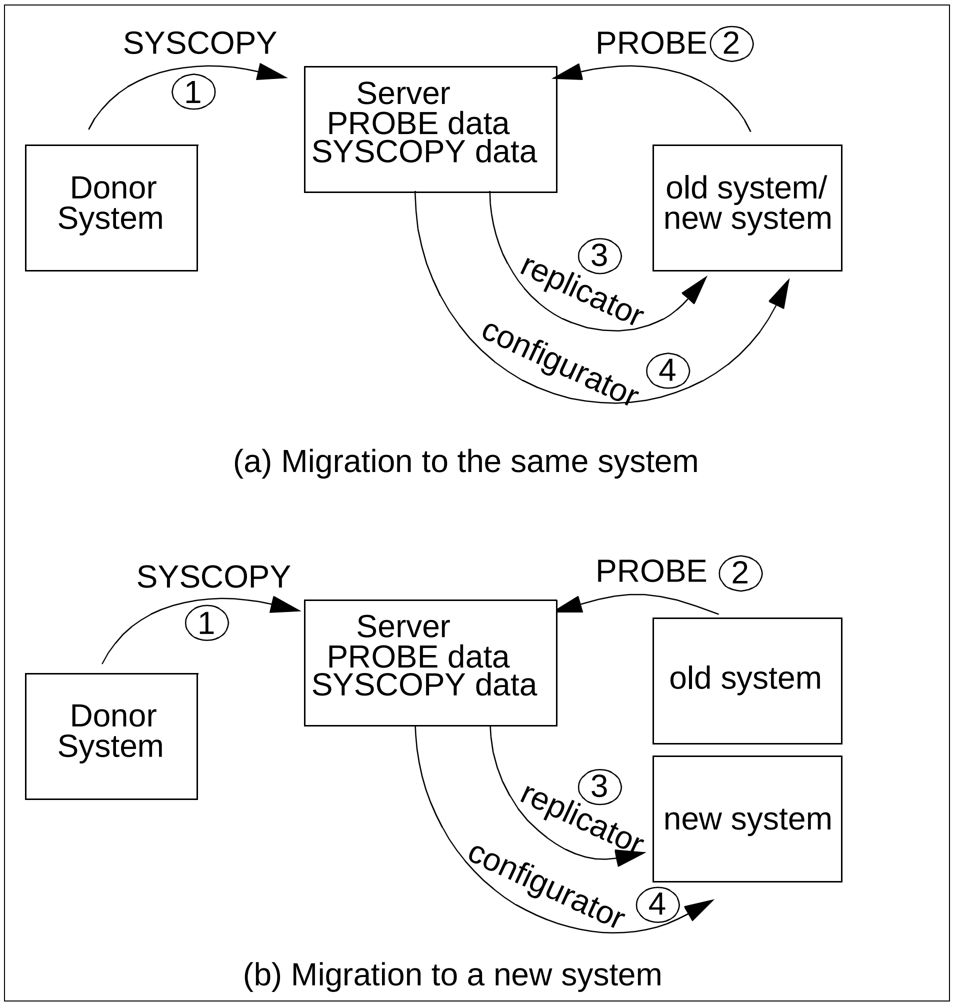


Figure 76. Various Migration Scenarios

As shown in Figure 76, both scenarios use exactly the same RDT tools. The difference is the target system. In the first scenario, the target system is the same as the old one. In the second scenario, we have a new system that gets configuration information retrieved from the old system.

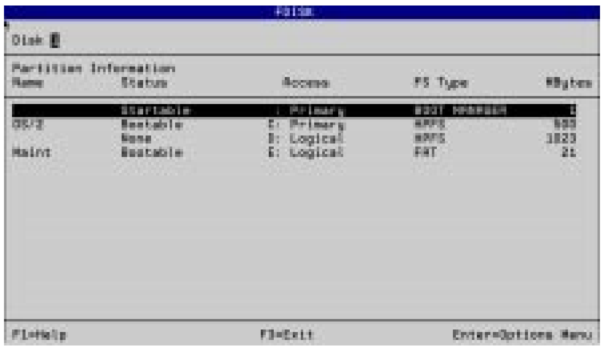
The following list details our migration scenario:

The system to be migrated is installed with OS/2 Warp Connect (including WIN-OS/2).

- IBM OS/2 LAN Adapter and Protocol Support (MPTS)
  CSD level: WR08600
- IBM TCP/IP Version 3.0 for OS/2
  CSD level: WR08600
- IBM OS/2 LAN Requester Version 4.00.1
  CSD level: IP08000
- IBM Peer for OS/2 Version 1.00
  CSD level: IP02000
- Netware Client for OS/2 Version 2.11
- IBM Net View DM/2 Change Control Client Version 2.10.1
  CSD level: IP02000
- IBM Communications Manager/2 Version 1.11
  CSD level: WR06150
- Netscape Navigator for OS/2 Warp Version 2.02

As shown in Figure 77, the hard disk is partitioned with Boot Manager and three additional partitions:

- Primary partition C: for the base operating system including components
- Logical partition D: for user data
- Boot logical partition E: (maintenance partition)



```
Disk 1

Partition Information
Name        Status           Access        FS Type        MBytes

            Startable         : Primary    BOOT MANAGER        1
OS/2        Bootable        C: Primary     HPFS              503
            None            D: Logical     HPFS             1023
Maint       Bootable        E: Logical     FAT                21
```

Figure 77. OS/2 Warp Connect Hard Disk Information (FDISK)

The following list details specific configuration information and other important data for each product installed in our system:

LAN Requester configuration information:

Computer name:	UZ1175J
Domain name:	ITSCAUS
Local Admin:	USERID/PASSWORD LOCADMIN/LOCADMIN

Communications Manager/2 configuration information:

Network ID:	USIBMAU1
Local node:	AU168000
Local node ID:	05d00000
LAN dest. address:	400021041064

Netview DM/2 configuration information:

CC client:	UZ1175J
CC domain:	NVDM1175

Novell NetWare configuration information:

Netware server:	uz1175nw
Context (OU):	ps1175

Peer Services user accounts definitions:

```
User ID       Password       Priviledge
---------------------------------------
GUEST
LOCADMIN      LOCADMIN       ADMIN
USERID        PASSWORD       ADMIN
UZ1175J1      UZ1175J1       USER
UZ1175J2      UZ1175J2       USER
```

Peer Services share definitions:

```
Share Name    Path           Max.Users
--------------------------------------
SHAREDIR      D:\SHAREDIR    No Limit
```

TCP/IP configuration information:

IP address:	9.3.1.139
Subnet mask:	255.255.255.0
Router/Getway:	9.3.1.74

DNS/NameServer:	9.3.1.74
Host name:	UZ1175J

The tools we used in our migration Surinamer are the very same ones we used in the pristine installation scenario. However, some of them are used in a slightly different way.

The first step in the migration process consists of preparing a customized donor system with all software and configuration required according to your needs. Refer Chapter 5.4, “Configuring the Donor System” on page 168 for detailed information on how to do it.

Once the donor system is customized the way you want it, create an system image using the `SYSCOPY` tool and store the image on the software distribution server.

To do a migration, you need to obtain configuration data from our old system, to configure the new system. The `PROBE` tool retrieves data from the old system, gathers configuration information, generates bundle files, and stores the files on the code server. As soon as the donor system image is generated by the `SYSCOPY` tool, it can be installed on the target workstation using the replicator tool, `REPLICAT`, which inputs the bundle files generated by the `PROBE` tool and configures the new system the way the old system was configured. In addition to the RDT tools, we use other tools to accomplish the migration process.

NetView DM/2 is our software distribution server. You need to have the set of NVDM/2 boot diskettes to start the target workstation. For information on how to do it, refer to Chapter 3.4, “Creating Boot Diskettes with NetView DM/2 Client” on page 129.

Our disk partitioning and formatting REXX command file `DISKPREP.CMD` takes care of repartitioning and formatting of the target system.

`CUBE.CMD` is used to modify some files at the target system before the target systems reboots for the first time after the donor system image has been distributed.

Additional self-written programs help enhancing the migration and configuration tasks. These programs are described in detail in the corresponding sections.

8.1 Step-by-Step Guidance

The following sections illustrate the migration process in detail in a step-by-step fashion.

8.1.1 Creating the Donor System Image Using SYSCOPY

As mentioned before, SYSCOPY creates a donor system image and stores it on the NetViewDM/2 Server. For detailed information on how to use it, refer to “Creating an Image of the Donor System” on page 185.

8.1.2 Creating the Configuration Bundle Files Using PROBE

After the donor system image has been created using SYSCOPY, we need to retrieve configuration information from the old system using PROBE. Table 26 lists the file names and the lines that carry configuration information that need to be retrieved by PROBE.

Note: In this step, we do not migrate the PROTOCOL.INI file which contains the MPTS configuration information. Migrating this file from the old systems using PROBE can result in missing important protocol and network adapter configuration information. We assume the donor system immage was configured with the correct protocol and adapter information. CUBE will take care of locally administered addresses (LAA) in a later process.

Table 26. Configuration Information to be Retrieved by PROBE

Category	Description	Location	File	Line
LAN Requester	Computer Name	C:\IBMLAN	IBMLAN.INI	`COMPUTERNAME=`
	Default Domain	C:\IBMLAN	IBMLAN.INI	`DOMAIN=`
	Name description	C:\IBMLAN	IBMLAN.INI	`SRVCOMMENT=`
	NetBIOS resources	C:\IBMLAN	IBMLAN.INI	`net1 =`
TCP/IP	Host name	C:\	CONFIG.SYS	`SET HOSTNAME=`
	IP address	C:\MPTN\BIN	SETUP.CMD	`ifconfiglan0`
	IP net mask	C:\MPTN\BIN	SETUP.CMD	`netmask`
	IP default router	C:\MPTN\BIN	SETUP.CMD	`route add default`
	Net address for router	C:\MPTN\BIN	SETUP.CMD	`route add net`
	IP Domain Name	C:\MPTN\ETC	RESOLV2	`domain`
	Domain Name Server Address	C:\MPTN\ETC	RESOLV2	`nameserver`
	IP Domain Name (DOS)	C:\TCPIP\DOS\ETC	RESOLV	`domain`
	Domain Name Server add. (DOS)	C:\TCPIP\DOC\ETC	RESOLV	`nameserver`
NVDM/2	Client Name	E:\IBMNVDM2	IBMNVDM2.INI	`ClientName`
	Server Name	E:\IBMNVDM2	IBMNVDM2.INI	`ServerName`
NetWare Requester	All	C:\\	NET.CFG	`*`

8.1.2.1 PROBE Scripts

In this section, we describe the `PROBE` scripts used to retrieve configuration information from the old system.

Table 27 on page 250 lists the names of `PROBE` command script files (.PRB) used in our scenario. A brief description about the purpose of each `PROBE` file is provided, too.

Table 27. PROBE Command Script Files

PROBE Script File	Category	Description
GETALL3.PRB	General	Retrieves all the configuration information from the old system.
LAN.PRB	LAN Requester	Retrieves all the LAN Requester configuration information.
NVDM.PRB	NVDM/2	Retrieves the NVDM/2 configuration information.
TCP.PRB	TCP/IP	Retrieves the TCP/IP configuration information.
NWNETCFG.PRB	Netware Requester	Copies entire NET.CFG.
LRDRIVE.PRB	LAN Requester	Gets IBMLAN drive from CONFIG.SYS and sets an FI variable.
LRCOMNAM.PRB	LAN Requester	Gets the computer name.
LRDOMNAM.PRB	LAN Requester	Gets the domain name.
LRNET1.PRB	LAN Requester	Gets values for adapter one.
LRCOMMEN.PRB	LAN Requester	Gets the workstation description.
MAINTDRV.PRB	NVDM/2	Sets an FI variable for the maintenance partition drive.
NVDMINI.PRB	NVDM/2	Gets NVDM/2 Client name and server name.
TCPDRIV.PRB	TCP/IP	Sets the TCP/IP drive and MPTN drive and sets two FI variables.
TCPRSLV2.PRB	TCP/IP	Copies RESOLV2 file.
TCPRSLV.PRB	TCP/IP	Copies RESOLV file.
TCPHSTNM.PRB	TCP/IP	Gets the host name.
TCPSETUP.PRB	TCP/IP	Gets IP address, IP mask, default router and default router net address.

As illustrated in Chapter 1.3, “The PROBE Tool” on page 15, each script file can contain inclusions of other command stream files using the `INCLUDE` command. We will notice later, that GETALL3.PRB, LAN.PRB, NVDM.PRB, and TCP.PRB are script files that consist of the inclusion of other command stream files depending on the category to which they belong.

Figure 78 on page 251 shows a chart that illustrates these nested command stream files.

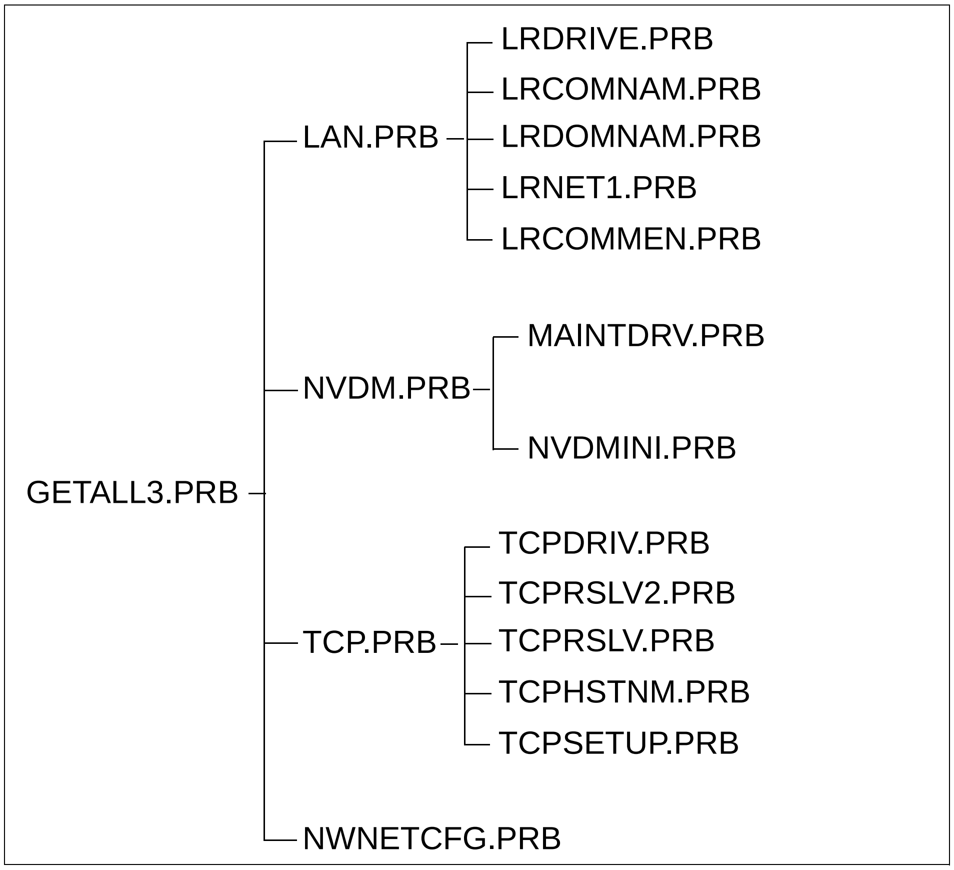


Figure 78. Structure of Nested Command Stream Files (GETALL3.PRB)

In the following sections, we provide a detailed description of each PROBE file we used in our migration scenario. Each section is divided according to the category of the information retrieved by the PROBE tool.

8.1.2.2 General Category Probe Command Stream File

GETALL3.PRB is the only script file categorized as general. This file includes LAN.PRB, NVDM.PRB, TCP.PRB and NWNETCFG.PRB, and is in charge of gathering all configuration information described in Table 26 on page 249. The configurator, CONFIGUR, will use these bundle files generated with this command stream file to configure the target workstation.

Figure 79 shows the GETALL3.PRB file.

```
// This script will gather all the information regarding Lan requester,
// NVDM,TCP/IP and Netware requester
// FIVARS {PRBPATH} = path where all the prb files are located

INCLUDE "{PRBPATH}\LAN.PRB"
INCLUDE "{PRBPATH}\NVDM.PRB"
INCLUDE "{PRBPATH}\TCP.PRB"
INCLUDE "{PRBPATH}\NWNETCFG.PRB"
```

Figure 79. GETALL3.PRB Command Stream File

As you can see in Chapter 8.1.2.7, “Executing PROBE” on page 267, we have defined the PRBPATH FI variable that will point to the directory where all the PROBE script files reside.

Editing PROBE Script Files

If you create your own PROBE command stream files don’t forget to end the file with a Return character, meaning you need to press the **Enter** key after the last line in the script. Otherwise, the PROBE command might fail to execute.

8.1.2.3 LAN Requester Category Probe Command Stream File

The following PROBE command stream files belong to the LAN Requester category:

- LAN.PRB

  This command stream file retrieves relevant LAN Requester configuration information. It includes a set of command stream files which we describe later.

  Figure 80 on page 253 shows the LAN.PRB file.

```
// LAN.PRB. Lan Requester configuraiton

INCLUDE "{PRBPATH}\LRDRIVE.PRB"     // Get IBMLAN Drive from config.sys
                                    //and set FI variable IBMLANDRIVE
INCLUDE "{PRBPATH}\LRCOMNAM.PRB"    // Computername
INCLUDE "{PRBPATH}\LRDOMNAM.PRB"    // Domain
INCLUDE "{PRBPATH}\LRNET1.PRB"      // NET1
INCLUDE "{PRBPATH}\LRCOMMEN.PRB"    //Computer name comment
```

Figure 80. LAN.PRB Command Stream File

- LRDRIVE.PRB

  This command stream file is used to set a FI variable called IBMLANDRIVE, which will be used by other command stream files. The IBMLANDRIVE FI variable points to the drive where the \IBMLAN directory is located.

  As shown in Figure 81, the value of this variable is retrieved through searching the CONFIG.SYS file.

```
// Category: LAN Server/Requester
// Name:    IBMLAN Drive
// Version:  1.0
// Description: This will set the FI Variable IBMLANDRIVE
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will determine which drive IBMLAN is installed on and
//set the FI variable IBMLANDRIVE
// FIVARS:{BOOTDRIVE} =  Drive system is booted
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   EDITLINE
      SOURCESTRING
         SETFIVAR "IBMLANDRIVE"
         LINEPATTERN "IBMLAN\\NETPROG\\NETWKSTA.200 /I:"
         MATCHPATTERN "[A-Z]{1}:"
      ENDSOURCESTRING
TARGETSTRING
         LOCATEDACTION NOTHING
         NOTLOCATEDACTION NOTHING
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 81. LRDRIVE.PRB Command Stream File

- LRCOMNAM.PRB

  This PROBE command stream file retrieves the LAN Requester computer name out of IBMLAN.INI. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE.PRB command stream file.

  Figure 82 shows the LRCOMNAM.PRB file.

```
// Category: LAN Server/Requester
// Name:    Computer Name
// Version:  1.0
// Description: Migrate Computername from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches
//                    "COMPUTERNAME" from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "COMPUTERNAME"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 82. LRCOMNAM.PRB Command Stream File.

- LRDOMNAM.PRB

  This PROBE command stream file retrieves the LAN Requester default domain name out of IBMLAN.INI. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE.PRB command stream file.

  Figure 83 shows the LRDOMNAM.PRB file.

```
// Category:LAN Server/Requester
// Name:    Domain Name
// Version:     1.0
// Description: Migrate Domainname from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:    3.0, 4.0, 5.0
// Details:  This will migrate the "Domainname" line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is
//installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "DOMAIN"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 83. LRDOMNAM.PRB Command Stream File

- LRNET1.PRB

  This PROBE command stream file retrieves the entire NET1 line out of IBMLAN.INI. This line contains relevant information about NetBIOS resources assigned to LAN Requester, such as NetBIOS commands, NetBIOS names and NetBIOS sessions. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE.PRB command stream file.

  Figure 84 shows the LRNET1.PRB file.

```
// Category: LAN Server/Requester
// Name:    NET1 Line
// Version:  1.0
// Description: Migrate NET1 Line from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire NET1 line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is
//installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "NETWORKS"
         KEY
            SOURCE "NET1"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 84. LRNET1.PRB Command Stream File

- LRCOMMEN.PRB

  This PROBE command stream file retrieves the LAN Requester workstation comment out of IBMLAN.INI. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE.PRB command stream file.

  Figure 85 shows the LRCOMMEN.PRB file.

```
// Category: LAN Server/Requester
// Name:    Computer Name Description
// Version:  1.0
// Description: Migrate computername description from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches "SRVCOMMENT"
//from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "PEER"
         KEY
            SOURCE "SRVCOMMENT"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 85. LRCOMMEN.PRB Command Stream File

8.1.2.4 NetView DM/2 Category Probe Scripts

The following PROBE command stream files belong to the NetView DM/2 category:

- NVDM.PRB

  NVDM.PRB retrieves relevant information regarding the NVDM/2 Client software in the maintenance partition. Later on, in “SETNVDM Using CUBE” on page 278, we show that the NVDM/2 Client information for the C: partition must be retrieved in a different way. NVDM.PRB includes a set of command stream files which we describe later.

  Figure 80 shows the NVDM.PRB file

```
INCLUDE "{PRBPATH}\MAINTDRV.PRB"   // Set Maintenance partition drive
INCLUDE "{PRBPATH}\NVDMINI.PRB" // ClientName and ServerName of
                                   //IBMNVDM2.INI
```

Figure 86. NVDM.PRB Command Stream File

- MAINTDRV.PRB

The command stream file, MAINTDRV.PRB, is used to set an FI variable called MAINTDRIVE, which will be used by NVDMINI.PRB command stream file. The MAINTDRIVE FI variable points to the drive where the maintenance partition is located.

As shown in Figure 87, we manually set the value of the variable.

```
// Category: Maintenance Partition
// Version:  1.0
// Description: Manually set Maintenance partition Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables MAINT
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance Partition

FIVAR
   NAME "MAINTDRIVE"
   VALUE "E:"
ENDFIVAR
```

Figure 87. MAINTDRV.PRB Command Stream File

- NVDMINI.PRB

  The command stream file, NVDMINI.PRB, migrates the NVDM/2 Client name and server name by extracting the values from the IBMNVDM2.INI file that resides in the maintenance partition. Note that this PROBE file is using the MAINTDRIVE FI variable created with the MAINTDRV.PRB command stream file.

  Figure 88 shows the NVDMINI.PRB file.

```
// Category: NVDM
// Name:    WORKSTATION AND SERVER NAMES
// Version:  1.0
// Description: Migrate WORKSTATION AND SERVER NAMES FROM IBMNVDM2.INI
// Details:  This will migrate the lines that matches ServerName and
ClientName
//              and ClientName in the IBMNVDM2.INI of the maintenance
partition
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance partition

TEXTFILE
   SOURCE  "{MAINTDRIVE}\IBMNVDM2\IBMNVDM2.INI"
   REPLACELINE
      SOURCELINE
        PATTERN "ServerName"
      ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE
        PATTERN "ClientName"
      ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 88. NVDMINI.PRB Command Stream File.

8.1.2.5 TCP/IP Category PROBE Scripts

The following PROBE command stream files belong to the TCP/IP category:

- TCP.PRB

  TCP.PRB retrieves relevant TCP/IP configuration information. It includes a set of command stream files which we describe later.

  Figure 89 shows the TCP.PRB file.

```
INCLUDE "{PRBPATH}\TCPDRIV.PRB"   // Drive letters TCPIPDRIVE&MPTNDRIVE
INCLUDE "{PRBPATH}\TCPRSLV2.PRB"  // Resolv2 file
INCLUDE "{PRBPATH}\TCPRSLV.PRB"   // Resolv file
INCLUDE "{PRBPATH}\TCPHSTNM.PRB"  // Host name
INCLUDE "{PRBPATH}\TCPSETUP.PRB"  // setup.cmd
```

Figure 89. TCP.PRB Command Stream File

- TCPDRIV.PRB

This command stream file sets two FI variables called TCPDRIVE and MPTNDRIVE, which will be used by other command stream files. The TCPDRIVE and MPTNDRIVE FI variables point to the drives where the \TCPIP and \MPTN directories are located.

As shown in Figure 90, we manually set the values of these variables.

```
// Category: TCPIP
// Name:    Installed Drives
// Version:  1.0
// Description: Manually set TCPIP Drive and MTPN Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables TCPIPDRIVE and MPTNDRIVE
// FIVARS:  {TCPIPDRIVE} = Drive where TCPIP is installed
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

FIVAR
   NAME "TCPIPDRIVE"
   VALUE "C:"
ENDFIVAR

FIVAR
   NAME "MPTNDRIVE"
   VALUE "C:"
ENDFIVAR
```

Figure 90. TCPDRIV.PRB Command Stream File

- TCPRSLV2.PRB

  This command stream file migrates the RESOLV2 file from the old system to the target system.

  Figure 91 shows the TCPRSLV2.PRB file.

```
// Category: TCPIP
// Name:    RESOLV2
// Version:  1.0
// Description: Migrate RESOLV2
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV2
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

COPYFILES
   SOURCE "{MPTNDRIVE}\MPTN\ETC"
   INCLUDEFILES
      PATTERN "RESOLV2"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 91. TCPRSLV2.PRB Command Stream File

- TCPRSLV.PRB

  This command stream file migrates the RESOLV file from the old system to the target system.

  Figure 92 shows the TCPRSLV.PRB file.

```
// Category: TCPIP
// Name:    RESOLV
// Version:  1.0
// Description: Migrate RESOLV
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV (for DOS sesions)
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

COPYFILES
   SOURCE "{TCPIPDRIVE}\TCPIP\DOS\ETC"
   INCLUDEFILES
      PATTERN "RESOLV"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 92. TCPRSLV.PRB Command Stream File

- TCPHSTNM.PRB

This PROBE file migrates the TCP/IP host name extracting it from the CONFIG.SYS file.

Figure 93 shows the TCPHSTNM.PRB file.

```
// Category: TCPIP
// Name:    HOSTNAME
// Version:  1.0
// Description: Migrate HOSTNAME from config.sys
// From:  2.0, 3.0
// To:  2.0, 3.0
// Details:  This will migrate the entire line that matches HOSTNAME
from config.sys
// FIVARS:  {BOOTDRIVE} = Drive system is booted from

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   REPLACELINE
      SOURCELINE PATTERN "HOSTNAME" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 93. TCPHSTNM.PRB Command Stream File

- TCPSETUP.PRB

  This PROBE file migrates the IP address, IP mask, IP default router, and the router net address by extracting this data from the \MPTN\ETC\SETUP.CMD file.

  Figure 94 shows the TCPSETUP.PRB file.

```
// Category:    TCP/IP
// Name:        SETUP.CMD
// Version:     1.0
// Description: This will modify strings on SETUP.CMD
// From:        4.0
// To:          4.1
// Details:     This will get the values for net address, mask and router
// FIVARS:      {MPTNDRIVE}  = Drive where MPTN is installed
TEXTFILE
  SOURCE  "{MPTNDRIVE}\MPTN\BIN\SETUP.CMD"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "ifconfig lan0 "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "ifconfig lan0"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "netmask"
         MATCHPATTERN "\ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "netmask"
         STRINGPATTERN "\ {1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
         STRINGSCOPE LAST
      ENDTARGETSTRING
   ENDEDITLINE
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add default "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add default"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add net "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add -net 9"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 94. TCPSETUP.PRB Command Stream File

We can see that the TCPSETUP.PRB command stream file is considerably different to the ones we discussed before. This command stream file is more complex because the SETUP.CMD files of TCP/IP 3.0 (the one shipped with OS/2 Warp Connect) and TCP/IP 4.1 (the one we migrate to) are slightly different.

We need the data of the older version to match the data in the newer one. This is accomplished by retrieving strings within certain lines using the SOURCESTRING command and replacing those strings in the target file with the TARGETSTRING command.

Syntax of PROBE Files

Remember that you use FI regular expressions on the target system, but you use XPG4 Extended Regular Expressions on the source system. You must be careful with the syntax you are using for each case. Refer to the appropiate documentation to see the differences on the syntax.

The following two figures help us to analyze the two different versions of SETUP.CMD. Figure 95 shows the SETUP.CMD file of TCP/IP 3.0, whereas Figure 96 shows the SETUP.CMD file of TCP/IP 4.1.

```
route -fh
arp -f
ifconfig lan0 9.3.1.146 netmask 255.255.255.0
REM ifconfig lan1
REM ifconfig lan2
REM ifconfig lan3
REM ifconfig lan4
REM ifconfig lan5
REM ifconfig lan6
REM ifconfig lan7
REM ifconfig sl0
route add net 9.3 9.3.1.74 1
route add default 9.3.1.74 1 *
route add net 9 9.3.1.74 1 *
ipgate off
```

Figure 95. SETUP.CMD of TCP/IP 3.0

```
route -fh
arp -f
ifconfig lo 127.0.0.1
ifconfig lan0 9.3.1.146 netmask 255.255.255.0
REM ifconfig lan1
REM ifconfig lan2
REM ifconfig lan3
REM ifconfig lan4
REM ifconfig lan5
REM ifconfig lan6
REM ifconfig lan7
rem ifconfig sl
route add default 9.3.1.74 -hopcount 1 *
route add -net 9 9.3.1.74 -netmask 255.0.0.0 -hopcount 1 *
```

Figure 96. SETUP.CMD of TCP/IP 4.1

If you compare both files and analyze the TCPSETUP.PRB file, you notice that the lines that have a leading asterisk are the ones that contain the strings that must be copied with the SOURCESTRING and TARGETSTRING commands.

8.1.2.6 NetWare Requester Category PROBE Scripts

The following PROBE command stream files belong to the NetWare Requester category:

- NWNETCFG.PRB

  We only need one script file for this category. The NWNETCFG.PRB file migrates the NET.CFG file from the old system to the target system.

  Figure 97 shows the NWNETCFG.PRB file.

```
/ Category: NETWARE
// Name:    NET.CFG
// Version:  1.0
// Description: Migrate NET.CFG
// Details:  This will migrate the file NET.CFG for Netware requester
// FIVARS:  {BOOTDRIVE} = Drive system is booted

COPYFILES
   SOURCE "{BOOTDRIVE}\"
   INCLUDEFILES
      PATTERN "NET.CFG"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 97. NWNETCFG.PRB Command Stream File

8.1.2.7 Executing PROBE

Now that we all PROBE script files are defined we are going to use, we are ready to execute the PROBE command to extract the configuration data from the old system. The way we are going to do this is by executing the PROBE command in all old systems that have the data we want to store using NetView Distribution Manager /2.

The Figure 98 shows the change file profile used to execute PROBE at the desired OS/2 Warp Connect workstations.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.PROBE.GETALL3.REF.1
End

Section Install
Begin
      Program = SA:\exe\probe.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
/b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir
/l:$(sb)\log\$(WorkStatName) /x:4 $(sa)\rsp\probe\getall3.prb"
End
```

Figure 98. Change File Profile PROBE3.PRO

The single-line PROBE command executed from the NVDM/2 Server is the following:

PROBE Execution

```
probe /v:{prbpath}=$(sa)\rsp\probe
      /e:$(WorkStatName)
      /b:$(sa)\img\config\$(WorkStatName)
      /w:$(sb)\workdir /l:$(sb)\log\$(WorkStatName)
      /x:4
      $(sa)\rsp\probe\getall3.prb
```

where:

`/v:{prbpath}=$(sa)\rsp\probe`	Defines an FI variable that points to the subdirectory that contains all the PROBE command stream files. In our particular case this subdirectory is D:\SHAREA\RSP\PROBE at the NVDM/2 Server. This FI variable is used by GETALL3.PRB, LAN.PRB, NVDM.PRB and, TCP.PRB,
`/e:$(WorkStatName)`	Specifies the client's name that corresponds to the NVDM/2 workstation name. This means that the name of the bundle files generated by PROBE correspond to the NVDM/2 workstation name with extensions of 000, 001 and so on. This parameter is used with the configurator tool at a later time.
`/b:$(sa)\img\config\$(WorkStatName)`	Specifies the fully qualified path where bundle files generated by PROBE will be stored, in our particular case, in D:\SHAREA\IMG\CONFIG\$(WorkStat Name).
`/w:$(sb)\workdir`	The working directory used by PROBE will be, in our particular case, in D:\SHAREB\WORKDIR.
`/l:$(sb)\log\$(WorkStatName)`	Specifies the fully qualified path and name of the log file.

`/x:4`	Specifies the highest logging level for test purposes. You may change it later.
`$(sa)\rsp\probe\getall3.prb`	We call GETALL3.PRB to extract the data from all systems.

8.1.3 CM2PCOMM and COPYCONF Utilities

We make use of these REXX programs to migrate from Communications Manager/2 to eNetwork Personal Communications for OS/2 Warp. If we had had Personal Communications on the old system, we would have been able to perform the migration by simply extracting the *.WS configuration files from the old system using the `PROBE` tool and copy them to the target system.

The program, `CM2PCOMM.CMD` extracts the Communications Manager/2 configuration information from the old system and converts it to the Personal Communications format.

It also creates a subdirectory in D:\SHAREA\IMG\CONFIG with the name of the NVDM/2 workstation. This subdirectory will have the information extracted by `CM2PCOMM` as well as the bundle files created by `PROBE`.

Refer to Chapter 6.1.1, “CM2PCOMM.CMD” on page 189, for detailed information about this utility.

`COPYCONF` is a small REXX program that copies the configuration files created by `CM2PCOMM` to the target system. Refer to Chapter 6.1.3, “COPYCONF.CMD” on page 205, for more information on this utility.

8.1.3.1 CM2PCOMM.PRO Change Profile

The following figure, Figure 99, displays the NetView DM/2 change profile, CM2PCOMM.PRO, that executes the `CM2PCOMM.CMD` REXX command file.

