z/OS

Distributed File Service SMB Administration

SC24-5918-04

z/OS

Distributed File Service SMB Administration

SC24-5918-04

Note!
Before using this information and the product it supports, read the information in Appendix G, “Notices” on page 161.

Fifth Edition (September 2002)

This document is a complete revision of SC24-5918-03.

This edition applies to Version 1 Release 4 of z/OS Distributed File Service (program number 5694-A01), Version 1 Release 4 of z/OS.e (program number 5655-G52), and to all subsequent releases until otherwise indicated in new editions.

IBM® welcomes your comments. You may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zosqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document.
- Page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this document

The purpose of this document is to provide complete and detailed guidance and reference information. This information is used by system and network administrators working with the Server Message Block (SMB)[1] support of the IBM z/OS Distributed File Service base element of z/OS and z/OS.e.

Distributed File Service includes an SMB function that is based on the X/Open SMB Version 2 specification and the IETF RFCs on Netbios over IP (RFC1001 and RFC1002).

The Distributed File Service base element supports both Distributed Computing Environment (DCE) DFS file protocols and SMB file/print protocols. This document focuses on the SMB support of z/OS Distributed File Service. If you are using only SMB protocols, then you should use this document. If you are using DCE DFS protocols, then you need to refer to the z/OS Distributed File Service DFS Administration. If you are using both protocols (DCE DFS and SMB), you may need to refer to both documents.

Who should use this document

This document is intended for users and network and system administrators who understand the basic concepts of data communications. A knowledge of Transmission Control Protocol/Internet Protocol (TCP/IP) communications, z/OS UNIX operating system concepts, and Microsoft Windows (referred to as Windows throughout this document) operating system concepts can help you use this guide more effectively.

How this document is organized

The information in this document is divided into the following parts, each part divided into chapters.

Part 1, “SMB support administration guide” on page 1 discusses guidance information. The chapters in that part begin with a short introduction and are followed by detailed information.

Part 2, “SMB support reference” on page 71 discusses reference information.

Conventions used in this document

This document uses the following typographic conventions

Bold	**Bold** words or characters represent system elements that you must enter into the system literally, such as commands.
Italic	Italicized words or characters represent values for variables that you must supply.
`Example Font`	Examples and information displayed by the system are printed using an example font that is a `constant width typeface`.
[ ]	Optional items found in format and syntax descriptions are enclosed in brackets.
{ }	A list from which you choose an item found in format and syntax descriptions are enclosed by braces.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the name of a key on a keyboard.
...	Horizontal ellipsis points indicated that you can repeat the preceding item one or more times.

1. Server Message Block (SMB) is a protocol for remote file/print access used by Microsoft Windows clients. This protocol is also known as Common Internet File System (CIFS).

**** A backslash is used as a continuation character when entering commands from the shell that exceed one line (255 characters). If the command exceeds one line, use the backslash character \ as the last non-blank character on the line to be continued, and continue the command on the next line.

Note: When you enter a command from this document that uses the backslash character (\) make sure you immediately press the Enter key and then continue with the rest of the command. In most cases, the backslash has been positioned for ease of readability.

A pound sign is used to indicate a command is entered from the shell, specifically where **root** authority is needed (**root** refers to a user with a **UID = 0**).

This document used the following keying convention:

<Return> The notation **<Return>** refers to the key on your terminal or workstation that is labeled with either the word “Return” or “Enter”, with a left arrow.

Entering commands
When instructed to enter a command, type the command name and then press **<Return>**.

Where to find more information

Where necessary, this document references information in other documents. For complete titles and order numbers for all elements of z/OS, refer to the *z/OS: Information Roadmap*, SA22-7500.

For information about installing Distributed File Service components, refer to the *z/OS: Program Directory*, GI10-0669.

Information concerning Distributed File Service-related messages can be found in the *z/OS: Distributed File Service Messages and Codes* document.

Softcopy publications

The z/OS Distributed File Service library is available on a CD-ROM, *z/OS Collection*, SK3T-4269. The CD-ROM online library collection is a set of unlicensed documents for z/OS and related products that includes the IBM Library Reader™. This is a program that enables you to view the BookManager® files. This CD-ROM also contains the Portable Document Format (PDF) files. You can view or print these files with the Adobe Acrobat reader.

Internet sources

The softcopy z/OS publications are also available for web-browsing and for viewing or printing PDFs using the following URL:

http://www.ibm.com/servers/eserver/zseries/zos/bkserv/

You can also provide comments about this document and any other z/OS documentation by visiting that URL. Your feedback is important in helping to provide the most accurate and high-quality information.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because, in most cases, LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/

or from anywhere in z/OS where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX Systems Services running OMVS). You can also download code from the (SK3T-4269) and the LookAt Web site that will allow you to access LookAt from a handheld computer (Palm PilotVIIx suggested).

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your (SK3T-4569) or from the **News** section on the LookAt Web site.

Some messages have information in more than one document. For those messages, LookAt displays a list of documents in which the message appears.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format

at the IBM Resource Link™ Web site at:

http://www.ibm.com/servers/resourcelink

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

http://www.ibm.com/servers/resourcelink

To register for access to the z/OS licensed documents:
1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this document, send your comments by using Resource Link at http://www.ibm.com/servers/resourcelink. Select Feedback on the Navigation bar on the left. Be sure to include the name of the document, the form number of the document, the version of the document, if applicable, and the specific location of the text you are commenting on (for example, a page number or table number.)

Summary of Changes

Summary of Changes
for SC24-5918-04
z/OS Version 1 Release 4

This document contains information previously presented in *z/OS Distributed File Service SMB Administration*, SC24-5918-03, which supports z/OS Version 1 Release 3 and subsequent releases.

The following summarizes the changes to that information.

New Information:

- Information is added to indicate this document supports z/OS.e.

Changed Information:

- The **smbidmap** file supports an asterisk for the SMB user ID on a Domain basis.
- Updates to the preface sections entitled ″Accessing z/OS licensed documents on the Internet″ and ″Using LookAt to look up message explanations″.

This document includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You may notice changes in the style and structure of some content in this document— for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of Changes
for SC24-5918-03
z/OS Version 1 Release 3

This document contains information previously presented in *z/OS Distributed File Service SMB Administration*, SC24-5918-02, which supports z/OS Version 1 Release 2 and subsequent releases.

The following summarizes the changes to that information.

New Information:

- The SMB server supports ACLs. That is, it reflects authorizations based on ACLs.
- An appendix with z/OS product accessibility information has been added.

This document includes terminology, maintenance, and editorial changes.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document— for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of Changes
for SC24-5918-02
z/OS Version 1 Release 2

This document contains information previously presented in SC24-5918-01, which supports z/OS Version 1 Release 1.

New Information:

- References are made to the zSeries File System (ZFS). For more information on the ZFS file system, refer to the *z/OS: Distributed File Service zSeries File System Administration* document.

Summary of Changes
for SC24-5918-01
z/OS Version 1 Release 2

This document contains information previously presented in Version 1 Release 1 of the *z/OS Distributed File Service SMB Administration*, SC24-5918-00.

New Information:

- The SMB server supports the ability to use a Windows NT or Windows 2000 Domain Controller for authentication.
- The SMB server supports the ability to recognize file tags when Enhanced ASCII is enabled in the SMB server.
- The SMB server supports absolute symbolic links.
- The SMB server supports the PC end of line characters (carriage return/line feed) in the SMB server configuration files.
- The SMB server can export non-owned Shared HFS file systems by attempting to move the ownership to the system the SMB server is running on.

This document includes terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of Changes
for SC24-5918-00

This document provides the following new information.

New Information:

- The SMB server supports Microsoft Windows 2000 Professional as an SMB client.
- The SMB server supports HFS automounted file systems by using a new capability called dynamic export.
- Permissions used when creating a new file or directory by using the SMB server can now be specified on a shared directory basis. They can be specified in the **smbtab**.
- Encrypted password support can be used without requiring OCSF when encryption hardware is not used.

Part 1. SMB support administration guide

This part of the document discusses guidance information for system administrators and Personal Computer (PC) users. Specifically, Chapter 1 is intended for PC users and system administrators, Chapters 2 through 8 are intended for system administrators, and Chapters 9 through 11 are for PC users.

Chapter 1. An overview of SMB support

The Distributed File Service Server Message Block (SMB)[2] support provides a server that makes Hierarchical File System (HFS) files and data sets available to SMB clients. The data sets supported include sequential data sets (on DASD)[3], partitioned data sets (PDS), partitioned data sets extended (PDSE) and Virtual Storage Access Method (VSAM) data sets. The data set support is usually referred to as Record File System (RFS) support. The SMB protocol is supported through the use of TCP/IP on z/OS. This communication protocol allows clients to access shared directory paths and shared printers. Personal Computer (PC) clients on the network use the file and print sharing functions that are included in their operating systems. Supported SMB clients include Microsoft Windows 98, Windows NT 4.0 Workstation, and Windows 2000 Professional. At the same time, these files can be shared with local z/OS UNIX applications and with DCE DFS clients.

Note: Throughout this document, references are made to HFS. Unless otherwise stated, HFS is a generic reference that includes HFS, ZFS, TFS, and AUTOMNT file system data. If you are in a sysplex with Shared HFS, SMB support of ZFS is limited to ZFS compatibility mode file systems.

In addition, Windows SMB clients can make remote print requests to z/OS printers that are connected to the Infoprint® Server for z/OS (OS/390® Version 2 Release 8 or later).

SMB support features

PC users work with files on their computers. Files are used to store data, programs, and other information. Files are stored on a disk on the computer. There may be several disks on the computer. Each of these disks are referred to by a different drive letter (for example, A:, B:, C:, D:, etc.). PC Users can read or write files on different disks on their computer by using the appropriate drive letter in the file name (for example, D:\dir1\file1).

PC users also work with printers attached to their computers. When a print request is made, the PC user can choose which printer the print request should go to.

Disclaimer

The z/OS SMB server supports basic file and print serving. It does not necessarily support all functions that a Windows file server supports. For example, the z/OS SMB server does not support Kerberos authentication or Dfs. There may be other functions that are not supported.

SMB support allows PC users to be able to access files that reside on a z/OS system remotely. That is, PC users can access files that are not located on their computer. Remote files simply appear to the PC user on one or more separate drive letters. PC users can “connect” an unused drive letter to a “shared resource” on a remote computer. This is sometimes referred to as “mapping a network drive”. This capability is provided by software that resides on the PC (the client), in combination with software that resides on the remote computer (the server). There must also be a TCP/IP network connection between the PC and the remote computer.

In addition, SMB support allows Windows PC users to be able to use remote printers that are attached to a z/OS system. Remote printers simply appear to be additional printers that are available to the PC user. Remote printers are installed on PCs using existing commands or install utilities.

2. Server Message Block (SMB) is a protocol for remote file/print access used by Windows clients. This protocol is also known as Common Internet File System (CIFS).

3. Direct Access Storage Device

SMB processes

SMB support provides a server process that makes file data and printers available to PC users. It allows an administrator to define shared directories and shared printers. It also handles PC requests to connect to the server process to satisfy file or print requests.

Another SMB process is the control process (also known as the DFS Control Task). It oversees the server process. When SMB support is started, it is really the DFS Control Task that is started. The DFS Control Task, in turn, starts the server process. If the server process ends abnormally for some reason, the DFS Control Task can automatically restart the server process.

Shared directories

In order to allow PCs to access remote files (located on a z/OS system), one or more shared directories must be created on the z/OS system. Distributed File Service administrators make files available to SMB clients by creating shared directories. A shared directory is given a share name and specifies a directory path name in a file system. Any directory can be shared with clients. To access shared directories from a PC, clients can map a network drive by choosing an available drive letter and mapping it to a computer name and a share name, or they can use Universal Naming Convention (UNC) mapping. (Refer to Chapter 10, “Accessing data” on page 61, for more information on UNC mapping.) The computer name is the name of the Distributed File Service SMB server and the share name is the name of the shared directory created on that server. After this is done, the remote files can be read or written as though they were local files.