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.CM2PCOMM.INST.REF.4
     Description="Migration from CM/2 to Personal Communications"
End

Section Install
Begin
      Program =  SA:\EXE\CM2PCOMM.CMD
      Parms= "C:\CMLIB $(WorkStatName)"
End
```

Figure 99. Change File Profile CM2PCOMM.PRO

8.1.3.2 COPYCONF.PRO Change Profile

The following figure, Figure 100, displays the NetView DM/2 change profile, COPYCONF.PRO, that executes the `COPYCONF.CMD` REXX command file.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.COPYCONF.REF.1
End

Section Install
Begin
      Program = SA:\exe\copyconf.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 100. Change File Profile COPYCONF.PRO

8.1.4 GETNVDM

We make use of this REXX program in order to obtain two values from the old system: The NVDM/2 Client name and the NVDM/2 Server name. This program creates a profile that CUBE will use to set these parameters on the donor system.

These IBMNVDM2.INI parameters must be modified at the target workstations before they reboot for the first time since all systems that were installed with the donor system image carry identical parameters. If we don't modify them, we will not be able to execute the rest of the commands at the target systems from the NVDM/2 Server since the NVDM/2 Clients won't come up due to identical names.

When the systems reboot, some error messages will be shown because of some duplicate parameters. Since we only want to connect to the NVDM/2 Server from the production system to run CONFIGUR to change rest of the parameters of the workstations we don't need to pay attention to these upcoming error messages.

As mentioned before, GETNVDM generates a profile called IBMNVDM2.PRO that contains the NVDM/2 parameters. This profile will be stored in \SHAREA\IMG\CONFIG directory, where the images of PROBE and CM2PCOMM are stored.

This profile will be used by SETNVDM to store this data at the new systems before the first reboot is performed. This allows the clients to come up with the proper NetView DM/2 Client names.

The following figure, Figure 101 on page 272, displays the contents of the GETNVDM.CMD REXX command file.

```
/* REXX program to extract the NVDM/2 Client and server name    */
/* from a workstation. The information is stored in a           */
/* CUBE profile in the correct directory on the NVDM/2 Server.  */

parse upper arg workstationname .
ECHO OFF
targetpath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'

logdir = 'X:\LOG\'||workstationname

RCode = 0
server = ''
client = ''

RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\GETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
if RCode == 0 then do
   do while lines(nvdmpath||'\IBMNVDM2.INI')
      line.linenum = Linein(nvdmpath||'\IBMNVDM2.INI')
      select
         when word(line.linenum, 1)=='ServerName' then server=line.linenum
         when word(line.linenum, 1)=='ClientName' then client=line.linenum
      otherwise NOP
      end  /* select */
   end /* do */

if RCode == 0 then do
   RC = STREAM(logdir||'\GETINFO.LOG', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'logdir||'\GETINFO.LOG'
   end /* do */
   RC = STREAM(targetpath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'targetpath||'\IBMNVDM2.PRO'
   end /* do */

   cubeline = 'RepLine "ServerName" WITH "'server'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'

   cubeline = 'RepLine "ClientName" WITH "'client'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'
   Call Stream targetpath||'\IBMNVDM2.PRO','c','close'
end /* do */
if RCode == 0 then do
   'ECHO GETINFO ended successful >> 'logdir||'\GETINFO.LOG'
end /* do */
else do
   'ECHO GETINFO ended unsuccessful with >> 'logdir||'\GETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
Call Stream logpath||'\GETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 101. GETNVDM.CMD REXX Command File

8.1.4.1 GETNVDM.PRO Change Profile

The following figure, Figure 102, displays the NetView DM/2 change profile, GETNVDM.PRO, that executes the GETNVDM.CMD REXX command file.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.GETNVDM3.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\getnvdm.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 102. Change File Profile GETNVDM3.PRO

8.1.5 DISKPREP.CMD

As mentioned earlier, besides the RDT, we are going to make use of other tools like DISKPREP and CUBE.

In our specific case we are going to use DISKPREP in the target systems to repartition and format the the hard disk on the target systems. We are doing so because we already backed up all the configuration information we need to keep from the old systems by using PROBE.

Old System=Target System?

If the "old system" is also "target system" and you plan to use DISKPREP.CMD, make sure that relevant information from your old systems is retrieved and backed up, since DISKPREP will format the partitions.

DISKPREP allows you to delete and create all necessary partitions in a given system using response files. It also takes care of the system reboot and continues to format all the partitions before exiting.

For detailed information about how DISKPREP works, refer to Appendix C.6, “DISKPREP.CMD” on page 390.

Because DISKPREP will be executed from the Netview DM/2 Server at the target systems, we need to boot the target system with the NVDM/2 boot diskettes. Refer to Chapter 3.4, “Creating Boot Diskettes with NetView DM/2 Client” on page 129.

Keep in mind that DISKPREP requires REXX support to be enabled. Before you execute DISKPREP you need to enable REXX support from the NetView DM/2 boot diskettes. For details on how to enable REXX support, refer to “Enabling REXX Support” on page 226.

8.1.5.1 Executing DISKPREP.CMD

The execution of DISKPREP from the NVDM/2 Server requires two steps:

- Enabling REXX support
- Execution of DISKPREP itself

Figure 103 and Figure 104 show the change file profiles used in the NVDM/2 Server.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = REXX Support for Pristine Clients
End

Section Install
Begin
      Program = SA:\EXE\cmd.exe
      Parms =  /C $(SA)\exe\detrexx.cmd $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 103. Change File Profile LOADREXX.PRO to Load REXX Support

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir)
/L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 104. Change File Profile DISKPREP.PRO

Note that in Figure 103 the REXX enablement requires a batch file called DETREXX.CMD. Figure 64 on page 227 displays this file.

8.1.6 Replicator

Once the hard disk on the target system(s) have been partitioned and formated, we are ready to copy the donor system image stored on the NVDM/2 Server using the replicator tool, REPLICAT.EXE.

Figure 105 shows the change file profile used in the NVDM/2 Server to run the REPLICAT.EXE command at the target system(s).

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
      Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
               /L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 105. FiIBM750.PRO Change File Profile

The following box illustrates the REPLICAT command executed at the NVDM/2 Server:

REPLICAT Execution

```
REPLICAT.EXE  /C:IBM750
              /B:$(SA)\IMG\FINANCE
              /F:$(SA)\RSP\FiIBM750.RSP
              /L:$(SB)\LOG\$(WorkStatName)
```

where:

/C:IBM750	For this specific scenario the class ID of the donor system is IBM750.
/B:$(SA)\IMG\FINANCE	Indicates the directory where the class IBM750 bundle files (generated by SYSCOPY) are located. This directory corresponds to D:\SHAREA\IMG\FINANCE on the NVDM/2 Server. In this specific scenario, this directory contains all class IDs for the finance department. The class ID itself identifies the system brand and model.

`/F:$(SA)\RSP\FiIBM750.RSP`	Specifies a command file with drive stanza directives. `REPLICAT.EXE`, by default, only restores images to a drive with the same partition size or larger. Since our scenario is replicating to a smaller partition size, this file is used to specify the minimum allowed partition size to make the restore and overwrite the default partition size.
`/L:$(SB)\LOG\$(WorkStatName)"`	Specifies the directory name where the log file will be located. `REPLICAT.EXE` names the log files with the same name as the class ID. In this particular case, it puts the log file in D:\SHAREB\LOG\ $(WORKSTATNAME) directory on the NVDM/2 Server.

The following figure shows the response file used in this scenario:

```
DRIVE
        DONORLETTER C:
        TARGETLETTER C:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER D:
        TARGETLETTER D:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER E:
        TARGETLETTER E:
        MINSIZE 18
ENDDRIVE
```

Figure 106. FIIBM750.RSP Response File

Log File Location

Care should be taken on the log file location. As `REPLICAT.EXE` always uses the class name for the log file, a unique location should be given to each workstation. Otherwise, `REPLICAT.EXE` will have problems accessing the log file. In our scenario, NetView DM/2 provides several variables, where the workstation name is one of them. This variable is used to identify the unique directory name where the log file will be located.

In addition, `REPLICAT.EXE` will not create the directory specified in the `/L:` parameter. The user has to create the directory before running `REPLICAT.EXE`.

8.1.7 SETNVDM Using CUBE

We use `CUBE` after the replication has finished and before the configurator is executed. With this REXX program, you can access and edit ASCII files at the target workstation.

The file that we need to modify at the target system(s) is IBMNVDM2.INI, which contains the NVDM/2 configuration, such as the NVDM/2 Client name and NVDM/2 Server name.

These two parameters (especially the NVDM/2 workstation name) in the IBMNVDM2.INI file must be modified before the target systems reboot for the first time because all the systems that were installed with the donor system image have the same parameters. If we don't modify them, we wont be able to execute the rest of the commands at all the target systems from the NVDM/2 Server since the NVDM/2 Clients won't come up due to replicated names.

The reason to use `CUBE` instead of `CONFIGUR` is that the latter one requires Presentation Manager to run. Presentation Manager means the target systems would have needed to reboot prior to those changes and would make it impossible to connect to the NetView DM/2 Server meaning no `CONFIGUR` commands could be run at the target workstations.

After `CUBE` changes the NVDM/2 parameters, the target systems will be able to connect after reboot and will have Presentation Manager up and running to run `CONFIGUR`, which changes the rest of parameters gathered by `PROBE`.

For migration to OS/2 Warp 4.0 purposes, the `SETNVDM.CMD` REXX command file was written. This REXX program makes use of `CUBE` and changes parameters on the target systems. SETNVDM makes use of the IBMNVDM2.PRO profile that was generated by GETNVDM.

Figure 107 shows the contents of SETNVDM.CMD.

```
/* REXX program to set the information extracted by GETNVDM */

parse upper arg workstationname .

ECHO OFF

sourcepath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'
mptspath = 'C:\IBMCOM'

logdir = 'X:\LOG\'||workstationname

RCode = 0

RC = STREAM(logdir||'\SETINFO.LOG', 'C', 'QUERY EXISTS')
if RC <> '' then do
   'DEL 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(sourcepath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.PRO does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */

if RCode == 0 then do
   cubearg = sourcepath||'\ibmnvdm2.pro 'nvdmpath||'\IBMNVDM2.INI'
   RC = 'cube.cmd'(cubearg)
   if RC <> 0 then RCode = RC
   if RCode <> 0 then do
      say 'One or more errors were found executing the changes'
      say 'to 'nvdmpath||'\IBMNVDM2.INI and 'mptspath||'\PROTOCOL.INI'
      RCode = 4
   end
end /* do */
if RCode == 0 then do
   'ECHO SETINFO ended successful >> 'logdir||'\SETINFO.LOG'
end /* do */
else do
   'ECHO SETINFO ended unsuccessful with >> 'logdir||'\SETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
Call Stream logpath||'\SETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 107. SETNVDM.CMD REXX Command File

8.1.7.1 SETNVDM Change File Profile

The following figure, Figure 108, shows the change file profile used at the NVDM/2 Server:

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.SETINFO.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\setnvdm.cmd
      Parms = "$(WorkStatName)"
      PhaseEnd = yes
End
```

Figure 108. SETNVDM3.PRO Change File Profile

After the execution of SETNVDM, the machine must be rebooted. After the reboot, when Presentation Manager is active, CONFIGUR can be executed at the target systems to change the rest of the parameters.

8.1.8 Configurator

Once CUBE changes the NVDM/2 workstation name, the target system is ready to receive the rest of the configuration information that was extracted by PROBE from the old system using the CONFIGUR configurator tool.

Running CONFIGUR more than Once

At the time this redbook was writen, CONFIGUR had some problems updating the information stored in the bundle files when it was run more than once in the same machine. This is due to the FI version used. To avoid this problem, the machine must be rebooted every time CONFIGUR is executed.

Figure 109 shows the change file profile used in the NVDM/2 Server to run the CONFIGUR command at the target system(s).

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.CONFIGUR.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\configur.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
              /b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir
              /l:$(sb)\log\$(WorkStatName) /x:4"
End
```

Figure 109. CONFIGUR3.PRO Change File Profile

The following command is executed at the NVDM/2 Server:

CONFIGUR Execution

```
CONFIGUR /v:{prbpath}=$(sa)\rsp\probe
         /e:$(WorkStatName)
         /b:$(sa)\img\config\$(WorkStatName)
         /w:$(sb)\workdir
         /l:$(sb)\log\$(WorkStatName)
          /x:4
```

where:

`/v:{prbpath}=$(sa)\rsp\probe`

Defines an FI variable that points to the subdirectory that contains all the `PROBE` command stream files. In our case, this directory is D:\SHAREA\RSP\PROBE on the NVDM/2 Server. This FI variable is used by GETALL3.PRB, LAN.PRB, NVDM.PRB, and TCP.PRB.

`/e:$(WorkStatName)`

The client name corresponds with the NVDM/2 workstation name. This name was defined during the execution of `PROBE`, which means that the configurator will search the bundle files generated by probe that correspond to the NVDM workstation name with the extensions of 000, 001 and so on.

`/b:$(sa)\img\config\$(WorkStatName)`

This is the location of the bundle files generated by `PROBE`. In our case, this is the D:\SHAREA\IMG\CONFIG\$(WorkStatName) directory.

`/w:$(sb)\workdir`

The working directory used by `CONFIGUR`. In our case, this is the D:\SHAREB\WORKDIR directory.

`/l:$(sb)\log\$(WorkStatName)`

Specifies the fully qualified path of the log file directory.

`/x:4` Specifies the highest logging level for test purposes. You can change it later.

8.2 Navigating through NetView DM/2

In this section, we summarize the activities necessary at the NetView DM/2 Server to perform a migration from OS/2 Warp Connect to OS/2 Warp 4. We assume a connectivity between the clients, also known as CC clients (Change Control clients), and server, also known as CC server, has been established either through boot diskettes or through the maintenance partition. Otherwise, you won’t be able to install so-called change files to the target workstations.

Before you continue, verify that the CC client is attached to the CC server. This can be done through the command line by issuing `CDM LIST_WS` or through the PM dialog. Return to the CDM Catalog window and initiate software installation on the CC client using the CDM Catalog window.

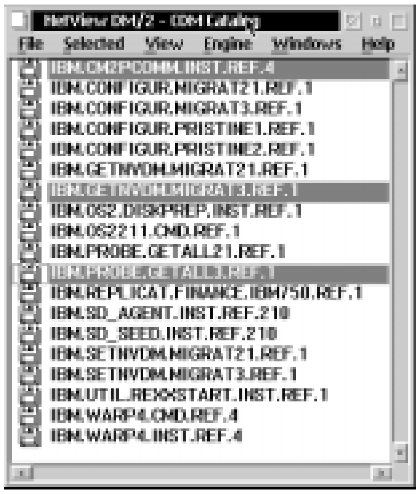
8.2.1 Retrieving Configuration Data from the Old System

To extract configuration information from the old systems, perform the following steps:

1. As shown in Figure 110 on page 283, at the CDM Catalog window, select the following change files and prepare them to install as a corequisite group:
   - IBM.CM2PCOMM.INST.REF.4

- IBM.PROBE.GETALL3.REF.1
- IBM.GETNVDM.MIGRATE3.REF.1

.



NetView DM/2 - CDM Catalog
File Selected View Engine Windows Help

IBM.CM2PCOMM.INST.REF.4
IBM.CONFIGUR.MIGRAT21.REF.1
IBM.CONFIGUR.MIGRAT3.REF.1
IBM.CONFIGUR.PRISTINE1.REF.1
IBM.CONFIGUR.PRISTINE2.REF.1
IBM.GETNVDM.MIGRAT21.REF.1
IBM.GETNVDM.MIGRAT3.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.OS2211.CMD.REF.1
IBM.PROBE.GETALL21.REF.1
IBM.PROBE.GETALL3.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SD_AGENT.INST.REF.210
IBM.SD_SEED.INST.REF.210
IBM.SETNVDM.MIGRAT21.REF.1
IBM.SETNVDM.MIGRAT3.REF.1
IBM.UTIL.REXXSTART.INST.REF.1
IBM.WARP4.CMD.REF.4
IBM.WARP4.INST.REF.4

Figure 110. Selecting Change Files to Install on the Old System

2. Click on **Selected** from the action bar to display the pull-down menu.

   Select **Install** from the pull-down menu. The Install window will be displayed with all the selected objects listed.

   Note: If you are uncertain whether or not the selected software exists on the target workstation, select **Force** as installation method. NetView DM/2 catalogs distributed software in the DB2 database, thus preventing a software product from being distributed more than once. Force overrides this.

3. Define installation options.

   Select **Options** from the Install window. The Options screen will be displayed.

   Select **Install as a coreq group**.

   The purpose of "corequisite groups" is to bundle the installation requests of several objects together into one request. Reboot requests of the single objects are queued until a change file with the statement PhaseEnd=Yes or the last object of the corequisite group is installed.

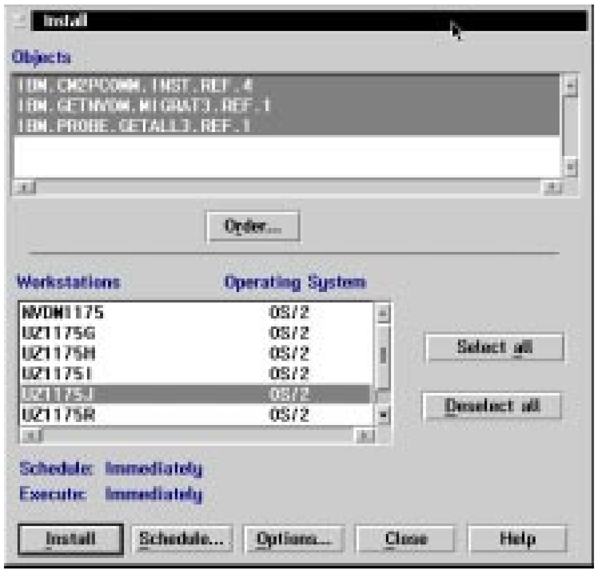
   Corequisite groups may consist of a maximum of seven change files; they are used to install several pieces of software that depend on each other. If one of the installs in a coreq group fails, the complete group will receive the status "failed", even if installs prior to the failed one were successful.

Click on **Set** to return to the Install window.

4. At the Install window, as shown in Figure 111, select the client's name from the workstation list to install the objects on the client workstation and then select **Install** to execute the command.

   You will receive a message pop-up; check if everything went OK and select **OK** to continue.

   Select **Close** at the Install screen to return to the Catalog menu.



IBM.CM2PCOMM.INST.REF.4
IBM.GETNVDM.MIGRAT3.REF.1
IBM.PROBE.GETALL3.REF.1

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

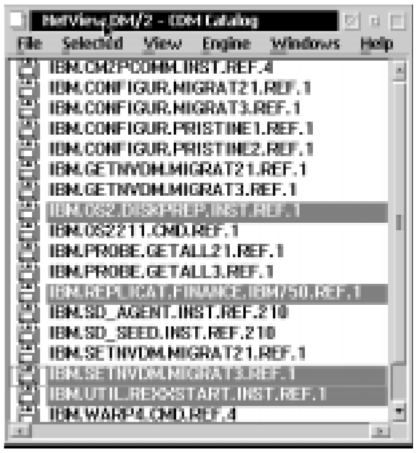
Figure 111. Install Change Files as Corequisite Group

8.2.2 Replicating and Configuring the New System

Once all necessary configuration information is retrieved from the old system, we can start the replication and configuration process on the target workstations. Perform the following steps:

1. As shown in Figure 112 on page 285, at the CDM Catalog window, select the following change files and prepare them to install as a corequisite group:
   - IBM.OS2.DISKPREP.INST.REF.1

- IBM.REPLICAT.FINANCE.IBM750.REF.1
- IBM.SETNVDM.MIGRAT3.REF.1
- IBM.UTIL.REXXSTART.INS.REF.1



IBM.CM2PCOMM.INST.REF.4
IBM.CONFIGUR.MIGRAT21.REF.1
IBM.CONFIGUR.MIGRAT3.REF.1
IBM.CONFIGUR.PRISTINE1.REF.1
IBM.CONFIGUR.PRISTINE2.REF.1
IBM.GETNVDM.MIGRAT21.REF.1
IBM.GETNVDM.MIGRAT3.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.OS2211.CMD.REF.1
IBM.PROBE.GETALL21.REF.1
IBM.PROBE.GETALL3.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SD_AGENT.INST.REF.210
IBM.SD_SEED.INST.REF.210
IBM.SETNVDM.MIGRAT21.REF.1
IBM.SETNVDM.MIGRAT3.REF.1
IBM.UTIL.REXXSTART.INST.REF.1
IBM.WARP4.CMD.REF.4

Figure 112. Selecting Change Files to Install on Target Workstations

2. Click on **Selected** from the action bar to display the pull-down menu.

   Select **Install** from the pull-down menu. The Install window will be displayed with all the selected objects listed.

   Note: If you are uncertain whether or not the selected software exists on the target workstation, select **Force** as the installation method. NetView DM/2 catalogs distributed software in the DB2 database, thus preventing a software product from being distributed more than once. Force overrides this.

3. Define the installation order.

   Click on **Order...** to display the Install Order window.

   Select **IBM.UTIL.REXXSTART.INST.REF.1**

   Select **Up** to move it up to the first position.

   Move other objects to arrange in this order:

   IBM.UTIL.REXXSTART.INS.REF.1
   IBM.OS2.DISKPREP.INST.REF.1

IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SETNVDM.MIGRAT3.REF.1

4. At the Install Order window, select **OK** to return to the Install window.

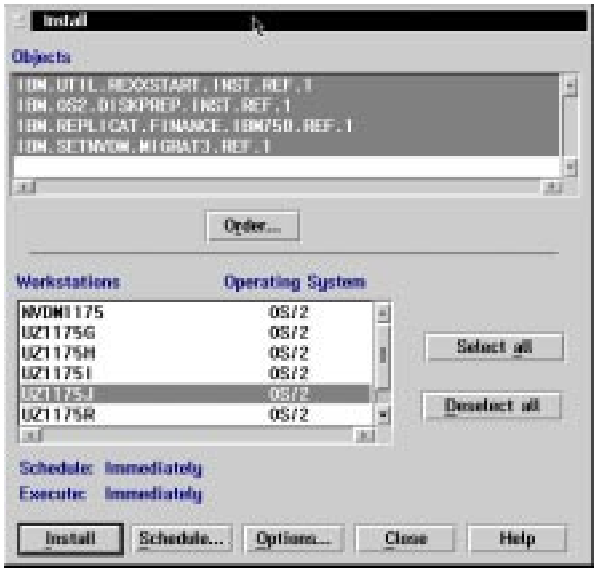
   Select client's name from the workstation list to install the objects on client workstation.

5. Define installation options.

   Select **Options** from the Install window. The Options screen will be displayed.

   Select **Install as a coreq group**.

.



IBM.UTIL.REXXSTART.INST.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SETNVDM.MIGRAT3.REF.1

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

Figure 113. Install Change Files as Corequisite Group

6. At the Install windows, as shown in Figure 113, select **Install** to execute the command.

   You will receive a message pop-up; check if everything went OK and select **OK** to continue.

   Select **Close** at the Install screen to return to the Catalog menu.

7. Check the status of the install request for your client.

At the CDM Catalog window:

Select **Engine** to display the menu.

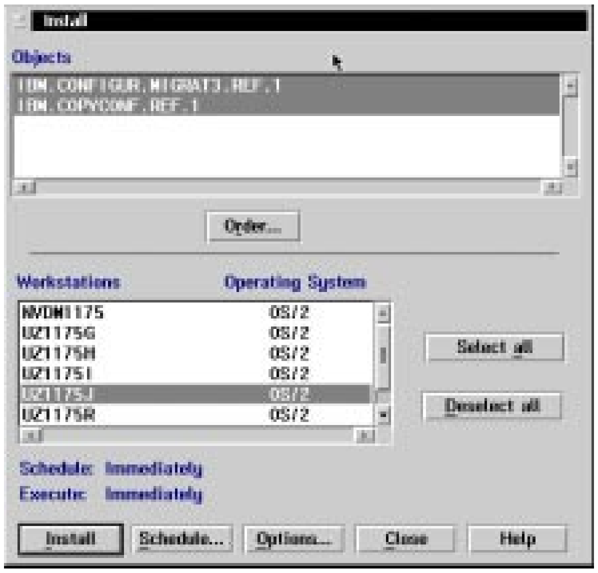
Select **All Pending Requests** to display the Pending Request menu.

Select the client's name.

Select **Details** to display the details.

8. You may immediately send another installation request to the previously selected target workstations. The next installation step configures the target workstations. As shown in Figure 114, send the following change files as corequisite group:

   IBM.CONFIGUR.MIGRAT3.REF.1
   IBM.COPYCONF.REF.1


Install

Objects

IBM.CONFIGUR.MIGRAT3.REF.1
IBM.COPYCONF.REF.1

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

Figure 114. Installing Configuration Change Files at the Target Workstations

After the previously mentioned installation requests, the target workstations are installed and configured properly.

Chapter 9. Migrating from OS/2 V2.11

This chapter illustrates the migration scenario from OS/2 V2.11 to OS/2 Warp 4, using the Rapid Deployment Tools. There are many customers out there who did not migrate their OS/2 V2.11 workstation to OS/2 Warp Connect, for whatever reason.

The scenario illustrated in this chapter deals with the migration from OS/2 V2.11 directly to OS/2 Warp 4.0. In our example, the OS/2 V2.11 system comes with a lot of networking features:

- MPTS
- TCP/IP 2.0 including NFS
- LAN Requester 3.01

Additional software products installed on the machine are:

- Communications Manager/2 Version 1.3
- NetView DM/2 CC client

Relevant configuration information from our "old system" is retrieved and stored on our software distribution server. This data can be used later to configure the new system.

The hard disk is set up with the following partitions:

- Boot manager
- Primary C: partition of 400 MB for the base operating system including components
- Logical D: partition of 600 MB for applications and data
- Boot logical E: partition of 20 MB for the maintenance partition

The maintenance partition on this system is set up in the same way as described in 5.4.6, “Setting Up the Maintenance Partition” on page 175. Please refer to this chapter for further information about maintenance partitions.

The NetView DM/2 CC Client Version 2.1 is installed on the C: partition and is used to allow `PROBE` to extract workstation configuration data. This kind of NetView DM/2 CC client must be present because `PROBE` needs the Presentation Manager up and running to run successfully. You cannot use the NetView DM/2 CC client from boot diskettes or from the maintenance partition.

Command Line Interpreter from the NVDM/2 Client

When you want to run the command line interpreter inside the NVDM/2 Client, you have to create a change file on the server that runs the command line interpreter Version 2.11. You cannot run the CMD version 4 like in Warp3.

9.1 Extracting the Workstation Configuration

Figure 115 on page 291 shows the procedure of performing a migration to the same system and for a migration to a new system.

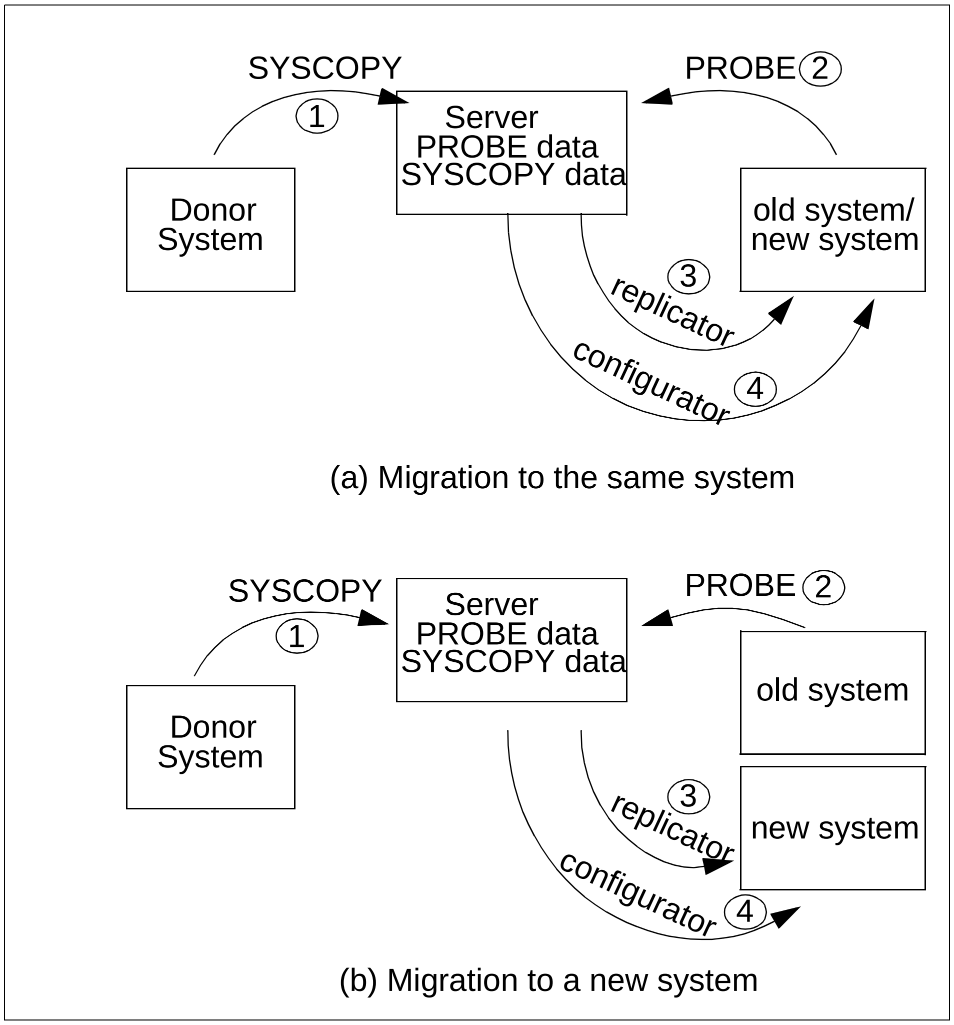


Figure 115. Various Migration Scenarios

As you can see in this figure, the tools to be used in both scenarios are exactly the same. What differentiates them is the target system. In the first scenario, the target system is the same as the old system; in the second one, we have a new system that will have the configuration gotten from the old system.

The following list details specific configuration information and other important data for each product installed in our systems:

LAN Requester configuration information:

Computer name:	UZ1175J
Domain name:	ITSCAUS
Local Admin:	USERID/PASSWORD LOCADMIN/LOCADMIN

Communications Manager/2 configuration information:

Network ID:	USIBMAU1
Local node:	AU168000
Local node ID:	05d00000
LAN dest. address:	400021041064

NetView DM/2 configuration information:

CC client:	UZ1175J
CC domain:	NVDM1175

TCP/IP configuration information:

IP address:	9.3.1.139
Subnet mask:	255.255.255.0
Router/Gateway:	9.3.1.74
DNS/NameServer:	9.3.1.74
Host name:	UZ1175J

9.2 Introducing the Tools to be Used

The tools to be used for a migration scenario are the same that were used in the pristine installation, but some of them are used in a slightly different way.

The first step for the migration consists of preparing a customized donor system with all the software and configuration required according your needs. Refer to NetView for detailed information on how to configure the donor system.

Once we have our donor system customized just as we want it, we have to create an image of this system with the SYSCOPY tool and store it in the server.

Because we are doing a migration, we have to obtain the configuration data on our old system, in order to use it in the new system. To achieve this, we will use the PROBE tool to retrieve the data in the old system. The gathered information will be stored in bundle files generated by the PROBE tool. As soon as the image generated by SYSCOPY is installed in the proper systems by the replicator tool, the bundle files generated by PROBE will be used by the configurator tool to put the configuration data back in the new system.

Besides the RDT we will make use of other tools to accomplish the migration. First, we have NetView DM/2 to distribute the software, so we have to create the NVDM/2 boot diskettes. Refer to 3.4, “Creating Boot Diskettes with NetView DM/2 Client” on page 129 for detailed information on this topic.

Another of the tools that may be used is DISKPREP.CMD. This REXX program is going to be used only if you plan to repartition and format the target systems.

We will use CUBE.CMD to modify some files before the target systems reboot for the first time.

Additionally, we will use some self-written programs to do some migration and configuration tasks. These programs are described in the corresponding chapters.

9.3 Step-by-Step Guidance

The following sections describe in detail all the steps for our migration scenario.

9.3.1 Creating the Donor System Image using SYSCOPY

As mentioned before, we will use SYSCOPY tool to store a donor image in the NetView DM/2 Server. For detailed information on how to do it, refer to 5.4.9, “Creating an Image of the Donor System” on page 185.

9.3.2 Creating the Configuration Bundle Files Using PROBE

Having used SYSCOPY tool to store the donor system’s image in the NetView DM/2 Server, the next step for the migration consists of backing up the configuration files, or some important data inside them, on the old system using the PROBE tool. Table 28 on page 294 lists the files and the critical lines inside those files that need to be backed up by PROBE.

Note: In this step, we do not migrate the PROTOCOL.INI file which contains the MPTS configuration information. Migrating this file from the old systems using PROBE can result in missing important protocol and network adapter configuration information. We assume the donor system image was configured with the correct protocol and adapter information. CUBE will take care of locally administered addresses (LAA) in a later process.

Table 28. Configuration Information to be Extracted by PROBE

Category	Description	Location	File	Line
LAN Requester	Computer name	C:\IBMLAN	IBMLAN.INI	COMPUTERNAME=
	Default Domain	C:\IBMLAN	IBMLAN.INI	DOMAIN=
TCP/IP	Hostname	C:\	CONFIG.SYS	SET HOSTNAME=
	IP Address	C:\MPTN\BIN	SETUP.CMD	ifconfiglan0
	IP Mask	C:\MPTN\BIN	SETUP.CMD	netmask
	IP Default Router	C:\MPTN\BIN	SETUP.CMD	route add default
	Net add. for router	C:\MPTN\BIN	SETUP.CMD	route add net
	IP Domain Name	C:\MPTN\ETC	RESOLV2	domain
	Domain Name Server Address	C:\MPTN\ETC	RESOLV2	nameserver
	IP Domain Name (DOS)	C:\TCPIP\DOS\ETC	RESOLV	domain
	Domain Name Server add. (DOS)	C:\TCPIP\DOS\ETC	RESOLV	nameserver
NVDM/2	Client Name	E:\IBMNVDM2	IBMNVDM2.INI	ClientName
	Server Name	E:\IBMNVDM2	IBMNVDM2.INI	ServerName

9.3.2.1 PROBE Scripts

In this section, we describe the PROBE scripts that are used to back up the configuration information in the old system.

Table 29 on page 294 lists the names of all the PRB files used in our scenario, and we describe briefly the purpose of each file:

Table 29. PROBE Command Stream Files

Probe script file	Category	Description
GETALL21.PRB	General	Retrieves all the configuration information from the old system
LAN21.PRB	LAN Requester	Retrieves all the LAN Requester configuration information

Probe script file	Category	Description
NVDM21.PRB	NVDM/2	Retrieves all the NVDM/2 configuration information
TCP21.PRB	TCP/IP	Retrieves all the TCP/IP configuration information
LRDRIVE21.PRB	LAN Requester	Gets IBMLAN drive from config.sys and sets an FI variable
LRCOMNAM21.PRB	LAN Requester	Gets the Computer name
LRDOMNAM21.PRB	LAN Requester	Gets the Domain Name
MAINTDRV21.PRB	NVDM/2	Sets an FI variable for the maintenance partition drive
NVDMINI21.PRB	NVDM/2	Gets NVDM/2 Client name and Server Name
TCPDRIV21.PRB	TCP/IP	Sets the TCP/IP drive and MPTN drive and sets two FI variables
TCPRSLV221.PRB	TCP/IP	Copies RESOLV2 file
TCPRSLV21.PRB	TCP/IP	Copies RESOLV file
TCPHSTNM21.PRB	TCP/IP	Gets the host name
TCPSETUP21.PRB	TCP/IP	Gets IP address, IP mask, Default router and Default router net address

As we mentioned in Chapter 1.3, “The PROBE Tool” on page 15, each script file can contain inclusions of other command stream files by using the `INCLUDE` command. We will notice later that GETALL21.PRB, LAN21.PRB and TCP21.PRB are script files consisting of the inclusion of other command script files depending on the category which they belong to. Figure 116 on page 296 shows a chart that illustrates these nested command stream files.

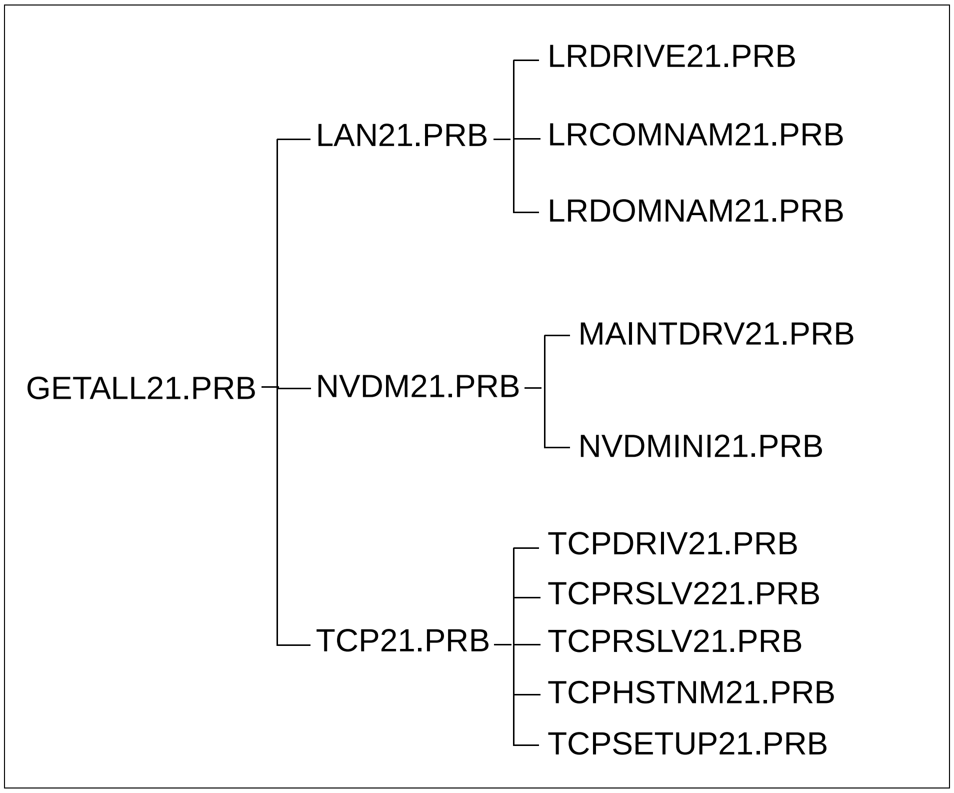


Figure 116. Structure of Nested Command Stream Files (GETALL21.PRB)

In the following sections, we give the detailed description of each PROBE file used for our migration scenario. We divided each section according to the category of the information backed up by the PROBE tool.

9.3.2.2 General Category Probe Scripts

GETALL21.PRB is the only script file categorized as general. It includes LAN21.PRB, NVDM21.PRB and TCP21.PRB. The bundle files generated with this command script file will be the ones to be used by the configurator tool to configure the target system.

```
// This script will gather all the information regarding LAN requester,
// TCPIP and Netware requester
// {PRBPATH = path where all the prb files are located

INCLUDE "{PRBPATH}\LAN21.PRB"
INCLUDE "{PRBPATH}\TCP21.PRB"
INCLUDE "{PRBPATH}\NVDM21.PRB"
```

Figure 117. GETALL21.PRB command script file

As you can see in “Executing PROBE” on page 309, we have defined the PRBPATH FI variable that will point to the subdirectory where all the PROBE script files reside.

Editing PROBE Script Files

If you create your own PROBE command script files, don’t forget to end the file with a Return character. This means you have to press the **Enter** key after the last line in the script. Otherwise, the PROBE command might fail to execute.

9.3.2.3 LAN Requester Category PROBE Scripts

The following PROBE command stream files belong to the LAN Requester category:

- LAN21.PRB

  This command stream file retrieves relevant LAN Requester configuration information. It includes a set of command stream files which we describe later.

  Figure 118 shows the LAN21.PRB file.

```
INCLUDE "{PRBPATH}\LRDRIVE21.PRB" // Get IBMLAN Drive from
                                  //config.sys and set FI variable
                                  //IBMLANDRIVE
INCLUDE "{PRBPATH}\LRCOMNAM21.PRB"          // Computername
INCLUDE "{PRBPATH}\LRDOMNAM21.PRB"          // Domain
```

Figure 118. LAN21.PRB Command Script File

- LRDRIVE21.PRB

This command stream file is used to set a FI variable called IBMLANDRIVE, which will be used by other command stream files. The IBMLANDRIVE FI variable points to the drive where the \IBMLAN directory is located.

As shown in Figure 119, the value of this variable is retrieved through searching the CONFIG.SYS file.

```
// Category: LAN Server/Requester
// Name:    IBMLAN Drive
// Version:  1.0
// Description: This will set the FI Variable IBMLANDRIVE
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will determine which drive IBMLAN is installed on and
set the FI variable IBMLANDRIVE
// FIVARS:{BOOTDRIVE} =  Drive system is booted
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   EDITLINE
      SOURCESTRING
         SETFIVAR "IBMLANDRIVE"
         LINEPATTERN "IBMLAN\\NETPROG\\NETWKSTA.200 /I:"
         MATCHPATTERN "[A-Z]{1}:"
      ENDSOURCESTRING
      TARGETSTRING
         LOCATEDACTION NOTHING
         NOTLOCATEDACTION NOTHING
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 119. LRDRIVE21.PRB Command Script File

- LRCOMNAM21.PRB

  This PROBE command stream file retrieves the LAN Requester computer name out of IBMLAN.INI. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE21.PRB command stream file.

  Figure 120 on page 299 shows the LRCOMNAM21.PRB file.

```
// Category: LAN Server/Requester
// Name:    Computer Name
// Version:  1.0
// Description: Migrate Computername from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches
"COMPUTERNAME" from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "COMPUTERNAME"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 120. LRCOMNAM21.PRB Command Script File.

- LRDOMNAM21.PRB

  This PROBE command stream file retrieves the LAN Requester default domain name out of IBMLAN.INI. Note that this PROBE file uses the IBMLANDRIVE FI variable created by the LRDRIVE21.PRB command stream file.

  Figure 121 on page 300 shows the LRDOMNAM21.PRB file.

```
// Category:LAN Server/Requester
// Name:    Domain Name
// Version:     1.0
// Description: Migrate Domainname from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:    3.0, 4.0, 5.0
// Details:  This will migrate the "Domainname" line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is
installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "DOMAIN"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 121. LRDOMNAM21.PRB Command Script File

9.3.2.4 NetView DM/2 Category PROBE Scripts

The following PROBE command stream files belong to the NetView DM/2 category:

- NVDM21.PRB

  NVDM21.PRB retrieves relevant information regarding the NVDM/2 Client software in the maintenance partition. Later on, in “SETNVDM Using CUBE” on page 321, we show that the NVDM client information for the C: partition must be retrieved in a different way. NVDM21.PRB includes a set of command stream files which we describe later.

  Figure 122 shows the NVDM21.PRB file.

```
INCLUDE "{PRBPATH}\MAINTDRV21.PRB"         // Set Maintenance
                                           //partition drive
INCLUDE "{PRBPATH}\NVDMINI21.PRB"   // ClientName and ServerName of
                                    //IBMNVDM2.INI
```

Figure 122. NVDM21.PRB Command Stream File

- MAINTDRV21.PRB

The command stream file, MAINTDRV.PRB, is used to set an FI variable called MAINTDRIVE, which will be used by NVDMINI21.PRB command stream file. The MAINTDRIVE FI variable points to the drive where the maintenance partition is located.

As shown in Figure 123, we manually set the value of the variable.

```
// Category: Maintenance Partition
// Version:  1.0
// Description: Manually set Maintenance partition Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables MAINT
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance Partition

FIVAR
   NAME "MAINTDRIVE"
   VALUE "E:"
ENDFIVAR
```

Figure 123. MAINTDRV21.PRB Command Stream File

- NVDMINI21.PRB

  The command stream file, NVDMINI.PRB, migrates the NVDM/2 Client name and server name by extracting the values from the IBMNVDM2.INI file that resides in the maintenance partition. Note that this PROBE file is using the MAINTDRIVE FI variable created with the MAINTDRV21.PRB command stream file.

  Figure 124 on page 302 shows the NVDMINI21.PRB file.

```
// Category: NVDM
// Name:    WORKSTATION AND SERVER NAMES
// Version:  1.0
// Description: Migrate WORKSTATION AND SERVER NAMES FROM IBMNVDM2.INI
// Details:  This will migrate the lines that matches ServerName and
ClientName
//              and ClientName in the IBMNVDM2.INI of the maintenance
partition
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance partition

TEXTFILE
   SOURCE  "{MAINTDRIVE}\IBMNVDM2\IBMNVDM2.INI"
   REPLACELINE
      SOURCELINE
        PATTERN "ServerName"
      ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE
        PATTERN "ClientName"
      ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 124. NVDMINI21.PRB Command Stream File.

9.3.2.5 TCP/IP Category PROBE Scripts

The following PROBE command stream files belong to the TCP/IP category:

- TCP21.PRB

  TCIP21.PRB retrieves relevant TCP/IP configuration information. It includes a set of command stream files which we describe later.

  Figure 125 shows the TCP21.PRB file.

```
INCLUDE "{PRBPATH}\TCPDRIV21.PRB"          // Drive letters TCPIPDRIVE
                                         //& MPTNDRIVE
INCLUDE "{PRBPATH}\TCPRSLV21.PRB"           // Resolv2 file
INCLUDE "{PRBPATH}\TCPRSLV221.PRB"          //Resolv file
INCLUDE "{PRBPATH}\TCPHSTNM21.PRB"           // Host name
INCLUDE "{PRBPATH}\TCPSETUP21.PRB"           // setup.cmd
```

Figure 125. TCP21.PRB Command Script File

- TCPDRIV21.PRB

  This command stream file sets two FI variables, called TCPDRIVE and MPTNDRIVE, which will be used by other command stream files. The TCPDRIVE and MPTNDRIVE FI variables point to the drives where the \TCPIP and \MPTN directories are located.

  As shown in Figure 126, we manually set the values of these variables.

```
// Category: TCPIP
// Name: Installed Drives
// Version: 1.0
// Description: Manually set TCPIP Drive and MTPN Drive
// From: ANY
// To: ANY
// Details: Manually set the fi variables TCPIPDRIVE and MPTNDRIVE
// FIVARS: {TCPIPDRIVE} = Drive where TCPIP is installed
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

FIVAR
   NAME "TCPIPDRIVE"
   VALUE "C:"
ENDFIVAR

FIVAR
   NAME "MPTNDRIVE"
   VALUE "C:"
ENDFIVAR
```

Figure 126. TCPDRIV21.PRB Command Script File

- TCPRSLV221.PRB

  This command script file migrates the RESOLV2 file from the old system to the target system.

  Figure 127 shows the TCPRSLV221.PRB file.

```
// Category: TCPIP
// Name:    RESOLV2
// Version:  1.0
// Description: Migrate RESOLV2
// From:  2.0
// To:  4.0
// Details:  This will migrate the file RESOLV2
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

TEXTFILE
   SOURCE  "{TCPIPDRIVE}\TCPIP\ETC\RESOLV"
   TARGET  "{MPTNDRIVE}\MPTN\ETC\RESOLV2"
   REPLACELINE
      SOURCELINE PATTERN "DOMAIN" ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE PATTERN "NAMESERVER" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 127. TCPRSLV221.PRB Command Script File

- TCPRSLV21.PRB

  This command script file migrates the RESOLV file from the old system to the target system.

  Figure 128 on page 305 shows the TCPRSLV21.PRB file

```
// Category: TCPIP
// Name:    RESOLV
// Version:  1.0
// Description: Migrate RESOLV
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV (for DOS sesions)
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

COPYFILES
   SOURCE "{TCPIPDRIVE}\TCPIP\DOS\ETC"
   INCLUDEFILES
      PATTERN "RESOLV"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 128. TCPRSLV21.PRB Command Script File

- TCPHSTNM21.PRB

  This PROBE file migrates the TCP/IP host name extracting it from the CONFIG.SYS file.

  Figure 129 shows the TCPHSTNM21.PRB file.

```
// Category: TCPIP
// Name:    HOSTNAME
// Version:  1.0
// Description: Migrate HOSTNAME from config.sys
// From:  2.0, 3.0
// To:  2.0, 3.0
// Details:  This will migrate the entire line that matches HOSTNAME
from config.sys
// FIVARS:  {BOOTDRIVE} = Drive system is booted from

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   REPLACELINE
      SOURCELINE PATTERN "HOSTNAME" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 129. TCPHSTNM21.PRB Command Script File

- TCPSETUP21.PRB

This `PROBE` file migrates the IP address, IP mask, IP default router, and the router net address by extracting this data from the \MPTN\ETC\SETUP.CMD file.

Figure 130 shows the TCPSETUP21.PRB file.

```
// Category:    TCP/IP
// Name:        SETUP.CMD
// Version:     1.0
// Description: This will modify strings on SETUP.CMD
// From:        4.0
// To:          4.1
// Details:     This will get the values for net address, mask and router
// FIVARS:      {MPTNDRIVE}  = Drive where MPTN is installed
TEXTFILE
  SOURCE  "{MPTNDRIVE}\MPTN\BIN\SETUP.CMD"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "ifconfig lan0 "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "ifconfig lan0"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "netmask"
         MATCHPATTERN "\ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "netmask"
         STRINGPATTERN "\ {1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
         STRINGSCOPE LAST
      ENDTARGETSTRING
   ENDEDITLINE
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add default "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add default"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add net "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add -net 9"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 130. TCPSETUP21.PRB Command Stream File

We can see that the TCPSETUP21.PRB command stream file is considerably different from the ones we discussed before. This command stream file is more complex because the SETUP.CMD files of TCP/IP 2.0 and TCP/IP 4.1 (the one we migrate to) are slightly different.

We need the data of the older version to match the data in the newer one. This is accomplished by retrieving strings within certain lines using the SOURCESTRING command and replacing those strings in the target file with the TARGETSTRING command.

Syntax of PROBEFiles

Remember that you use FI regular expressions on the target system, but you use XPG4 Extended Regular Expressions on the source system. You must be careful with the syntax you are using for each case. Refer to the appropiate documentation to see the differences on the syntax.

The following two figures help us to analyze the two different versions of SETUP.CMD. Figure 131 and Figure 132 show the SETUP.CMD for TCP/IP V2 and SETUP.CMD for TCP/IP V4.1, respectively.

```
route -fh
arp -f
ifconfig lan0 9.3.1.146 netmask 255.255.255.0
REM ifconfig lan1
REM ifconfig lan2
REM ifconfig lan3
REM ifconfig lan4
REM ifconfig lan5
REM ifconfig lan6
REM ifconfig lan7
REM ifconfig sl0
route add net 9.3 9.3.1.74 1
route add default 9.3.1.74 1 *
```

Figure 131. SETUP.CMD for TCP/IP V2

```
route -fh
arp -f
ifconfig lo 127.0.0.1
ifconfig lan0 9.3.1.146 netmask 255.255.255.0
REM ifconfig lan1
REM ifconfig lan2
REM ifconfig lan3
REM ifconfig lan4
REM ifconfig lan5
REM ifconfig lan6
REM ifconfig lan7
rem ifconfig sl
route add default 9.3.1.74 -hopcount 1 *
route add -net 9 9.3.1.74 -netmask 255.0.0.0 -hopcount 1 *
```

Figure 132. SETUP.CMD for TCP/IP V4.1

If you compare both files and analyze the TCPSETUP.PRB file you notice that the lines that have a leading asterisk are the ones that contain the strings that must be copied with the SOURCESTRING and TARGETSTRING commands.

9.3.2.6 Executing PROBE

Now that all the PROBE script files we are going to use are defined. We are ready to execute the PROBE command to extract the configuration data from the old system. The way we are going to do this is by executing the PROBE command in all old systems that have the data we want to store using NetView Distribution Manager /2.

Figure 133 shows the change file profile used to execute PROBE at the desired OS/2 V2.11 workstations.

```
Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.PROBE.GETALL21.REF.1
End

Section Install
Begin
      Program = SA:\exe\probe.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
        /b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir
        /l:$(sb)\log\$(WorkStatName) /x:4
        $(sa)\rsp\probe\getall21.prb"
End
```

Figure 133. Change File Profile PROBE21.PRO

The program we are executing from the NVDM/2 Server is the following:

PROBE Execution

```
probe /v:{prbpath}=$(sa)\rsp\probe
      /e:$(WorkStatName)
      /b:$(sa)\img\config\$(WorkStatName)
      /w:$(sb)\workdir
      /l:$(sb)\log\$(WorkStatName)
      /x:4 $(sa)\rsp\probe\getall21.prb
```

where:

`/v:{prbpath}=$(sa)\rsp\probe`	Defines an FI variable that points to the subdirectory that contains all the PROBE command stream files. In our particular case this subdirectory is D:\SHAREA\RSP\PROBE at the NVDM/2 Server. This FI variable is used by GETALL21.PRB, LAN21.PRB, NVDM21.PRB, and TCP21.PRB,
`/e:$(WorkStatName)`	Specifies the client's name that corresponds to the NVDM/2 workstation name. This means that the name of the bundle files generated by PROBE correspond to the NVDM/2 workstation name with extensions of

	000, 001 and so on. This parameter is used with the configurator tool at a later time.
`/b:$(sa)\img\config\$(WorkStatName)`	Specifies the fully qualified path where bundle files generated by PROBE will be stored, in our particular case, in D:\SHAREA\IMG\CONFIG\$(WorkStat Name)
`/w:$(sb)\workdir`	The working directory used by PROBE will be, in our particular case, in D:\SHAREB\WORKDIR.
`/l:$(sb)\log\$(WorkStatName)`	Specifies the fully qualified path and name of the log file.
`/x:4`	Specifies the highest logging level for test purposes. You may change it later.
`$(sa)\rsp\probe\GETALL21.PRB`	We call GETALL21.PRB to extract the data from all systems.

9.3.3 CM2PCOMM and COPYCONF Utilities

We make use of these REXX programs to migrate from Communications Manager/2 to eNetwork Personal Communications for OS/2 Warp. If we had had Personal Communications on the old system, we would have been able to perform the migration by simply extracting the *.WS configuration files from the old system using the PROBE tool and copy them to the target system.

The program, CM2PCOMM.CMD extracts the Communications Manager/2 configuration information from the old system and converts it to the Personal Communications format.

It also creates a subdirectory in D:\SHAREA\IMG\CONFIG with the name of the NVDM/2 workstation. This subdirectory will have the information extracted by CM2PCOMM as well as the bundle files created by PROBE.

Refer to 6.1.1, “CM2PCOMM.CMD” on page 189, for detailed information about this utility.

COPYCONF is a small REXX program that copies the configuration files created by CM2PCOMM to the target system. Refer to 6.1.3, “COPYCONF.CMD” on page 205, for more information on this utility.

9.3.3.1 CM2PCOMM.PRO Change Profile

The following figure, Figure 134, displays the NetView DM/2 change profile CM2PCOMM.PRO that executes the CM2COMM.CMD REXX command file.

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.CM2PCOMM.INST.REF.4
     Description="Migration from CM/2 to Personal Communications"
End

Section Install
Begin
      Program =  SA:\EXE\CM2PCOMM.CMD
      Parms= "C:\CMLIB $(WorkStatName)"
End
```

Figure 134. Change File Profile CM2PCOMM.PRO

9.3.3.2 COPYCONF.PRO Change Profile

The following figure, Figure 135, displays the NetView DM/2 change profile, COPYCONF.PRO, that executes the COPYCONF.CMD REXX command file.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.COPYCONF.REF.1
End

Section Install
Begin
      Program = SA:\exe\copyconf.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 135. Change File Profile COPYCONF.PRO

9.3.4 GETNVDM

We make use of this REXX program in order to obtain two values from the old system: the NVDM/2 Client name and the NVDM/2 Server name. This program creates a profile that CUBE will use to set these parameters on the donor system.

These IBMNVDM2.INI parameters must be modified at the target workstations before they reboot for the first time since all systems that were installed with the donor system image carry identical parameters. If we don't modify them, we will not be able to execute the rest of the commands at the target systems from the NVDM/2 Server since the NVDM/2 Clients won't come up due to identical names.

When the systems reboot, some error messages will be shown because of some duplicate parameters. Since we only want to connect to the NVDM/2 Server from the production system to run CONFIGUR to change rest of the parameters of the workstations we don't need to pay attention to these upcoming error messages.

As mentioned before, GETNVDM generates a profile called IBMNVDM2.PRO that contains the NVDM/2 parameters. This profile will be stored in \SHAREA\IMG\CONFIG directory, where the images of PROBE and CM2PCOMM are stored.

This profile will be used by `SETNVDM` to store this data at the new systems before the first reboot is performed. This allows the clients to come up with the proper NetView DM/2 Client names.

The following figure, Figure 136 on page 315, displays the contents of the `GETNVDM.CMD` REXX command file.

```
/* REXX program to extract the NVDM/2 Client and server name    */
/* from a workstation. The information is stored in a           */
/* CUBE profile in the correct directory on the NVDM/2 Server.  */

parse upper arg workstationname .
ECHO OFF
targetpath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'

logdir = 'X:\LOG\'||workstationname

RCode = 0
server = ''
client = ''

RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\GETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
if RCode == 0 then do
   do while lines(nvdmpath||'\IBMNVDM2.INI')
      line.linenum = Linein(nvdmpath||'\IBMNVDM2.INI')
      select
         when word(line.linenum, 1)=='ServerName' then server=line.linenum
         when word(line.linenum, 1)=='ClientName' then client=line.linenum
      otherwise NOP
      end  /* select */
   end /* do */

if RCode == 0 then do
   RC = STREAM(logdir||'\GETINFO.LOG', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'logdir||'\GETINFO.LOG'
   end /* do */
   RC = STREAM(targetpath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'targetpath||'\IBMNVDM2.PRO'
   end /* do */

   cubeline = 'RepLine "ServerName" WITH "'server'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'

   cubeline = 'RepLine "ClientName" WITH "'client'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'
   Call Stream targetpath||'\IBMNVDM2.PRO','c','close'
end /* do */
if RCode == 0 then do
   'ECHO GETINFO ended successful >> 'logdir||'\GETINFO.LOG'
end /* do */
else do
   'ECHO GETINFO ended unsuccessful with >> 'logdir||'\GETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
Call Stream logpath||'\GETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 136. GETNVDM.CMD REXX Command File

9.3.4.1 GETNVDM21.PRO Change Profile

The following figure, Figure 137, displays the NetView DM/2 change profile, GETNVDM21.PRO, that executes the GETNVDM.CMD REXX command file.

t

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.GETNVDM21.MIGRAT21.REF.1
End

Section Install
Begin
      Program = SA:\exe\getnvdm.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 137. Change File Profile GETNVDM21.PRO

9.3.5 DISKPREP

As mentioned earlier, besides the RDT, we are going to make use of other tools like DISKPREP and CUBE.

In our specific case we are going to use DISKPREP in the target systems to repartition and format the hard disk on the target systems. We are doing so because we already backed up all the configuration information we need to keep from the old systems by using PROBE.

Old System=Target System?

If the "old system" is also "target system", and if you plan to use DISKPREP.CMD, make sure that relevant information from your old systems is retrieved and backed up, since DISKPREP will format the partitions.

DISKPREP allows you to delete and create all necessary partitions in a given system using response files. It also takes care of the system reboot and continues to format all the partitions before exiting.

For detailed information about how DISKPREP works, refer to NetView.

Because DISKPREP will be executed from the NetView DM/2 Server at the target systems, we need to boot the target system with the NVDM/2 boot diskettes. Refer to 3.4, “Creating Boot Diskettes with NetView DM/2 Client” on page 129.

Keep in mind that DISKPREP requires REXX support to be enabled. Before you execute DISKPREP you need to enable REXX support from the NetView DM/2 boot diskettes. For details on how to enable REXX support, refer to 7.2.2, “Enabling REXX Support” on page 226.

9.3.5.1 Executing DISKPREP.CMD

The execution of DISKPREP from the NVDM/2 Server requires two steps:

- Enabling REXX support
- Execution of DISKPREP itself

Figure 138 and Figure 139 show the change file profiles used at the NVDM/2 Server.

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = REXX Support for Pristine Clients
End

Section Install
Begin
      Program = SA:\EXE\cmd.exe
      Parms =  /C $(SA)\exe\detrexx.cmd $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 138. Change File Profile LOADREXX.PRO to Enable REXX Support

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir)
/L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 139. Change File Profile DISKPREP.PRO

Note that in Figure 138, REXX enablement requires a batch file called `DETREXX.CMD`. Figure 64 on page 227 displays this file.

9.3.6 Replicator

Once the hard disk on the target system(s) have been partitioned and formatted, we are ready to copy the donor system image stored on the NVDM/2 Server using the replicator tool, `REPLICAT.EXE`.

Figure 140 shows the change file profile used in the NVDM/2 Server to run the `REPLICAT.EXE` command at the target system(s).

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
     Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
              /L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 140. FiIBM750.PRO Change File Profile

The following box illustrates the REPLICAT command executed at the NVDM/2 Server:

REPLICAT Execution

```
REPLICAT.EXE  /C:IBM750
              /B:$(SA)\IMG\FINANCE
              /F:$(SA)\RSP\FiIBM750.RSP
              /L:$(SB)\LOG\$(WorkStatName)
```

where:

`/C:IBM750`	For this specific scenario, the class ID of the donor system is IBM750.
`/B:$(SA)\IMG\FINANCE`	Indicates the directory where the class IBM750 bundle files (generated by SYSCOPY) are located. This directory corresponds to D:\SHAREA\IMG\FINANCE on the NVDM/2 Server. In this specific scenario, this directory contains all class IDs for the finance department. The class ID itself identifies the system brand and model.

`/F:$(SA)\RSP\FiIBM750.RSP`	Specifies a command file with drive stanza directives. `REPLICAT.EXE`, by default, only restores images to a drive with the same partition size or larger. Since our scenario is replicating to a smaller partition size, this file is used to specify the minimum allowed partition size to make the restore and overwrite the default partition size.
`/L:$(SB)\LOG\$(WorkStatName)"`	Specifies the directory name where the log file will be located. `REPLICAT.EXE` names the log files with the same name as the class ID. In this particular case, it puts the log file in D:\SHAREB\LOG\ $(WORKSTATNAME) directory on the NVDM/2 Server.

The following figure shows the response file used in this scenario:

```
DRIVE
        DONORLETTER C:
        TARGETLETTER C:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER D:
        TARGETLETTER D:
        MINSIZE 400
ENDDRIVE

DRIVE
        DONORLETTER E:
        TARGETLETTER E:
        MINSIZE 18
ENDDRIVE
```

Figure 141. FIIBM750.RSP Response File

Log File Location

Care should be taken on the log file location. Since REPLICAT.EXE always uses the class name for the log file, a unique location should be given to each workstation. Otherwise, REPLICAT.EXE will have problems accessing the log file. In our scenario, NetView DM/2 provides several variables, where the workstation name is one of them. This variable is used to identify the unique directory name where the log file will be located.

In addition, REPLICAT.EXE will not create the directory specified in the /L: parameter. The user has to create the directory before running REPLICAT.EXE.

9.3.7 SETNVDM Using CUBE

We use CUBE after the replication has finished and before the configurator is executed. With this REXX program, you can access and edit ASCII files at the target workstation.

The file that we need to modify at the target system(s) is IBMNVDM2.INI which contains the NVDM/2 configuration, such as the NVDM/2 Client name and NVDM/2 Server name.

These two parameters (especially the NVDM/2 workstation name) in the IBMNVDM2.INI file must be modified before the target systems reboot for the first time because all the systems that were installed with the donor system image have the same parameters. If we don't modify them, we wont be able to execute the rest of the commands at all the target systems from the NVDM/2 Server since the NVDM/2 Clients won't come up due to replicated names.

The reason to use CUBE instead of CONFIGUR is that the latter one requires Presentation Manager to run. Presentation Manager means the target systems would have needed to reboot prior to those changes and would make it impossible to connect to the NetView DM/2 Server meaning no CONFIGUR commands could be run at the target workstations.

After CUBE changed the NVDM/2 parameters, the target systems will be able to connect after reboot and will have Presentation Manager up and running to run CONFIGUR which changes the rest of parameters gathered by PROBE.

For migration to OS/2 Warp 4.0 purposes, the SETNVDM.CMD REXX command file was written. This REXX program makes use of CUBE and changes parameters on the target systems. SETNVDM makes use of the IBMNVDM2.PRO profile that was generated by GETNVDM.

Figure 142 shows the contents of SETNVDM.CMD.

```
/* REXX program to set the information extracted by GETNVDM */

parse upper arg workstationname .

ECHO OFF

sourcepath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'
mptspath = 'C:\IBMCOM'

logdir = 'X:\LOG\'||workstationname

RCode = 0

RC = STREAM(logdir||'\SETINFO.LOG', 'C', 'QUERY EXISTS')
if RC <> '' then do
   'DEL 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(sourcepath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.PRO does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */

if RCode == 0 then do
   cubearg = sourcepath||'\ibmnvdm2.pro 'nvdmpath||'\IBMNVDM2.INI'
   RC = 'cube.cmd'(cubearg)
   if RC <> 0 then RCode = RC
   if RCode <> 0 then do
      say 'One or more errors were found executing the changes'
      say 'to 'nvdmpath||'\IBMNVDM2.INI and 'mptspath||'\PROTOCOL.INI'
      RCode = 4
   end
end /* do */
if RCode == 0 then do
   'ECHO SETINFO ended successful >> 'logdir||'\SETINFO.LOG'
end /* do */
else do
   'ECHO SETINFO ended unsuccessful with >> 'logdir||'\SETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
Call Stream logpath||'\SETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 142. SETNVDM.CMD REXX Command File

9.3.7.1 SETNVDM21 Change File Profile

The following figure, Figure 143, shows the change file profile used at the NVDM/2 Server:

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.SETINFO.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\setinfo.cmd
      Parms = "$(WorkStatName)"
      PhaseEnd = yes
End
```

Figure 143. Change File Profile SETNVDM21.PRO

After the execution of SETNVDM, the machine must be rebooted. After the reboot, when Presentation Manager is active, CONFIGUR can be executed at the target systems to change the rest of the parameters.

9.3.8 Configurator

Once CUBE changes the NVDM/2 workstation name, the target system is ready to receive the rest of the configuration information that was extracted by PROBE from the old system using the CONFIGUR configurator tool.

Running CONFIGUR More Than Once

At the time this redbook was writen, CONFIGUR had some problems updating the information stored in the bundle files when it was run more than once in the same machine. This is due to the FI version used. To avoid this problem, the machine must be rebooted every time CONFIGUR is executed.

Figure 144 shows the change file profile used in the NVDM/2 Server to run CONFIGUR command at the target system(s).

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.CONFIGUR.MIGRAT21.REF.1
End

Section Install
Begin
      Program = SA:\exe\configur.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
              /b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir
              /l:$(sb)\log\$(WorkStatName) /x:4"
End
```

Figure 144. CONFIGUR21.PRO Change File Profile

The following command is executed at the NVDM/2 Server:

CONFIGUR Execution

```
CONFIGUR /v:{prbpath}=$(sa)\rsp\probe
         /e:$(WorkStatName)
         /b:$(sa)\img\config\$(WorkStatName)
         /w:$(sb)\workdir
         /l:$(sb)\log\$(WorkStatName)
          /x:4
```

where:

`/v:{prbpath}=$(sa)\rsp\probe`

Defines an FI variable that points to the subdirectory that contains all the PROBE command stream files. In our case, this directory is D:\SHAREA\RSP\PROBE on the NVDM/2 Server. This FI variable is used by GETALL3.PRB, LAN.PRB, NVDM.PRB, and TCP.PRB.