Shared printers

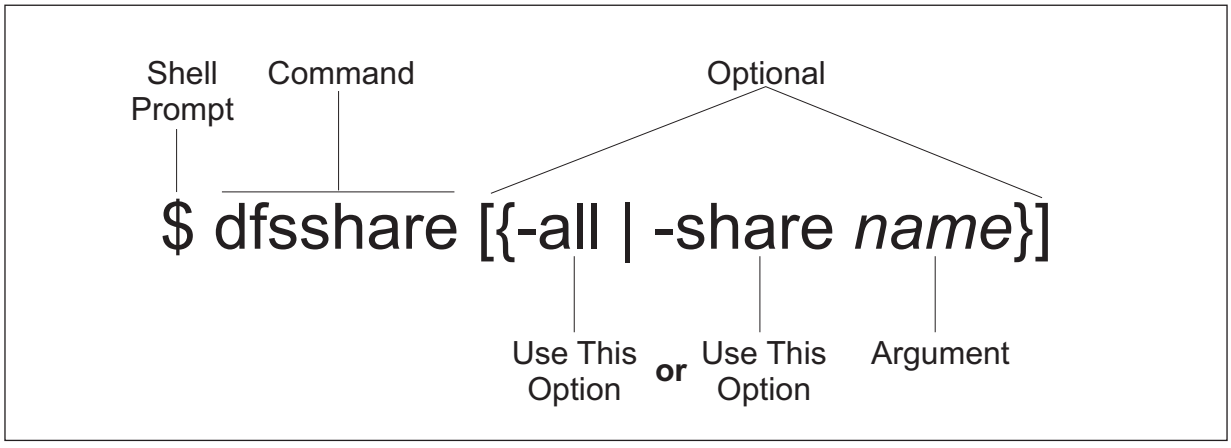
Distributed File Service administrators make z/OS printers available to Windows PCs by creating shared printers. A shared printer is given a share name and specifies a z/OS Infoprint Server printer definition name. To access a remote printer from a Windows PC, clients can install a remote printer or they can use commands. After this is done, the remote printers can be used as though they were local printers.

Command structure and help

The SMB commands share a similar structure. The following example shows the basic format of an SMB command:

```
$ command [{-option1 | -option2 argument}]
```

The following example illustrates the elements of an SMB command:



The following list summarizes the elements of an SMB command:

Command — A command consists of the command name. This name directs the server process or program to perform a specific action. The command name always appears in bold font.

Options — Command options appear in bold font, are always preceded by a **-** (dash), and are often followed by arguments. In the previous example, **-all** and **-share** are options, and *name* is the argument. The { | } (braces separated by a vertical bar) indicate that you can enter only one of two possible options.

An option and its arguments tell the server process or program which entities to manipulate when executing the command (for example, the name to assign to the shared directory or printer). In general, you should provide the options for a command in the order presented in the documentation.

Arguments — Arguments for options always appear in italic font.

Optional Information — Some commands can have optional, as well as required, options. Optional information is enclosed in [ ] (brackets). The **-all** and the **-share** with its *name* argument in the previous example are optional.

Enter each SMB command and its options and arguments on a single line followed by a carriage return at the end of the line. Use a space to separate each element (command name, options, and arguments) on the command line. Also use spaces to separate multiple arguments. Do not use a space to separate an option from its - (dash).

Command shortcuts

When supplying an argument (such as *name* in the previous example), you can omit the option (such as **-share** in the example) associated with the argument if:

- All arguments supplied with the command are entered in the order in which they appear in the command's syntax. The syntax for each command is presented with its description in Part 2, "SMB support reference" on page 71.
- Arguments are supplied for all options that precede the option to be omitted.
- All options that precede the option to be omitted accept only a single argument.
- No options, either those that accept an argument or those that do not, are supplied before the option to be omitted.

In the case where two options are presented in { | } (braces separated by a vertical line), the option associated with the first argument can be omitted if that argument is provided; however, the option associated with the second argument is required if that argument is provided.

If you must provide an option, you can abbreviate it to the shortest possible form that distinguishes it from other options of the command. For example, the **-share** option can typically be omitted or abbreviated to be simply **-s**.

The following example illustrates three acceptable ways to enter the same **dfsshare** command:

- Complete command: $ **dfsshare -share** *name*
- Abbreviated command name and abbreviated option: $ **dfsshare -s** *name*
- Abbreviated command name and omitted option: $ **dfsshare** *name*

Receiving help

You can receive help on the SMB commands by using the **-help** option:

$ **command -help**

Chapter 2. Considerations for a new SMB release

If you have an earlier release of the Distributed File Service installed on your system and you plan to install this release of the Distributed File Service to use the SMB support, there are some important items to consider. This chapter contains Distributed File Service release migration considerations for SMB in the following sections:

New release considerations

When migrating to a new release of Distributed File Service, it is not necessary to copy and save your customized files prior to installing. The customizable files are listed in Appendix D, "Customizable files" on page 141.

Some of the customizable files listed may not be present on your system when installing a new release of Distributed File Service (unless you are reinstalling this version) because the files are new for this release. Most customizable files that do not exist are created as part of Distributed File Service post installation by the **dfs_cpfiles** program from sample files provided in the **/opt/dfsglobal/examples** directory. There are several files that you need to create (if this is your first installation of Distributed File Service) in order to specify the HFS data and shared printers to be made available to PC clients. These include the **smbtab**, **dfstab**, and the **devtab** files. Also, the **smbidmap** file must be created to map PC user IDs to z/OS user IDs.

To install a new release of Distributed File Service:

1. DCE DFS Considerations

   If you previously used DCE DFS, refer to the *z/OS: Distributed File Service DFS Customization* document, "DFS Migration Considerations" chapter. It is recommended that you migrate the DFS in a DCE environment before completing any additional steps to migrate the SMB support to the new release.

   If you have already migrated DFS to the new release and you want to activate SMB support for the first time, you should follow the "SMB file/print server installation and configuration steps" on page 11.

2. Symbolic Link Considerations

   Prior to OS/390 Version 2 Release 6, all Distributed File Service customizable files resided in a separate HFS data set mounted at **/usr/lpp/dfs/local**. Starting in OS/390 Version 2 Release 6, the Distributed File Service customizable files reside in the path **/etc/dfs**. During installation, symbolic links are deleted and recreated to link to files in **/etc/dfs**.

   If you have replaced any Distributed File Service symbolic links, they are deleted during installation. You should save the file data before installing the Distributed File Service. These files are listed in the *z/OS: Distributed File Service DFS Customization* document, "Directories and Files" appendix.

3. Installing the Distributed File Service

   If you have not already done so, install this release of Distributed File Service by referring to the instructions in the *z/OS: Program Directory*, GI10-0669.

   Note: The following instructions assume that you have copied the preexisting **/etc** file system for use on the target image of the release installation as recommended in the *z/OS: Program Directory*, GI10-0669.

4. Stopping the Distributed File Service server

   If the Distributed File Service server is running on the target image, stop the server at this time using the operator command **stop dfs**. Refer to Chapter 4, "Managing SMB processes" on page 17 for more information.

5. RACF® Database Considerations

If you are installing into a target image that uses the same RACF database as the production image, the preexisting SMB configuration files that were included when the production **/etc** file system was copied by using the instructions in the *z/OS: Program Directory*, GI10-0669, do not need to be modified to change the RACF user IDs. You need to only review and update the target image copy of the configuration files with optional parameters new for this release later in the migration process.

If the target image uses a different RACF database, the **smbidmap** file defined by the **_IOE_SMB_IDMAP** environment variable in the **/opt/dfslocal/home/dfskern/envar** file must be updated to define the correct user identifiers. Refer to Chapter 6, “Mapping SMB user IDs to z/OS user IDs” on page 29 for more details. You may also need to review the **_IOE_MVS_DFSDFLT** environment variable in the **/opt/dfslocal/home/dfskern/envar** file. Refer to “Logon considerations” on page 48 for more details.

6. Exported Data Considerations

   If you are installing into a target image that can access the same exported file data as the production image, the preexisting **smbtab**, **dfstab**, and **devtab** files in the **/opt/dfslocal/var/dfs** directory can be used.

   If the target image cannot access the same exported file data as the production image, or if you do not want to export the production data files during the testing of the new z/OS release on the target image, you must update these files to export and share only test data.

7. Printer Considerations

   If you are installing into a target image that can access the same printers as the production image, the preexisting **smbtab** file in the **/opt/dfslocal/var/dfs** directory can be used.

   If the target image cannot access the same printers as the production image, or if you do not want to share the production printers during the testing of the new z/OS release on the target image, you must update this file to create different shared printers.

8. Running the **/opt/dfsglobal/scripts/dfs_cpfiles** program

   Some of the customizable files listed in Appendix D, “Customizable files” on page 141, may not exist in the **/etc/dfs** file system on the target image when installing a new release of the Distributed File Service (unless you are reinstalling this release). Customizable files that do not exist are created as part of post installation processing by the **dfs_cpfiles** program from the sample files in the **/opt/dfsglobal/examples** directory.

   If you have not already run the **dfs_cpfiles** program per the instructions in the *z/OS: Program Directory*, GI10-0669, you should run it now against the target image **/etc/dfs** to ensure that all new configuration files for the new release are created.

   More information on the **dfs_cpfiles** program can be found in Chapter 3, “Post installation processing” on page 11.

   Note: The **dfs_cpfiles** program does not create all the files necessary for SMB support. Refer to the section “SMB configuration file considerations” on page 9 for more information.

9. Updating Preexisting Configuration Files and Server Startup Parameters

   Compare your target system customizable files with the example **ioepdcf** and **envar** files in the **/opt/dfsglobal/examples** directory for this release to determine if any new parameters or variables for this release are applicable to your system.

10. Authorized Program Considerations

    With OS/390 Version 2 Release 9, the list of authorized programs for the Distributed File Service is:
    - IOEGRWAG
    - IOENEWAG
    - IOESALVG
    - SMBPW.

    Refer to the *z/OS: Program Directory*, GI10-0669, for a description of the PARMLIB member updates required for the member IKJTSOxx.

11. SMB File/Print Considerations

For the optional SMB support, the **_IOE_PROTOCOL_SMB** environment variable in the **/opt/dfslocal/home/dfskern/envar** file for the **dfskern** process must be updated to specify **_IOE_PROTOCOL_SMB=ON**.

Insure that the **LIBPATH** environment variable in the **/opt/dfslocal/home/dfskern/envar** file is updated. If SMB print serving support is used, specify the **LIBPATH** environment variable to identify the z/OS Infoprint Server library. If the SMB support for encrypted passwords is used, update the **LIBPATH** to include the path **/usr/lib** so the OCSF library can be found. For example, the envar can specify **LIBPATH=/usr/lpp/Printsrv/lib:/usr/lib**.

The **smbtab**, **dfstab**, and **devtab** files must be created if they do not exist and updated to specify the HFS data to be made available to PC clients. The **smbtab** file must be created and updated to specify the shared printers to be made available to PC clients. These files reside in the **/opt/dfslocal/var/dfs** directory. Refer to Chapter 7, “Sharing files” on page 33 and Chapter 8, “Sharing printers” on page 53 for more information on these topics.

SMB configuration file considerations

Certain SMB configuration files contain system specific information. These HFS files, if copied to another system, must be modified to contain or point to data on that system. In particular, the **smbtab**, **dfstab**, and **devtab** files point to the data that is to be made available to PC clients. The **smbtab** may also refer to shared printers that are to be made available to PC clients. The **smbidmap** file and the **_IOE_MVS_DFSDFLT dfskern** environment variable both contain z/OS user IDs that may need to be changed. Other **dfskern** environment variables that may need to be changed include:

- **_IOE_SMB_COMPUTER_NAME**
- **_IOE_SMB_DOMAIN_NAME**
- **_IOE_SMB_IDMAP**
- **_IOE_SMB_PRIMARY_WINS**
- **_IOE_SMB_SECONDARY_WINS**
- **_IOE_SMB_WINS_PROXY**.

The **dfs_cpfiles** program is run during Distributed File Service installation. The **dfs_cpfiles** program creates (**not replaces**) certain customizable files with default information if the file does not exist. The files that are copied are listed in Chapter 3, “Post installation processing” on page 11.

Chapter 3. Post installation processing

The SMB File/Print Server function is part of the Distributed File Service base element of z/OS. Before using the SMB support, you must install the z/OS release, the Distributed File Service and the other base elements of z/OS using the appropriate release documentation, the *z/OS: Program Directory*, GI10-0669, or the *ServerPac: Installing Your Order*. During the installation of the z/OS release, you must perform actions to activate the SMB support.

Note: If you are only using the SMB File/Print Server support in the Distributed File Service (and not the DCE DFS support), DCE does not need to be configured and started. DCE only needs to be installed.