`/e:$(WorkStatName)`

The client name corresponds with the NVDM/2 workstation name. This name was defined during the execution of PROBE, which means that the configurator will search the bundle files generated by `PROBE` that correspond to the NVDM workstation name with the extensions of 000, 001 and so on.

`/b:$(sa)\img\config\$(WorkStatName)`

This is the location of the bundle files generated by `PROBE`. In our case, this is the D:\SHAREA\IMG\CONFIG\$(WorkStatName) directory.

`/w:$(sb)\workdir`

The working directory used by `CONFIGUR`. In our case, this is the D:\SHAREB\WORKDIR directory.

`/l:$(sb)\log\$(WorkStatName)`

Specifies the fully qualified path of the log file directory.

`/x:4` Specifies the highest logging level for test purposes. You can change it later.

9.4 Navigating through NetView DM/2

In this section, we summarize the activities necessary at the NetView DM/2 Server to perform a migration from OS/2 V2.11 to OS/2 Warp 4. We assume a connectivity between the clients, also known as CC clients (Change Control clients), and server, also known as CC server, has been established either through boot diskettes or through the maintenance partition. Otherwise, you won’t be able to install so-called change files to the target workstations.

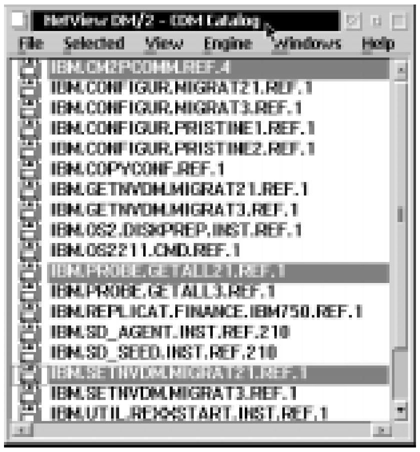
Before you continue, verify that the CC client is attached to the CC server. This can be done through the command line by issuing `CDM LIST_WS` or through the PM dialog. Return to the CDM Catalog window and initiate software installation on the CC client using the CDM Catalog window.

9.4.1 Retrieving Configuration Data from the Old System

To extract configuration information from the old systems, perform the following steps:

1. As shown in Figure 145 on page 326, at the CDM Catalog window, select the following change files and prepare them to install as a corequisite group:
   - CM2PCOM.PRO
   - PROBE21.PRO

- GETNVDM21.PRO



IBM.CM2PCOMM.REF.4
IBM.CONFIGUR.MIGRAT21.REF.1
IBM.CONFIGUR.MIGRAT3.REF.1
IBM.CONFIGUR.PRISTINE1.REF.1
IBM.CONFIGUR.PRISTINE2.REF.1
IBM.COPYCONF.REF.1
IBM.GETNVDM.MIGRAT21.REF.1
IBM.GETNVDM.MIGRAT3.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.OS2211.CMD.REF.1
IBM.PROBE.GETALL21.REF.1
IBM.PROBE.GETALL3.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SD_AGENT.INST.REF.210
IBM.SD_SEED.INST.REF.210
IBM.SETNVDM.MIGRAT21.REF.1
IBM.SETNVDM.MIGRAT3.REF.1
IBM.UTIL.REXXSTART.INST.REF.1

Figure 145. Selecting Change Files to Install on the Old System

2. Click on **Selected** from the action bar to display the pull-down menu.

   Select **Install** from the pull-down menu. The Install window will be displayed with all the selected objects listed.

   Note: If you are uncertain whether or not the selected software exists on the target workstation, select **Force** as installation method. NetView DM/2 catalogs distributed software in the DB2 database, thus preventing a software product from being distributed more than once. Force overrides this.

3. Define installation options.

   Select **Options** from the Install window. The Options screen will be displayed.

   Select **Install as a coreq group**.

   The purpose of "corequisite groups" is to bundle the installation requests of several objects together into one request. Reboot requests of the single objects are queued until a change file with the statement PhaseEnd=Yes or the last object of the corequisite group is installed.

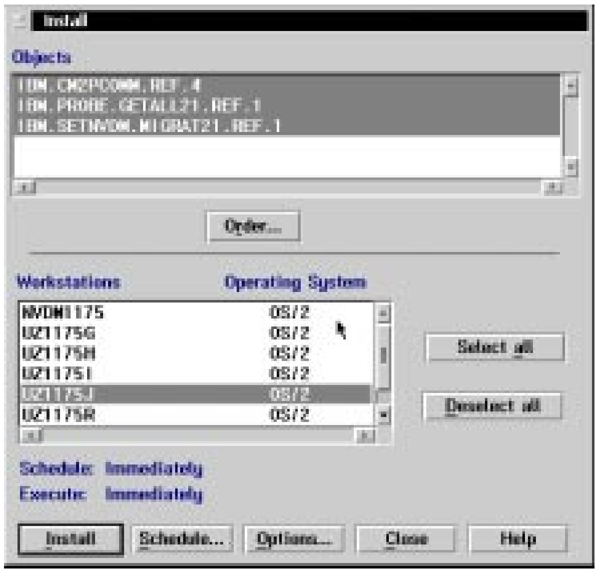
   Corequisite groups may consist of a maximum of seven change files; they are used to install several pieces of software that depend on each other. If one of the installs in a coreq group fails, the complete group will receive the status "failed", even if installs prior to the failed one were successful.

Click on **Set** to return to the Install window.

4. At the Install window, as shown in Figure 146, select the client's name from the workstation list to install the objects on client workstation and then select **Install** to execute the command.

   You will receive a message pop-up; check if everything went OK and select **OK** to continue.

   Select **Close** at the Install screen to return to the Catalog menu.



IBM.CM2PCOMM.REF.4
IBM.PROBE.GETALL21.REF.1
IBM.SETNVDM.MIGRAT21.REF.1

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

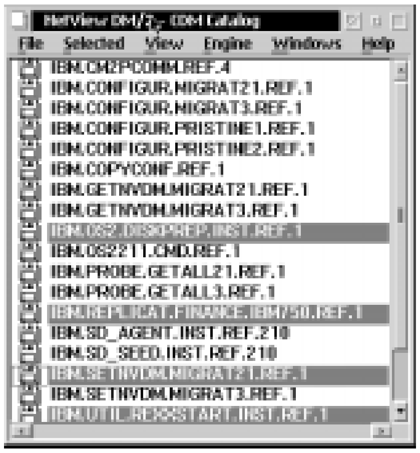
Figure 146. Install Change Files as Corequisite Group

9.4.2 Replicating and Configuring the New System

Once all necessary configuration information is retrieved from the old system, we can start the replication and configuration process on the target workstations. Perform the following steps:

1. As shown in Figure 147 on page 328, at the CDM Catalog window, select the following change files and prepare them to install as a corequisite group:
   - LOADREXX.PRO
   - DISKPREP.PRO.PRO
   - FiIBM750.PRO

- SETNVDM21.PRO
- CONFIGUR21.PRO


NetView DM/2 - CDM Catalog
File Selected View Engine Windows Help

IBM.CM2PCOMM.REF.4
IBM.CONFIGUR.MIGRAT21.REF.1
IBM.CONFIGUR.MIGRAT3.REF.1
IBM.CONFIGUR.PRISTINE1.REF.1
IBM.CONFIGUR.PRISTINE2.REF.1
IBM.COPYCONF.REF.1
IBM.GETNVDM.MIGRAT21.REF.1
IBM.GETNVDM.MIGRAT3.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.OS2211.CMD.REF.1
IBM.PROBE.GETALL21.REF.1
IBM.PROBE.GETALL3.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SD_AGENT.INST.REF.210
IBM.SD_SEED.INST.REF.210
IBM.SETNVDM.MIGRAT21.REF.1
IBM.SETNVDM.MIGRAT3.REF.1
IBM.UTIL.REXXSTART.INST.REF.1

Figure 147. Selecting Change Files to Install on Target Workstations

2. Click on **Selected** from the action bar to display the pull-down menu.

   Select **Install** from the pull-down menu. The Install window will be displayed with all the selected objects listed.

   Note: If you are uncertain whether or not the selected software exists on the target workstation, select **Force** as installation method. NetView DM/2 catalogs distributed software in the DB2 database, thus preventing a software product from being distributed more than once. Force overrides this.

3. Define the installation order.

   Click on **Order...** to display the Install Order window.

   Select **IBM.UTIL.REXXSTART.INST.REF.1**

   Select **Up** to move it up to the first position.

   Move other objects to arrange in this order:

   IBM.UTIL.REXXSTART.INS.REF.1
   IBM.OS2.DISKPREP.INST.REF.1
   IBM.REPLICAT.FINANCE.IBM750.REF.1
   IBM.SETNVDM.MIGRAT21.REF.1

4. At the Install Order window, select **OK** to return to the Install window.

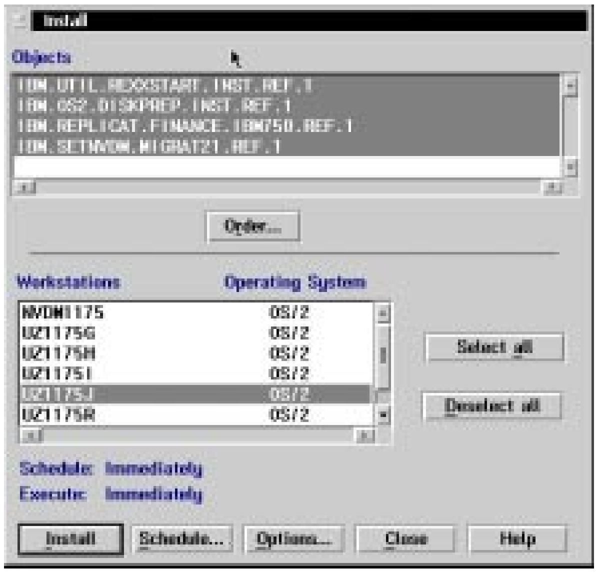
   Select client's name from the workstation list to install the objects on client workstation.

5. Define installation options.

   Select **Options** from the Install window. The Options screen will be displayed.

   Select **Install as a coreq group**.

.



```
IBM.UTIL.REXXSTART.INST.REF.1
IBM.OS2.DISKPREP.INST.REF.1
IBM.REPLICAT.FINANCE.IBM750.REF.1
IBM.SETNVDM.MIGRAT21.REF.1
```

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

Figure 148. Install Change Files as Corequisite Group

6. At the Install windows, as shown in Figure 148, select **Install** to execute the command.

   You will receive a message pop-up; check if everything went OK and select **OK** to continue.

   Select **Close** at the Install screen to return to the Catalog menu.

7. Check the status of the install request for your client.

   At the CDM Catalog window:

   Select **Engine** to display the menu.

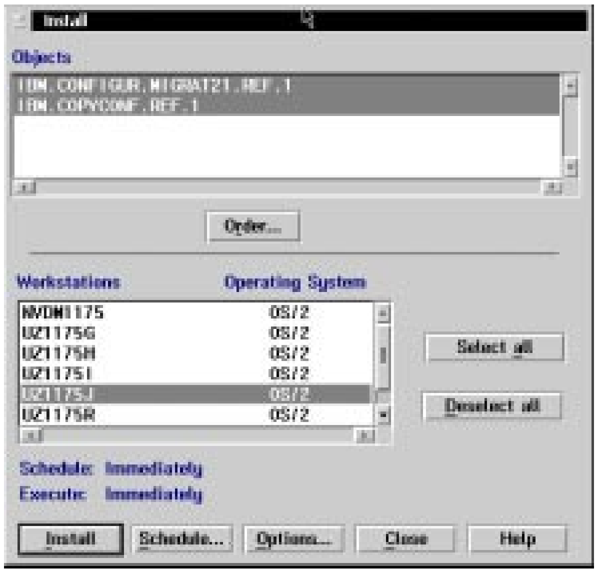
Select **All Pending Requests** to display the Pending Request menu.

Select the client’s name.

Select **Details** to display the details.

8. You may immediately send another installation request to the previously selected target workstations. The next installation step configures the target workstations. As shown in Figure 149, send the following change files as corequisite group:

   IBM.CONFIGUR.MIGRAT21.REF.1
   IBM.COPYCONF.REF.1



IBM.CONFIGUR.MIGRAT21.REF.1
IBM.COPYCONF.REF.1

Workstations	Operating System
NVDM1175	OS/2
UZ1175G	OS/2
UZ1175H	OS/2
UZ1175I	OS/2
UZ1175J	OS/2
UZ1175R	OS/2

Schedule: Immediately
Execute: Immediately

Figure 149. Installing Configuration Change Files at the Target Workstations

After the previously mentioned installation requests, the target workstations are installed and configured properly.

Part 3. Appendixes

Appendix A. Probe Stream Command Files

A.1 Pristine Installation

A.1.1 PRISTINE.PRB

```
// Define FI Variables to be use in this Workstation
// As the actual FI Variable will be defined at
// CONFIGUR.EXE run time, the FI Variable values are
// being set to themselves. The main purpose is to
// create them, so they exist when CONFIGUR.EXE is run.

FIVAR
   NAME "COMPUTERNAME"
   VALUE "{COMPUTERNAME}"
ENDFIVAR

FIVAR
   NAME "DOMAIN"
   VALUE "{DOMAIN}"
ENDFIVAR

FIVAR
   NAME "SRVCOMMENT"
   VALUE "{SRVCOMMENT}"
ENDFIVAR

FIVAR
   NAME "PCommDestSAP"
   VALUE "{PCommDestSAP}"
ENDFIVAR

FIVAR
   NAME "PCommID"
   VALUE "{PCommID}"
ENDFIVAR

FIVAR
   NAME "HOSTNAME"
   VALUE "{HOSTNAME}"
ENDFIVAR

FIVAR
   NAME "IPDOMAIN"
   VALUE "{IPDOMAIN}"
ENDFIVAR

FIVAR
   NAME "IP_DNS"
   VALUE "{IP_DNS}"
ENDFIVAR

FIVAR
   NAME "IP_DNS"
   VALUE "{IP_DNS}"
ENDFIVAR
```

Figure 150. PRISTINE.PRB (Part 1 of 2)

```
FIVAR
   NAME "IPMask"
   VALUE "{IPMask}"
ENDFIVAR

FIVAR
   NAME "IPRouter"
   VALUE "{IPRouter}"
ENDFIVAR

FIVAR
   NAME "IPNetAddress"
   VALUE "{IPNetAddress}"
ENDFIVAR

FIVAR
   NAME "IPNetMask"
   VALUE "{IPNetMask}"
ENDFIVAR

FIVAR
   NAME "IPHostSocksNameServer"
   VALUE "{IPHostSocksNameServer}"
ENDFIVAR

FIVAR
   NAME "IPSocksNameServer"
   VALUE "{IPSocksNameServer}"
ENDFIVAR

// Now include all the control files needed
// to configure this workstation

// Call the Probe Stream File to configure the LAN Requester
include "w:\rsp\probe\pris-IBMLAN.prb"

// Call the Probe Stream File to configure PCom
include "w:\rsp\probe\pris-PCom.prb"

// Call the Probe Stream File to make the changes in C:\CONFIG.SYS
include "w:\rsp\probe\pris-config.prb"

// Call the Probe Stream File to configure TCP/IP
// (This includes all values, except HOSTNAME, which
// it’s done in C:\CONFIG.SYS
include "w:\rsp\probe\pris-tcpip.prb"
```

Figure 151. PRISTINE.PRB (Part 2 of 2)

A.1.2 PRIS-IBMLAN.PRB

```
// Details:  This will set the value for COMPUTERNAME and DOMAIN in
//repective keys in the appl section REQUESTER
//on the TARGET systems ibmlan\IBMLAN.INI. Also it will
//              set the value for SRVCOMMENT in the appl section PEER.

STANZAINI
   TARGET  "C:\IBMLAN\IBMLAN.INI"  // No source file because you are
// supplying the value explicitly
   APPL
      TARGET "REQUESTER"// No source APPL because you are
// supplying the value explicitly
         KEY
            TARGET "COMPUTERNAME"// No source APPL because you are
// supplying the value explicitly
    VALUE "{COMPUTERNAME}"// My explicit value for the KEY
                                        // supplied through the FI Variable
                                        // {COMPUTERNAME}.
         ENDKEY
   ENDAPPL
ENDSTANZAINI

STANZAINI
   TARGET  "C:\IBMLAN\IBMLAN.INI"
   APPL
      TARGET "REQUESTER"
         KEY
            TARGET "DOMAIN"
    VALUE "{DOMAIN}"
         ENDKEY
   ENDAPPL
ENDSTANZAINI

STANZAINI
   TARGET  "C:\IBMLAN\IBMLAN.INI"
   APPL
      TARGET "PEER"
         KEY
            TARGET "SRVCOMMENT"
    VALUE "{SRVCOMMENT}"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 152. PRIS-IBMLAN.PRB

A.1.3 PRIS-PCOM.PRB

```
// Details:     This will take the value specified by REPLACEMENT
//              and replace the matching lines DestinationSAP and Identifier
//              on C:\TCPIP\PCOMOS2\PRIVATE\SNA.WS of the target system.


TEXTFILE
   SOURCE  "C:\PCOMOS2\PRIVATE\SNA.WS"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         REPLACEMENT "DestinationSAP={PCommDestSAP}"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "DestinationSAP=400021041062"
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\PCOMOS2\PRIVATE\SNA.WS"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         REPLACEMENT "Identifier={PCommID}"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "Identifier=05D5A0A3"
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 153. PRIS-PCOM.PRB

A.1.4 PRIS-CONFIG.PRB

```
// Details:     This will take the value specified by REPLACEMENT
//              and replace the matching line HOSTNAME on c:\config.sys
//              of the target system.

TEXTFILE
   SOURCE  "C:\CONFIG.SYS"      // Target file will be the same as
                                // source
   REPLACELINE
      SOURCELINE
         REPLACEMENT "SET HOSTNAME={HOSTNAME}"  // This will cause a
                                        // replacement of
                                        // SET HOSTNAME={HOSTNAME} on the
                                        // target system specified by
                                        // TARGETLINE. Where {HOSTNAME}
                                        // will be replace with its
                                        // value at execution time.
      ENDSOURCELINE
      TARGETLINE
         PATTERN "HOSTNAME"
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\CONFIG.SYS"      // Target file will be the same as
                                // source
   REPLACELINE
      SOURCELINE
         DELETELINE            // This will delete this line
                               // on the target system.
      ENDSOURCELINE
      TARGETLINE
         PATTERN "PAUSEONERROR"
      ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 154. PRIS-CONFIG.PRB

A.1.5 PRIS-TCPIP.PRB

```
// Details: This will take the value specified by REPLACEMENT
//          and replace the respective matching lines with the
//          relevant values in the files "C:\MPTN\ETC\RESOLV2",
//          "c:\tcpip\dos\etc\resolv", "C:\MPTN\BIN\SETUP.CMD",
//          "C:\MPTN\ETC\SOCKS.CFG" and "C:\MPTN\ETC\SOCKS.ENV"
//          to set up the TCP/IP Values: Domain, IP Address,
//          IP Address Mask, Default Router, Default Router Net
//          Address, Default Router Net Address Mask, Domain Name
//          Server Address, Socks Name Server and Sock Name Server
//          Address.

TEXTFILE
   SOURCE  "C:\MPTN\ETC\RESOLV2"         // Target file will be the same
                                         // as the source
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "domain {IPDOMAIN}"  // This will cause a
                                            // replacement of the line
                                            // identified by "domain"
                                            // on the target system
                                            // specified by TARGETLINE.
                                            // Where {IPDOMAIN} will be
                                            // replaced with its value at
                                            // execution time.
        ENDSOURCELINE
        TARGETLINE
           PATTERN "domain"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\ETC\RESOLV2"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "nameserver {IP_DNS}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "nameserver"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "c:\tcpip\dos\etc\resolv"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "domain {IPDOMAIN}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "domain"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 155. PRIS-TCPIP.PRB (Part 1 of 3)

```
TEXTFILE
   SOURCE  "c:\tcpip\dos\etc\resolv"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "nameserver {IP_DNS}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "nameserver"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "ifconfig lan0 {IPAddress} netmask {IPMask}"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "lan0"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
           REPLACEMENT "route add default {IPRouter} -hopcount 1"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "route add default"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\BIN\SETUP.CMD"
   REPLACELINE
        CASESENSITIVE
        SOURCELINE
          REPLACEMENT "route add -net {IPNetAddress} {IPRouter} -netmask {IPNetMask}
-hopcount 1"
        ENDSOURCELINE
        TARGETLINE
           PATTERN "route add -net 9 9.3.1.74"
        ENDTARGETLINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 156. PRIS-TCPIP.PRB (Part 2 of 3)

```
TEXTFILE
   SOURCE  "C:\MPTN\ETC\SOCKS.CFG"
   EDITLINE
        CASESENSITIVE
        SOURCESTRING
           REPLACEMENT "{IPHostSocksNameServer}"
        ENDSOURCESTRING
        TARGETSTRING
           LINEPATTERN "sockd @=socks.austin.ibm.com"
           STRINGPATTERN "socks.austin.ibm.com"
        ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE

TEXTFILE
   SOURCE  "C:\MPTN\ETC\SOCKS.ENV"
   EDITLINE
        CASESENSITIVE
        SOURCESTRING
           REPLACEMENT "{IPSocksNameServer}"
        ENDSOURCESTRING
        TARGETSTRING
           LINEPATTERN "9.14.1.30"
           STRINGPATTERN "SOCKS_NS"
        ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 157. PRIS-TCPIP.PRB (Part 3 of 3)

A.2 Migration from OS/2 Warp 3 to OS/2 Warp 4

A.2.1 GETALL3.PRB

```
// This script will gather all the information regarding Lan requester, NVDM,
// TCPIP and Netware requester
// {PRBPATH = path where all the prb files are located

INCLUDE "{PRBPATH}\LAN.PRB"
INCLUDE "{PRBPATH}\NVDM.PRB"
INCLUDE "{PRBPATH}\TCP.PRB"
INCLUDE "{PRBPATH}\NWNETCFG.PRB"
```

Figure 158. GETALL3.PRB

A.2.2 LAN.PRB

```
// Get IBMLAN Drive from config.sys and set FI variable IBMLANDRIVE
INCLUDE "{PRBPATH}\LRDRIVE.PRB"
// Computername
INCLUDE "{PRBPATH}\LRCOMNAM.PRB"
// Domain
INCLUDE "{PRBPATH}\LRDOMNAM.PRB"
// NET1
INCLUDE "{PRBPATH}\LRNET1.PRB"
//Computer name comment
INCLUDE "{PRBPATH}\LRCOMMEN.PRB"
```

Figure 159. LAN.PRB

A.2.3 LRDRIVE.PRB

```
// Category: LAN Server/Requester
// Name:    IBMLAN Drive
// Version:  1.0
// Description: This will set the FI Variable IBMLANDRIVE
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will determine which drive IBMLAN is installed
//              on and set the FI variable IBMLANDRIVE
// FIVARS:{BOOTDRIVE} =  Drive system is booted
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   EDITLINE
      SOURCESTRING
         SETFIVAR "IBMLANDRIVE"
         LINEPATTERN "IBMLAN\\NETPROG\\NETWKSTA.200 /I:"
         MATCHPATTERN "[A-Z]{1}:"
      ENDSOURCESTRING
      TARGETSTRING
         LOCATEDACTION NOTHING
         NOTLOCATEDACTION NOTHING
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 160. LRDRIVE.PRB

A.2.4 LRCOMNAM.PRB

```
// Category: LAN Server/Requester
// Name:    Computer Name
// Version:  1.0
// Description: Migrate Computername from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches
//              "COMPUTERNAME" from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "COMPUTERNAME"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 161. LRDRIVE.PRB

A.2.5 LRDOMNAM.PRB

```
// Category:LAN Server/Requester
// Name:    Domain Name
// Version:     1.0
// Description: Migrate Domainname from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:    3.0, 4.0, 5.0
// Details:  This will migrate the "Domainname" line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "DOMAIN"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 162. LRDRIVE.PRB

A.2.6 LRNET1.PRB

```
// Category: LAN Server/Requester
// Name:    NET1 Line
// Version:  1.0
// Description: Migrate NET1 Line from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire NET1 line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "NETWORKS"
         KEY
            SOURCE "  net1"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 163. LRNET1.PRB

A.2.7 LRCOMMEN.PRB

```
// Category: LAN Server/Requester
// Name:    Computer Name Description
// Version:  1.0
// Description: Migrate computername description from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches
//              "SRVCOMMENT" from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "PEER"
         KEY
            SOURCE "SRVCOMMENT"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 164. LRCOMMEN.PRB

A.2.8 NVDM.PRB

```
INCLUDE "{PRBPATH}\MAINTDRV.PRB"       // Set Maintenance partition drive
INCLUDE "{PRBPATH}\NVDMINI.PRB"        // ClientName and ServerName of IBMNVDM2.INI
```

Figure 165. NVDM.PRB

A.2.9 MAINTDRV.PRB

```
// Category: Maintenance Partition
// Version:  1.0
// Description: Manually set Maintenance partition Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables MAINT
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance Partition

FIVAR
   NAME "MAINTDRIVE"
   VALUE "E:"
ENDFIVAR
```

Figure 166. MAINDRV.PRB

A.2.10 NVDMINI.PRB

```
// Category: NVDM
// Name:    WORKSTATION AND SERVER NAMES
// Version:  1.0
// Description: Migrate WORKSTATION AND SERVER NAMES FROM IBMNVDM2.INI
// Details:  This will migrate the lines that matches ServerName and ClientName
//              and ClientName in the IBMNVDM2.INI of the maintenance partition
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance partition

TEXTFILE
   SOURCE  "{MAINTDRIVE}\IBMNVDM2\IBMNVDM2.INI"
   REPLACELINE
      SOURCELINE
        PATTERN "ServerName"
      ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE
        PATTERN "ClientName"
      ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 167. NVDMINI.PRB

A.2.11 TCP.PRB

```
INCLUDE "{PRBPATH}\TCPDRIV.PRB"           // Drive letters TCPIPDRIVE & MPTNDRIVE
INCLUDE "{PRBPATH}\TCPRSLV2.PRB"           // Resolv2 file
INCLUDE "{PRBPATH}\TCPRSLV.PRB"            //Resolv file
INCLUDE "{PRBPATH}\TCPHSTNM.PRB"           // Host name
INCLUDE "{PRBPATH}\TCPSETUP.PRB"           // setup.cmd
```

Figure 168. TCP.PRB

A.2.12 TCPDRIV.PRB

```
// Category: TCPIP
// Name:    Installed Drives
// Version:  1.0
// Description: Manually set TCPIP Drive and MTPN Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables TCPIPDRIVE and MPTNDRIVE
// FIVARS:  {TCPIPDRIVE} = Drive where TCPIP is installed
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

FIVAR
   NAME "TCPIPDRIVE"
   VALUE "C:"
ENDFIVAR

FIVAR
   NAME "MPTNDRIVE"
   VALUE "C:"
ENDFIVAR
```

Figure 169. TCPDRIVI.PRB

A.2.13 TCPRSLV2.PRB

```
// Category: TCPIP
// Name:    RESOLV2
// Version:  1.0
// Description: Migrate RESOLV2
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV2
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

COPYFILES
   SOURCE "{MPTNDRIVE}\MPTN\ETC"
   INCLUDEFILES
      PATTERN "RESOLV2"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 170. TCPRSLV2I.PRB

A.2.14 TCPRSLV.PRB

```
// Category: TCPIP
// Name:    RESOLV
// Version:  1.0
// Description: Migrate RESOLV
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV (for DOS sesions)
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

COPYFILES
   SOURCE "{TCPIPDRIVE}\TCPIP\DOS\ETC"
   INCLUDEFILES
      PATTERN "RESOLV"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 171. TCPRSLV.PRB

A.2.15 TCPHSTNM.PRB

```
// Category: TCPIP
// Name:    HOSTNAME
// Version:  1.0
// Description: Migrate HOSTNAME from config.sys
// From:  2.0, 3.0
// To:  2.0, 3.0
// Details:  This will migrate the entire line that matches HOSTNAME from config.sys
// FIVARS:  {BOOTDRIVE} = Drive system is booted from

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   REPLACELINE
      SOURCELINE PATTERN "HOSTNAME" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 172. TCPHSTNM.PRB

A.2.16 TCPSETUP.PRB

```
// Category:    TCP/IP
// Name:        SETUP.CMD
// Version:     1.0
// Description: This will modify strings on SETUP.CMD
// From:        4.0
// To:          4.1
// Details:     This will get the values for net address, mask and router
// FIVARS:      {MPTNDRIVE}  = Drive where MPTN is installed
// Changes:

TEXTFILE
  SOURCE  "{MPTNDRIVE}\MPTN\BIN\SETUP.CMD"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "ifconfig lan0 "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "ifconfig lan0"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE

   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "netmask"
         MATCHPATTERN "\ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "netmask"
         STRINGPATTERN "\ {1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
         STRINGSCOPE LAST
      ENDTARGETSTRING
   ENDEDITLINE

   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add default "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add default"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
```

Figure 173. TCPSETUP.PRB (Part 2 of 2)

```
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add net "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add -net 9"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE

ENDTEXTFILE
```

Figure 174. TCPSETUP.PRB (Part 2 of 2)

A.3 Migration from OS/2 V2.11 to OS/2 Warp 4

A.3.1 GETALL21.PRB

```
// This script will gather all the information regarding Lan requester, MPTS,
// TCPIP and Netware requester
// {PRBPATH = path where all the prb files are located

INCLUDE "{PRBPATH}\LAN21.PRB"
INCLUDE "{PRBPATH}\TCP21.PRB"
INCLUDE "{PRBPATH}\NVDM21.PRB"
```

Figure 175. GETALL21.PRB

A.3.2 LAN21.PRB

```
INCLUDE "{PRBPATH}\LRDRIVE21.PRB"           // Get IBMLAN Drive from config.sys
//                                             and set FI variable IBMLANDRIVE
INCLUDE "{PRBPATH}\LRCOMNAM21.PRB"           // Computername
INCLUDE "{PRBPATH}\LRDOMNAM21.PRB"           // Domain
```

Figure 176. LAN21.PRB

A.3.3 LRDRIVE21.PRB

```
// Category: LAN Server/Requester
// Name:    IBMLAN Drive
// Version:  1.0
// Description: This will set the FI Variable IBMLANDRIVE
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will determine which drive IBMLAN is installed on and set the FI
variable IBMLANDRIVE
// FIVARS:{BOOTDRIVE} =  Drive system is booted
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   EDITLINE
      SOURCESTRING
         SETFIVAR "IBMLANDRIVE"
         LINEPATTERN "IBMLAN\\NETPROG\\NETWKSTA.200 /I:"
         MATCHPATTERN "[A-Z]{1}:"
      ENDSOURCESTRING
      TARGETSTRING
         LOCATEDACTION NOTHING
         NOTLOCATEDACTION NOTHING
      ENDTARGETSTRING
   ENDEDITLINE
ENDTEXTFILE
```

Figure 177. LRDRIVE21.PRB

A.3.4 LRCOMNAM21.PRB

```
// Category: LAN Server/Requester
// Name:    Computer Name
// Version:  1.0
// Description: Migrate Computername from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches "COMPUTERNAME" from
IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "COMPUTERNAME"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 178. LRCOMNAM21.PRB

A.3.5 LRDOMNAM21.PRB

```
// Category:LAN Server/Requester
// Name:    Domain Name
// Version:     1.0
// Description: Migrate Domainname from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:    3.0, 4.0, 5.0
// Details:  This will migrate the "Domainname" line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "REQUESTER"
         KEY
            SOURCE "DOMAIN"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 179. LRDOMNAM21.PRB

A.3.6 LRNET1.PRB

```
// Category: LAN Server/Requester
// Name:    NET1 Line
// Version:  1.0
// Description: Migrate NET1 Line from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire NET1 line from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBM LAN Requester or Server is installed

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "NETWORKS"
         KEY
            SOURCE "  net1"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 180. LRNET1.PRB

A.3.7 LRCOMMEN.PRB

```
// Category: LAN Server/Requester
// Name:    Computer Name Description
// Version:  1.0
// Description: Migrate computername description from IBMLAN.INI
// From:  3.0, 4.0, 5.0
// To:  3.0, 4.0, 5.0
// Details:  This will migrate the entire line that matches
//              "SRVCOMMENT" from IBMLAN.INI
// FIVARS:  {IBMLANDRIVE} = Drive IBMLAN is installed
// Changes:

STANZAINI
   SOURCE  "{IBMLANDRIVE}\IBMLAN\IBMLAN.INI"
   APPL
      SOURCE "PEER"
         KEY
            SOURCE "SRVCOMMENT"
         ENDKEY
   ENDAPPL
ENDSTANZAINI
```

Figure 181. LRCOMMEN.PRB

A.3.8 NVDM21.PRB

```
INCLUDE "{PRBPATH}\MAINTDRV21.PRB"     // Set Maintenance partition drive
INCLUDE "{PRBPATH}\NVDMINI21.PRB"      // ClientName and ServerName of IBMNVDM2.INI
```

Figure 182. NVDM21.PRB

A.3.9 MAINTDRV21.PRB

```
// Category: Maintenance Partition
// Version:  1.0
// Description: Manually set Maintenance partition Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables MAINT
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance Partition

FIVAR
   NAME "MAINTDRIVE"
   VALUE "E:"
ENDFIVAR
```

Figure 183. MAINTDRV21.PRB

A.3.10 NVDMINI21.PRB

```
// Category: NVDM
// Name:    WORKSTATION AND SERVER NAMES
// Version:  1.0
// Description: Migrate WORKSTATION AND SERVER NAMES FROM IBMNVDM2.INI
// Details:  This will migrate the lines that matches ServerName and ClientName
//              and ClientName in the IBMNVDM2.INI of the maintenance partition
// FIVARS:  {MAINTDRIVE} = Drive of the maintenance partition

TEXTFILE
   SOURCE  "{MAINTDRIVE}\IBMNVDM2\IBMNVDM2.INI"
   REPLACELINE
      SOURCELINE
        PATTERN "ServerName"
      ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE
        PATTERN "ClientName"
      ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 184. NVDMINI21.PRB

A.3.11 TCP21.PRB