To use the SMB File/Print support, you may also need to install and configure the Open Cryptographic Service Facility (OCSF) or the Infoprint Server Feature of z/OS.

- If you plan to use encrypted passwords and you want to exploit hardware encryption, you need to install and configure the Open Cryptographic Services Facility (OCSF) base component of the Cryptographic Services element. If you plan to use password encryption and you do not want to use hardware encryption, you do not need to install and configure OCSF and you should set the **dfskern** environment variable **_IOE_SMB_OCSF=OFF**.
- If you plan to use the SMB print serving support, you need to install and configure the Infoprint Server feature.

If you are migrating from a prior release of Distributed File Service and you want to use the SMB function, please review Chapter 2, “Considerations for a new SMB release” on page 7.

To use the SMB support, you must configure the support on the system. Configuration includes administrative actions such as:

- Activating the SMB file/print serving function
- Defining SMB administrators
- Specifying optional DFS process options
- Defining SMB users
- Defining HFS and RFS data sets to export and directories to share for access by SMB clients
- Defining printers to share for access by SMB clients
- Updating SMB client PC machines running Windows, Windows NT, or other operating systems that issue requests to file/print servers using the Server Message Block (SMB) protocol.

This chapter contains the information to assist and guide you in completing the installation and configuration of the Distributed File Service SMB File/Print Server support.

SMB file/print server installation and configuration steps

To install, configure, and access the Distributed File Service server (**dfskern**) for SMB File/Print Server operation, you must perform the following administrative steps:

1. Install and perform post-installation of the Distributed File Service by following the applicable instructions in the *z/OS: Program Directory*, GI10-0669, or the *ServerPac: Installing Your Order*.

   If you plan to use encrypted passwords (recommended) and optionally, you want to exploit OCSF and hardware encryption, you must ensure that the proper authorizations have been given to the DFS server user ID to use OCSF services. Refer to the “Cryptographic Services OCSF Customization Considerations” section in the *z/OS: Program Directory*, GI10-0669, and the “Configuring and Getting Started” section in the *z/OS: Open Cryptographic Services Facility Application Programming*, SC24-5899 for information on this topic.

If you are using ICSF, you may need to PERMIT the user ID DFS READ access to the profiles in the CSFSERV general resource class. Refer to the *z/OS: ICSF Administrator's Guide*, SA22-7521, for more information on the CSFSERV resource class.

The SMB server uses an Event Notification Facility (ENF) exit for event code 51 (allocation contention). When this event occurs, an SRB is scheduled to queue a request to the SMB server. If the SMB server is at too low a dispatching priority, the requests may become backed up and the system may eventually run out of resources. The SMB server should have a dispatching priority that is sufficiently high to allow these requests to be processed in a timely manner.

2. Stop the Distributed File Service server (**dfskern**), if it is already running, by following the applicable instructions in Chapter 4, "Managing SMB processes" on page 17.
3. Define administrators on the host system following the applicable instructions in "Defining SMB administrators" on page 14.
4. Create the default DFS configuration files using the **/opt/dfsglobal/scripts/dfs_cpfiles** shell script, if they were not created during the installation process.

   These configuration files, required by SMB File/Print Server, are usually created before the Distributed File Service installation is verified by the **/opt/dfsglobal/scripts/dfs_cpfiles** shell script, as indicated in the *z/OS: Program Directory*, GI10-0669. **dfs_cpfiles** is described in further detail, refer to "Using dfs_cpfiles to create default DFS configuration files" on page 14.
5. Modify the **/opt/dfslocal/home/dfskern/envar** file to activate SMB File/Print Server by setting the environment variable (envar) **_IOE_PROTOCOL_SMB=ON**. Assuming that you do not want to also use DCE DFS file serving, you should also set the environment variable **_IOE_PROTOCOL_RPC=OFF** in the file **/opt/dfslocal/home/dfskern/envar**.

   If you are using OCSF, ensure that the **/opt/dfslocal/home/dfskern/envar** file has a LIBPATH that adds the directory that contains the OCSF DLLs. Be sure that the directory added is the directory indicated in the *z/OS: Open Cryptographic Services Facility Application Programming*, SC24-5899.

   If you are using the print capability of the SMB File/Print Server, ensure that the Infoprint Server is installed and customized by following the applicable instructions in the *z/OS: Program Directory*, GI10-0669. In addition, ensure that the **/opt/dfslocal/home/dfskern/envar** file has a LIBPATH entry that adds the directory that contains the Infoprint Server DLLs. Be sure that the directory added is the directory indicated in the "Infoprint Server Customization Considerations" section of the *z/OS: Program Directory*, GI10-0669.

   For example, a LIBPATH that specifies both the OCSF DLL directory and the Infoprint Server DLL directory might look like **LIBPATH=/usr/lib:/usr/lpp/Printsrv/lib**.
6. Since the SMB File/Print Server runs as an APF-authorized server, you must ensure that any DLLs that are used by the SMB File/Print Server are APF-authorized. This can be accomplished by using the OMVS **extattr +a** command. If you are using the Infoprint Server or OCSF, refer to the "Infoprint Server Customization Considerations" section in the *z/OS: Program Directory*, GI10-0669, and the "Cryptographic Services OCSF Customization Considerations" section in the *z/OS: Open Cryptographic Services Facility Application Programming*, SC24-5899, for information on the location of the DLLs and setting the APF-authorized extended attribute. The DFS load library is called hlq.SIOELMOD.
7. PC clients must be able to find the server on the network in order to use the shares that the SMB server makes available. If you are using Windows 2000 clients, you should ensure that your computer name (specified in the **_IOE_SMB_COMPUTER_NAME** environment variable in the **/opt/dfslocal/home/dfskern/envar**) file is the same as your TCP/IP hostname. Refer to Chapter 5, "Networking considerations" on page 27.
8. Define SMB users by modifying the **smbidmap** file identified by the **_IOE_SMB_IDMAP** environment variable of **dfskern**. Map SMB users to z/OS users on the host system by following the applicable instructions in Chapter 6, "Mapping SMB user IDs to z/OS user IDs" on page 29.

   In addition, z/OS users that do not use DCE, should put the following line in their HFS **.profile** file in their home directory:

   ```
   export _EUV_AUTOLOG=NO
   ```

Alternatively, if no z/OS users are using DCE, the above line may be placed in **/etc/profile**. It is then set for all z/OS UNIX users.

9. Determine whether you intend to use passthrough authentication. Refer to “Using passthrough authentication” on page 49 for information on passthrough authentication. Users in the domain will be authenticated using the Windows Domain Controller. Users that are not in the domain and that fail the domain authentication will additionally attempt local authentication (at the SMB server). This local authentication will use clear or encrypted passwords based on what the Domain Controller chose (most likely encrypted passwords) independent of the **_IOE_SMB_CLEAR_PW** environment variable. In the case of encrypted passwords, those users that get authenticated locally will need to store their SMB password in their RACF DCE segment.
10. Determine whether you intend to use password encryption. Refer to the **_IOE_SMB_CLEAR_PW** environment variable on page 126 and “Logon considerations” on page 48 for information on password encryption. Before you enable password encryption, your PC users must store their SMB password into their RACF DCE segment. Otherwise, they are not able to logon except possibly as a guest user.
11. Determine whether you intend to allow guest users. Guest users are PC users that have (limited) access to files and printers on the SMB server without identifying themselves. Guest users are allowed when the **_IOE_MVS_DFSDFLT** environment variable in the **dfskern** process is set to a valid z/OS user ID. Guest users can access any data or files that that z/OS user ID can access. If guest users are allowed, users that specify an incorrect password or no password become the guest user ID. It is better to disallow guest users until you are certain you need this capability and that it meets your security guidelines.
12. Determine whether you intend to use the dynamic export capability. It is controlled by the **_IOE_DYNAMIC_EXPORT** environment variable of **dfskern**. The default is **OFF** meaning that dynamic export is not enabled. Dynamic export allows the SMB server to support file systems mounted by using the z/OS Automount Facility. Refer to *z/OS: UNIX System Services Planning* document, GA22-7800, for information on the automount facility. Dynamic export also allows the SMB server to dynamically “discover” mounted file systems without the need to provide **dfstab** and **devtab** entries for the file systems. Refer to “Dynamic export for HFS” on page 38 for information on using the dynamic export capability of the SMB server.
13. Define shared directories if the SMB File/Print Server is run on the host system to export file data sets for access by PC clients by updating the **smbtab**, **dfstab**, and **devtab** files and optionally, for RFS, by specifying an **rfstab** file in the **/opt/dfslocal/var/dfs** directory. Define file systems and filesets following the applicable instructions in Chapter 7, “Sharing files” on page 33. For RFS, the DFS server user ID (usually DFS) must have RACF ALTER authority to the data sets that are made available to PC users. Alternatively, you may give the DFS server user ID the OPERATIONS attribute. If you specify a single level prefix in the **devtab**, you must use the OPERATIONS attribute since you cannot create a data set profile that covers a single level prefix. (The OPERATIONS attribute can be limited so that the DFS server user ID does not have authority to all required data sets. Refer to the *z/OS: Security Server RACF Security Administrator’s Guide*, SA22-7683, for information on the OPERATIONS attribute).
14. Define shared printers if the SMB File/Print Server is run on the host system to export Infoprint Server printers for access by PC clients. Define the print shares by updating the file **/opt/dfslocal/var/dfs/smbtab**. Refer to Chapter 8, “Sharing printers” on page 53 for more information.
15. We have determined that SMB server performance is significantly enhanced through the use of the LE HEAPPOOLS(ON) parameter. Refer to “ioepdcf” on page 96 on how to specify HEAPPOOLS for the SMB server. Refer to the *z/OS: Language Environment Programming Guide*, SA22-7561, for information on HEAPPOOLS.
16. Start the Distributed File Service server (**dfskern**) by following the applicable instructions in Chapter 4, “Managing SMB processes” on page 17.
17. Configure PC client workstations to access the SMB File/Print Server by following the instructions in Chapter 9, “Locating the SMB server” on page 55.

Important Note to Users

If you modify the RACF FSSEC class to activate or deactivate ACL checking, the SMB server must be restarted. The SMB server caches permissions and does not get notified of changes to the FSSEC class.

Defining SMB administrators

There are two types of users who can start or stop the Distributed File Service server address space and the processes controlled by DFSCNTL, (refer to Chapter 4, “Managing SMB processes” on page 17):

- A user with z/OS operator privileges.
- A user who has update privilege to the **DFSKERN.START.REQUEST** profile in the RACF FACILITY class. This profile is created during the installation of DFS. For more information on this, refer to the *z/OS: Program Directory*, GI10-0669.

In addition, the SMB Administrator must have **root** authority on the z/OS machine. This means a z/OS user ID with a **UID=0**. This is required in order to issue certain commands and to define shared directories and shared printers.

If the SMB Administrator does not use DCE facilities, the following environment variable should be specified in the SMB Administrator’s OMVS environment (for example, in the SMB Administrator’s **$HOME/.profile** file):

```
export _EUV_AUTOLOG=NO
```

Using dfs_cpfiles to create default DFS configuration files

The **/opt/dfsglobal/scripts/dfs_cpfiles** program is a shell script that creates customizable configuration files in **/opt/dfslocal** subdirectories. **dfs_cpfiles** copies IBM-supplied files from the **/opt/dfsglobal/examples** directory to **/opt/dfslocal** subdirectories. The **dfs_cpfiles** program creates files that do not exist. It does not replace an existing file to preserve any installation configuration data from a previous release.

/opt/dfslocal is a symbolic link to **/etc/dfs**. Therefore all the files created by the **dfs_cpfiles** program are actually created in the **/etc** file system. Refer to the *z/OS: Distributed File Service DFS Customization* document, for more information on the symbolic links defined to identify the configuration files.

Steps for using dfs_cpfiles program

Before you begin: You need to have all the configuration files reside in the **/opt/dfslocal** subdirectories in the **/etc** file system even though some configuration files can be created in other directories and identified by **envar** specifications.

Perform the following steps to invoke **dfs_cpfiles**:

1. Log in as **root** on the local machine (**UID=0**).
2. Enter the following in the shell environment to invoke the **dfs_cpfiles** program:

   ```
   /opt/dfsglobal/scripts/dfs_cpfiles
   ```

   Note: An existing DFS configuration file is not replaced in order to ensure that any existing user customized data is not overlaid.