```
INCLUDE "{PRBPATH}\TCPDRIV21.PRB"           // Drive letters TCPIPDRIVE & MPTNDRIVE
INCLUDE "{PRBPATH}\TCPRSLV221.PRB"           // Resolv2 file
INCLUDE "{PRBPATH}\TCPRSLV21.PRB"            //Resolv file
INCLUDE "{PRBPATH}\TCPHSTNM21.PRB"           // Host name
INCLUDE "{PRBPATH}\TCPSETUP21.PRB"           // setup.cmd
```

Figure 185. TCP21.PRB

A.3.12 TCPDRIV21.PRB

```
// Category: TCPIP
// Name:    Installed Drives
// Version:  1.0
// Description: Manually set TCPIP Drive and MTPN Drive
// From:  ANY
// To:  ANY
// Details:  Manually set the fi variables TCPIPDRIVE and MPTNDRIVE
// FIVARS:  {TCPIPDRIVE} = Drive where TCPIP is installed
// FIVARS:{MPTNDRIVE}  = Drive where MPTN is installed

FIVAR
   NAME "TCPIPDRIVE"
   VALUE "C:"
ENDFIVAR

FIVAR
   NAME "MPTNDRIVE"
   VALUE "C:"
ENDFIVAR
```

Figure 186. TCPDRIV21.PRB

A.3.13 TCPRSLV221.PRB

```
// Category: TCPIP
// Name:    RESOLV2
// Version:  1.0
// Description: Migrate RESOLV2
// From:  2.0
// To:  4.0
// Details:  This will migrate the file RESOLV2
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

TEXTFILE
   SOURCE  "{TCPIPDRIVE}\TCPIP\ETC\RESOLV"
   TARGET  "{MPTNDRIVE}\MPTN\ETC\RESOLV2"
   REPLACELINE
      SOURCELINE PATTERN "DOMAIN" ENDSOURCELINE
   ENDREPLACELINE
   REPLACELINE
      SOURCELINE PATTERN "NAMESERVER" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 187. TCPRSLV221.PRB

A.3.14 TCPRSLV21.PRB

```
// Category: TCPIP
// Name:    RESOLV
// Version:  1.0
// Description: Migrate RESOLV
// From:  3.0
// To:  4.0
// Details:  This will migrate the file RESOLV (for DOS sesions)
// FIVARS:{TCPIPDRIVE} = Drive where TCPIP is installed

COPYFILES
   SOURCE "{TCPIPDRIVE}\TCPIP\DOS\ETC"
   INCLUDEFILES
      PATTERN "RESOLV"
   ENDINCLUDEFILES
ENDCOPYFILES
```

Figure 188. TCPRSLV21.PRB

A.3.15 TCPHSTNM21.PRB

```
// Category: TCPIP
// Name:    HOSTNAME
// Version:  1.0
// Description: Migrate HOSTNAME from config.sys
// From:  2.0, 3.0
// To:  2.0, 3.0
// Details:  This will migrate the entire line that matches HOSTNAME from config.sys
// FIVARS:  {BOOTDRIVE} = Drive system is booted from

TEXTFILE
   SOURCE  "{BOOTDRIVE}\CONFIG.SYS"
   REPLACELINE
      SOURCELINE PATTERN "HOSTNAME" ENDSOURCELINE
   ENDREPLACELINE
ENDTEXTFILE
```

Figure 189. TCPHSTNM21.PRB

A.3.16 TCPSETUP21.PRB

```
// Category:    TCP/IP
// Name:        SETUP.CMD
// Version:     1.0
// Description: This will modify strings on SETUP.CMD
// From:        4.0
// To:          4.1
// Details:     This will get the values for net address, mask and router
// FIVARS:      {MPTNDRIVE}  = Drive where MPTN is installed
// Changes:

TEXTFILE
  SOURCE  "{TCPIPDRIVE}\TCPIP\BIN\SETUP.CMD"
  TARGET  "{MPTNDRIVE}\MPTN\BIN\SETUP.CMD"
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "ifconfig lan0 "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "ifconfig lan0"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE

   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "netmask"
         MATCHPATTERN "\ [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "netmask"
         STRINGPATTERN "\ {1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
         STRINGSCOPE LAST
      ENDTARGETSTRING
   ENDEDITLINE

   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add default "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add default"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE
```

Figure 190. TCPSETUP21.PRB (Part 1 of 2)

```
   EDITLINE
      CASESENSITIVE
      SOURCESTRING
         LINEPATTERN "route add net "
         MATCHPATTERN "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"
      ENDSOURCESTRING
      TARGETSTRING
         LINEPATTERN "route add -net 9"
         STRINGPATTERN "{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]\.{1,3}[0-9]"
      ENDTARGETSTRING
   ENDEDITLINE

ENDTEXTFILE
```

Figure 191. TCPSETUP21.PRB (Part 2 of 2)

Appendix B. NetView DM/2 Change Profile Files

B.1 Pristine Installation

B.1.1 LOADREXX.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = REXX Support for Pristine Clients
End

Section Install
Begin
      Program = SA:\EXE\cmd.exe
      Parms =  /C $(SA)\exe\detrexx.cmd $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 192. LOADREXX.PRO

B.1.2 DISKPREP.PRO

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir) /L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 193. DISKPREP.PRO

B.1.3 FiIBM750.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
      Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
/L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 194. FiIBM750.PRO

B.1.4 PRISTINE1.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = Software
      GlobalName = IBM.CONFIGUR.PRISTINE1.REF.1
      Description = Configure a Pristine WS (Phase 1)
End

Section Install
Begin
      Program = $(SA)\EXE\WSCONFIG.CMD
      Parms = "$(WorkStatName) PRISTINE"
      WorkingDir = C:\
End
```

Figure 195. PRISTINE1.PRO

B.1.5 PRISTINE2.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = Software
      GlobalName = IBM.CONFIGUR.PRISTINE2.REF.1
      Description = Configure a Pristine WS (Phase 2)
End

Section Install
Begin
      Program = $(SB)\LOG\$(WorkStatName)\WSDETAILS.CMD
      WorkingDir = C:\
End
```

Figure 196. PRISTINE2.PRO

B.2 Migration from OS/2 Warp 3 to OS/2 Warp 4

B.2.1 PROBE3.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.PROBE.GETALL3.REF.1
End

Section Install
Begin
      Program = SA:\exe\probe.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
/b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir /l:$(sb)\log\$(WorkStatName)
/x:4 $(sa)\rsp\probe\getall3.prb"
End
```

Figure 197. PROBE3.PRO

B.2.2 CM2PCOMM.PRO

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.CM2PCOMM.REF.4
     Description="Migration from CM/2 to Personal Communications"
End

Section Install
Begin
      Program =  SA:\EXE\CM2PCOMM.CMD
      Parms= "C:\CMLIB $(WorkStatName)"
End
```

Figure 198. CM2PCOMM.PRO

B.2.3 GETNVDM3.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.GETNVDM.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\getinfo.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 199. GETNVDM3.PRO

B.2.4 LOADREXX.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = REXX Support for Pristine Clients
End

Section Install
Begin
      Program = SA:\EXE\cmd.exe
      Parms =  /C $(SA)\exe\detrexx.cmd $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 200. LOADREXX.PRO

B.2.5 DISKPREP.PRO

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir) /L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 201. DISKPREP.PRO

B.2.6 FiIBM750.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
      Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
/L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 202. FIIBM750.PRO

B.2.7 SETNVDM3.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.SETNVDM.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\setinfo.cmd
      Parms = "$(WorkStatName)"
      PhaseEnd = yes
End
```

Figure 203. SETNVDM3.PRO

B.2.8 CONFIGUR3.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.CONFIGUR.MIGRAT3.REF.1
End

Section Install
Begin
      Program = SA:\exe\configur.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
/b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir /l:$(sb)\log\$(WorkStatName)
/x:4"
End
```

Figure 204. CONFIGUR3.PRO

B.2.9 COPYCONF.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.COPYCONF.REF.1
End

Section Install
Begin
      Program = SA:\exe\copyconf.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 205. COPYCONF.PRO

B.3 Migration from OS/2 V2.11 to OS/2 Warp 4

B.3.1 PROBE21.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.PROBE.GETALL21.REF.1
End

Section Install
Begin
      Program = SA:\exe\probe.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
/b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir /l:$(sb)\log\$(WorkStatName)
/x:4 $(sa)\rsp\probe\getall21.prb"
End
```

Figure 206. PROBE21.PRO

B.3.2 CM2PCOMM.PRO

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.CM2PCOMM.REF.4
     Description="Migration from CM/2 to Personal Communications"
End

Section Install
Begin
      Program =  SA:\EXE\CM2PCOMM.CMD
      Parms= "C:\CMLIB $(WorkStatName)"
End
```

Figure 207. CM2PCOMM.PRO

B.3.3 GETNVDM21.PRO

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.CM2PCOMM.REF.4
     Description="Migration from CM/2 to Personal Communications"
End

Section Install
Begin
      Program =  SA:\EXE\CM2PCOMM.CMD
      Parms= "C:\CMLIB $(WorkStatName)"
End
```

Figure 209. GETNVDM21.PRO

B.3.4 LOADREXX.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.UTIL.REXXSTART.INST.REF.1
      Description = REXX Support for Pristine Clients
End

Section Install
Begin
      Program = SA:\EXE\cmd.exe
      Parms =  /C $(SA)\exe\detrexx.cmd $(SourceDir) $(WorkingDir)
      SourceDir = SA:\EXE
      WorkingDir = SA:\EXE
End
```

Figure 210. LOADREXX.PRO

B.3.5 DISKPREP.PRO

```
TargetDir=C:\

Section Catalog
Begin
   ObjectType  = SOFTWARE
   GlobalName  = IBM.OS2.DISKPREP.INST.REF.1
   Description = Automated Partitioning for OS/2 PCs
End

Section Install
Begin
   Program      = SA:\EXE\DISKPREP.CMD
   Parms        = /R:$(SA)\RSP /E:$(WorkingDir) /S:$(SourceDir) /L:$(LogFile1)
   SourceDir    = SA:\IMG\Warp4
   WorkingDir   = SA:\EXE
   LogFile1     = SB:\LOG\$(WorkStatName)\DISKPREP.LOG
End
```

Figure 211. DISKPREP.PRO

B.3.6 FiIBM750.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 4

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.REPLICAT.FINANCE.IBM750.REF.1
      Description = Load the replica from SYSCOPY for Finance IBM-750
End

Section Install
Begin
      Program = W:\EXE\REPLICAT.EXE
      Parms = "/C:IBM750 /B:$(SA)\IMG\FINANCE /F:$(SA)\RSP\FiIBM750.RSP
/L:$(SB)\LOG\$(WorkStatName)"
End
```

Figure 212. FiIBM750.PRO

B.3.7 SETNVDM21.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.SETNVDM.MIGRAT21.REF.1
End

Section Install
Begin
      Program = SA:\exe\setinfo.cmd
      Parms = "$(WorkStatName)"
      PhaseEnd = yes
End
```

Figure 213. SETNVDM21.PRO

B.3.8 CONFIGUR21.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.CONFIGUR.MIGRAT21.REF.1
End

Section Install
Begin
      Program = SA:\exe\configur.exe
      Parms = "/v:{prbpath}=$(sa)\rsp\probe /e:$(WorkStatName)
/b:$(sa)\img\config\$(WorkStatName) /w:$(sb)\workdir /l:$(sb)\log\$(WorkStatName)
/x:4"
End
```

Figure 214. CONFIGUR21.PRO

B.3.9 COPYCONF.PRO

```
TargetDir = "C:\IBMNVDM2"
CompNameLen = 3

Section Catalog
Begin
      ObjectType = SOFTWARE
      GlobalName = IBM.COPYCONF.REF.1
End

Section Install
Begin
      Program = SA:\exe\copyconf.cmd
      Parms = "$(WorkStatName)"
End
```

Figure 215. COPYCONF.PRO

B.4 Miscellaneous

B.4.1 CMD.PRO

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.WARP4.CMD.REF.4
     Description="OS/2 Warp 4 Command Prompt"
End

Section Install
Begin
      Program =  SA:\IMG\WARP4\CMD.EXE
End
```

Figure 216. CMD.PRO

B.4.2 CMD211.PRO

```
TargetDir=C:\

Section Catalog
Begin
     ObjectType=Software
     GlobalName=IBM.OS2211.CMD.REF.1
     Description="OS/2 2.11 Command Prompt"
End

Section Install
Begin
      Program =  SA:\EXE\CMD211.EXE
End
```

Figure 217. CMD211.PRO

Appendix C. Program Source Code Listings

This Appendix lists the source code of all our programs used in the order of appearance in the book.

C.1 COPYNVDM.CMD

```
/* REXX cmd file that copies the NVDM/2 software Distribution                 */
/* Agent to a maintenance partition and creates a IBMNVDM2.INI file with      */
/* the correct workstation name and path statements                           */
/* Additionaly the program copies the REXX Support to the maintenance Partion */
/* and updates the CONFIG.SYS of the maintenance partition                    */
/* Needed Files in the directory this program is located:                     */
/*                                                                            */
/* BOS2REXX EXE                                                               */
/* REXX     211, REXX     3, REXX     4                                       */
/* REXXAPI  DLL, REXXAPI  3, REXXAPI  4                                       */
/* REXXINIT 211, REXXINIT 3, REXXINIT 4                                       */
/* CUBE     CMD                                                               */
/* CONFIG   PRO                                                               */
/* IBMNVDM2 PRO                                                               */
/* XCOPY    EXE                                                               */




PARSE UPPER ARG source target version .


RCode = 0
'ECHO OFF'
'ECHO COPYNVDM >> output.txt'

IF (source == '') | (target == '') | (version == '') then do
   say 'INCORRECT PARAMETERS !'
   say 'Syntax : COPYNVDM <sourcedrive> <targetdrive> <OS/2-version>'
   say ''
   say 'where sourcedrive is the drive the installed NVDM/2 Client,'
   say 'is located, targetdrive the drive to copy the client to and'
   say 'OS/2-version is the Version of OS/2 you use on this maschine,'
   say 'valid are 211, 3 and 4.'
   say ''
   say 'Example: COPYNVDM C: E: 4'
   RCode = 4
end

if length(source) == 1 then source = source||':'
if length(target) == 1 then target = target||':'

if RCode == 0 then do
   if (version == '4') | (version == '3') | (version == '211') then NOP
   else do
      say 'OS/2 Version is not valid !'
```

Figure 218. COPYNVDM.CMD (Part 1 of 3)

```
      say 'Allowed are the values 4, 3 and 211'
      RCode = 4
   end /* do */
end /* do */

if RCode == 0 then do
   say 'COPYNVDM is working. One Moment please...'
   'CD 'target||'\OS2'
   RCode = RC
   if RCode <> 0 then do
      say 'No OS2 subdirectory found on targetdrive 'target
      RCode = 8
   end /* do */
end

CALL buildprofile

if RCode == 0 then do
   'XCOPY 'source||'\IBMNVDM2\*.* 'target||'\IBMNVDM2\*.* /s /e >> output.txt'
   if RCode <> 0 then do
      say 'No NVDM Client was found on the C Partition !'
      RCode = 8
   end
end

if RCode == 0 then do
   'XCOPY BOS2REXX.EXE 'target||'\OS2 >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXX.'||version' 'target||'\OS2\REXX.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXXINIT.'||version' 'target||'\OS2\REXXINIT.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   'COPY REXXAPI.'||version' 'target||'\OS2\REXXAPI.DLL >> output.txt'
   if RC <> 0 then RCode = RC
   if RCode <> 0 then do
      say 'Copy of the REXX Support File failed!'
      RCode = 8
   end
end

if RCode == 0 then do
   RC = STREAM(target||'\IBMNVDM2\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
   if RC == '' then RCode = 1
   RC = STREAM(target||'\CONFIG.SYS', 'C', 'QUERY EXISTS')
   if RC == '' then RCode = 1
   if RCode <> 0 then do
      say target||'\CONFIG.SYS or 'target||'\IBMNVDM2\IBMNVDM2.INI'
      say 'were not found!'
      RCode = 8
   end
   if RCode == 0 then do
      cubearg = 'ibmnvdm2.pro 'target||'\IBMNVDM2\IBMNVDM2.INI'
      RC = 'cube.cmd'(cubearg)
      if RC <> 0 then RCode = RC
      cubearg = 'config.pro 'target||'\CONFIG.SYS'
      RC = 'cube.cmd'(cubearg)
      if RC <> 0 then RCode = RC
      if RCode <> 0 then do
```

Figure 219. COPYNVDM.CMD (Part 2 of 3)

```
         say ’One or more Errors were found executing the changes’
         say ’to E:\CONFIG.SYS and E:\IBMNVDM2\IBMNVDM2.INI’
         RCode = 8
      end
   end
end

if RCode == 0 then do
   say ’’
   say ’COPYNVDM ended successfull’
end /* do */
else do
   say ’’
   say ’COPYNVDM ended unsucessfull with RCode = ’RCode
end /* do */
’DEL OUTPUT.TXT’
’ECHO ON’
return RCode

buildprofile:
RC = STREAM(’CONFIG.PRO’, ’C’, ’QUERY EXISTS’)
if RC <> ’’ then ’DEL CONFIG.PRO’
RC = STREAM(’IBMNVDM2.PRO’, ’C’, ’QUERY EXISTS’)
if RC <> ’’ then ’DEL IBMNVDM2.PRO’

cubeline = ’RepString "’source’" WITH "’target’" ( ALL’
’ECHO ’cubeline’ >>IBMNVDM2.PRO’
cubeline = ’RepString "ZYXWVUTSRQPONMLKJIHGF" WITH "ZYWXVUTSRQPONMLKJIHGF" ( ALL’
’ECHO’ cubeline’ >>IBMNVDM2.PRO’
Call Stream ’IBMNVDM2.PRO’,’c’,’close’

cubeline = ’AddLine "DEVICE=’target||’\IBMNVDM2\BIN\anxifpid.sys" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddLine "DEVICE=’target||’\IBMNVDM2\BIN\anxifcom.sys" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddLine "IFS=’target||’\IBMNVDM2\BIN\anxifcom.ifs" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddLine "DEVICE=’target||’\IBMNVDM2\BIN\ANXACAIP.SYS" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddLine "RUN=’target||’\OS2\BOS2REXX.EXE" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’RepString "’target||’\OS2\CMD.EXE" WITH "’
cubeline2 = target||’\IBMNVDM2\BIN\ANXCMCLC.EXE" IN "SET OS2_SHELL"’
’ECHO’ cubeline||cubeline2’ >> CONFIG.PRO’
cubeline = ’AddString ";’target||’\IBMNVDM2\DLL;W:\DLL;"  IN "LIBPATH" ( IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddString ";’target||’\IBMNVDM2\BIN;W:\EXE;W:\CMD;"  IN "SET PATH" (
IFNEW AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’AddString ";’target||’\IBMNVDM2;W:\EXE;W:\CMD;"  IN "SET DPATH" ( IFNEW
AFTER’
’ECHO’ cubeline’ >> CONFIG.PRO’
cubeline = ’RepString ";;" WITH ";" ( ALL’
’ECHO’ cubeline’ >> CONFIG.PRO’
Call Stream ’CONFIG.PRO’,’c’,’close’
RETURN
```

Figure 220. COPYNVDM.CMD (Part 3of 3)

C.2 CM2PCOMM.CMD

```
/****************************************************************************/
/*                                                                          */
/* REXX Procedure to migrate the CM/2 Configuration to a PCOMM Configuration */
/*                                                                          */
/****************************************************************************/



parse arg cmpath workstationname cmconfigfile


/****************************************************************************/
/* Variables Dokumentation                                                  */
/*                                                                          */
/* RCode             Returncode CM2PCOMM gives back to the invoking program  */
/* RC                Returncode CM2PCOMM receives from programs it invokes   */
/* workstationname   NVDM/2 name of the workstation                         */
/* cmconfigfile      Name of the CM/2 configuration File                    */
/* cmpath            full qualified path in which CM/2 is installed         */
/* drive             driveletter + colon extracted from cmpath              */
/* path              path extracted from cmpath without leading backslash    */
/* slashpos          position of the first backslash in cmpath              */
/* point             position of the point in cmconfigfile                  */
/* configname        name of the CM/2 configurationfile without the point    */
/*                   and the extension                                      */
/* namelength        length of cmconfigfile                                 */
/* rsp_file          holds the cmpath and the configname with extension .RSP */
/* session3270       counter for the difines 3270 Sessions in the CM/2       */
/*                   Response File                                          */
/* session5250       counter for the difines 5250 Sessions in the CM/2       */
/*                   Response File                                          */
/* type1             counter for the Sessions with Type 1 (Terminal Sessions)*/
/* type2             counter for the Sessions with Type 2 (Printer Sessions) */
/* localcp           counter for the LOCAL_CP Section in the CM/2 Response   */
/*                   File                                                   */
/* logicallink       counter for the LOGICAL_LINK Section in the CM/2        */
/*                   Response File                                          */
/* argument          holds the part of the Line read from the RSP File       */
/*                   before the '=' Sign                                    */
/* wert              holds the part of the Line read from the Response File  */
/*                   after the '=' Sign                                     */
/* PU                holds the value of the PU                              */
/* nodeid            holds the value of the NODE_ID                         */
/* address           holds the value of the DESTINATION_ADDRESS             */
/* tmpPU             holds a temporary value of the PU                      */
/* trash             Variable to store unneeded information from the parse   */
/*                   command                                                */
/* numberOfSessions  stores the number of the defined 3270 Sessions in the   */
/*                   CM/2 Response File                                     */
/* destpath          drive and path on the Code Server where the PComm Files */
/*                   are copied                                             */
/* i                 loop counter                                           */
/****************************************************************************/
```

Figure 221. CM2PCOMM (Part 1 of 7)

```
'ECHO OFF'

RCode = 0
destpath = 'W:\IMG\CONFIG\'||workstationname
logpath = 'X:\LOG\'||workstationname||'\'

say 'Checking the Arguments .... '

if cmpath <> '' then do
    slashpos=pos('\',cmpath)
    drive=substr(cmpath,1,slashpos-1)
    path=substr(cmpath,slashpos+1,length(cmpath))
    RC = STREAM(cmpath||'\CMRECORD.EXE', 'C', 'QUERY EXISTS')
    if RC == '' then do
       say ''
       say 'The specified directory does not exist'
       say 'or CM/2 not found in the specified directory...'
       call argerror
    end /* do */
end /* do */
else do
   say ''
   say 'CM/2 install path was not specified ...'
   call argerror
end /* do */

if RCode == 0 then do
    if (cmconfigfile <> '')then do
       cmconfigfile=strip(cmconfigfile)
       point=lastpos(".", cmconfigfile)
       if point <> 0 then do
          configname=substr(cmconfigfile,1,point-1)
       end /* do */
       else do
          configname = cmconfigfile
          cmconfigfile = cmconfigfile||'.cfg'
       end /* do */
       namelength=length(configname)
       if namelength > 8 then do
          say ''
          say 'The specified CM/2 configuration file is not valid ...'
          call argerror
       end /* do */
    end /* do */
    else do
       say ''
       say 'CM/2 configuration file was not specified'
       say 'the default configuration file will be extracted...'
    end /* do */
    if cmconfigfile <> '' then do
       'DIR 'cmpath||'\'||cmconfigfile' >> output.txt'
       if RC <> 0 then do
          say ''
          say 'The specified CM/2 configuration file does not exist ...'
          call argerror
       end /* do */
    end /* do */
end /* do */
```

Figure 222. CM2PCOMM (Part 2 of 7)

```
if RCode == 0 then do
   say 'Arguments OK ....'
   say ''
end /* do */


if RCode == 0 then do
   say 'Extracting CM/2 Configuration ....'
   if cmconfigfile <> '' then do
      ' 'cmpath||'\CMRECORD 'cmpath||'\'||configname' >> output.txt'
      if RC <> 0 then do   end /* do */
         say 'Extraction of the configuration failed !!!!!!!!!!!!!!!'
         RCode = 8
      end /* do */
      else do
         say 'Configuration extracted ....'
         say ''
      end
   end /* do */
   else do
      ' 'cmpath||'\CMRECORD /D >> output.txt'
      if RC <> 0 then do   end /* do */
         say 'Extraction of the configuration failed !!!!!!!!!!!!!!!'
         RCode = 8
      end /* do */
      else do
         say 'Configuration extracted ....'
         say ''
         found = 0
         do while found == 0
            line.linenum = Linein(output.txt)
            if substr(line.linenum,1, 26) == 'Recording of configuration' then do
               cmconfigfile = word(line.linenum, 4)
               slashpos = lastpos("\", cmconfigfile)
               configname = substr(cmconfigfile, slashpos+1, length(cmconfigfile))
               configname = strip(configname)
               found = 1
               CALL Lineout(output.txt)
            end /* do */
         end
      end
   end /* do */
end /* do */

if RCode == 0 then do
   say 'Retrieving configuration values ....'
   rsp_file = cmpath||'\'||configname||'.rsp'
   session3270 = 0
   session5250 = 0
   type1 = 0
   type2 = 0
   localcp = 0
   logicallink = 0
   argument = ''
   wert = ''
   PU = ''
```

Figure 223. CM2PCOMM (Part 3 of 7)

```
 nodeid = ''
   address = ''

   do while lines(rsp_file)
      line.linenum = Linein(rsp_file)
      if session5250 == 0 then do
         select
           when line.linenum == '3270_SESSION = (' then session3270 = session3270 + 1
            when line.linenum == '     SESSION_TYPE = 1' then do
               type1 = type1 + 1
            end /* do */
           when line.linenum == '5250_SESSION = (' then session5250 = session5250 + 1
            when line.linenum == '     SESSION_TYPE = 2' then type2 = type2 + 1
         otherwise NOP
         end  /* select */
      end /* do */
      if (logicallink < 2) | (localcp < 2) then do
         select
            when line.linenum == 'LOCAL_CP = (' then localcp = localcp + 1
           when line.linenum == 'LOGICAL_LINK = (' then logicallink = logicallink + 1
         otherwise NOP
         end  /* select */
         parse VALUE line.linenum with argument '=' wert
         select
            when strip(argument) == 'NAME' then do
               PARSE VALUE wert with trash '.' tmpPU
               if tmpPU <> '' then PU = tmpPU
               PU = strip(PU)
            end /* do */
            when strip(argument) == 'NODE_ID' then do
               nodeid = substr(wert, 5,length(wert))
               nodeid = strip(nodeid)
            end /* do */
            when strip(argument) == 'DESTINATION_ADDRESS' then do
               address = strip(wert)
            end /* do */
         otherwise NOP
         end  /* select */
      end /* do */
   end /* do */
   type1 = type1 - type2
   numberOfSessions = type1
end /* do */

if RCode == 0 then do
   do i = 1 to (length(nodeid))
      rc = verify(substr(nodeid,i,1),'0123456789ABCDEF')
      if RC <> 0 then do
         RCode = 12
         say ''
         say '*********************************************************************'
         say '*                                                                   *'
         say '* The CM/2 Values are not compatible with Personal Communications ! *'
         say '*                                                                   *'
         say '* PComm requires hexadecimal values, please contact your            *'
         say '* Administrator to receive valid definitions.                       *'
         say '*                                                                   *'
        say '* For further information refer to the CM/2 and PComm Doccumentation.*'
```

Figure 224. CM2PCOMM (Part 4 of 7)

```
         say '*                                                                    *'
         say '**********************************************************************'
         say ''
      end /* do */
   end /* do */
end /* do */

if RCode == 0 then do
   say 'Configuration values retrieved ...'
   say ''
end /* do */

if RCode == 0 then do
   curdir = DIRECTORY()
   RC = DIRECTORY(destpath)
   CALL DIRECTORY curdir
   if RC == '' then 'MD 'destpath

   say 'Building PComm Files ....'
   RC = STREAM(destpath||'\SNA.WS', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'destpath||'\SNA.WS'
   RC = STREAM(destpath||'\SNA.BCH', 'C', 'QUERY EXISTS')
   if RC <> '' then 'DEL 'destpath||'\SNA.BCH'

   'ECHO [Profile] >> 'destpath||'\SNA.WS'
   'ECHO ID=WS >> 'destpath||'\SNA.WS'
   'ECHO Description= >> 'destpath||'\SNA.WS'
   'ECHO [Translation] >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultView=Y >> 'destpath||'\SNA.WS'
   'ECHO DefaultView= >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultDBCS=Y >> 'destpath||'\SNA.WS'
   'ECHO DefaultDBCS= >> 'destpath||'\SNA.WS'
   'ECHO [Communication] >> 'destpath||'\SNA.WS'
   'ECHO AutoConnect=Y >> 'destpath||'\SNA.WS'
   'ECHO Link=slan >> 'destpath||'\SNA.WS'
   'ECHO Session=3270 >> 'destpath||'\SNA.WS'
   'ECHO [SLAN] >> 'destpath||'\SNA.WS'
   'ECHO Adapter=0 >> 'destpath||'\SNA.WS'
   'ECHO DestinationSAP=04 >> 'destpath||'\SNA.WS'
   'ECHO GatewayAddress='||address' >> 'destpath||'\SNA.WS'
   'ECHO Identifier=061'||nodeid' >> 'destpath||'\SNA.WS'
   'ECHO LinkStations=1 >> 'destpath||'\SNA.WS'
   'ECHO ReceiveBufferCount=12 >> 'destpath||'\SNA.WS'
   'ECHO ReceiveBufferSize=2012 >> 'destpath||'\SNA.WS'
   'ECHO SourceSAP=04 >> 'destpath||'\SNA.WS'
   'ECHO TraceName=snalantrace0 >> 'destpath||'\SNA.WS'
   'ECHO TransmitFrameSize=2012 >> 'destpath||'\SNA.WS'
   'ECHO CPName= >> 'destpath||'\SNA.WS'
   'ECHO ExchangeID=N >> 'destpath||'\SNA.WS'
   'ECHO [LU] >> 'destpath||'\SNA.WS'
   'ECHO CompEnabled=N >> 'destpath||'\SNA.WS'
   'ECHO CompBufSize=4096 >> 'destpath||'\SNA.WS'
   'ECHO SLE=N >> 'destpath||'\SNA.WS'
   'ECHO [3270] >> 'destpath||'\SNA.WS'
   'ECHO ScreenSize=24x80 >> 'destpath||'\SNA.WS'
   'ECHO SessionType=Display >> 'destpath||'\SNA.WS'
   'ECHO HostGraphics=Y >> 'destpath||'\SNA.WS'
   'ECHO NativeGraphics=N >> 'destpath||'\SNA.WS'
```

Figure 225. CM2PCOMM (Part 5 of 7)

```
   'ECHO LoadPSS=N >> 'destpath||'\SNA.WS'
   'ECHO QueryReplyMode=Auto >> 'destpath||'\SNA.WS'
   'ECHO CharacterCellSize=9x16 >> 'destpath||'\SNA.WS'
   'ECHO HostCodePage=037-U >> 'destpath||'\SNA.WS'
   'ECHO LtNumber=FF >> 'destpath||'\SNA.WS'
   'ECHO LuNumber=FF >> 'destpath||'\SNA.WS'
   'ECHO [Transfer] >> 'destpath||'\SNA.WS'
   'ECHO HostSystem=1 >> 'destpath||'\SNA.WS'
   'ECHO DefaultVMDisk=a >> 'destpath||'\SNA.WS'
   'ECHO DefaultDataSet= >> 'destpath||'\SNA.WS'
   'ECHO DefaultLibrary=qgpl >> 'destpath||'\SNA.WS'
   'ECHO DefaultDirectory=C:\PCOMOS2 >> 'destpath||'\SNA.WS'
   'ECHO ResumeSendFileName= >> 'destpath||'\SNA.WS'
   'ECHO ResumeRecvFileName= >> 'destpath||'\SNA.WS'
   'ECHO [Keyboard] >> 'destpath||'\SNA.WS'
   'ECHO TypeAHead=Y >> 'destpath||'\SNA.WS'
   'ECHO CuaKeyboard=3 >> 'destpath||'\SNA.WS'
   'ECHO ResetInsertbyAttn=N >> 'destpath||'\SNA.WS'
   'ECHO SpotConversion=Y >> 'destpath||'\SNA.WS'
   'ECHO CaretSizeControl=Y >> 'destpath||'\SNA.WS'
   'ECHO CheckCodepage=Y >> 'destpath||'\SNA.WS'
   'ECHO Language=United-States >> 'destpath||'\SNA.WS'
   'ECHO IBMDefaultKeyboard=N >> 'destpath||'\SNA.WS'
   'ECHO DefaultKeyboard=C:\pcomos2\PRIVATE\CM101_3.KMP >> 'destpath||'\SNA.WS'

   'ECHO [Profile] >> 'destpath||'\SNA.BCH'
   'ECHO id=BCH >> 'destpath||'\SNA.BCH'
   'ECHO Description= >> 'destpath||'\SNA.BCH'
   'ECHO [Batch] >> 'destpath||'\SNA.BCH'
   'ECHO Run1=;If you want to modify an existing batch file, >> 'destpath||'\SNA.BCH'
   'ECHO Run2=;please choose [File] - [Open]. >> 'destpath||'\SNA.BCH'
   'ECHO Run3=;  >> 'destpath||'\SNA.BCH'
   'ECHO Run4=;If you want WorkStation Profiles or other >> 'destpath||'\SNA.BCH'
   'ECHO Run5=;programs to be included in a batch file, >> 'destpath||'\SNA.BCH'
   'ECHO Run6=;you must enter them in the edit area below. >> 'destpath||'\SNA.BCH'
   'ECHO Run7=;To do this, double-click the filename in >> 'destpath||'\SNA.BCH'
   'ECHO Run8=;the listbox or select it and choose Add. >> 'destpath||'\SNA.BCH'
   'ECHO Run9=;  >> 'destpath||'\SNA.BCH'
   'ECHO Run10=;If you want to suppress the IBM logo, add >> 'destpath||'\SNA.BCH'
   'ECHO Run11=;/Q to the command for the first session. >> 'destpath||'\SNA.BCH'
   'ECHO Run12=; >> 'destpath||'\SNA.BCH'
   'ECHO Run13=;Do NOT delete the remarks above - they will >> 'destpath||'\SNA.BCH'
   'ECHO Run14=;be ignored when the batch file is run. >> 'destpath||'\SNA.BCH'
   'ECHO Run15=;  >> 'destpath||'\SNA.BCH'
   linenumber = 16
   do i=1 to numberOfSessions
    echoline = 'Run'||linenumber||'=c:\pcomos2\pcsws.exe
"c:\pcomos2\private\sna.ws"'
    'ECHO 'echoline' >> 'destpath||'\SNA.BCH'
    linenumber = linenumber + 1
   end /* do */
   say 'PComm Files built ....'
   say ''
   Call Stream destpath||'\SNA.WS','c','close'
   Call Stream destpath||'\SNA.BCH','c','close'
end /* do */

 Call Stream rsp_file,'c','close'
```

Figure 226. CM2PCOMM (Part 6 of 7)

```
 'DEL' rsp_file
 'DEL output.txt'
 if RCode == 0 then say 'CM2PCOMM ended successful.'
 else say 'CM2PCOMM ended unsuccessful with ReturnCode 'RCode
 exit RCode



argerror:
say ''
say '***************************************************************************'
say '*                                                                         *'
say '* The Arguments are not valid !!                                          *'
say '*                                                                         *'
say '* Syntax:                                                                 *'
say '* cm2pcomm <CM/2 install path> <NVDM/2 workstationname> ...              *'
say '* <Name of the active Configurationfile>                                  *'
say '*                                                                         *'
say '* NVDM/2 workstationname is the name you defined for this workstation   *'
say '* at the NVDM/2 server.                                                   *'
say '* The Name of the Configurationfile has to be a valid 8.3 Filename      *'
say '* and all directories specified may have up to 8 characters only        *'
say '*                                                                         *'
say '* Example:                                                                *'
say '* cm2pcomm c:\cmlib myconfig                                              *'
say '*                                                                         *'
say '***************************************************************************'
say ''

RCode = 4
return RCode
```

Figure 227. CM2PCOMM (Part 7 of 7)

C.3 MAKECONF.CMD

```
/*************************************************************************/
/*                                                                       */
/* REXX Procedure to create a PCOMM Configuration with usergiven values */
/*                                                                       */
/*************************************************************************/

parse upper arg inputfile .

/******************************************************************************/
/* Variables Dokumentation                                                    */
/*                                                                            */
/* RCode             Returncode CM2PCOMM gives back to the invoking program  */
/* RC                Returncode CM2PCOMM receives from programs it invokes   */
/* workstationname   NVDM/2 name of the workstation                          */
/* nodeid            holds the value of the NODE_ID                          */
/* address           holds the value of the DESTINATION_ADDRESS              */
/* numberOfSessions  stores the number of the difines 3270 Sessions in the   */
/*                   CM/2 Response File                                       */
/* destpath          drive and path on the Code Server where the PComm Files */
/*                   are copied without the <workstationname> subdirectory   */
/* targetpath        drive and path on the Code Server where the PComm Files */
/*                   are copied with the <workstationname> subdirectory      */
/* i                 loop counter                                             */
/* inputfile         name of the file which contains the configurationdata   */
/*                   for many maschines                                       */
/* curdir            name of the directory in which this program is invoked  */
/* Error             memory variable for unsuccessful execution for single   */
/*                   workstations.                                            */
/* answer            the value of this variable is read from the keyboard,   */
/*                   the value is not important the purpose is to wait for an*/
/*                   user response.                                           */
/* success           counter for number of configurations successful built   */
/* unsuccess         counter for number of configurations unsuccessful built */
/* echoline          contains the line to write to the a file. It is used to */
/*                   avoid lines which are too long for viewing              */
/* pcomdir           PComm install directory                                 */
/******************************************************************************/


ECHO OFF
RCode = 0
Error = 0
success = 0
unsuccess = 0
destpath = 'W:\IMG\CONFIG'   /* without "\" at the end !! */
pcomdir = 'C:\PCOMOS2'       /* without "\" at the end !! */
logdir = 'X:\LOG'            /* without "\" at the end !! */

RC = STREAM(logdir||'\MAKECONF.LOG', 'C', 'QUERY EXISTS')
if RC <> '' then 'DEL 'logdir||'\MAKECONF.LOG'

if inputfile == '' then do
   'CLS'
   say 'You are about to create PComm configurationfiles for use in the'
   say 'migration process.'
```

Figure 228. MAKECONF.CMD (Part 1 of 6)

```
   say ’The following values are needed:’
   say ’’
   say ’Destination Address, NodeID, Number of 3270 Terminal Sessions to’
   say ’configure and the NVDM/2 workstationname you have given to this’
   say ’maschine in NVDM/2. The configurationfiles will be stored in’
   say destpath||’\<workstationname> on your NVDM/2 Server.’
   say ’’
   say ’Enter Destination Address :’
   say ’123456789012 (12 Digits)’
   pull address
   say ’Enter NodeID              :’
   say ’12345 (5 Digits)’
   pull nodeid
   say ’Enter Number of Sessions  :’
   say ’1 (1 Digit)’
   pull numberOfSessions
   say ’Enter Workstationname     :’
   say ’12345678 (up to 8 Digits)’
   parse upper pull workstationname
   CALL checkvalues

   if RCode == 0 then do
      call buildprofile
   end /* do */
   if RCode == 0 then do
      ’ECHO    configuration for ’workstationname’ built >> ’logdir||’\MAKECONF.LOG’
      echoline = ’*********************************************************’
      ’ECHO ’echoline’ >> ’logdir||’\MAKECONF.LOG’
      success = 1
   end /* do */
   else do
      echoline = ’   configuration for ’workstationname’ not built’
      ’ECHO ’echoline’ >> ’logdir||’\MAKECONF.LOG’
      echoline = ’*********************************************************’
      ’ECHO ’echoline’ >> ’logdir||’\MAKECONF.LOG’
      unsuccess = 1
      Error = 1
   end /* do */
end /* do */

else do
   RC = STREAM(inputfile, ’C’, ’QUERY EXISTS’)
   if RC == ’’ then do
      ’ECHO The specified inputfile does not exist... >> ’logdir||’\MAKECONF.LOG’
      RCode = 8
      ’ECHO RCode = ’RCode’ >> ’logdir||’\MAKECONF.LOG’
   end /* do */
   if RCode == 0 then do
      do while lines(inputfile)
         line.linenum = Linein(inputfile)
         if (substr(line.linenum, 1, 1)==’;’)|(strip(line.linenum)==’’) then iterate
         else do
            workstationname = strip(word(line.linenum, 1))
            address = strip(word(line.linenum, 2))
            nodeid = strip(word(line.linenum, 3))
            numberOfSessions = strip(word(line.linenum, 4))
            CALL checkvalues
            if RCode == 0 then do
```

Figure 229. MAKECONF.CMD (Part 2 of 6)

```
            call buildprofile
            end /* do */
            if RCode == 0 then do
               echoline '   configuration for 'workstationname' built'
               'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
               echoline = '*******************************************************'
               'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
            end /* do */
            else do
               echoline = '   configuration for 'workstationname' not built'
               'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
               echoline = '*******************************************************'
               'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
            end /* do */
         end /* do */
         if RCode <> 0 then do
            Error = Error + 1
            unsuccess = unsuccess + 1
         end /* do */
         else success = success + 1
         RCode = 0
      end /* do */
   end /* do */
end /* do */

CALL LINEOUT inputfile
if success == 0 then RCode = 4
if RCode == 0 then do
   'ECHO MAKECONF ended successful. >> 'logdir||'\MAKECONF.LOG'
   'ECHO 'success' configuration(s) successful built >> 'logdir||'\MAKECONF.LOG.'
   'ECHO 'unsuccess' configuration(s) failed. >> 'logdir||'\MAKECONF.LOG'
   say ''
   say success' configuration(s) successful built.'
   say unsuccess' configuration(s) failed.'
   if Error <> 0 then do
      RCode = 2
      say ''
      say '**********************************************************'
      say '* MAKECONF ended successful.                             *'
      say '* But errormessages were logged for several workstations!  *'
      say '* Check the Logfile...                                   *'
      say '* MAKECONF ended with Rcode 'RCode'                          *'
      say '**********************************************************'
      pull answer
   end /* do */
end /* do */
else do
   echoline = 'MAKECONF ended unsuccessful with ReturnCode 'RCode
   'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
   say 'MAKECONF ended unsuccessful with ReturnCode 'RCode'!!'
end /* do */
ECHO ON
exit RCode


buildprofile:
if RCode == 0 then do
   curdir = DIRECTORY()
```

Figure 230. MAKECONF.CMD (Part 3 of 6)

```
    RC = DIRECTORY(destpath)
    CALL DIRECTORY curdir
    if RC == '' then do
       RCode = 8
       'ECHO    The directory 'destpath' does not exist! >> 'logdir||'\MAKECONF.LOG'
       'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
    end /* do */
    if RCode == 0 then do
     curdir = DIRECTORY()
     targetpath = destpath||'\'||workstationname
     RC = DIRECTORY(targetpath)
     CALL DIRECTORY curdir
     if RC == '' then 'MD 'targetpath
     RC = STREAM(targetpath||'\SNA.WS', 'C', 'QUERY EXISTS')
     if RC <> '' then 'DEL 'targetpath||'\SNA.WS'
     RC = STREAM(targetpath||'\SNA.BCH', 'C', 'QUERY EXISTS')
     if RC <> '' then 'DEL 'targetpath'\SNA.BCH'

     'ECHO [Profile] >> 'targetpath||'\SNA.WS'
     'ECHO ID=WS >> 'targetpath||'\SNA.WS'
     'ECHO Description= >> 'targetpath||'\SNA.WS'
     'ECHO [Translation] >> 'targetpath||'\SNA.WS'
     'ECHO IBMDefaultView=Y >> 'targetpath||'\SNA.WS'
     'ECHO DefaultView= >> 'targetpath||'\SNA.WS'
     'ECHO IBMDefaultDBCS=Y >> 'targetpath||'\SNA.WS'
     'ECHO DefaultDBCS= >> 'targetpath||'\SNA.WS'
     'ECHO [Communication] >> 'targetpath||'\SNA.WS'
     'ECHO AutoConnect=Y >> 'targetpath||'\SNA.WS'
     'ECHO Link=slan >> 'targetpath||'\SNA.WS'
     'ECHO Session=3270 >> 'targetpath||'\SNA.WS'
     'ECHO [SLAN] >> 'targetpath||'\SNA.WS'
     'ECHO Adapter=0 >> 'targetpath||'\SNA.WS'
     'ECHO DestinationSAP=04 >> 'targetpath||'\SNA.WS'
     'ECHO GatewayAddress='||address' >> 'targetpath||'\SNA.WS'
     'ECHO Identifier=061'||nodeid' >> 'targetpath||'\SNA.WS'
     'ECHO LinkStations=1 >> 'targetpath||'\SNA.WS'
     'ECHO ReceiveBufferCount=12 >> 'targetpath||'\SNA.WS'
     'ECHO ReceiveBufferSize=2012 >> 'targetpath||'\SNA.WS'
     'ECHO SourceSAP=04 >> 'targetpath||'\SNA.WS'
     'ECHO TraceName=snalantrace0 >> 'targetpath||'\SNA.WS'
     'ECHO TransmitFrameSize=2012 >> 'targetpath||'\SNA.WS'
     'ECHO CPName= >> 'targetpath||'\SNA.WS'
     'ECHO ExchangeID=N >> 'targetpath||'\SNA.WS'
     'ECHO [LU] >> 'targetpath||'\SNA.WS'
     'ECHO CompEnabled=N >> 'targetpath||'\SNA.WS'
     'ECHO CompBufSize=4096 >> 'targetpath||'\SNA.WS'
     'ECHO SLE=N >> 'targetpath||'\SNA.WS'
     'ECHO [3270] >> 'targetpath||'\SNA.WS'
     'ECHO ScreenSize=24x80 >> 'targetpath||'\SNA.WS'
     'ECHO SessionType=Display >> 'targetpath||'\SNA.WS'
     'ECHO HostGraphics=Y >> 'targetpath||'\SNA.WS'
     'ECHO NativeGraphics=N >> 'targetpath||'\SNA.WS'
     'ECHO LoadPSS=N >> 'targetpath||'\SNA.WS'
     'ECHO QueryReplyMode=Auto >> 'targetpath||'\SNA.WS'
     'ECHO CharacterCellSize=9x16 >> 'targetpath||'\SNA.WS'
     'ECHO HostCodePage=037-U >> 'targetpath||'\SNA.WS'
     'ECHO LtNumber=FF >> 'targetpath||'\SNA.WS'
     'ECHO LuNumber=FF >> 'targetpath||'\SNA.WS'
```

Figure 231. MAKECONF.CMD (Part 4 of 6)

```
    'ECHO [Transfer] >> 'targetpath||'\SNA.WS'
     'ECHO HostSystem=1 >> 'targetpath||'\SNA.WS'
     'ECHO DefaultVMDisk=a >> 'targetpath||'\SNA.WS'
     'ECHO DefaultDataSet= >> 'targetpath||'\SNA.WS'
     'ECHO DefaultLibrary=qgpl >> 'targetpath||'\SNA.WS'
     'ECHO DefaultDirectory='||pcomdir||' >> 'targetpath||'\SNA.WS'
     'ECHO ResumeSendFileName= >> 'targetpath||'\SNA.WS'
     'ECHO ResumeRecvFileName= >> 'targetpath||'\SNA.WS'
     'ECHO [Keyboard] >> 'targetpath||'\SNA.WS'
     'ECHO TypeAHead=Y >> 'targetpath||'\SNA.WS'
     'ECHO CuaKeyboard=3 >> 'targetpath||'\SNA.WS'
     'ECHO ResetInsertbyAttn=N >> 'targetpath||'\SNA.WS'
     'ECHO SpotConversion=Y >> 'targetpath||'\SNA.WS'
     'ECHO CaretSizeControl=Y >> 'targetpath||'\SNA.WS'
     'ECHO CheckCodepage=Y >> 'targetpath||'\SNA.WS'
     'ECHO Language=United-States >> 'targetpath||'\SNA.WS'
     'ECHO IBMDefaultKeyboard=N >> 'targetpath||'\SNA.WS'
     'ECHO DefaultKeyboard='||pcomdir||'\CM101_3.KMP >> 'targetpath||'\SNA.WS'


     'ECHO [Profile] >> 'targetpath||'\SNA.BCH'
     'ECHO id=BCH >> 'targetpath||'\SNA.BCH'
     'ECHO Description= >> 'targetpath||'\SNA.BCH'
     'ECHO [Batch] >> 'targetpath||'\SNA.BCH'
     echoline = 'Run1=;If you want to modify an existing batch file,'
     'ECHO echoline  >> 'targetpath||'\SNA.BCH'
     'ECHO Run2=;please choose [File] - [Open]. >> 'targetpath||'\SNA.BCH'
     'ECHO Run3=;  >> 'targetpath||'\SNA.BCH'
     'ECHO Run4=;If you want WorkStation Profiles or other >> 'targetpath||'\SNA.BCH'
     'ECHO Run5=;programs to be included in a batch file, >> 'targetpath||'\SNA.BCH'
    'ECHO Run6=;you must enter them in the edit area below. >> 'targetpath||'\SNA.BCH'
     'ECHO Run7=;To do this, double-click the filename in >> 'targetpath||'\SNA.BCH'
     'ECHO Run8=;the listbox or select it and choose Add. >> 'targetpath||'\SNA.BCH'
     'ECHO Run9=;  >> 'targetpath||'\SNA.BCH'
     'ECHO Run10=;If you want to suppress the IBM logo, add >> 'targetpath||'\SNA.BCH'
     'ECHO Run11=;/Q to the command for the first session. >> 'targetpath||'\SNA.BCH'
     'ECHO Run12=; >> 'targetpath||'\SNA.BCH'
   'ECHO Run13=;Do NOT delete the remarks above - they will >> 'targetpath||'\SNA.BCH'
     'ECHO Run14=;be ignored when the batch file is run. >> 'targetpath||'\SNA.BCH'
     'ECHO Run15=;  >> 'targetpath||'\SNA.BCH'
     linenumber = 16
     do i=1 to numberOfSessions
      echoline1 = 'Run'||linenumber||'='||pcomdir||'\pcsws.exe '
      echoline2 = '"'||pcomdir||'\private\sna.ws"'
      'ECHO 'echoline1||echoline2' >> 'targetpath||'\SNA.BCH'
      linenumber = linenumber + 1
     end /* do */
    end /* do */
end /* do */
CALL LINEOUT targetpath||'\SNA.WS'
CALL LINEOUT targetpath||'\SNA.BCH'
return RCode

checkvalues:
'ECHO workstation 'workstationname' >> 'logdir||'\MAKECONF.LOG'
do i = 1 to (length(nodeid))
   rc = verify(substr(nodeid,i,1),'0123456789ABCDEF')
   if RC <> 0 then do
```

Figure 232. MAKECONF.CMD (Part 5 of 6)

```
      RCode = 12
      'ECHO    The nodeid is not compatible with PComm! >> 'logdir||'\MAKECONF.LOG'
      'ECHO    Digit number 'i' is not a hex-value! >> 'logdir||'\MAKECONF.LOG'
      'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
   end /* do */
end /* do */
if RCode == 0 then do
   do i = 1 to (length(address))
      rc = verify(substr(address,i,1),'0123456789ABCDEF')
      if RC <> 0 then do
         RCode = 12
         'ECHO    The address is not compatible with PComm! >>
'logdir||'\MAKECONF.LOG'
         'ECHO    Digit number 'i' is not a hex-value! >> 'logdir||'\MAKECONF.LOG'
         'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
      end /* do */
   end /* do */
end /* do */
address = strip(address)
if length(address) <> 12 then do
   RCode = 8
   'ECHO    The address has to be a 12 digit value! >> 'logdir||'\MAKECONF.LOG'
   'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
end /* do */
numberOfSessions = strip(numberOfSessions)
rc = verify(numberOfSessions, '123456789')
if RC <> 0 then do
   RCode = 8
   echoline = '   numberOfSessions has to be a number between 1 and 9!'
   'ECHO 'echoline' >> 'logdir||'\MAKECONF.LOG'
   'ECHO    RCode ='RCode' >> 'logdir||'\MAKECONF.LOG'
end /* do */
return RCode
```

Figure 233. MAKECONF.CMD (Part 6 of 6)

C.4 COPYCONF.CMD

```
/***********************************************************/
/* Cmd File to copy the generated PComm configuration files */
/* to the correct directory on the target system            */
/***********************************************************/

parse upper arg workstationname

/****************************************************************************/
/* Variables Dokumentation                                                  */
/*                                                                          */
/* RCode            Returncode CM2PCOMM gives back to the invoking program  */
/* RC               Returncode CM2PCOMM receives from programs it invokes   */
/* workstationname  NVDM/2 name of the workstation                          */
/* sourcepath       drive and path on the Code Server where the PComm Files */
/*                  are copied without the <workstationname> subdirectory   */
/* i                loop counter                                            */
/* curdir           name of the directory in which this program is invoked  */
/* echoline         contains the line to write to the a file. It is used to */
/*                  avoid lines which are too long for viewing              */
/* pcompath         PComm install directory                                 */
/* logpath          Path the logfiles are located                           */
/****************************************************************************/


ECHO OFF
RCode = 0

pcommpath = 'C:\PCOMOS2\PRIVATE'
sourcepath = 'W:\IMG\CONFIG\'||workstationname
logpath = 'X:\LOG\'||workstationname


curdir = DIRECTORY()
RC = DIRECTORY(sourcepath)
CALL DIRECTORY curdir
if RC == '' then do
   RCode = 8
   'ECHO    The directory 'sourcepath' does not exist! >> logpath||.\COPYCONF.LOG'
end /* do */

if RCode == 0 then do
   RC = STREAM(sourcepath||'\SNA.WS', 'C', 'QUERY EXISTS')
   if RC == '' then do
      RCode = 8
      'ECHO The sourcefile SNA.WS is missing ! >> 'logpath||'\COPYCONF.LOG'
   end /* do */
   RC = STREAM(sourcepath||'\SNA.BCH', 'C', 'QUERY EXISTS')
   if RC == '' then do
      RCode = 8
      'ECHO The sourcefile SNA.WS is missing ! >> 'logpath||'\COPYCONF.LOG'
   end /* do */
   if RCode == 0 then do
      'COPY 'sourcepath||'\SNA.WS 'pcommpath
      if RC == 0 then do
         'COPY 'sourcepath||'\SNA.BCH 'pcommpath
```

Figure 234. COPYCMF.CMD (Part 1 of 2)

```
         if RC <> 0 then do
            RCode = 4
            'ECHO Copy of  sourcefile SNA.WS failed ! >> 'logpath||'\COPYCONF.LOG'
         end /* do */
      end /* do */
      else do
         RCode = 4
         'ECHO Copy of sourcefile SNA.BCH failed ! >> 'logpath||'\COPYCONF.LOG'
      end /* do */
   end /* do */
end /* do */

if RCode == 0 then do
   echoline = 'COPYCONF.CMD ended successful for workstation 'workstationname
   'ECHO 'echoline' >> 'logpath||'\COPYCONF.LOG'
   'ECHO RCode = 'RCode' >> 'logpath||'\COPYCONF.LOG'
end /* do */
else do
   'ECHO COPYCONF.CMD ended unsuccessful >> 'logpath||'\COPYCONF.LOG'
   'ECHO RCode = 'RCode' >> 'logpath||'\COPYCONF.LOG'
end /* do */
ECHO ON
exit RCode
```

Figure 235. COPYCMF.CMD (Part 2 of 2)

C.5 DETREXX.CMD

```
@echo off

SET PATH=%PATH%;%2;
set helper=%1\SRVREXX.EXE

if not exist %helper% goto error
detach %helper%
call %2\casckrex
goto end

:error
@echo  .
@echo  %helper% not found. NDM-Server updated for REXX-Support  ?
@cmd
@goto end

:end
```

Figure 236. DETREXX.CMD

C.6 DISKPREP.CMD

```
/*******************************************************************
 ****  File          : DISKPREP.CMD                            ****
 ****  Function    : Automated Partitioning of Harddisks         ****
 ****                                                           ****
 ****  Prerequisites:                                           ****
 ****  The file PIPE.CMD has to be in the SourcePath. The Input- ****
 ****  file for FDISK  m u s t  create a Partition named OS/2    ****
 ****  because otherwise the routine will loop.                 ****
 ****                                                           ****
 ****                                                           ****
 ****                                                           ****
 ****  RC 0x0000  = Success:                                    ****
 ****  RC 0xFE00  = Success: Reboot Required                    ****
 ****  RC 0x0800  = Error : Data ressource not found            ****
 ****  RC 0x1200  = Error : IO-Error                            ****
 ****  RC 0x0808  = Error : Access denied, missing authorization ****
 ****  RC 0x0812  = Error : datapath not found                  ****
 ****  RC 0x1600  = Error : Incorrect Program invocation         ****
 ****  RC 0x1604  = Error : Uncexpected condition               ****
 ****                                                           ****
 *******************************************************************/

call CidInit NORXFUNC

Par.FilePath  = ''
Par.ExePath   = ''
Par.SPath     = ''
Ctrl.LogFile  = ''
Ctrl.ErrDesc  = ''
Par.FDiskMode = ''

arg Param


w='FIRST'
i = 1
do until w==''
  w=translate(word(Param, i,))
  if left(w,3)=="/?" then
     do
         call Help
         exit Error.Incorrect_Invocation
     end
  if left(w,3)=="/R:"then do
        w=right(w,length(w)-3)
        Par.FilePath  = w
  end  /* Do */
  if left(w,3)=="/E:"then do
        w=right(w,length(w)-3)
        Par.ExePath  = w
  end
  if left(w,3)=="/S:"then do
        w=right(w,length(w)-3)
        Par.SPath  = w
  end
```

Figure 237. DISKPREP.CMD (Part 1 of 8)

```
 if left(w,3)=="/L:"then do
       w=right(w,length(w)-3)
       Ctrl.LogFile = w
  end
    if left(w,3)=="/F:"then do
       w=right(w,length(w)-3)
       Par.FDIskMode = w
  end
  i = i+1
end

ParmsOk = 0
if Ctrl.LogFile=="" then do
  say "Parameter-Error: LogFile missing!"
  ParmsOk = 1
end
else
do
/* You can delete the first part of the log if you want to save space:

    if stream(Ctrl.LogFile, 'c', 'query exists')\='' then
        '@del 'Ctrl.LogFile' >Nul' */
  Log='ok'
end

if Par.FilePath=="" then do
  say "Parameter-Error: Responsefile-Path missing!"
  if Log='ok' then
    call Log "Parameter-Error: Responsefile-Path missing!"
  ParmsOk = 1
end
if Par.ExePath=="" then do
  say "Parameter-Error: EXE-Path missing "
  if Log='ok' then
     call Log "Parameter-Error: EXE-Path missing "
  ParmsOk = 1
end
if Par.SPath=="" then do
  say "Parameter-Error: OS2-Source Directory missing !"
  if Log='ok' then
     call Log "Parameter-Error: OS2-Source Directory missing !"
  ParmsOk = 1
end
if Par.FDiskMode=="" then
   Par.FDiskMode = 'Y'

if ParmsOk == 1 then
  do
     call Help
     exit Error.Incorrect_Invocation
  end

call Log 'DISKPREP'
call Log '--------'
call Log 'Responsefile Path  : 'Par.FilePath
call Log 'EXE-Directory  :     'Par.ExePath
call Log 'Source-Directory     'Par.SPath
call Log 'LogFile:             'Ctrl.LogFile
```

Figure 238. DISKPREP.CMD (Part 2 of 8)

```
call Log 'FDiskMode:           'Par.FDiskMode
call Log ' '

Par.PipeIt='| '||Par.ExePath||'\pipe.cmd'



/* Start of Partitioning Process */

if NAMECHECK()==1 then do
   if Par.FDiskMode == 'Y' then do
        rc2 = FIRSTFORMAT()
        if rc2 <> 0 then
        do
           call log "Error while formatting volumes.."
           exit Error.IO
        end
        else
           exit Error.Success
    end
    else
         exit Error.Success
end
else
do
   call DELETEPART
   rc = Cmd(Par.SPath'\disk_2\fdisk /newmbr') /* New MasterBoot */
                                              /* Record         */
   call Log 'Query result of Partitioning ...'
   Par.SPath'\disk_2\fdisk /query >>'Ctrl.LogFile
   exit Error.SuccessReboot
end

/****************************************************************/
NAMECHECK:
/****************************************************************/

/* If a Partition named OS/2 exists this routine returns 1     */
/* otherwise 0                                                 */

   call Log 'Checking if Partitioning is already done...'
   Par.SPath'\disk_2\fdisk /query 'Par.PipeIt

   OS2 = 0
   do while queued() > 0
     call GetFDiskLine
     if Name=='OS/2' then do
          OS2 = 1
     end
   end  /* Do */
   if OS2==1 then do
        call Log 'OS/2-Partition already exists !'
        return 1
   end
   else
        return 0
```

Figure 239. DISKPREP.CMD (Part 3 of 8)

```
Help: Procedure Expose Ctrl. Error.

  say 'DISKPREP'
  say "--------"
  say "Program-Invocation: "
  say "================== "
  say ''
  say "DISKPREP /R:FilePath /E:ExePath /S:SourcePath"
  say "           /L:LogFile /F:Formatmode"
  say "With: "
  say "         FilePath:    Response-File Directory."
  say "         ExePath:     Directory for SETBOOT.EXE and PIPE.CMD."
  say "         SourcePath:  Source-Directory (OS/2 Image)."
  say "         LogFile:     Logfile."
  say "         Formatmode:  Y(es) format Volumes, N(o) do not format."
  return


GetFDiskLine: Procedure Expose name part drive vtype fstype status ,
  start size

  name  = ''
  drive = ''
  vtype = ''
  fstype= ''
  status= ''
  start = 0
  size  = 0
  if queued()>0 then do
     pull Line
     drive = strip(substr(Line, 1, 5))
     if datatype(drive, 'N') then do
          name  = strip(substr(Line, 7, 10))
          Line =  strip(substr(Line, 19))
          part  = word(Line, 1)
          vtype = word(Line, 2)
          fstype= word(Line, 3)
          status= word(Line, 4)
          start = word(Line, 5)
          size  = word(Line, 6)
     end
  end
  return

/*********************************************************************/
CheckDrives: procedure expose maxdrives Error. Ctrl. Par. Vol.
/*********************************************************************/

/* Procedure to get the number of installed Harddrives     */
   call Log 'Checking for number of drives ...'
   volumes = XRANGE('C','Z')
   maxdrives=1
   maxvolumes = 0
   Par.SPath'\disk_2\fdisk /query 'Par.PipeIt
   do while queued() > 0
        call GetFDiskLine
        if datatype(drive, 'Whole number') then do
           if drive > 0 & drive < 9 then
```

Figure 240. DISKPREP.CMD (Part 4 of 8)

```
             maxdrives = drive
             if pos(left(Part,1),volumes) <> 0 then
             do
                   maxvolumes = maxvolumes + 1
                   Vol.0 = maxvolumes
                   Vol.maxvolumes = Part
             end /* do */
          end  /* if */
     end /* do */
     call Log ’There are ’maxdrives’ Drives in the system’
     i = 1
     do maxvolumes
          call log ’Found volume < ’ Vol.i ’ >’
          i = i + 1
     end /* do */
     return


  /*********************************************************************/
  DELETEPART: procedure expose Error. Par. Ctrl.
  /*********************************************************************/

  /* Existing partitions will be deleted an a new Partition-tabel is
     created with use of response-files:
     We handled 2 cases:
     a) only one physical Harddrive is installed, so the FDSKHD1.RSP
          is used to create a physical Drive C: , a logigal Drive D:
          for Service and ad logical Drive E: for Data

     b) there are two physical Harddrives installed, so the FDSKHD2.RSP
          is used to create the same structure as mentioned in a) on the
          first drive two additional logical drives F: and G: on the
          second drive.

     If you plan to install a DB2/2-System you should take care of
     correct time-settings because otherwise you may run into problems
     if the current system-time is set to a future time.
  */

     call CheckDrives
     count = 1

     fstype.0 = 4
     fstype.1 = ’FAT’
     fstype.2 = ’HPFS’
     fstype.3 = ’DOS’
     fstype.4 = ’H0A’ /* Boot-Manager */

     /* Now the existing partitions are selectively deleted to avoid the
          deletion of the Reference-partition. */
     do maxdrives
        i=1
        do fstype.0   /* for all defined Filesystem-Types */
           cmdline = Par.Spath’\disk_2\fdisk /delete:all /disk:’count,
                          ’ /fstype:’fstype.i
           i=i+1
           rc2 = Cmd(cmdline)
           /* Possible Return-Codes of FDISK for successful execution:
```

Figure 241. DISKPREP.CMD (Part 5 of 8)

```
             1400 = 5120
             1C00 = 7168
             1E00 = 7680
             1800 = 6144
        */

        if rc2<>7168 & rc2<>7680 & rc2<>6144 & rc2<>5120 & rc2<>0
        then do
           call Log 'Error deleting Disk 'count' FSTYPE: 'fstype.i,
                    '. Rc: 'rc2
           exit Error.IO
        end  /* if */
     end /* do */
     count = count + 1
  end    /* do */

  /* Start of Partitioning: */
  /* 1. Boot-Manager */
  call Log 'Creating Bootmanager Partition...'
  rc2 = Cmd(Par.SPath'\disk_2\fdisk /create /bootmgr /start:t /disk:1')
  if rc2 <> 7168 & rc2 <> 6144 then do
     call Log 'Error Creating Bootmanager-Partition. Rc: 'rc2
     exit Error.IO
  end /* if */

  /* 2. Creating Partitions, dependend on number of Harddrives */

  call Log 'Starting Partitioning for 'maxdrives ' Harddisk(s) '
  rspfile = Par.FilePath || '\FDSKHD'maxdrives'.RSP'
  call Log 'ResponseFile: 'rspfile
  rc2 = Cmd(Par.SPath'\disk_2\fdisk /file:'||rspfile)
  if rc2 <> 0 then do
        call Log 'Error during Partitioning. Rc: 'rc2
        exit Error.IO
  end /* if */

  /*****************************************************************/
  /*        Subroutine to write banner to screen                   */
  /*****************************************************************/
  say '[0;7m';
  '@cls';                                       /* Clear the screen */
  Par.SPath'\disk_2\fdisk /query >>'Ctrl.LogFile
  say '[8;1H';                            /* Move cursor to line 8 */
  do while stream('a:\OS2BOOT', 'c', 'query exists')==''
       call Log 'Waiting for Reentering Boot-Disk !'
       'CLS'
       call PrettyBox 'Disk Partitioning successfully done !',
                 'Reenter Boot-Diskette into Drive A: ',,
                 'Press ENTER to continue.'
         pull Dummy
    end
  say "Now the System will be rebootet !!!! "
  Par.ExePath'\setboot /b'
  do forever
       nop   /* waiting forever*/
  end /* do */
  return  /* DELETEPART */
```