Result: dfs_cpfiles creates customizable files for all aspects of the Distributed File Service. A subset of these files are applicable to the SMB File/Print Server. The customizable files applicable to the SMB File/Print Server are:

```
/opt/dfslocal/etc/ioepdcf
/opt/dfslocal/var/dfs/devtab
/opt/dfslocal/var/dfs/dfstab
/opt/dfslocal/var/dfs/rfstab
/opt/dfslocal/var/dfs/smbtab
/opt/dfslocal/var/dfs/hfsattr
/opt/dfslocal/home/dfskern/smbidmap
/opt/dfslocal/home/daemonct/envar
/opt/dfslocal/home/dfscntl/envar
/opt/dfslocal/home/dfskern/envar
/opt/dfslocal/home/dfsexport/envar
```

The **smbtab**, **dfstab**, and **devtab** files in the **/opt/dfslocal/var/dfs** directory are created by **dfs_cpfiles**. They must be updated to define shared directories. Refer to Chapter 7, “Sharing files” on page 33 for more information.

- The **smbtab** file must be updated to define shared printers. Refer to Chapter 8, “Sharing printers” on page 53 for more information.
- The **smbidmap** file must be updated to map PC user IDs to z/OS user IDs. Refer to Chapter 6, “Mapping SMB user IDs to z/OS user IDs” on page 29 for more information.
- Optionally, the **hfsattr** and **rfstab** files can be updated to define HFS data translation by file name extension attributes and RFS attributes for RFS files, respectively. Refer to Chapter 7, “Sharing files” on page 33 for more information.

The other customizable files are used for DFS client and server support in a distributed computing environment.

Recommendation: The files exist as created by the **dfs_cpfiles** program, otherwise it can be ignored if only the SMB File/Print Server is used.

Figure 1 on page 16 displays the possible output after using **dfs_cpfiles** to create the customizable configuration files for the Distributed File Service.

```
*********************************************************************
**                  Distributed File Service                       **
**            Default Configuration Files Creation Program         **
*********************************************************************
    Attempt to Create envar Files....
       File /opt/dfslocal/home/bakserver/envar created
       File /opt/dfslocal/home/boserver/envar created
       File /opt/dfslocal/home/butc01/envar created
       File /opt/dfslocal/home/butc02/envar created
       File /opt/dfslocal/home/butc03/envar created
       File /opt/dfslocal/home/butc04/envar created
       File /opt/dfslocal/home/butc05/envar created
       File /opt/dfslocal/home/butc06/envar created
       File /opt/dfslocal/home/butc07/envar created
       File /opt/dfslocal/home/butc08/envar created
       File /opt/dfslocal/home/daemonct/envar created
       File /opt/dfslocal/home/dfscm/envar created
       File /opt/dfslocal/home/dfscntl/envar created
       File /opt/dfslocal/home/dfsexport/envar created
       File /opt/dfslocal/home/dfskern/envar created
       File /opt/dfslocal/home/flserver/envar created
       File /opt/dfslocal/home/ftserver/envar created
       File /opt/dfslocal/home/growaggr/envar created
       File /opt/dfslocal/home/newaggr/envar created
       File /opt/dfslocal/home/repserver/envar created
       File /opt/dfslocal/home/salvage/envar created
       File /opt/dfslocal/home/upclient/envar created
       File /opt/dfslocal/home/upserver/envar created

     Attempt to Create Miscellaneous Configuration Files....
      File /opt/dfslocal/etc/ioepdcf created
```

```
File /opt/dcelocal/etc/CacheInfo created
File /opt/dfslocal/var/dfs/devtab created
File /opt/dfslocal/var/dfs/dfstab created
File /opt/dfslocal/var/dfs/rfstab created
File /opt/dfslocal/var/dfs/smbtab created
File /opt/dfslocal/var/dfs/cmattr created
File /opt/dfslocal/var/dfs/hfsattr created
File /opt/dfslocal/home/dfskern/dfsidmap created
File /opt/dfslocal/home/dfskern/smbidmap created
```

Figure 1. Example output of dfs_cpfiles

Figure 1 shows the **dfs_cpfiles** messages when the files are created.

- If a file already exists, an example of this message is:

  ```
  File /opt/dfslocal/etc/ioepdcf already exists.
  ```

- If an error occurs creating the file, an example of this message is:

  ```
  File /opt/dfslocal/etc/ioepdcf not created
  ```

Tips:

- If you are migrating to this release of the Distributed File Service from an earlier release and **dfs_cpfiles** created a new set of customizable files, you may need to add your customization data to the newly created files.
- If you are migrating to this release of z/OS from an earlier release and new customizable configuration files were not created by **dfs_cpfiles**, you may want to update preexisting customizable files with new customization options available with this release of Distributed File Service. Refer to Chapter 2, “Considerations for a new SMB release” on page 7 for more information on what is new in this release.
- The SMB user identity mapping file is identified by the envar _**IOE_SMB_IDMAP**. It is recommended that the **/opt/dfslocal/home/dfskern/smbidmap** file created by the **dfs_cpfiles** program should be used by the installation for user identity mapping.

When all the configuration files required by SMB file/print processing have been created and updated, you can proceed to the next step of configuration. Refer to the *z/OS: Distributed File Service DFS Customization* document.

Chapter 4. Managing SMB processes

This chapter briefly describes the SMB server address space and then discusses starting the server, stopping the server, and other activities required to manage the server.

The SMB daemons run as separate processes within the DFS address space shown in Figure 2.

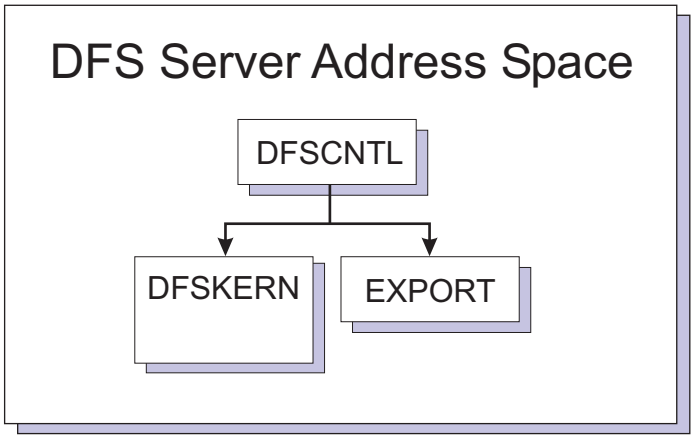


Figure 2. DFS server address space

Figure 2 shows **dfscntl** (the DFS Control Task) acting as the **parent process** to all the DFS server daemons. The DFS server daemons (**dfskern** and **export**) run as child processes of **dfscntl**. The **export** daemon communicates with **dfskern** to make the specified file systems available on the network and then stops. The **dfskern** daemon remains active to handle incoming file and print requests. **dfskern** can be run in the DFS Server Address Space or in its own address space. This is controlled by the, **_IOE_DAEMONS_IN_AS**, environment variable which is set in the **/opt/dfslocal/home/dfscntl/envar** file. If this environment variable is not specified, **dfskern** runs in the DFS Server Address Space. If it is specified as **_IOE_DAEMONS_IN_AS=DFSKERN**, **dfskern** runs in it's own address space (called the **dfskern** Address Space) as shown in Figure 3 on page 18. Running **dfskern** in it's own address space may reduce contention for resources and provide better failure recovery. IBM recommends that **dfskern** be run in its own address space. **MODIFY** commands are unchanged whether **dfskern** runs in its own address space or not. Ensure that the **dfskern** JCL is available and the **daemonct** envar file is in the **daemonct** home directory (**/opt/dfslocal/home/daemonct**) if you want to run **dfskern** in its own address space. Then, if you want to change where **dfskern** runs, add or remove the **dfscntl** environment variable (in the **/opt/dfslocal/home/dfscntl/envar** file), **_IOE_DAEMONS_IN_AS=DFSKERN**, stop DFS if it is running, and then restart DFS.

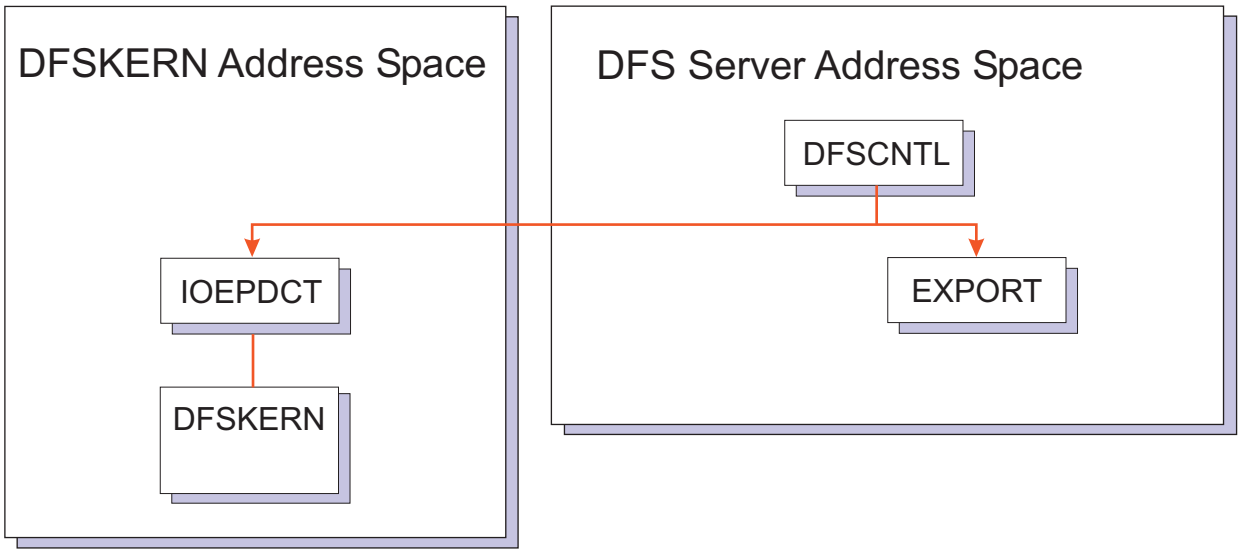


Figure 3. DFSKERN in a separate address space

After the DFS server address space is configured, both daemons (**dfskern** and **export**) are started automatically when the DFS server address space is started.

All requests to DFS are directed to **dfscntl**, that performs the requested action. The DFS server daemons can be started and stopped using an operator command. In starting and stopping the daemons, **dfscntl** uses a **Daemon Configuration File** (that is, the **ioepdcf** file) that contains information on the daemons that were previously configured on the host. The Daemon Configuration File contains runtime options, startup parameters, and restart information for its subprocesses. For details on the Daemon Configuration File, refer to “Daemon configuration file” on page 22.

Besides starting and stopping the DFS server daemons, **dfscntl** can also detect a daemon that has prematurely stopped and tries to restart it automatically. The algorithm used by **dfscntl** in starting and restarting the DFS server daemons is summarized in “How dfscntl starts the DFS server daemons” on page 23.

The following are the PDS member names for the DFS processes started by **dfscntl**:

dfskern The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

export The **export** process is started by **dfscntl** and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

Who can start and stop DFS server daemons?

There are two types of users who can start and stop the DFS server daemons in DFS:

- A user with z/OS operator privileges.
- A user who has update privilege to the **DFSKERN.START.REQUEST** profile in the RACF FACILITY class. This profile is created during the installation of DFS. For information on this, refer to the *z/OS: Program Directory*, GI10-0669.

Starting DFS server daemons

DFS server daemons are started in any of the following ways:
- During the system IPL
- Using the **MODIFY DFS** operator command
- Using the **START** operator command.

Based upon the control files for **dfscntl** set up by the DFS administrator, all of the DFS server daemons can be automatically started when the DFS started task is started.

The DFS daemons can all run in one address space (except for possibly **dfskern**) which is a started task (default name DFS). A user with z/OS operator privileges can start and stop the DFS started task. The DFS server address space may be started automatically during system IPL. To start the DFS server address space, use the z/OS **START** command. For example,

```
start dfs
```

Ideally, the daemons run continuously in the background and do not need to be started or stopped again. However, the DFS server daemons may have to be started manually in certain situations, for example, if a daemon ends abnormally. You can use the **MODIFY** operator command to manually start or stop the DFS server daemons.

Each of these alternatives is discussed in the following sections.

The MODIFY DFS operator command

The DFS server daemons (processes) can be started or stopped using the **MODIFY DFS** operator command. Using **MODIFY DFS**, you can also view the status of the DFS server daemons. Following is the syntax of the **MODIFY DFS** command:

MODIFY DFS,*command daemon[,options]*

where:

DFS is the name of the DFS server address space.

command Is the action that is to be performed on the SMB server daemon or daemons. It can have any of the following values:

start	Starts the DFS server daemon or daemons.
stop	Stops the DFS server daemon or daemons.
query	Displays the state of the DFS server daemon or daemons.
send	Sends requests to the DFS server daemon or daemons.

daemon Is the name of the DFS server daemon for which the action is being requested. It can have any of the following values:

dfskern	DFS kernel program (includes the SMB File/Print Server).
export	Export program to make file systems available for exporting and shares available to PC users.
unexport	Unexport program to unexport file systems and delete shares.
all	All the DFS server daemons (that is, **dfskern** and **export**).

options Values that are passed to the daemons.

Using MODIFY DFS to start DFS server daemons:

With the **MODIFY DFS** operator command, you have the option of starting an individual daemon or starting all the daemons using a single command.

For example, to start the **dfskern** daemon, enter the following:

```
modify dfs,start dfskern
```

To start all the daemons, enter the following:

```
modify dfs,start all
```

Note: Do not use the **MODIFY** command to start the DFS server daemons while the DFS server address space is still initializing. During initialization, DFS attempts to start all the DFS server daemons that have been configured on the z/OS host. If you issue the **MODIFY** command while DFS is initializing, the DFS server daemons may be started out of order or stopped erroneously. This may lead to unexpected errors during initialization and cause DFS to end abnormally.

It is recommended that you wait until DFS has issued a log message indicating that DFS server initialization has completed before using the **MODIFY** commands.

Order of starting DFS server daemons

When DFS server daemons are started manually, the successful startup of some daemons depend on the availability of the services provided by other daemons. This implies that the DFS server daemons must be started in a particular order.

Following is the sequence by which DFS server daemons should be started.

Note: This is only applicable if you need to start any of the DFS server daemons individually. If the DFS server daemons are started collectively, (for example, using the **start all** option of the **MODIFY DFS** command) DFS ensures that the correct starting sequence is followed.

1. **dfskern**
2. **export**

For example, to successfully start the **export** daemon, the **dfskern** daemon must already be up and running.

If you are using the DFS server's SMB capability to do printing, the Infoprint Server should be started before the **dfskern** server daemon. Otherwise, you need to issue the **dfsshare** command to share the printers defined in **smbtab**.

Stopping DFS

To stop the DFS server address space, use the **STOP** operator command to ensure the normal shutdown of the address space.

To stop the DFS server address space and all DFS server daemons, enter the following z/OS operator command:

```
stop dfs
```

To stop the DFS server daemons, but not the DFS server address space, use the **STOP ALL** command. For example:

```
modify dfs,stop all
```

The **STOP ALL** command causes **dfscntl** to stop all daemons that it controls.

Using MODIFY DFS to stop DFS server daemons

You can use the **MODIFY DFS** system command to stop a DFS server daemon or all daemons that are configured on the host.

For example, to stop the **dfskern** daemon, enter:

```
modify dfs,stop dfskern
```

To stop all DFS server daemons on the host, enter:

```
modify dfs,stop all
```

Viewing the status of DFS server daemons

You can query the status of the DFS server daemons using the **query** option of the **MODIFY** system command. You do not need the special privileges of a DFS administrator or an operator to use the **QUERY** option.

For example, to query the status of the **dfskern** daemon, enter the following:

```
modify dfs,query dfskern
```

A message about the status of the daemon is written on the system log. This message also contains the **process ID** of the daemon.

The status of the daemon can be any of the following:

READY Indicates that the daemon is running, has been initialized, and is ready to receive and process incoming requests.

ACTIVE Indicates that a manual process is running. When an active process stops, it is never considered an error and it is never restarted automatically.

INITIALIZING Indicates that the daemon has been started, but is not yet ready to receive and process incoming requests.

STOPPING Indicates that a request to stop the daemon has been received and that the daemon is in the process of stopping.

DOWN Indicates that the daemon is not active.

UNKNOWN Indicates that the status of the daemon cannot be determined. This can occur if the daemon was started, but no response was received by the system indicating a change in its status.

Note: You can issue a command to stop a daemon if it is in the UNKNOWN state.

The status of DFS daemons controlled by **dfscntl** can be queried by z/OS operator commands. For example:

```
modify dfs,query all
modify dfs,query dfskern
```

Following is an example of a query command to **dfscntl**. The output is sent to the z/OS Operator's console:

```
modify dfs,query dfskern
IOEP00022I DFS daemon DFSKERN status is READY and process id is 781.
```

Starting DFS server daemons during IPL

Because the DFS server daemons are contained in the DFS server address space (except for possibly **dfskern**), these daemons are started during the initialization of DFS. This allows you to configure the host to automatically start the DFS server address space during the system IPL.

dfscntl uses the Daemon Configuration File to determine which daemons can be started, and the parameters to pass to the daemon load module when starting the daemon. The DFS server address space may be started automatically during system IPL. To start the DFS server address space, use the z/OS **START** command. For example,

```
start dfs
```

Daemon configuration file

The Daemon Configuration File is used by **dfscntl** to obtain necessary information when starting the DFS server daemons. The Daemon Configuration File contains the following information:

- The name of the process to be started.
- A parameter that specifies actions to be taken when the process is started. It can have the following values:

  Y Start the process during initialization. If the process ends abnormally, then restart it automatically.

  N Do not start the process automatically or manually.

  I Start the process during initialization. The process can be started manually. If the process ends, it does not restart.

  M Can be started manually.
- Parameters that are passed to the load module when a daemon is started, called the argument list (including LE/370 runtime options).
- The **Minimum Restart Interval. dfscntl** attempts to restart a daemon that ends abnormally only if the daemon was running for at least this time interval. If a daemon ends during this time interval, it is not be restarted.
- The **Time-out Period**, which is the maximum time interval that **dfscntl** waits for the daemon to complete its initialization after it has been started. When this time interval elapses, and **dfscntl** has not received confirmation from the daemon that initialization has completed, the status of the daemon is set to **UNKNOWN**.

The path name to the Daemon Configuration file is **/opt/dfslocal/etc/ioepdcf.** Figure 4 on page 23 shows the typical contents of the Daemon Configuration file.

```
DFSKERN  CONFIGURED=Y LMD=DFSKERN  ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern')/-admingroup
subsys/dce/dfs-admin>DD:DFSKERN2>&1" RESTART=300 TIMEOUT=300
EXPORT CONFIGURED=I LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-all -verbose
>DD:EXPORT  2>&1" RESTART=300 TIMEOUT=300
UNEXPORT CONFIGURED=M LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-detach -all -ver
>DD:UNEXPORT2>&1" RESTART=300 TIMEOUT=300
BOSERVER CONFIGURED=N LMD=BOSERVER ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/boserver')/ >DD:BOSERVER 2>&1"
RESTART=300 TIMEOUT=300
BUTC01 CONFIGURED=M LMD=BUTC01 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc01')/ -tcid   >DD:BUTC01 2>&1"
RESTART=300 TIMEOUT=300
BUTC02 CONFIGURED=M LMD=BUTC02 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc02')/ -tcid 1 >DD:BUTC02 2>&1"
RESTART=300 TIMEOUT=300
BUTC03 CONFIGURED=M LMD=BUTC03 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc03')/ -tcid 2 >DD:BUTC03 2>&1"
RESTART=300 TIMEOUT=300
BUTC04 CONFIGURED=M LMD=BUTC04 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc04')/ -tcid 3 >DD:BUTC04 2>&1"
RESTART=300 TIMEOUT=300
BUTC05 CONFIGURED=M LMD=BUTC05 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc05')/ -tcid 4 >DD:BUTC05 2>&1"
RESTART=300 TIMEOUT=300
BUTC06 CONFIGURED=M LMD=BUTC06 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc06')/ -tcid 5 >DD:BUTC06 2>&1"
RESTART=300 TIMEOUT=300
BUTC07 CONFIGURED=M LMD=BUTC07 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc07')/ -tcid 6 >DD:BUTC07 2>&1"
RESTART=300 TIMEOUT=300
BUTC08 CONFIGURED=M LMD=BUTC08 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc08')/ -tcid 7 >DD:BUTC08 2>&1"
RESTART=300 TIMEOUT=300
```

Figure 4. Daemon configuration file

In the **ARG** (argument list) field of this example, **ENVAR** indicates the Language Environment® (LE) environment variables used, in this case, the **_EUV_HOME** environment variable is specified to identify the daemon's home directory. (The home directory contains the daemon's **envar** file. Refer to "envar" on page 93 for more information.) Anything after the "/" character are program parameters. The ">" character is the redirection character which indicates that the output is redirected to the DD name that follows.

Important Note to Users:
Under normal circumstances, you do **not** need to edit the Daemon Configuration File. Although there may be certain situations when the Daemon Configuration File has to be modified, it is recommended that you do so under the supervision of an IBM service representative.

The Daemon Configuration File can only be modified when the DFS server address space is not running.

The Daemon Configuration File is optional. When it is omitted, the defaults are as listed in Figure 4 except that BOSERVER CONFIGURED=M. When using only the SMB capability of the Distributed File Service (and not the DCE DFS capability), there is no need to start or reference the BOSERVER or the BUTC0n servers.

How dfscntl starts the DFS server daemons

When a request arrives to start DFS server daemons, **dfscntl** looks at the Daemon Configuration File to see if the particular daemon or daemons are configured on the host. If the daemon is configured and is not running, **dfscntl** starts it and waits for the daemon to initialize successfully.

In case of the abnormal ending of any DFS server daemon, **dfscntl** tries to restart the daemon. **dfscntl** attempts to restart the daemon only if the daemon was running for at least the duration of the Minimum Restart Interval, as specified in the Daemon Configuration File. If the daemon ended within this time interval, it is not restarted.

Note: When **dfscntl** restarts an abnormally terminated daemon, it does not correct the problem that caused the daemon to end unexpectedly. Thus, depending on the cause of the abnormal ending, the daemon may fail again after restarting because of the same error condition.

Using the -nodfs option to start dfscntl

You can start the DFS server address space without starting the configured DFS server daemons by using the **-nodfs** option of the **START** operator command, for example:

```
start dfs,parm='-nodfs'
```

Changing environment variables

Each SMB server process uses environment variables to control how it behaves. When any processes environment variables are changed in the **envar** file in the home directory of the process, the process must be restarted to make them take effect.

Changing mappings

The **dfskern** process optionally supports a mapping between SMB user IDs and z/OS user IDs. The **smbidmap** file is located by means of the **_IOE_SMB_IDMAP** environment variable of **dfskern**. If the **smbidmap** file is updated (to add, change, or delete mappings), you must issue the **modify dfs,send dfskern,reload,smbmap** operator command to make them take effect. If you change the location of the **smbidmap** file, then the **_IOE_SMB_IDMAP** environment variable must be updated and the **dfskern** process must be restarted. It is recommended that the **smbidmap** file is located in the **/opt/dfslocal/var/dfs** directory so that it is contained with the other customizable files.

Changing shared directories or shared printers

Shared directories and shared printers are defined in the **smbtab** file. If the **smbtab** file is changed to add or delete a share, the **dfsshare** command must be issued to make it take effect. Use the **-share** parameter of the **dfsshare** command to add a new share defined in **smbtab**. Use the **-detach** parameter of the **dfsshare** command to delete a share. The **dfsshare -detach** for the share must be issued before removing the share from the **smbtab**.

If the shared directory is contained in a file system that has not been exported before, or if there are additional file systems (below the file system containing the shared directory) that you want available to PC users, then you should add those file systems to the **dfstab** and the **devtab** and then issue the **dfsexport** command before issuing the **dfsshare** command.

If, however, you are using the dynamic export capability of the SMB server, **dfstab** and **devtab** entries are not required for file systems that are mounted below the file system containing the shared directory. Refer to “Dynamic export for HFS” on page 38 for information on the dynamic export capability.

Shared printers do not need entries in the **dfstab** nor the **devtab** and do not require the **dfsexport** command.