Figure 242. DISKPREP.CMD (Part 6 of 8)

```
/*******************************************************************/
FIRSTFORMAT: procedure expose Error. Ctrl. Par. maxdrives Vol.
/*******************************************************************/
   rc2 = 0

   call CheckDrives
   call directory Par.SPath     /* Change to Sourcepath ...        */
   call directory 'disk_2'      /* ... and to SubDirectory Disk_2  */

   /* Now formatting of every drive starts ... */
   /* The advantage of doing the FORMAT here ist that the Parameter
      /L can be set to do a fully format. The Format-Options in
      the OS/2-Response-file lead to a quick-format of the
      partitions. */

   volno = 1
   do Vol.0    /* Vol.0 = Number of Volumes */
        call log 'Now foramtting Drive: 'Vol.volno
        cmdline = 'echo Y | format ' || Vol.volno || ' /FS:HPFS /P /L '
        /* Parameter /P: not documented, suppresses Question for
           Volume-label. */
        rc2 = Cmd(cmdline)
        if rc2 <> 0 then
        do
           call Log "Error formatting "Vol.volno ". Rc: "rc2
           return Error.IO
        end
        else
          volno = volno + 1
   end /* do */

   call Log 'Formatting of Harddrives successfully completed.'
   return rc2           /* FIRSTFORMAT */

/*****************************************************************/

PrettyBox: procedure
   /* Put message in a pretty box */
   say         ' É' || left('Í',76,'Í')   || '»'
/*   say     ' º' || left(' ',76)         || 'º' */
   do i = 1 to arg()
        say    ' º' || center(arg(i) ,76) || 'º';
   end /* do */
   say         ' È' || left('Í',76,'Í')   || '¿'
   return
/*****************************************************************/

/* Initialising of Error.- and Ctrl.-Structures                 */

CidInit: procedure expose Ctrl. Error.
  if (Arg(1)\='NORXFUNC') then do
    call RxFuncAdd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
    call SysLoadFuncs
  end
  Error.Success                 = x2d('0000')
  Error.SuccessReboot           = x2d('FE00')
  Error.Data_Resource_Not_Found = x2d('0800')
  Error.Incorrect_Invocation    = x2d('1600')
```

Figure 243. DISKPREP.CMD (Part 7 of 8)

```
 Error.IO                      = x2d('1200')
  Error.NoDiskSpace             = x2d('1208')
  Error.Access                  = x2d('0808')
  Error.UnexpectedCondition     = x2d('1604')

  Ctrl.LogFile = ''
  success = 0
  return

/* Logging-Procedure: Adds a line to the logfile */

Log: procedure expose Ctrl. Error.

   /* Creating Log-Line-Header */
   success = 0
   header = '['date('E') time() ' DISKPREP]'
   text = arg(1)
   if text = '' then
        header = left('', 77, '-')
   rc2 = stream(Ctrl.LogFile, 'c', 'open write')
   if rc2 <> 'READY:' then
        text = 'open' Ctrl.LogFile '=' rc2'.' text
   else do
        rc2 = stream(Ctrl.LogFile, 'c', 'seek < 0')  /* Set         */
                                                     /* File-Cursor */
                                                     /* to EOF */
        if rc2 = ''
        then rc2 = 0
        if \datatype(rc2, 'Whole number') then
           text = 'seek' Ctrl.LogFile '=' rc2'.' text
        else do
           rc2 = lineout(Ctrl.LogFile, header text)
           if rc2 <> 0 then
                text = 'lineout' Ctrl.LogFile '=' rc2'.' text
           else do
                success = 1
           end  /* Do */
        end  /* Do */
        call stream Ctrl.LogFile, 'c', 'close'
   end  /* Do */
   return

Cmd: Procedure Expose Ctrl.
  /* Arg(1) : command                                          */
  /* Arg(2) : NOCMDLOG: No errorlog while executing command. */
  rc2=0
  if Arg(2)=="NOCMDLOG" then
     (Arg(1))
  else
     (Arg(1))' 2>>'Ctrl.Logfile /* Stdout and StdErr to Logfile */
  rc2=rc
  call Log Arg(1)'. Rc: 'rc2
  return rc2
KillQueue: Procedure
  do while queued()>0
    pull a
  end
return
```

Figure 244. DISKPREP.CMD (Part 8 of 8)

C.7 WSCONFIG.CMD

```
/*****************************************************************/
/* WSCONFIG.CMD:                                                 */
/* This procedure read configuration details from an ASCII       */
/* file, with fixed length field, then pass these values to      */
/* CONFIGUR.EXE to process. The following item are configured by */
/* this program:                                                 */
/*                                                               */
/*      1.- LAN Requester Computername                           */
/*      2.- LAN Requester default Domain                         */
/*      3.- LAN Requester Computername Description               */
/*      4.- IP Hostname (same as LAN Requester Computername)     */
/*      5.- IP Address                                           */
/*      6.- IP Mask                                              */
/*      7.- IP Default Router                                    */
/*      8.- IP Net Address for Router                            */
/*      9.- IP Net Mast                                          */
/*     10.- IP Domain name                                       */
/*     11.- IP Domain Name Server address                        */
/*     12.- IP Hostname Socks Name Server                        */
/*     13.- IP Socks Name Server address                         */
/*     14.- NIC MAC Address                                      */
/*     15.- PCOM Destination MAC Address (Mainframe MAC address) */
/*     16.- PCOM Physical Unit ID                                */
/*                                                               */
/*****************************************************************/

/* Initialize local variables... */

WSDataBase = "W:\WSDetails\LocalWS.DAT"  /* File with Workstation details */
CurrentLine = " "                        /* Holds the line being read from */
                                         /* WSDataBase */
FoundRecord = 0                          /* Boolean variable regarding */
                                         /* record found in WSDataBase */
ConfigurCtrlFile = ""                    /* Location for the REXX script */
                                         /* that sets FI Variable values */
                                         /* and runs CONFIGUR.EXE */
ConfigurFileRec = ""                     /* Holds record to be written */
                                         /* into ConfigurCtrlFile */
ComputerName = ""                        /* LAN Requester Computername, */
                                         /* NvDM/2 Workstation Name and */
                                         /* Hostname Name */
NVDMServerName = ""                      /* NVDM/2 Server Name */
ComputerClass = ""                       /* Computer Class */
ClassPath = ""                           /* Path for the Computer Class */
                                         /* Bundle files */
ShareB = ""                              /* Netview DM/2 ShareB drive */
LRDomain = " "                           /* LAN Requester default Domain */
LRDescription = " "                      /* LAN Requester Computername */
                                         /* Description */
IPAddress = " "                          /* IP Address */
IPMask = " "                             /* IP Mask */
IPRouter = " "                           /* IP default Router */
IPNetAddress = " "                       /* IP Net Address */
IPNetMask = " "                          /* IP Router Net Mask */
IPDomain = " "                           /* IP Domain Name */
```

Figure 245. WSCONFIG.CMD (Part 1 of 5)

```
IP_DNS = " "                              /* IP Domain Name Server */
IPHostSocksNameServer = " "               /* IP Sostname Socks Name Server */
                                          /* address */
IPSocksNameServer = " "                   /* IP Socks Name Server address */
LAMacAddress = " "                        /* Localy Administ. MAC Address */
PCommDestinationSAP = " "                 /* Destination Address for PComm */
                                          /* SNA Session */
PCommIdentifier = " "                     /* PU Id for PComm SNA Session */
WSConfig_rc = 0                           /* Return Code */
tmp_rc = 0                                /* Temp to hold return codes */
tmp_various = ""                          /* Temp to hold various values */

/* Get NVDM/2 WorkStation Name */

parse UPPER arg ComputerName ProcessMode dummy    /* dummy will separate */
                                                  /* any unwanted        */
                                                  /* parameters from the */
                                                  /* ComputerName */
If ComputerName = ""
then do
   WSConfig_rc = 2
   say "Required Computer Name was not provided."
   say "The systax is WSCONFIG computername process-mode"
   say "Example: WSCONFIG UZ1175E PRISTINE"
   say "Contact LAN Administrator to obtain a Computer Name."
end /* then do */

If (ProcessMode <> "PRISTINE") & (ProcessMode <> "MIGRATION")
then do
   WSConfig_rc = 2
   say "Required Process Mode was not provided."
   say "The systax is WSCONFIG computername process-mode"
   say "Example: WSCONFIG UZ1175E PRISTINE"
end /* then do */

/* Define the location of the REXX script to run CONFIGUR.EXE */
ConfigurCtrlFile = "X:\LOG\"ComputerName"\WSDetails.cmd"

/* Find WorkStation details */

if WSConfig_rc = 0
then do
   say "Searching for "ComputerName" details in data file "WSDataBase"..."
   Do While Lines(WSDataBase) & FoundRecord = 0
      CurrentLine = LineIn(WSDataBase)
      if Strip(SubStr(CurrentLine,1,15),B," ") = ComputerName
      then do
         LRDomain = Strip(SubStr(CurrentLine,16,15),B," ")
         LRDescription = Strip(SubStr(CurrentLine,31,48),B," ")
         IPAddress = Strip(SubStr(CurrentLine,79,15),B," ")
         IPMask = Strip(SubStr(CurrentLine,93,15),B," ")
         IPRouter = Strip(SubStr(CurrentLine,109,15),B," ")
         IPNetAddress = Strip(SubStr(CurrentLine,124,15),B," ")
         IPNetMask = Strip(SubStr(CurrentLine,139,15),B," ")
         IPDomain = Strip(SubStr(CurrentLine,154,64),B," ")
         IP_DNS = Strip(SubStr(CurrentLine,218,15),B," ")
         IPHostSocksNameServer = Strip(SubStr(CurrentLine,233,64),B," ")
         IPSocksNameServer = Strip(SubStr(CurrentLine,297,15),B," ")
```

Figure 246. WSCONFIG.CMD (Part 2 of 5)

```
         LAMacAddress = SubStr(CurrentLine,312,12)
         NVDMServerName = Strip(SubStr(CurrentLine,324,15),B," ")
         PCommDestinationSAP = SubStr(CurrentLine,339,12)
         PCommIdentifier = Strip(SubStr(CurrentLine,351,8),B," ")
         FoundRecord = 1
         say "Details for "ComputerName" have been found."
      end /* do */
   end /* do */
   if FoundRecord = 0
   then do
      WSConfig_rc = 2
      say "No data for Computer Name ’"ComputerName"’ was found in ’",
           WSDataBase"’."
      say "Contact LAN Administrator to obtain a Computer Name."
   end /* do */
end /* do */


/*********************************************************************/
/* Modify C:\IBMNVDM2\IBMNVDM2.INI to configure the NVDM/2 WorkStation */
/* Name and NVDM/2 Server Name. Also do the same to                   */
/* E:\IBMNVDM2\IBMNVDM2.INI.                                          */

if WSConfig_rc = 0
then do
   if Stream(TmpCubeCmdsFile,’c’,’query exists’) <> ""
   then do
      Call Stream TmpCubeCmdsFile,’c’,’close’ /* Just in case the file */
                                              /* is open (this is the  */
                                              /* first time the file is */
                                              /* used.                  */
      ’DEL ’TmpCubeCmdsFile               /* To have a new clean file */
   end /* then do */
   TmpCubeFileRec = ’RepLine "ServerName" WITH "ServerName     = ’||,
                    NVDMServerName’" ( *ID ’
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   TmpCubeFileRec = ’RepLine "ClientName" WITH "ClientName     = ’||,
                    ComputerName’" ( *ID ’
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   Call Stream TmpCubeCmdsFile,’c’,’close’
   CubeParamenters = TmpCubeCmdsFile’ C:\IBMNVDM2\IBMNVDM2.INI ’||,
                     ’C:\IBMNVDM2\IBMNVDM2.CUB’

   /* Make sure file exists... */
   if Stream(’C:\IBMNVDM2\IBMNVDM2.INI’,’c’,’query exists’) <> ""
   then WSConfig_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else WSConfig_rc = ’4’x

   CubeParamenters = TmpCubeCmdsFile’ E:\IBMNVDM2\IBMNVDM2.INI ’||,
                                    ’ E:\IBMNVDM2\IBMNVDM2.CUB’

   /* Make sure file exists... */
   tmp_various = Stream(’E:\IBMNVDM2\IBMNVDM2.INI’,’c’,’query exists’)
   if (WSConfig_rc = 0) & (tmp_various <> "")
   then WSConfig_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else WSConfig_rc = ’4’x
end /* do */
```

Figure 247. WSCONFIG.CMD (Part 3 of 5)

```
if WSConfig_rc = 0
then do
   if Stream(TmpCubeCmdsFile,'c','query exists') <> ""
   then do
      'DEL 'TmpCubeCmdsFile             /* To have a new clean file  */
   end /* then do */
   TmpCubeFileRec = 'AddLine ~   NETADDRESS = "'LAMacAddress||,
                  '"~ ( After ~[IBMTOK_nif]~ IFNEW'
   Call LineOut TmpCubeCmdsFile,TmpCubeFileRec
   Call Stream TmpCubeCmdsFile,'c','close'
   CubeParamenters = TmpCubeCmdsFile' C:\IBMCOM\PROTOCOL.INI '||,
                     'C:\IBMCOM\PROTOCOL.CUB ( DLM ~ )'

   /* Make sure file exists... */
   if Stream('C:\IBMCOM\PROTOCOL.INI','c','query exists') <> ""
   then WSConfig_rc = "W:\EXE\CUBE.CMD"(CubeParamenters)
   else WSConfig_rc = '4'x
end /* do */


/**********************************************************/
/* Last, but not least, delete the temp file being use to */
/* hold the CUBE.CMD commands.                            */

if Stream(TmpCubeCmdsFile,'c','query exists') <> ""
then 'DEL 'TmpCubeCmdsFile


/************************************************************/
/* Once the Workstation details have been found and minimun */
/* values have been set, write a REXX script file to run    */
/* CONFIGUR.EXE with all the Workstation values.            */

if (WSConfig_rc = 0) & (ProcessMode = "PRISTINE")
then do
   if Stream(ConfigurCtrlFile,'c','query exists') <> ""
   then do
      Call Stream ConfigurCtrlFile,'c','close' /* Just in case the   */
                                               /* file is open (this */
                                               /* is the first time  */
                                               /* the file is used. */
      'DEL 'ConfigurCtrlFile              /* To have a new clean file */
   end /* do */

   ConfigurFileRec = "/* Sets FI Variable values and runs CONFIG.EXE */"
   WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = " "
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "'CONFIGUR.EXE "||,
                     'E:'ComputerName||,
                     '/B:W:\IMG\CONFIG\'ComputerName||,
                     '/W:C:\'||,
                     '"/V:{COMPUTERNAME}='ComputerName'"'||,
                     '"/V:{DOMAIN}='LRDomain'"'||,
                     '"/V:{SRVCOMMENT}='LRDescription'"'||,
                     '"/V:{PCOMMDESTSAP}='LAMacAddress'"'||,
```

Figure 248. WSCONFIG.CMD (Part 4 of 5)

```
                     '"/V:{PCOMMID}='PCommIdentifier'"'||,
                     '"/V:{HOSTNAME}='ComputerName'"'||,
                     '"/V:{IPDOMAIN}='IPDomain'"'||,
                     '"/V:{IP_DNS}='IP_DNS'"'||,
                     '"/V:{IPADDRESS}='IPAddress'"'||,
                     '"/V:{IPMASK}='IPMask'"'||,
                     '"/V:{IPROUTER}'IPRouter'"'||,
                     '"/V:{IPNETADDRESS}='IPNetAddress'"'||,
                     '"/V:{IPNETMASK}='IPNetMask'"'||,
                     '"/V:{IPHOSTSOCKSNAMESERVER}='IPSocksNameServer'"'||,
                     '"/V:{IPSOCKSNAMESERVER}='IPSocksNameServer'"'||,
                     '/L:X:\LOG\'ComputerName'"'||,
                     '/X:4'
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "tmp_rc = rc"
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)
   ConfigurFileRec = "exit tmp_rc"
   if WSConfig_rc = 0
   then WSConfig_rc = LineOut(ConfigurCtrlFile,ConfigurFileRec)

   /* All lines have been written to the REXX script... */
   /* So, close the file.                               */

   if WSConfig_rc = 0
   then do
       WSConfig_rc = Stream(ConfigurCtrlFile,'c','close')
       if WSConfig_rc = "READY:"
       then WSConfig_rc = 0
   end /* then do */

end /* do */


/************************************************************/
/* If there were no errors, return a CID rc for succesful */
/* completition with reboot request.                      */

if WSConfig_rc = 0
then do
   WSConfig_rc = 'fe00'x
   say 'WSCONFIG.CMD has exited with return code 0, CID rc = fe00.'
end
else do
   say 'WSCONFIG.CMD failed with rc = 'WSConfig_rc'.'
end
exit WSConfig_rc
```

Figure 249. WSCONFIG.CMD (Part 5 of 5)

C.8 GETNVDM.CMD

```
/* REXX program to extract the NVDM/2 Client name and the NVDM/2 server name */
/* from a workstation and store this information in a                        */
/* cube profile in the correct directory on the NVDM/2 Server                */


parse upper arg workstationname .

ECHO OFF

targetpath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'

logdir = 'X:\LOG\'||workstationname

RCode = 0
server = ''
client = ''

RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\GETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
if RCode == 0 then do
   do while lines(nvdmpath||'\IBMNVDM2.INI')
      line.linenum = Linein(nvdmpath||'\IBMNVDM2.INI')
      select
         when word(line.linenum, 1)=='ServerName' then server=line.linenum
         when word(line.linenum, 1)=='ClientName' then client=line.linenum
      otherwise NOP
      end  /* select */
   end /* do */


if RCode == 0 then do
   RC = STREAM(logdir||'\GETINFO.LOG', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'logdir||'\GETINFO.LOG'
   end /* do */
   RC = STREAM(targetpath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
   if RC <> '' then do
      'DEL 'targetpath||'\IBMNVDM2.PRO'
   end /* do */

   cubeline = 'RepLine "ServerName" WITH "'server'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'

   cubeline = 'RepLine "ClientName" WITH "'client'"'
   'ECHO' cubeline' >> 'targetpath||'\IBMNVDM2.PRO'
   Call Stream targetpath||'\IBMNVDM2.PRO','c','close'
end /* do */
```

Figure 250. GETNVDM.CMD (Part 1 of 2)

```
if RCode == 0 then do
   'ECHO GETINFO ended successful >> 'logdir||'\GETINFO.LOG'
end /* do */
else do
   'ECHO GETINFO ended unsuccessful with >> 'logdir||'\GETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\GETINFO.LOG'
end /* do */
Call Stream logpath||'\GETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 251. GETNVDM.CMD (Part 2 of 2)

C.9 SETNVDM.CMD

```
/* REXX program to set the information extracted with GETINFO */

parse upper arg workstationname .

ECHO OFF

sourcepath = 'W:\IMG\CONFIG\'||workstationname
nvdmpath = 'C:\IBMNVDM2'
mptspath = 'C:\IBMCOM'

logdir = 'X:\LOG\'||workstationname

RCode = 0

RC = STREAM(logdir||'\SETINFO.LOG', 'C', 'QUERY EXISTS')
if RC <> '' then do
   'DEL 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(nvdmpath||'\IBMNVDM2.INI', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.INI does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
RC = STREAM(sourcepath||'\IBMNVDM2.PRO', 'C', 'QUERY EXISTS')
if RC == '' then do
   'ECHO IBMNVDM2.PRO does not exist... >> 'logdir||'\SETINFO.LOG'
   RCode = 8
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */

if RCode == 0 then do
   cubearg = sourcepath||'\ibmnvdm2.pro 'nvdmpath||'\IBMNVDM2.INI'
   RC = 'cube.cmd'(cubearg)
   if RC <> 0 then RCode = RC
   if RCode <> 0 then do
      say 'One or more Errors were found executing the changes'
      say 'to 'nvdmpath||'\IBMNVDM2.INI and 'mptspath||'\PROTOCOL.INI'
      RCode = 4
   end
end /* do */
if RCode == 0 then do
   'ECHO SETINFO ended successful >> 'logdir||'\SETINFO.LOG'
end /* do */
else do
   'ECHO SETINFO ended unsuccessful with >> 'logdir||'\SETINFO.LOG'
   'ECHO RCode = 'RCode' >> 'logdir||'\SETINFO.LOG'
end /* do */
Call Stream logpath||'\SETINFO.LOG','c','close'
ECHO ON
exit RCode
```

Figure 252. SETNVDM.CMD

Appendix D. CUBE Source Code and Documentation

CUBE is a REXX procedure used to modify a CONFIG.SYS-like ASCII file (the Target File), based on a set of CUBE's commands (the Procedure File).

CUBE was initially developped to automatically create/update CONFIG.SYS or STARTUP.CMD files. It may be integrated in dynamic processes used to customize workstation configurations. Any other ASCII file (such as profiles, commands files, etc...) may also be customized with CUBE. As a matter of fact, in our scenarios we used CUBE to modify the IBMNVDM2.INI file to represent the correct workstation name at boot-up time so that the Configurator, CONFIGUR.EXE, is able to perform the rest of necessary configuration changes.

CUBE has its own set of commands that provide editing functions at the line and string levels. This set of commands includes ADD, REPLACE and DELETE functions.

Most of CUBE's commands require identification of the Target File line they act upon: the line will be identify by its leftmost characters (as many as necessary for a precise or generic identification) starting from column 1. This was inherited from the CONFIG.SYS type of file CUBE was created for. However, the option '*ID', wherever available, tells CUBE to identify the Target File line as any line containing the identification string, starting at any position. Furthermore, the command LINEID may specify that leading characters in Target File lines should be ignored (stripped) for identification.

Commands in a Procedure File may be further customized at CUBE's execution time: strings may contain 'variables' names that will be replaced by their current value at execution time. This is known as 'substitution' and comes in two flavors: command line substitution and environment variable substitution. Command line substitution: values of variables are passed in the command line. Environment variable substitution: values of variables are those of currently defined OS/2 environment variables at CUBE's execution time. Substitution in CUBE's commands always occurs, while substitution in a Target File line only occurs if this line is processed by a CUBE's command. Variable names are always identified by delimiters: any single character you wish, unique for a given variable name, to avoid conflict with other characters in a string or system interpretation. Variable names ARE case sensitive.

D.1 Illustrating CUBE

The following syntax disgram displays the calling syntax of the CUBE utility:

CUBE Calling Syntax

```
CUBE <procfname>
     <targetfname>
     [backupfname]
     [options]
      [>log-fname ]
```

where:

procfname	Specifies the name of the procedure file containing CUBE's commands or { single cube command } or QUEUE Notes: 1) { and } are required delimiters for single cube command passed on command line. 2) QUEUE (uppercase!) is the required keyword when CUBE is called from a Rexx command that have 'queued' cube commands in the session queue, rather than used a procedure file.
targetfname	Specifies the name of the target file to be updated.
backupfname	If specified, it is the file name under which the target file will be saved, before CUBE is processing.

options

[PAUSE]	CUBE pauses after each command executed from Procfname
[CHECK]	Run CUBE but *DON'T SAVE* the target file. Return code will reflect the number of changes that would occur in target file.
[MAKE code]	Implements the 'conditionnal command execution'. CUBE will only execute the profile's commands that are under the control of a matching WHEN

	command. See WHEN command for explanations. Code can be any string, default is *.
[DLM char]	Define a new delimiter for CUBE's commands strings. See definition of d-string below.

[RS(#varnam1#=value1 #varnam2#=value2 ....)]

	Specifies a set of 'variablename=value' . CUBE will replace each occurrence of a variable, within each procedure file command, with its corresponding value. Substitution is done before commands processing. varnames must be defined with delimiters (# above).

Examples:

```
CUBE c:\config.pro config.sys config.bak >cube.log
CUBE p.cub config.sys (rs(#drive1#=D #drive2#=E) pause
CUBE {RS "#var2#" (RS#) all} startup.cmd >startup.log
CUBE QUEUE appl.pro
```

D.1.1 CUBE Procedure File Commands Syntax

Within CUBE's commands, any string representing the actual string to be looked for, replaced and/or inserted in the Target File, must be enclosed between two identical characters (delimiters). These strings will be referred to as "d-strings" (delimited strings). The default delimiter is " (double-quote). DLM option can be used to define your own delimiter for a given CUBE execution.

Commands will be interpreted/executed sequentially from Procedure File. Commands may be upper or lower case. d-strings too, but will be handled according to the last CASE command encountered (see CASE command).

Commands may span on more than one line. A ',' must end a command that is to be continued on next line. ',' within d-strings will not be interpreted as a continuation characters.

Blank (null) lines are accepted. Comments lines may be inserted, with either '*' or '--' as their first character(s). Blank and comments lines may NOT appear in between continuation lines of a command.

ONERROR	Specifies what action to be taken in case of syntax error while processing a command of the current procedure file.

Syntax:

`ONERROR { CONTINUE | STOP }`

`CONTINUE`	Issue error message and skip to next command.
`STOP`	Issue error message and stop CUBE processing. All updates to Target File will be lost in this case (even those applied before the `ONERROR STOP` command). Default is `STOP` if no `ONERROR` command issued.

`CASE` Specifies how procedure and target strings are handled for strings comparisons. Once set, it applies for both line identifications and strings identifications within line.

Syntax:

`CASE { SENSITIVE | IGNORE }`

`SENSITIVE`	Strings comparisons will be case sensitive.
`IGNORE`	Strings comparisons will NOT be case sensitive. For example, "AbCdef" matches "ABCDEf". Default is `IGNORE` if no `CASE` command issued.

`WHEN` Following CUBE's commands will only be executed if the CUBE's command line option `MAKE` matches one of the 'codes' specified. Applies until another `WHEN` command is found. `WHEN` has no effect on `ONERROR`, `CASE`, `LINEID` , `STRINGDELIMITER` and `WHEN` commands: these will always get executed.

Syntax:

`WHEN` { *code1* ... *coden*

coden	Any string, case insensitive. If * then following commands will all get executed, whatever was specified in `MAKE` option. * always matches the `MAKE` option. Default is * (no `WHEN` command specified)}

`LINEID` Specifies that identification of target file lines should begin at column 1 (CUBE's default) or ignore leading characters

Syntax:

`LINEID { NOSTRIP | STRIP "X" }`

NOSTRIP	Return to default CUBE identification (column 1).
STRIP "X"	Strip leading x's in Target File lines for line identification. Default is NOSTRIP if no LINEID command issued.

REPLINE — REPLACE an entire LINE of target file with a new text. The d-string used to identify the line to be replaced, corresponds to the leftmost characters of the line. If the line is not found within target file, the new line may be added to the target file.

Syntax:

REPLINE *lineid* WITH *replacement* [(*options*]

Command Abbreviation	RL
lineid	d-string to identify line
replacement	d-string representing the new line
options:	
[ALL | FIRST | LAST]	ALL replaces all occurences of 'lineid' (default) FIRST only replaces 1st occurences of *lineid* LAST only replaces last occurence of *lineid*
[ADDTOP | ADDBOTTOM | DONTADD]	
	ADDTOP: if no *lineid*, add replacement at top of file ADDBOTTOM : if no *lineid*, add replacement at bottom DONTADD: if no *lineid*, don't add replacement (default)
[*ID]	Target line identification anywhere in the line.
[RS(x)]	Replace string(s) within the new line. x is a single character used as a string delimiter

for the duration of this command only; all strings delimited by a pair of x, in the new line, will be interpreted as environment variable names replaced by their value. This occurs before any other REPLINE processing.

[IF "x"]	Replace only if a line identified by "x" exists in the target file.
[IFNOT "x"]	Replace only if a line identified by "x" doesn't exist in target file.

ADDLINE Conditionnally adds a LINE to the target file. Position of the line may be specified by reference to an existing line.

Syntax:

ADDLINE *line* [(*options*]

Command Abbreviation	AL
line	d-string representing the line to be added

Options:

[AFTER x [ONLY] | BEFORE x [ONLY]]

AFTER x means add after line identified by d-string x. If x is not specified or not present in target file, line will be added at bottom of Target File.
BEFORE x means add before line identified by d-string x. If x is not specified or not present in target file, line will be added at top of target file.
ONLY with AFTER/BEFORE means add only if x there.

[IFNEW | ALWAYS]	IFNEW means add line only if not already there (default) ALWAYS means always add line (even if already there)
[*ID]	Target line identification anywhere in the line.
[RS(x)]	Replace string(s) within the line to be added. x is a single character used as a string delimiter for the duration of this

	command only; all strings delimited by a pair of x, in the line to be added, will be interpreted as environment variable names replaced by their value. This occurs before any other ADDLINE processing.
[IF "x"]	Add only if a line identified by "x" exists in the target file.
[IFNOT "x"]	Add only if a line identified by "x" doesn’t exist in target file.

DELLINE DELETE a given LINE from target file.

Syntax:

DELLINE *lineid* [(*options*]

Command Abbreviation	DL
lineid	id-string identifying the line to be deleted. Correspond to the lefmost characters of the line.
Options:	
[ALL | FIRST | LAST]	ALL deletes all occurences of ’lineid’ (default) FIRST only deletes 1st occurence of ’lineid’ LAST only deletes last occurence of ’lineid’
[*ID]	Target line identification anywhere in the line.
[RS(x)]	Replace string(s) within lineid to identify the line to be deleted. Standard RS() pre-processing; refer to addline.
[IF "x"]	Delete only if a line identified by "x" exists in the target file.
[IFNOT "x"]	Delete only if a line identified by "x" doesn’t exist in target file.

COMMENTLINE Comment out a line in target file. In fact, it places a user-specified character string at the beginning of a line.

Syntax:

COMMENTLINE *lineid* WITH *type* [(*options*]

Command Abbreviation	CL
lineid	d-string identifying the line to be commented out.
type	d-string representing comment character(s)
Options:	
[ALL | FIRST | LAST]	ALL comments all occurences of *lineid* (default) FIRST only comments 1st occurence of *lineid* LAST only comments last occurence of *lineid*
[*ID]	Target line identification anywhere in the line.
[IF "X"]	Comment only if a line identified by "x" exists in the target file.
[IFNOT "X"]	Comment only if a line identified by "x" doesn’t exist in target file.

ADDSTRING Add a string within a given line of the target file. The position of the string is specified by reference to an existing string within the line.

Syntax:

ADDSTRING *string* IN *lineid* ({ AFTER S | BEFORE S} [*Options*]

Command Abbreviation	AS
string	d-string representing the string to be added
lineid	d-string identifying the line where string must be added. Correspond to the leftmost characters of the line.
AFTER [s]	Add string after d-string s in line. If s is not specified or not present in line, string will be added at end of line.

BEFORE [s]	Add string before d-string s in line. If s is not specified or not present in line, string will be added at beginning of line, but after the identifier (*lineid*).
Options:	
[IFNEW | ALWAYS]	IFNEW means add string only if not already there (default) ALWAYS means always add string (even if already there)
[ALL | FIRST | LAST]	ALL means add to all occurences of *lineid* FIRST means add only to 1st occurence of *lineid* LAST means add only to last occurence of *lineid*
[*ID]	Target line identification anywhere in the line.
[RS(x)]	Replace substring(s) within the new string. x is a single character used as a string delimiter for the duration of this command only; all substrings delimited by a pair of x, in the new string, will be interpreted as environment variable names replaced by their value. This occurs before any other ADDSTRING processing.
[ADDTOP | ADDBOTTOM]	ADDTOP means if lineid not found, adds line *lineid* || *string* at TOP of target file. ADDBOTTOM means if lineid not found, adds line *lineid* || *string* at BOTTOM of target file.

REPSTRING REPLACE a given STRING by another, within a given line of target file. If no line is specified (lineid) then all occurences of string within target file will be replaced with the new string. All occurences of the string in a given line are replaced.

Syntax:

REPSTRING *ostring* WITH *nstring* [IN *lineid*] [(*Options*]

Command Abbreviation	RS

ostring	d-string representing the string to be replaced.
nstring	d-string representing the new string
lineid	d-string identifying the line where string must be replaced. Optional.
Options:	
[ALL | FIRST | LAST]	ALL replaces in all occurences of *lineid* FIRST replaces only in 1st occurence of *lineid* LAST replaces only in last occurence of *lineid*
[*ID]	Target line identification anywhere in the line.
[RS(x)]	Replace substring(s) within the new string. x is a single character used as a string delimiter for the duration of this command only; all substrings delimited by a pair of x, in the new string, will be interpreted as environment variable names replaced by their value. This occurs before any other ADDSTRING processing.

DELSTRING

DELETE a given STRING in a given line of target file. If no line is specified, then all occurrences of string within target file will be deleted. All occurences of the string in a given line are deleted.

Syntax:

DELSTRING *string* [IN *lineid*] [(*Options*]

Command Abbreviation	DS
string	d-string representing the string to be deleted.
lineid	d-string identifying the line where string must be deleted. Optional.
Options:	
[ALL | FIRST | LAST]	ALL deletes in all occurences of *lineid* FIRST deletes only in 1st occurence of *lineid*

	`LAST` deletes only in last occurence of *lineid*
[`*ID`]	Target line identification anywhere in the line.
[`RS(x)`]	Replace substring(s) within string to be deleted. Standard RS() pre-processing; refer to addstring.

Examples:

1. `ONError CONTINUE`
   Continue processing even in case of error(s).
2. `REPline "COUNTRY=" with "COUNTRY=001,C:\OS2\SYSTEM\COUNTRY.SYS" (ADDBOTTOM`
   replaces all occurences of lines starting with "COUNTRY=" adds new line at bottom of Target File if no line starting with "COUNTRY=" exists.
3. `ADDLINE "CODEPAGE=437,850" (AFTER "COUNTRY=" IFNEW`
   adds a line "CODEPAGE=437,850" after the line starting with "COUNTRY=" , only if "CODEPAGE=437,850" doesn't already exist in Target File. If it is to be added and no line starting with "COUNTRY=" exist, then add at bottom of file (implied by AFTER).
4. `DELLINE "SET=" (first`
   deletes the first line starting with "SET=" .
5. `DELLINE "SET=" (first  ifnot "REQUIRESET"`
   deletes the first line starting with "SET=" only if Target File does not contain a line starting "REQUIRESET"
6. `COMMENTLINE "ifs=c:\os2\hpfs.ifs" with "rem "`
   comments the line starting with "IFS=C:\OS2\HPFS.IFS" with "REM ". Line will then be: REM IFS=C:\OS2\HPFS.IFS ....
7. `ADDSTRING "C:\MYDLL;" IN "LIBPATH=" (FIRST IFNEW BEFORE "C:\OS2\DLL;"`
   adds "C:\MYDLL;", if it doesn't already exist, within the line starting with "LIBPATH=". Adds the string before "C:\OS2\DLL;" if it exists; otherwise adds the string after "LIBPATH=".
8. `AS "MYNAME" IN "USERID" (AFTER *ID`
   adds "MYNAME", if it doesn't already exist, within any line containing "USERID". Adds the string after "USERID".
9. `AS "#indirect# IN "SET VALUES=" (AFTER RS(#)`
   assuming that the command 'SET INDIRECT=3' has been executed prior to CUBE execution, appends "3", if it doesn't already exists, to "SET VALUES=".

10. `REPSTRING "D:\TOOLKT13\IPFC;" WITH "D:\TK13\IPFC" IN "SET HELP=" (LAST`
    replaces the string "D:\TOOLKT13\IPFC; with "D:\TK13\IPFC;" within the last line starting with "SET HELP=".
11. `REPSTRING "D:\TOOLKT13" WITH "D:\TK13" (all`
    replaces all occurences of "D:\TOOLKT13" with "D:\TK13".
12. `AL "USER=#name#,NODE=#node#" (AFTER IFNEW`
    assuming CUBE has been invoked with option (RS(#name#=ME #node#=HERE), will adds a line "USER=ME,NODE=HERE" .
13. `WHEN C`
    assuming CUBE has been invoked with option (MAKE C , the following commands will be executed.
14. `DELSTRING _xxx"xxxxxxx_ (All`
    deletes all occurences of xxx"xxxxxxx if option DLM _ was used on CUBE's command line

D.1.2 CUBE.CMD Source Code

The following figures display the source code of the CUBE utility.