Changing the hfsattr or the rfstab

The **hfsattr** contains directives that map a file name extension (suffix) to an indication whether the data should be translated from ASCII to EBCDIC and vice versa. Its location is specified in **_IOE_HFS_ATTRIBUTES_FILE** environment variable of the **dfskern** process. The **rfstab** contains creation (and other) attributes for RFS files. Its location is globally specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable of the **dfskern** process. It can also be specified on an RFS file system basis in the **devtab attrfile** parameter. When the **hfsattr** or the global **rfstab** file is modified, the **dfskern** process must be restarted to make it take effect. If an RFS file system’s **rfstab** is

modified, the file system must be re-exported. That is, if it is already exported, it must be unexported and then exported. This is accomplished with the **dfsexport** command.

Changing the Infoprint Server DLL

If a new release of the Infoprint Server is installed or it is enabled, it can be activated for the SMB server by the **modify dfs,send dfskern,reload,print** system command. You also need to issue the **dfsshare** command to share the printers defined in **smbtab**.

Chapter 5. Networking considerations

In order to use the shares that the SMB server makes available, PC clients must be able to find the server on the network. The communications mechanism used is TCP/IP, and the methods of server discovery follow:

- The Domain Name Service (DNS)
- The Windows Internet Naming Service (WINS)
- Clients on the same workgroup and same subnet as the server
- The LMHOSTS file.

The order in which a server name is resolved to a TCP/IP address may vary depending on the client software and the service pack level of the Windows client software.

TCP/IP networks can use the Domain Name Service (DNS) to map server system names to IP addresses. In a DNS network, an entry tells clients in the network how to map the name of the server to its proper TCP/IP address.

If you want PC clients to access the SMB server by using DNS, then you must ensure that the hostname and IP address are added to the DNS database. Using DNS is generally the easiest way for clients to access the SMB server on a distributed network. In this case, you should ensure that the (TCP/IP) hostname and the (SMB) computer name are the same. (This is the default if you do not specify a computer name for the SMB server by using the **_IOE_SMB_COMPUTER_NAME** environment variable of **dfskern**.) Also, if you have Windows 2000 clients, you should ensure that the SMB computer name is the same as the TCP/IP hostname.

Microsoft Windows NT/2000 servers can provide the Windows Internet Naming Service (WINS) which allows clients to map a computer name to the computer's actual TCP/IP address. If you use a WINS server in your network, you can configure the SMB server to announce itself to the WINS server. Then you can configure PC clients to connect to the SMB server by using the WINS server. The SMB server announces itself to the primary WINS server (identified by the **_IOE_SMB_PRIMARY_WINS** environment variable of **dfskern**). If the primary WINS server cannot be contacted, the SMB server announces itself to the secondary WINS server (identified by the **_IOE_SMB_SECONDARY_WINS** environment variable of **dfskern**). The SMB server does not, itself, act as a WINS server. It can, however, act as a WINS proxy. That is, it can accept WINS requests from PC clients and forward them to a WINS server if the **_IOE_SMB_WINS_PROXY** environment variable of **dfskern** is specified as **ON**. The PC clients would have the IP address of the SMB server specified as the WINS Server IP address.

In the **_IOE_SMB_DOMAIN_NAME** environment variable of **dfskern**, you can specify the name of the domain or workgroup that the SMB server should be a member of. This can be the name of an existing domain or workgroup in your LAN environment. If possible, put your SMB server in the same domain or workgroup as your client PCs.

An SMB server that is in the same workgroup and the same subnet as the PC clients appear in the Windows Network Neighborhood without any additional configuration on the server or those PC clients. An SMB server that is on the same subnet as a Primary Domain Controller that is acting as a WINS server appears in the Network Neighborhood of PCs that contain the WINS server IP address. The SMB server announces itself to the Browser by using subnet broadcast on UDP port 138. It does this every Browser announcement interval (specified by the **_IOE_SMB_BROWSE_INTERVAL** environment variable of **dfskern**). Those PC clients can also find the SMB server by using subnet broadcast to UDP port 137. The SMB server responds to this broadcast. PC clients that want to use the Network Neighborhood function should have the NetBEUI protocol installed.

PC client operating systems can provide static configuration files that can map server system names to TCP/IP addresses. These files are typically more difficult to manage than a solution that involves more centralized control (for example, a DNS or WINS server). This is because the network administrator must

configure each PC client individually. Static configuration files are very useful, however, in large, distributed networks. In this environment, clients and servers exist in different subnets (network segments) and possibly different workgroups (domains). Static configuration files help clients locate servers.

Note: In order to enable the Network Neighborhood function, there must be at least one Windows NT or Windows 2000 Domain Controller on the same subnet as the SMB server. The Windows NT or Windows 2000 Domain Controller must be acting as a Domain Master Browser (as it usually does). Refer to “SMB server does not show up in Network Neighborhood” on page 137 for more information on Network Neighborhood.

Windows 9x and Windows NT/2000 clients provide the LMHOSTS file that can map server computer names to IP addresses. LMHOSTS contains IP addresses and server computer names for which to map those addresses. You can use these files to map the IP address of the SMB server for clients. This allows clients to find the SMB server in a large, distributed network environment.

You can find more information about LMHOSTS files in the sample LMHOSTS file that is provided with your Windows operating system. Additional information is available in your PC operating system documentation.

Chapter 6. Mapping SMB user IDs to z/OS user IDs

The local security subsystem (for example, RACF) determines if the client is authorized to access HFS or RFS directories and files. Because the local security subsystem does not recognize SMB user IDs, the SMB user ID that comes to the SMB server must be mapped to a local user ID. This mapping is defined in the **smbidmap** file, which is read during **dfskern** initialization or as a result of the **modify dfs,send dfskern,reload,smbmap** operator command. The **smbidmap** file is an HFS file, and its location is specified in the **_IOE_SMB_IDMAP** environment variable of **dfskern**. It contains SMB user IDs and their corresponding z/OS user IDs. This information is used by **dfskern** to map SMB user IDs to z/OS user IDs. The following procedure may be used to map SMB user IDs to z/OS user IDs:

- Create an **smbidmap** file
- Set the **_IOE_SMB_IDMAP** environment variable
- Stop and restart the **dfskern** process.

Note: If **_IOE_SMB_IDMAP** environment variable already has the name of the **smbidmap** file and the **smbidmap** is just being updated, the **modify dfs,send dfskern,reload,smbmap** command can be used to activate the updated mappings.

Each of these procedures are described in the following sections.

Creating an smbidmap file

The **smbidmap** file is a text file that the administrator creates and maintains. It must be an HFS file. Any editor available on z/OS UNIX may be used (for example, oedit, vi, etc.). The **smbidmap** file contains one or more mapping declarations and has the following general format:

```
SMB-user-ID1
z/OS-user-ID1

SMB-user-ID2
z/OS-user-ID2
...
```

Each entry has two elements: SMB user ID and z/OS user ID. A blank line is optional between entries (but is recommended for readability). The following explains each element in an SMB user ID mapping entry:

*SMB-user-ID or Domain/SMB-user-ID or Workgroup/SMB-user-ID or Domain/**
: Specifies the client's SMB identity. This may either be a simple SMB user ID (when you do not care what the domain of the SMB user ID is) or a fully qualified name (for clients within and outside the domain/workgroup) or a domain name with an asterisk (for all clients in a particular domain). The SMB user ID can be up to 20 characters in length. A Domain (Workgroup) name can be up to 15 characters in length.
  - *SMB-user-ID* is assumed to be in any domain.
  - *Domain/SMB-user-ID* is assumed to be in the specified domain.
  - *Workgroup/SMB-user-ID* is assumed to be in the specified workgroup.

  Note: When a PC user is in a workgroup, the PC client sends the computer name as the domain name (not the workgroup name).

z/OS-user-ID
: Specifies the z/OS user ID of the client. All potential SMB clients must have z/OS user IDs on the system where the SMB server is running.

  Note: This field is case sensitive. The case of the z/OS user ID is important when using the **&USERID** keyword of the **smbtab** file. When a PC user does a **net use** to a shared directory that has **&USERID** in the directory path name field of the **smbtab** entry, the z/OS user ID is taken from the **smbidmap** entry for that PC user and is used to reference (with the exact case specified for the z/OS user ID) the directory.

When this directory reference causes automount to occur, a directory with this exact case is created by automount. Make sure you specify the z/OS user ID with the proper case so that the directory created by automount will have the correct name. (For example, it may need to match the user ID in the definition of the user's home directory in the RACF OMVS segment of the user's profile.) The z/OS user ID will be folded to upper case when used to logon to z/OS.

Another entry that is allowed in **smbidmap** is:

*

=

This means that if a z/OS user ID cannot be determined, the SMB user ID should be used as the z/OS user ID. This only occurs if the SMB user ID is eight characters or less.

Note: The SMB user ID will be used (as received over the wire) for the z/OS user ID for an **&USERID** value specified in the **smbtab**. In this case, the PC user ID as specified on the Windows logon (or on the **net use** command) will be used by automount to create a directory. This is case sensitive in that the directory will be created with the exact case as received from the PC. Some clients may fold the PC user ID to upper case before sending it to the SMB server. In this case, if the directory created by automount needs to be lower case or mixed case, the administrator needs to add an entry to the **smbidmap** file to map the upper case PC user ID to the correct case z/OS user ID.

Each SMB user can only have one mapping to a z/OS user. However, different SMB users can be mapped to the same z/OS user, if desired.

Setting the _IOE_SMB_IDMAP environment variable

The **_IOE_SMB_IDMAP** environment variable must be set to the name of the **smbidmap** file used by the SMB server. The declaration of this environment variable can be made in the **envar** file of the **dfskern** process located in **/opt/dfslocal/home/dfskern/envar**.

For example, if the HFS path name of the **smbidmap** file is **/opt/dfslocal/home/dfskern/smbidmap**, this variable is set by the following entry in the **envar** file:

```
_IOE_SMB_IDMAP=/opt/dfslocal/home/dfskern/smbidmap
```

Modifying and deleting identity mapping entries

Edit the **smbidmap** file to modify or delete mapping entries. For these changes to take effect, you have to either restart the SMB server or reload the **smbidmap** file by using the **modify dfs,send dfskern,reload,smbmap** command. Refer to Chapter 12, "z/OS system commands" on page 73 for more information on the **modify** command.

Determining the z/OS user ID from the SMB user ID

The following decisions are made for how **dfskern** determines the z/OS user ID from the SMB user ID:

- The **smbidmap** table is read during SMB server initialization or due to a **modify dfs,send dfskern,reload,smbmap** operator command.
- When an SMB comes to the SMB server,
  - If an SMB user ID and domain are in the SMB, then search for a match in **smbidmap** in a case insensitive manner.
  - If no match, use just the SMB user ID from the SMB and search for a match in the SMB user ID and "don't care" domain in a case insensitive manner. (These are entries in **smbidmap** that do not have a domain.)

- If there is still no match, see if there is an *** =** entry. If so, use the SMB user ID from the SMB as a z/OS user ID.
- If a match was found, attempt a logon with the mapped ID and the password.
- If there is still no z/OS user ID, or if the logon attempt was unsuccessful, see if the **_IOE_MVS_DFSDFLT** environment variable is specified. If so, use its value as the z/OS user ID (without a password). This is sometimes known as a guest login.

The SMB client request is denied if the SMB server cannot determine a mapping to a z/OS user ID. For example, if the SMB user ID is unspecified or not mapped, or mapped but not in RACF and if **_IOE_MVS_DFSDFLT** is not specified or the **_IOE_MVS_DFSDFLT** user ID is not in RACF, the client request is denied.

How the SMB user ID is determined

The SMB user ID is determined from the user ID specified when the user logged in to Windows. (Windows NT/2000 provides some other options.) This user ID is mapped to a z/OS user ID, and the password is taken as the password for the z/OS user ID (when using clear passwords) or the user's SMB password in their RACF DCE segment (when using encrypted passwords). Refer to "Logon considerations" on page 48 for information on clear passwords and encrypted passwords.

The simplest method for using the SMB server is to logon to Windows with your SMB user ID that the SMB server can map to a z/OS user ID. When a drive letter is mapped to a shared directory, that user ID is sent to the SMB server. If you are prompted to enter your password, you should enter your z/OS password (when using clear passwords) or your SMB password from your RACF DCE segment (when using encrypted passwords).

The password that you logged onto Windows with is sent to the SMB server. If your z/OS password (when using clear passwords) or your SMB password from your RACF DCE segment (when using encrypted passwords) is different than your Windows password, you should specify your z/OS or SMB password on the **net use** command. If your Windows password is different than your z/OS or SMB password and you do not specify it on the **net use** command (or you specify it incorrectly), you are logged in as the DFSDFLT user ID, if the **_IOE_MVS_DFSDFLT** environment variable is specified on the SMB server. If the **_IOE_MVS_DFSDFLT** environment variable is not specified and you specified an incorrect password, you may be prompted for the correct password or denied.

On Windows NT/2000, you can specify your SMB user ID on the map network drive pull down and you can specify your SMB user ID (and password) on the **net use** command. This allows you to use a different SMB user ID than the one you used to logon to Windows NT or Windows 2000.

Chapter 7. Sharing files

Before PCs can access files, a shared directory must be created. A shared directory represents the starting point or top directory of a “tree” of directories and files. A PC user can access the shared directory and the subdirectories and files based on the user’s authorization to those items. A PC user cannot reference a directory (or file) that is higher than the shared directory (except by accessing an absolute symbolic link when the **_IOE_SMB_ABS_SYMLINK** environment variable in the **dfskern** process is **ON**).

Before a shared directory can be created, the file system containing the directory to be shared must be exported. (A file system is identified by its file system name.) The following discussion assumes that you are familiar with HFS file systems, z/OS UNIX concepts, and data sets. The file system types supported by the SMB server are HFS, ZFS, TFS, and AUTOMNT. If you are in a sysplex with Shared HFS files, SMB support of ZFS is limited to ZFS compatibility mode file systems. The NFS and DFSC file system types are not supported. Unless otherwise noted, when the term HFS is used, it includes all the supported file system types.

Note: In order to export an HFS file system, that file system must be owned by the system that the SMB server is running on. When export is attempted on a (sysplex) shared file system that is owned by a system other than the one that the SMB server is running on, the SMB server attempts to move the ownership of a shared HFS file system to the system that the SMB server is running on (when the **_IOE_MOVE_SHARED_FILESYSTEM** environment variable is **ON** in the **dfskern** process). If this is unsuccessful (when, for example, the file system is being exported by another SMB server on the other system), the SMB server is unable to export that file system and it is not available to PC clients. When the SMB server is running with dynamic export enabled and the move of ownership of file systems enabled, you should run a single SMB server on a sysplex. Otherwise, each SMB server may try to export (and move the ownership of) the same file system and one of them will fail.

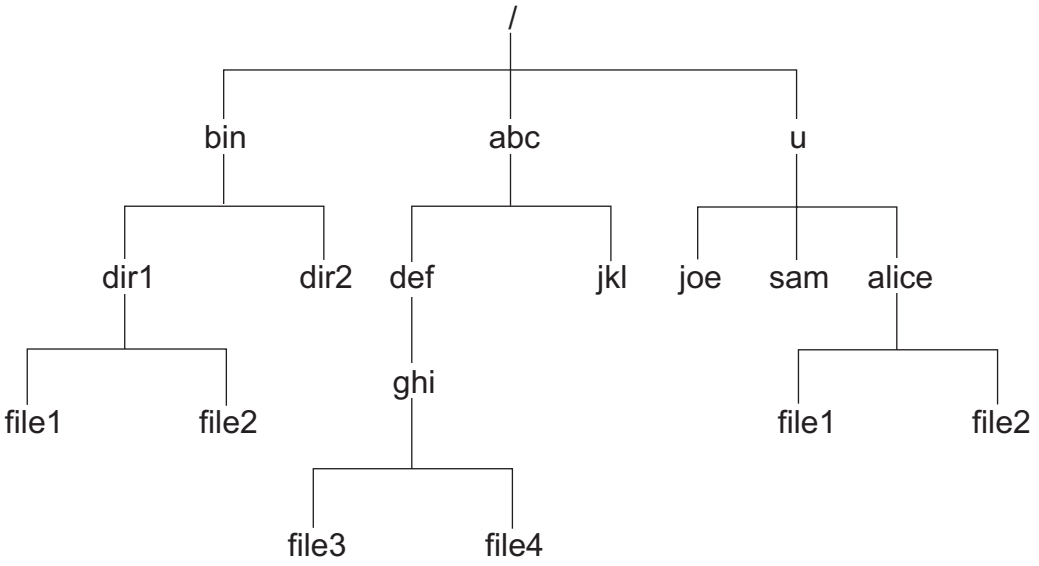
Sharing HFS files

This section discusses the HFS file system.

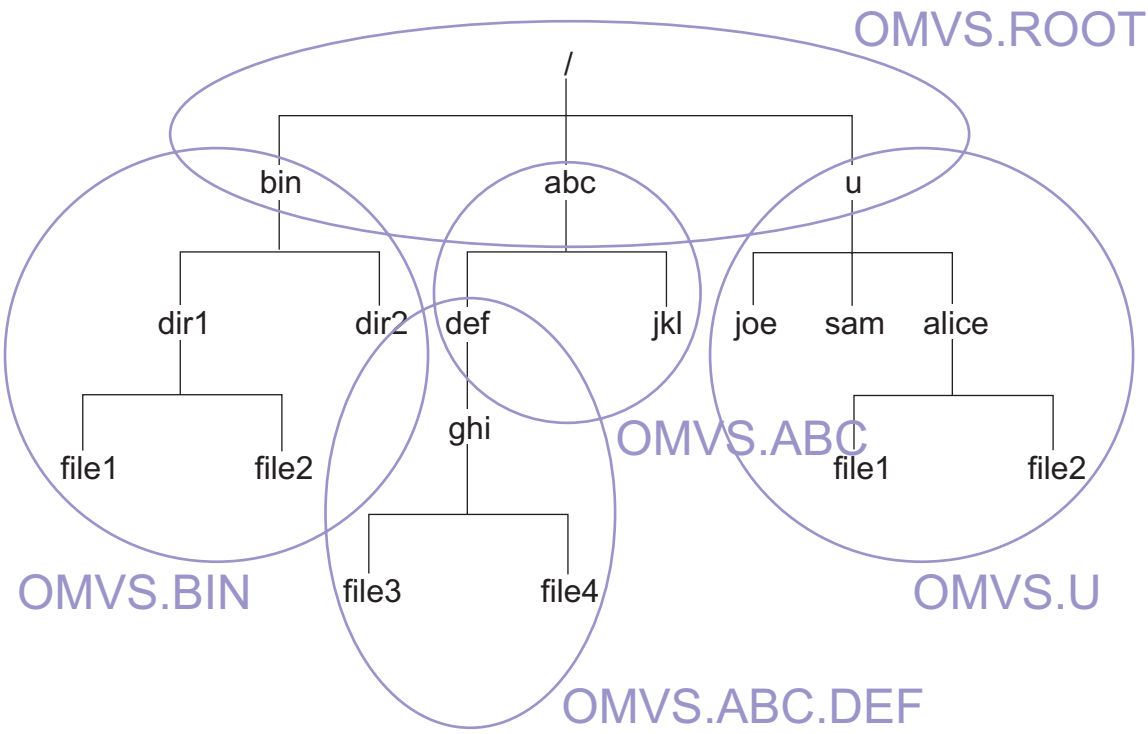
Exporting and sharing HFS file systems

An HFS file system must be exported before a directory contained in it can be shared. Exporting is done on an HFS file system basis and tells z/OS UNIX that a file system is being made available to clients by a File Exporter (that is, the Distributed File Services SMB File/Print Server).

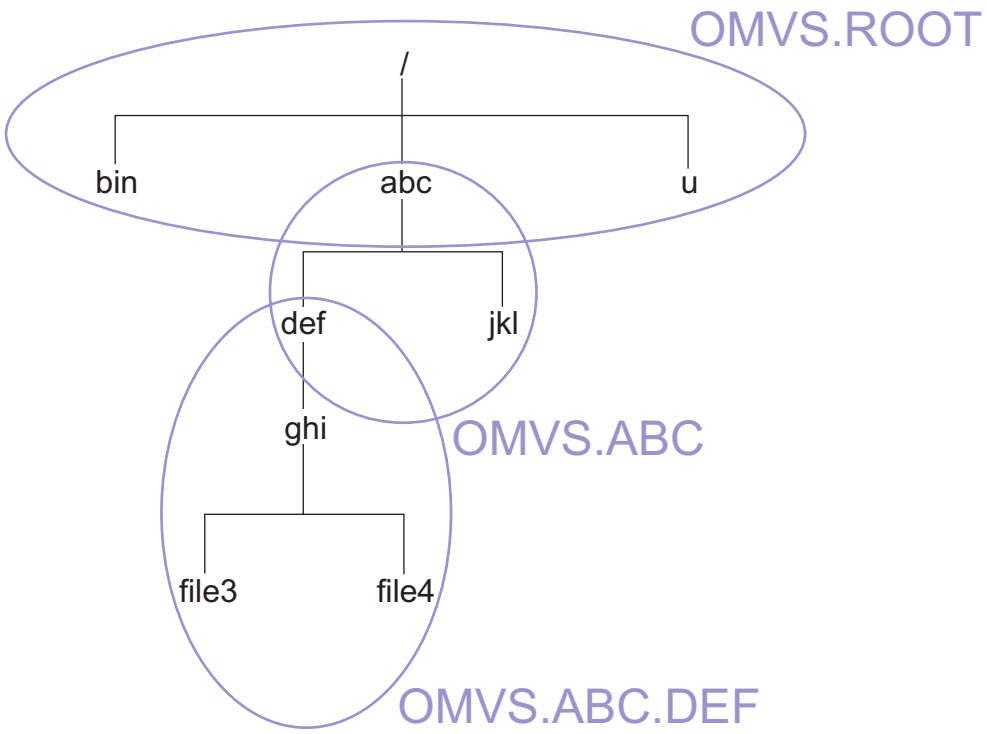
Sharing is done on a directory tree basis and allows PC clients to access data on HFS. In order to allow a directory to be shared, the file system that the directory is contained in must be exported. Suppose we had the following (purposely simplified) entire file hierarchy on a z/OS system:



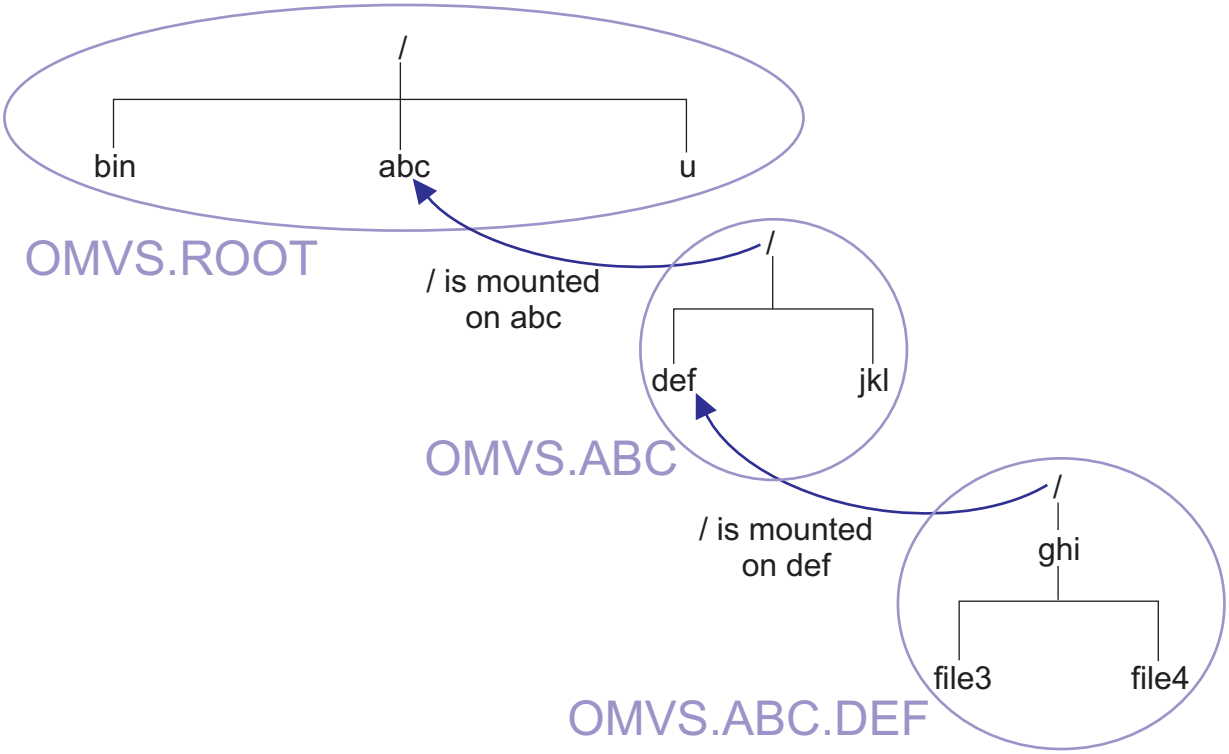
This file hierarchy is made up of separate file systems mounted together. Refer to the following representation of this:



The ovals represent the individual file systems. There are five file systems in this hierarchy. Each of these file systems has a data set name. The top file system is referred to as the **root** file system. Its data set name is OMVS.ROOT. Each of the lower file systems are mounted on top of a directory in the file system above it. There are three file systems mounted on three directories that are contained in the root file system. One is mounted on the directory **/bin** (data set OMVS.BIN), another is mounted on **/abc** (data set OMVS.ABC), and the third is mounted on **/u** (data set OMVS.U). The fifth file system is mounted on **/abc/def** (data set OMVS.ABC.DEF). The data set name of a file system can be displayed with the **df** command. The following diagram examines the **/abc/def/ghi** path a little more closely.