```
/*  +--------------------------------------------------------------------+
    |                    Config Update/Batch Editing                     |
    |                                                                    |
    |  Batch update of CONFIG.SYS-like files. CUBE modifies a Target ASCII  |
    |  file, given a set of commands in a Procedure file.                |
    |                                                                    |
    +--------------------------------------------------------------------+
*/
'@echo off'
version = '2.6'
pf = 0
if left(arg(1),1) = '{' then parse arg '{'PFile'}' TFile Bkup . '(' Opt
else do
  parse arg PFile TFile Bkup . '(' Opt
  pf = 1
end

parse upper source source                          /* who am I ?           */

if pf then do
 if PFile = '' then call Exit 0 'no procedure !'  /* no or missing PFile  */
 if Pfile <> 'QUEUE' then if exists(PFile)='' then call Exit 0 PFile 'not found'
end
if TFile = ''  then call Exit 0 'no target !'      /* no or missing Tfile  */
if exists(TFile)='' then call Exit 0 TFile 'not found'

pause=(wordpos('PAUSE',translate(Opt))>0)       /* Pause mode ?            */
chkmd=(wordpos('CHECK',translate(Opt))>0)       /* Check mode ?            */
ap = wordpos('MAKE',translate(Opt))             /* make  specified ?       */
if ap>0 then Make=translate(word(Opt,ap+1))     /* ...when                 */
        else Make = '*'                         /* default make = all      */
MWhen='*'                                       /* Default when = all      */
dlm = wordpos('DLM',translate(Opt))             /* New delimiter specified?*/
if dlm>0 then _d_=left(word(Opt,dlm+1),1)       /*   yes use it            */
         else _d_ = '"'                         /* else use default        */
OnErr = 'STOP'                                  /* Default OnError setting */
LStrip= ''                                      /* No lineid strip         */
CaseM = 'I'                                     /* String compare default  */
call get_cmdrs(Opt)
NumberOfChanges = 0
say arg
say 'CUBE' version 'applying' PFile 'to' TFile 'on' date() time()
if Bkup <> "" then do
   address cmd 'copy' Tfile Bkup '1>nul 2>nul'
   if rc = 0 then say Tfile 'backup is:' Bkup
end

Proc. = ''
if pf then do
   If PFile = 'QUEUE' then Do
     i = 1
     Do Queued()
       Parse Pull procline
       proc.i = proc.i || upkw(procline)
       if right(Proc.i,1) = ','                       /* continuation char? */
         then proc.i=left(proc.i,length(proc.i)-1)' ' /* yes: blank it out  */
         else  i = i + 1                              /* no: new Proc line  */
     end
```

Figure 253. CUBE.CMD Source Code (Part 1 of 13)

```
     Proc.0 = i-1                                 /* Proc.0 = # of lines*/
     if Proc.0 <= 0 then call Exit 0 PFile 'empty'
  end
  Else do
     i = 1 ;                                 /* current Proc line: null */
     do while lines(PFile)                   /* for all PFile's lines   */
       Proc.i = Proc.i || upkw(linein(PFile))    /*   concat to Proc line   */
       if right(Proc.i,1) = ','              /*   continuation char ?   */
         then proc.i=left(proc.i,length(proc.i)-1)' ' /* yes: blank it out  */
         else  i = i + 1                          /*  no: new Proc line */
     end
     Proc.0 = i-1                             /* Proc.0 = # of lines    */
     call close PFile
     if Proc.0 <= 0 then call Exit 0 PFile 'empty'
  End
end
else do
  Proc.0 = 1
  Proc.1 = upkw(Pfile)
end

i = 0
do while lines(TFile)                        /* for all TFile's lines   */
  i = i + 1                                  /*   get line in           */
  Target.i = linein(TFile)                   /*   Target. stem          */
end
Target.0 = i                                 /* Target.0 = # of lines   */
call close Tfile

/*  +------------------------------------------------------------------+
    |The real thing: go thru procedure file, interpret/execute its commands |
    |sequentially.                                                     |
    +------------------------------------------------------------------+
*/

p = 0                                        /* Proc lines index        */
do while p <= Proc.0                         /* for all PFile's lines   */
  p = p + 1                                  /*   index next line       */
  if Proc.p = '' then iterate                /*   ignore null lines     */
  parse var Proc.p Verb Parms                /*   Isolate command verb  */
  say ''
  say '>>>' Proc.p
  Verb = translate(Verb)
  Select                                     /*   Process verb          */
    When left(Verb,1) = '*'   then iterate
    When left(Verb,2) = '--'  then iterate
               /* commands that always get executed  */
    When Verb = 'WHEN'        then call APPLYWHEN
    When Verb = 'ONERROR'     then call ONERROR
    When Verb = 'CASE'        then call CASE
    When Verb = 'LINEID'      then call SLINEID
    When wordpos(Make,MWhen)=0 then iterate
               /* commands executed when WHEN/MAKE match */
    When Verb = 'REPLINE'     | verb = 'RL'  then call REPLINE
    When Verb = 'ADDLINE'     | verb = 'AL'  then call ADDLINE
    When Verb = 'ADDSTRING'   | verb = 'AS'  then call ADDSTRING
    When Verb = 'DELSTRING'   | verb = 'DS'  then call DELSTRING
    When Verb = 'REPSTRING'   | verb = 'RS'  then call REPSTRING
```

Figure 254. CUBE.CMD Source Code (Part 2 of 13)

```
    When Verb = 'COMMENTLINE' | verb = 'CL'  then call COMMENTL
    When Verb = 'DELLINE'     | verb = 'DL'  then call DELLINE
    Otherwise rc=OnErrorDo(p,"Don't know what to do")
  end
  if pause then Pull .
end
if chkmd = 0 then call exit 1 source 'ended.'    /* It's OVER !! and OK !!  */
if chkmd = 1 then call exit 2 source 'ended.'    /*                         */

/* +-----------------------------------------------------------------------+
   |  Error report and action (based on Onerr setting, from ONERROR cmd)   |
   +-----------------------------------------------------------------------+
*/
OnErrorDo:
  parse arg line,msg
  say PFile', line' line':' msg
  if OnErr = 'STOP' then call Exit 0 source 'stopped.'
                  else return 0

/* +-----------------------------------------------------------------------+
   |  Searches All or First or Last lines in Target starting with string   |
   |  Returns the line number(s) found.                                    |
   +-----------------------------------------------------------------------+
*/
Whereis: procedure expose Target. CaseM LStrip
 parse arg string,direction,mode
 if wordpos(direction,'F A')>0 then do; de=1; a=Target.0; par=1; end
                               else do; de=Target.0; a=1; par=-1; end
 stringlength=length(string); ret = ''
 do i = de to a by par
   If CaseM = 'S' then do; T = Target.i ; S = string ; end
      else do; T = translate(Target.i) ; S = translate(string) ; end
   If length(LStrip) = 1 then T = strip(T,'L',Lstrip)
   if mode=1 then do
     if left(T,stringlength)=S then do
       ret = ret i
       if direction \= 'A' then leave
     end
   end
   else do
     if pos(S,T) > 0 then do
       ret = ret i
       if direction \= 'A' then leave
     end
   end
 end
 return ret

/* +-----------------------------------------------------------------------+
   |  Update Target file from Target. stem. Remove '       ' lines |
   +-----------------------------------------------------------------------+
*/
SaveFile:
 address cmd 'erase' TFile
 src = rc
 if src = 0 then do
   do i = 1 to Target.0
     if Target.i = '       ' then iterate
```

Figure 255. CUBE.CMD Source Code (Part 3 of 13)

```
     rc=lineout(TFile,Target.i)
   end
   call close Tfile
 end
 return src

/* +--------------------------------------------------------------------+
   |  Insert a line in Target file (stem) after line number i.          |
   +--------------------------------------------------------------------+
*/
Insert: procedure  expose Target. NumberOfChanges
 parse arg i string
 if i = Target.0 then k = Target.0 + 1
 else do
   do j = Target.0 to i+1 by -1
     k = j + 1
     Target.k = Target.j
   end
   k = i + 1
 end
 Target.k = string
 Target.0 = Target.0 + 1
 say 'Inserted line' k ': "'Target.k'"'
 NumberOfChanges = NumberOfChanges + 1
 return

/* +--------------------------------------------------------------------+
   |  returns a procedure command line  with all strings uppercased, except|
   |  doubled-quoted strings.                                            |
   +--------------------------------------------------------------------+
*/
upkw: procedure expose vn. vv. _d_
  parse arg sentence
  sentence = strip(sentence)
  phrase = ""
  do forever
    if sentence = '' then leave
    if left(word(sentence,1),1) = _d_ then do
       parse var sentence (_d_) y (_d_) sentence
       phrase = phrase _d_ || y || _d_
    end
    else do
       parse var sentence y sentence
       phrase = phrase translate(y)
    end
  end
  return phrase

/* +--------------------------------------------------------------------+
   |  apply command line-specified substitutions within a string         |
   +--------------------------------------------------------------------+
*/
Cmdrs: procedure expose vn. vv.
 parse arg y
 do i = 1 to vn.0
   out = ''
   do forever
     if pos(vn.i,y) > 0 then do
```

Figure 256. CUBE.CMD Source Code (Part 4 of 13)

```
     parse var y x (vn.i) y
        out = out || x || vv.i
     end
     else leave
   end
 y = out || y
 end
 return y

/* +------------------------------------------------------------------------+
   |  apply env. variables substitutions to STRING if req. in OPTION.      |
   +------------------------------------------------------------------------+
*/
Envrs: procedure
  Parse arg String,Option
  out = ''
  parse var Option x 'RS('c')' .
  if length(c) = 1 then do
     do forever
        parse var String x (c) name (c) String
        if name = "" then leave
        out = out || x || value(name,,'OS2ENVIRONMENT')
     end
     String = out || x
  end
  return String

/* +------------------------------------------------------------------------+
   | All that must be done to quit and more: say msg, save Target file if  |
   | necessary (type=1).                                                    |
   +------------------------------------------------------------------------+
*/
Exit:
 parse arg type msg
 src=0
 Select
   When type=1 then do
        src = SaveFile()
        if src <> 0 then msg = msg 'Error writing' TFile
                    else msg = msg NumberOfChanges 'changes applied'
        end
   When type=2 then do
        src = NumberOfChanges
        msg = msg NumberOfChanges 'changes applied'
        end
   Otherwise nop
 End
 say msg
 Exit src

/*      +-----------------------------------------------------------+
        | ONERROR [CONTINUE] [STOP] : what to do on syntax errors |
        +-----------------------------------------------------------+
*/
ONERROR:
 if wordpos(translate(Parms),'CONTINUE STOP')>0 then OnErr = translate(Parms)
    else rc=OnErrorDo(p,'On Error what ?')
 return
```

Figure 257. CUBE.CMD Source Code (Part 5 of 13)

```
/*      +---------------------------------------------------------+
        | WHEN     ... wordlist of when codes ...                 |
        +---------------------------------------------------------+
*/
APPLYWHEN:
 MWhen = strip(translate(Parms),'B')
 if MWhen = '*' then MWhen = Make
 return

/*      +---------------------------------------------------------+
        | CASE [SENSITIVE] [IGNORE] : string compare mode         |
        +---------------------------------------------------------+
*/
CASE:
 if wordpos(translate(Parms),'SENSITIVE IGNORE')>0 then CaseM =
translate(left(Parms,1))
    else rc=OnErrorDo(p,'Case what ?')
 return

/*      +---------------------------------------------------------+
        | LINEID [NOSTRIP] [STRIP "x"]                            |
        +---------------------------------------------------------+
*/
SLINEID:
 Select
   When word(translate(Parms),1) = 'NOSTRIP' then Lstrip = ""
   When word(translate(Parms),1) = 'STRIP' then do
      Parse var Parms 'STRIP' (_d_) ww (_d_)
      if length(ww) <> 1 then rc=OnErrorDo(p,'Strip leading what ?')
                         else Lstrip = ww
      end
   Otherwise rc=OnErrorDo(p,'Lineid what ?')
 end
 return
/*      +---------------------------------------------------------+
        | REPLINE lineid WITH replacement [( options]             |
        +---------------------------------------------------------+
*/
REPLINE:
  parse var Parms (_d_) Lineid (_d_)  'WITH' (_d_) With (_d_)  '(' Opt
  if Lineid = '' then do                         /* No line identifier      */
    rc=OnErrorDo(p,'Replace what line ?')
    return
  end
  if With = '' then do                           /* No replacement string   */
    rc=OnErrorDo(p,'Replace line with ?')        /*   process error         */
    return                                       /*   ignore command        */
  end
  With = Cmdrs(With)                             /* cmd substitution        */
  With = Envrs(With,Opt)                         /* env substitution if req */
  dir = Searchdir(opt)                           /* What target lines ?     */
  mod = Lidmod(opt)                              /* floating line id ?      */
select                                         /* What if no target lines?*/
   when wordpos('ADDTOP',Opt)>0 then after=0     /* add after line 0        */
   when wordpos('ADDBOTTOM',Opt)>0 then after=Target.0 /* add after last l. */
   when wordpos('DONTADD',Opt)>0 then after=-1   /* don't add               */
   otherwise after=-1                            /* don't add is the default*/
  end
```

Figure 258. CUBE.CMD Source Code (Part 6 of 13)

```
  if ififnot() then return                      /* Process only when       */
  where = Whereis(Lineid,dir,mod)               /* Get target lines numbers*/
  if where \= '' then do                        /* if target(s) found      */
    do until where = ''                         /*   process all targets   */
      parse var where w where                   /*     1 at a time         */
      was = Target.w                            /* save old value for log  */
      Target.w = With                           /*     target = replacmnt. */
      call logrep w,was,Target.w                /*     log action          */
      if dir \= 'A' then leave                  /*     quit if not ALL     */
    end
  end
  else if after>-1 then call insert after With  /* if no target, try add   */
  return
/*      +-------------------------------------------------------------+
        | ADDLINE     line  [( options]                               |
        +-------------------------------------------------------------+
*/
ADDLINE:
  parse var Parms (_d_) Line (_d_)  '(' Opt
  if Line = '' then do                          /* No line identifier      */
    rc=OnErrorDo(p,'Add what line ?')           /*    process error        */
    return                                      /*    ignore command       */
  end
  select                                        /* When to add ?           */
    when wordpos('IFNEW',Opt)>0 then always=0   /*   if not already there  */
    when wordpos('ALWAYS',Opt)>0 then always=1  /*   even if already there */
    otherwise always=0                          /*   IFNEW is the default  */
  end
  Line = Cmdrs(Line)                            /* cmd substitution        */
  Line = envrs(Line,Opt)                        /* env substitution if req */
  mod = Lidmod(opt)                             /* floating line id ?      */
  exist = Whereis(Line,'F',mod)                 /* If this line exists and */
  if exist \= '' & always = 0 then return       /* IFNEW , don't add !     */
  if ififnot() then return
select                                        /* Where to add ?          */
    when wordpos('AFTER',Opt)>0 then do;        /* 1) After a given line   */
      parse var Opt 'AFTER' (_d_) astr (_d_)  . /*    line identifier ?    */
      if astr = '' then after = Target.0        /*    no id = add bottom   */
      else after = Whereis(astr,'F',mod)        /*    else get # of 1st    */
      parse var after after .                   /*    line with this id.   */
      if after='' then do                       /*    no match found       */
        if wordpos('ONLY',Opt)>0 then after=-1  /*      if ONLY, don't add */
           else after=Target.0                  /*      else add bottom    */
      end
    end
when wordpos('BEFORE',Opt)>0 then do;       /* 2) Before a given line  */
      parse var Opt 'BEFORE' (_d_) bstr (_d_)  . /*    line identifier ?    */
      if bstr = '' then after = 0               /*    no id = add top      */
      else after = Whereis(bstr,'F',mod)        /*    else get # of 1st    */
      parse var after after .                   /*    line with this id.   */
      if after ='' then do                      /*    no match found       */
        if wordpos('ONLY',Opt)>0 then after=-1  /*      if ONLY don't add  */
           else after=0                         /*      else add top       */
      end
      else after=max(0,after-1)                 /*    match found          */
    end
    otherwise after=Target.0                    /* 3) default = add bottom */
  end
```

Figure 259. CUBE.CMD Source Code (Part 7 of 13)

```
 if after \= -1 then call insert after Line     /* add the line            */
  return

/*      +----------------------------------------------------------+
        | ADDSTRING string IN lineid [(Options]                    |
        +----------------------------------------------------------+
*/
ADDSTRING:
  parse var Parms (_d_) With (_d_)  'IN' (_d_) Lineid (_d_)   '(' Opt
  if Lineid = '' then do                         /* No line identifier      */
    rc=OnErrorDo(p,'Add string where ?')         /*    process error        */
    return                                       /*    ignore command       */
  end
  if With = '' then do                           /* No string to add        */
    rc=OnErrorDo(p,'Add what string ?')          /*    process error        */
    return                                       /*    ignore command       */
  end
  With = Cmdrs(With)                             /* cmd substitution        */
  With = Envrs(With,Opt)                         /* env substitution if req */
  dir=Searchdir(opt)                             /* Which target line ?     */
  mod=Lidmod(opt)                                /* floating line id ?      */
  where = Whereis(Lineid,dir,mod)                /* Select target(s)        */
  if where \= '' then do                         /*   if target found       */
    do until where = ''                          /*     process target(s)   */
      parse var where w where                    /*     1 at a time         */
      if CaseM = 'S' then do
         Tar= Target.w; Wi = With; end           /*     string compare mode */
      else do
  Tar = translate(Target.w); Wi=translate(With); end /* string compare mode */
      if pos(Wi,Tar) > 0 & ,                     /* String already there &  */
         wordpos('ALWAYS',Opt) = 0 then leave    /* ALWAYS not specified.   */

      select                                     /* Where to add ?          */
        when wordpos('AFTER',Opt)>0 then do      /* 1) After a given string */
          astr=''                                /*    defaulted to null    */
      parse var Opt 'AFTER' (_d_) astr (_d_)  .  /*    what is this string  */
          If CaseM = 'I' then astr=translate(astr)
          was = Target.w                         /*    save for logging     */
          if astr = '' | pos(astr,Tar)=0         /* if no string or no match*/
             then Target.w = Target.w || With    /*    add at end of target */
          else do
             parse var Tar xx (astr) rest        /* insert string after     */
           Target.w = xx || astr || With || rest /*   specified string      */
          end
          call logrep w,was,Target.w             /*   log action            */
        end
        when wordpos('BEFORE',Opt)>0 then do     /* 2) Before a given string*/
          bstr=''                                /*    defaulted to null    */
       parse var Opt 'BEFORE' (_d_) bstr (_d_)  . /*    what is this string  */
          If CaseM = 'I' then bstr=translate(bstr)
          was = Target.w                         /*    save for logging     */
          if bstr = '' | pos(bstr,Tar)=0         /* if no string or no match*/
          then do                                /* add at beginning        */
            If CaseM = 'I' then Lid=translate(Lineid)
            Parse var Tar (Lid) rest             /* but after identifier    */
            Target.w = Lineid || With || rest    /*                         */
          end
```

Figure 260. CUBE.CMD Source Code (Part 8 of 13)

```
          else do
             parse var Tar xx (bstr) rest        /* insert string before    */
            Target.w = xx || With || bstr || rest /*   specified string      */
          end
          call logrep w,was,Target.w             /*   log action            */
        end
        otherwise nop
      end
      if dir \= ’A’ then leave                   /* leave if not ALL targets*/
    end
  end
  else do                                        /* no target : add line ?  */
    if wordpos(’ADDTOP’,Opt)>0 then call insert 0 Lineid || With
    if wordpos(’ADDBOTTOM’,Opt)>0 then call insert Target.0 Lineid || With
  end
  return

/*      +-----------------------------------------------------------+
        | DELSTRING string [IN lineid] [(Options]                   |
        +-----------------------------------------------------------+
*/
DELSTRING:
  parse var Parms (_d_) What (_d_)  ’IN’ (_d_) Lineid (_d_)  ’(’ Opt
  if Lineid = ’’ then parse var Parms (_d_) What (_d_)  ’(’ Opt
  if What = ’’ then do                           /* No string to del        */
    rc=OnErrorDo(p,’Delete what string ?’)       /*    process error        */
    return                                       /*    ignore command       */
  end
  What = Cmdrs(What)                             /* cmd substitution        */
  What = envrs(What,Opt)                         /* env substitution if req */
  dir=Searchdir(opt)                             /* Which target line ?     */
  mod=Lidmod(opt)                                /* floating line id ?      */
  where = Whereis(Lineid,dir,mod)                /* Select target(s)        */
  if where \= ’’ then do                         /*   if target found       */
    do until where = ’’                          /*     process target(s)   */
      parse var where w where                    /*     1 at a time         */
      do forever                                 /* for all occurences      */
        if CaseM = ’S’ then do
           Tar = Target.w; Wh=What; end          /*     string compare mode */
        else do
 Tar = translate(Target.w); Wh=translate(What); end  /* string compare mode */
        if pos(Wh,Tar) > 0 then do               /*     String is there     */
           was = Target.w                        /*     save for logging    */
           parse var Tar xx (Wh) rest            /*     isolate string      */
           Target.w = xx || rest                 /*     delete string       */
           call logrep w,was,Target.w            /*     log action          */
        end
        else leave
      end
      if dir \= ’A’ then leave                   /* leave if not ALL targets*/
    end
  end
  return

/*      +-----------------------------------------------------------+
        | REPSTRING ostring [WITH nstring] [IN lineid] [(Options]   |
        +-----------------------------------------------------------+
*/
```

Figure 261. CUBE.CMD Source Code (Part 9 of 13)

```
REPSTRING:
  parse var Parms (_d_) Ostr (_d_)  'WITH' (_d_) With (_d_)  'IN' (_d_) Lineid (_d_)
'(' Opt
  if Ostr = '' then do                         /* No old string specif.   */
    rc=OnErrorDo(p,'Replace what string ?')      /*    process error        */
    return                                     /*    ignore command       */
  end
  if With = '' then parse var Parms (_d_) Ostr (_d_)  'IN' (_d_) Lineid (_d_)  '(' Opt
  if Lineid = '' then parse var Parms (_d_) Ostr (_d_)  '(' Opt
  if With = '' then With = Ostr                /* No rep string specif.   */
  With = Cmdrs(With)                           /* cmd substitution        */
  With = envrs(With,Opt)                       /* env substitution if req.*/
  dir=Searchdir(opt)                           /* Which target line ?     */
  mod=Lidmod(opt)                              /* floating line id ?      */
  where = Whereis(Lineid,dir,mod)              /* Select target(s)        */
  do while where \= ''                         /*   if target found       */
      parse var where w where                  /*     1 at a time         */
      was = Target.w
      newtar = ''
      do forever                               /* for all occurences      */
        if CaseM = 'S' then do
           Tar = Target.w; Os=Ostr; end        /*     string compare mode */
        else do
 Tar = translate(Target.w); Os=translate(Ostr); end  /* string compare mode */
        if pos(Os,Tar) > 0 then do             /*     String is there     */
           parse var Tar xx (Os) rest          /*     isolate string      */
           newtar = newtar || xx || With       /*     replace occurrence  */
           Target.w = rest                     /*     next                */
        end
        else leave
      end
      Target.w = newtar || target.w
      call logrep w,was,Target.w               /* log action              */
      if dir \= 'A' then leave                 /* leave if not ALL targets*/
  end
  return

/*      +----------------------------------------------------------+
        | COMMENTLINE lineid WITH type [(options ]                 |
        +----------------------------------------------------------+
*/
COMMENTL:
  parse var Parms (_d_) Lineid (_d_)  'WITH' (_d_) cmnt (_d_)  '(' Opt
  if Lineid = '' then do                       /* No identifier           */
    rc=OnErrorDo(p,'Comment what ?')             /*    process error        */
    return                                     /*    ignore command       */
  end
  if cmnt = '' then do                         /* No comment string       */
    rc=OnErrorDo(p,'Comment how ?')              /*    process error        */
    return                                     /*    ignore command       */
  end
  dir=Searchdir(opt)                           /* Which target lines ?    */
  mod=Lidmod(opt)                              /* floating line id ?      */
  if ififnot() then return
  where= whereis(Lineid,dir,mod)               /* get target lines #s     */
if where \= '' then do                       /* if match(es) found      */
    do until where = ''                        /*   process target(s)     */
      parse var where w where                  /*   1 at a time           */
```

Figure 262. CUBE.CMD Source Code (Part 10 of 13)

```
      was = Target.w                             /*  save for logging       */
      Target.w = cmnt Target.w                   /*   comment target        */
      call logrep w,was,Target.w                 /*   log action            */
      if dir \= 'A' then leave                   /* leave if not ALL targets*/
    end
  end
  return

/*      +--------------------------------------------------------+
        | DELLINE lineid [(options ]                             |
        +--------------------------------------------------------+
*/
DELLINE:
  parse var Parms (_d_) Lineid (_d_)   '(' Opt
  if Lineid = '' then do                         /* No identifier           */
    rc=OnErrorDo(p,'Delete what line ?')         /*    process error        */
    return                                       /*    ignore command       */
  end
  Lineid = Cmdrs(Lineid)                         /* cmd substitution        */
  Lineid = envrs(Lineid,Opt)                     /* env substitution if req */
  dir=Searchdir(opt)                             /* Which target line(s) ?  */
  mod=Lidmod(opt)                                /* floating line id ?      */
  if ififnot() then return
  where= whereis(Lineid,dir,mod)                 /* Get target lin(s) #s    */
  if where \= '' then do                         /* if match(es) found      */
    do until where = ''                          /*    process all targets  */
      parse var where w where                    /*    one at a time        */
      say 'Deleted line' w
      say '  was:' Target.w
      NumberOfChanges = NumberOfChanges + 1
      Target.w = '       '               /*    mark for delete      */
      if dir \= 'A' then leave                   /* leave if not ALL targets*/
    end
  end
  return

/*      +--------------------------------------------------------+
        | SEARCHDIR: Direction for line search in Target File    |
        +--------------------------------------------------------+
*/
Searchdir: procedure
  select                                         /* What target lines ?     */
    when wordpos('LAST',arg(1))>0 then dir='L'   /* set reverse search      */
    when wordpos('FIRST',arg(1))>0 then dir='F'  /* set forward search      */
    otherwise dir='A'                            /* default is all lines    */
  end
  return dir

/*      +--------------------------------------------------------+
        | LIDMOD: Identify line at 1st col or anywhere in line    |
        +--------------------------------------------------------+
*/
Lidmod: procedure
  if wordpos('*ID',arg(1))>0 then return(0)
                             else return(1)
```

Figure 263. CUBE.CMD Source Code (Part 11 of 13)

```
/*      +-----------------------------------------------------------+
        | Stream functions (close & exists)                         |
        +-----------------------------------------------------------+
*/
close:
  return stream(arg(1),'C','CLOSE')

exists:
  return stream(arg(1),'C','QUERY EXISTS')

/*      +-----------------------------------------------------------+
        | get substitution variables and their values in cmdline. |
        | ( RS(#aaa#=aaa #bbb#=bbb)                                 |
        |Output: vn.i = number and names of subst. variables        |
        |        vv.i = values of these variables                   |
        |Adapted from Walter Pachl's                                |
        +-----------------------------------------------------------+
*/
get_cmdrs: procedure expose vn. vv.
  parse arg Opt
  vn.0 = 0
  p = pos('RS(',translate(Opt))
  if p > 0 then do
    rs = substr(Opt,p+3)
    Parse var rs rs ')'
    Do i=1 By 1 While rs<>''
      rs=strip(rs,'L')
      Parse Var rs vn '=' vv rs
      if left(vn,1) = right(vn,1) then do
        vn.i = vn
        vv.i = vv
      end
      else call Exit 0 'Invalid substitution variables ('vn vv')'
    End
    vn.0=i-1
  end
  Return

/*      +-----------------------------------------------------------+
        | Log a change in a target line                             |
        +-----------------------------------------------------------+
*/
logrep: procedure expose NumberOfChanges
parse arg ln,old,new
if old<>new then do
  say 'Changed line' ln
  say '  old: "'old'"'
  say '  new: "'new'"'
  NumberOfChanges = NumberOfChanges + 1
end
return

/*      +-----------------------------------------------------------+
        | Process IF/IFNOT logic for Lines commands                 |
        |    returns 1 if IF/IFNOT condition is false !!            |
        |    returns 0 otherwise                                    |
        +-----------------------------------------------------------+
*/
```

Figure 264. CUBE.CMD Source Code (Part 12 of 13)

```
ififnot:
if wordpos('IFNOT',Opt)>0 Then Do              /* Process only when       */
  parse var Opt 'IFNOT' (_d_) istr (_d_) .     /* another line doesn't    */
  iexist = Whereis(istr,'F',mod)               /* exists                  */
  if iexist \= '' then return(1)               /*                         */
end                                            /*                         */
if wordpos('IF',Opt)>0 Then Do                 /* Process only when       */
  parse var Opt 'IF' (_d_) istr (_d_) .        /* another line exists     */
  iexist = Whereis(istr,'F',mod)               /*                         */
  if iexist  = '' then return(1)               /*                         */
end                                            /*                         */
return(0)
```

Figure 265. CUBE.CMD Source Code (Part 13 of 13)

Appendix E. Special Notices

This redbook describes the CID (Configuration, Installation and Distribution) enhancements developed by the NCSD (Network Computing Software Division) Rapid Deployment Team (RDT) in Austin, Texas. It is also a handbook that provides step-by-step guidance in all phases of the usage of these new RDT Tools, such as Version-to-Version tools and Tribble Tools. Tribble Tools are also known as Enterprise Rescue Tools.

The new CID software distribution method, which is a sophisticated cloning technique, allows you to migrate from previous OS/2 versions, such as OS/2 V2.1x and OS/2 Warp 3, to OS/2 Warp 4, as well as to accomplish pristine installation scenarios. We use these tools in an OS/2 LAN environment with NetView DM/2, our software distribution manager of choice.

This document is intended for workstation specialists and system technical personnel responsible for mass distribution of OS/2 products in an OS/2 LAN. Some knowledge of LAN redirection principles and TCP/IP is assumed. This redbook solely focuses on the new RDT tools. If traditional CID information is needed or information about software distribution techniques other than those provided by the new RDT tools, you need to obtain the redbooks titled *The OS/2 Warp 4 CID Software Distribution Guide*, SG24-2010 and *OS/2 Installation Techniques: The CID Guide*, SG24-4295. The latter book concentrates on previous versions of OS/2 Warp 4.

This redbook provides a broad understanding of new software distribution features introduced with the Rapid Deployment Tools that were not previously available.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM

Appendix F. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 International Technical Support Organization Publications

For information on ordering these ITSO publications, see NetView.

- *The OS/2 Warp 4 CID Software Distribution Guide,* SG24-2010
- *OS/2 Installation Techniques: The CID Guide*, SG24-4295
- *Network Clients for OS/2 Warp Server: OS/2 Warp 4, DOS/Windows, Windows 95/NT, and Apple Macintosh*, SG24-2009
- *OS/2 Warp Server, Windows NT, and NetWare*: A Network Operating System Study, SG24-4786
- *Inside OS/2 Warp Server, Volume 1: Exploring the Core Components,* SG24-4602
- *Inside OS/2 Warp Server, Volume 2: System Management, Backup/Restore, Advanced Print Services,* SG24-4702
- *OS/2 Warp Server Functional Enhancements: Part 1,* SG24-2008
- *OS/2 Warp Server Functional Enhancements: Part 2,* SG24-2031 (in press)
- *Getting to Know OS/2 Warp 4,* SG24-4758
- *Prepare for OS/2 Engineer Certification,* SG24-4869

F.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	**Subscription Number**	**Collection Kit Number**
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041

CD-ROM Title	Subscription Number	Collection Kit Number
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
RS/6000 Redbooks Collection (PDF)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

F.3 Other Publications

These publications are also relevant as further information sources:

- *NetView Distribution Manager /2 Version 2.1: Change Distribution Manager User's Guide,* SH19-5048
- *Getting to Know OS/2 Warp 4*, ISBN 0-13-842147-1
- *Prepare for OS/2 Engineer Certification*, ISBN 9-655-61119-1
- *CID Enablement Guidelines*, S10H-9666

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** – to order hardcopies in United States
- **GOPHER link to the Internet** – type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks, type the following command::

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) – send orders to:

	IBMMAIL	**Internet**
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** – send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** – send orders to:

United States (toll free)	1-800-445-9269
Canada	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** – send note to `softwareshop@vnet.ibm.com`

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

ITSO Redbook Evaluation

The OS/2 Warp 4 CID Rapid Deployment Tools
SG24-2012-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

The OS/2 Warp 4 CID Rapid Deployment Tools Migration and Installation Scenarios **SG24-2012-00**

SG24-2012-00
Printed in the U.S.A.