The **/abc/def/ghi** path consists of three file systems mounted as follows:



If you want to create a share for directory **/abc/def/ghi**, you must export the file system where the (/ghi) directory resides (OMVS.ABC.DEF).

In addition, if there are other file systems below the file system containing the shared directory, you may want to export these also so that PC clients can reference data in those file systems. They should be added to **dfstab** and **devtab** and exported using the **dfsexport** command (refer to “smbtab, dfstab, and

devtab entries for HFS" for more information). Otherwise, a PC client is denied access to directories and files in file systems that are not exported.

smbtab, dfstab, and devtab entries for HFS

The files that the SMB File Server uses to relate the shared directory and its exported file system are the **smbtab**, the **dfstab**, and the **devtab**. (They are all located in **/opt/dfslocal/var/dfs.**) A common **minor device number** is used to tie related entries in these three files together. The **smbtab** is used to define the shared directory. The **dfstab** and the **devtab** are used to define the file system to be exported.

In **smbtab**, the minor device number is specified using the format **/dev/ufs***n*, where *n* is a locally assigned unique minor device number that must refer to the HFS file system data set where the root of the shared directory path name resides (that is the file system where the first / resides). This should always be the file system where the directory that you want to share resides. Note that when you are sharing a mount point, you would specify the file system that is mounted on the mount point in **devtab** and **/** as the directory in **smbtab**.

In **dfstab**, the minor device number is specified using the format **/dev/ufs***n* to identify the assigned unique identifiers for the HFS file system.

In **devtab**, the minor device number is specified using the format **define_ufs** *n* to identify the data set name for the HFS file system.

For example, if the shared directory is /ghi, then the **smbtab**, **dfstab**, and **devtab** entries might be:

smbtab:

```
/dev/ufs2  myshare  ufs  "My share description"  r/w  100  /ghi
```

dfstab:

```
/dev/ufs2  hfs2  ufs  101  0,,1715
```

devtab:

```
define_ufs 2
OMVS.ABC.DEF
```

Notice that you did not need to export the file systems that are above the OMVS.ABC.DEF file system. That is, you did not need to create **dfstab** and **devtab** entries for OMVS.ROOT and OMVS.ABC.

If, however, there were additional file systems below the shared directory, you may want to export those by specifying them in the **dfstab** and **devtab**, since PC users may want to access those directories and files. Alternatively, you can use the dynamic export capability. Refer to "Dynamic export for HFS" on page 38.

As another example, if you wanted to share the entire file hierarchy with PC users from the root on down, then the **smbtab**, **dfstab**, and **devtab** entries might be:

smbtab:

```
/dev/ufs4  hfsroot  ufs  "My share description"  r/w  100  /
```

dfstab:

```
/dev/ufs2  hfs2  ufs  101  0,,1715
/dev/ufs3  hfs3  ufs  102  0,,1718
/dev/ufs4  hfs4  ufs  103  0,,1721
/dev/ufs5  hfs5  ufs  104  0,,1724
/dev/ufs6  hfs6  ufs  105  0,,1727
```

devtab:

```
define_ufs 2
OMVS.ABC.DEF
define_ufs 3
OMVS.ABC
define_ufs 4
OMVS.ROOT
```

```
define_ufs 5
OMVS.BIN
define_ufs 6
OMVS.U
```

Notice that only one share is required (in **smbtab**) since only one directory is being shared but all the HFS file systems must be exported (by including them in the **dfstab** and **devtab** or by using dynamic export) since we want the PC user to be able to reference any file or directory below the root directory. If you are including them in **dfstab** and **devtab**, you should issue the **dfsexport -all** command to ensure that all the file systems are exported before the **dfsshare -share hfsroot** command is issued to share the directory. (For more information, refer to “dfsexport” on page 106 and “dfsshare” on page 109.)

Creating a shared directory for HFS

This section describes the steps that are involved in creating a shared directory for HFS data. The HFS file system must be locally mounted on the system. Refer to the *z/OS: UNIX System Services Planning* document, GA22-7800, for information on how to create and mount an HFS file system. Refer to the *z/OS: Distributed File Service zSeries File System Administration*, for information on how to create and mount a ZFS file system.

To create a shared directory, perform the following steps:

1. Choose the HFS directory that you want to share. (For example, **/abc/def/ghi**)
2. Determine the HFS file system data set name that the directory **/abc/def/ghi** resides in. If you do not know the HFS file system data set name, the z/OS UNIX **df** command can help with this task. The **df** command displays file system data set names and their mount points. Refer to the *z/OS: UNIX System Services Command Reference*, SA22-7802, for information on the **df** command. (For example, data set OMVS.ABC.DEF might be mounted on **/abc/def.**)

   The following is an example of the **df** command and may help with this determination:

   ```
   # df /abc/def/ghi
   Mounted on      Filesystem          Avail/Total   Files      Status
   /abc/def        (OMVS.ABC.DEF)      544/1440      4294967295 Available
   ```

   This shows the mount point and the HFS file system data set name of the file system mounted on that mount point. The HFS file system data set name is in parentheses (in this example, **OMVS.ABC.DEF**).
3. If there is no entry in **devtab** for this HFS file system, you must add an entry in **/opt/dfslocal/var/dfs/devtab** which maps a unique minor device number to the HFS file system you want to export and share. Choose a unique minor device number (for example, **2**) that is not in any other **define_ufs** statement and put it in the **define_ufs** statement. Put the HFS file system name (in this example, **OMVS.ABC.DEF**) on the next line. An example of an entry for an HFS file system might look like the following:

   ```
   * HFS devices
   define_ufs 2
   OMVS.ABC.DEF
   ```
4. If there is an entry in **devtab** for this HFS file system, you must add a corresponding entry in **dfstab**. Use the same minor device number (in this example, **2**) in the device parameter (so in this example, it would be /dev/ufs**2**). Use a unique file system name (for example, **hfs2**), a file system type of **ufs** and a unique file system ID (for example, **101**). Finally, use a unique fileset ID (for example, **0,,1715**). An example **dfstab** entry might look like this:

   ```
   /dev/ufs2  hfs2  ufs  101  0,,1715
   ```

   Note: If you are using only SMB protocols (and not DCE DFS protocols[4]), you can use the same number for all the numeric values in a **dfstab** entry as long as the number used is unique. For example, the **dfstab** entry might look like this:

4. If you are using both SMB and DCE DFS protocols, the file system ID must be assigned by the **fts crfldbentry** command. Refer to Appendix C, “Using both SMB and DCE DFS” on page 139 for more information.

```
/dev/ufs2  hfs2  ufs  2  0,,2
```

5. Add an entry in **smbtab** for the directory you want to share. Use the same minor device number (in this example, **2**) in the device name parameter (so in this example, it would be /dev/ufs**2**). Choose a unique share name (for example, **myshare**), a file system type of **ufs**, a description, share permission (for example, **r/w**), maximum users (for example, **100**) and the directory name (in this example **/ghi**). The directory name is relative to the root of the file system that the device name refers to.

   ```
   /dev/ufs2  myshare  ufs  "My share description"  r/w  100  /ghi
   ```

6. Issue the **dfsshare** command to cause the new share to be made available to PC users.

   ```
   # dfsshare -share myshare
   ```

At this point, a PC user can connect to this share.

If there were additional file systems below the shared directory, you may want to export those too since PC users may want to access those directories and files. That is, if there were one or more subdirectories below the **ghi** directory, and there were one or more file systems mounted on those directories or their subdirectories, those file systems could be exported to make them available to PC users of the share named **myshare**. Those file systems would be specified in **dfstab** and **devtab** with different unique minor device numbers. Alternatively, you can use the dynamic export capability. Refer to “Dynamic export for HFS”.

If there was an additional directory at the same level as the **ghi** directory and you wanted to share that directory, the minor device number would be the same as the share for the **ghi** directory (that is, it would be **2**) since that directory is also contained in the OMVS.ABC.DEF file system.

If you mount (or unmount) a file system on (from) a directory that is in the directory path specified on an **smbtab** entry and you have already shared the directory, then you should unshare (detach) and reshare the shared directory using the **dfsshare** command.

Removing a shared directory for HFS

A shared directory may be made unavailable by issuing the **dfsshare** command with the **-detach** option. For example, if you want to stop the shared directory named myshare from being available to PC clients, you would issue the following **dfsshare** command:

```
# dfsshare -share myshare -detach
```

This command makes the shared directory unavailable until the SMB server is restarted or the **dfsshare** command is issued to make it available. Since the shared directory is still in the **smbtab**, the shared directory would be made available again to PC clients. In order to make the shared directory permanently unavailable, it must be removed from the **smbtab** file.

Making a shared directory unavailable does not affect whether the underlying file systems are exported (that is, they remain exported). There may be other shares that apply to those underlying file systems. The **dfsexport** command may be used to unexport file systems.

Dynamic export for HFS

Previously, in order to make HFS file systems available to PC users via the SMB server, the administrator had to create a **dfstab** and **devtab** entry for each HFS file system. This allows the SMB server to export them at SMB server start up or on command. HFS file systems must also be mounted at the time the SMB server exports them. This meant that the SMB server could not support HFS automounted file systems.

The SMB server now has an optional function called dynamic export. Usage of dynamic export allows the SMB server to support HFS automounted file systems. That is, when dynamic export is used, PC clients are able to access data in HFS file systems that are automounted. (Refer to *z/OS: UNIX System Services Planning* document, GA22-7800 for information on automounted file systems.)

Note: The SMB server attempts to move the ownership of a shared HFS file system to the system that the SMB server is running on when the shared HFS file system is owned by a different system if the **_IOE_MOVE_SHARED_FILESYSTEM** environment variable is **ON** in the **dfskern** process. If this is unsuccessful (when, for example, the file system is being exported by another SMB server on the other system), the SMB server will not be able to export that file system and it will not be available to PC clients.

An environment variable (**_IOE_DYNAMIC_EXPORT=ON**) in the **dfskern** process, enables dynamic export. When dynamic export is enabled, the SMB server can ″discover″ file systems that are mounted but not yet exported and can dynamically export them. A file system is discovered by the SMB server when a PC user references a directory (for example, via the **cd** command) that is a mount point. This causes the SMB server to ″cross into″ the mounted file system. (The capability to cross file systems is controlled by the **_IOE_SMB_CROSS_MOUNTS** environment variable in the **dfskern** process.) If it is determined that the file system is not yet exported, the SMB server (dynamically) exports it. The SMB server then continues to handle the PC user request. This dynamic export occurs even though there may be no **dfstab** or **devtab** entry for the file system. The information that would be in the **dfstab** and **devtab** entry can be determined (or assigned) by the SMB server. Note that when a PC user references a remote directory that is under control of automount, this causes the SMB server to reference the directory and this in turn causes the correct file system to be automounted. So, the file system is mounted by the time the SMB server gets control back from referencing it.

Dynamic export is actually independent of whether automount is used. That is, the discovery and dynamic export of file systems occurs whether the file system was automounted or statically mounted. This means that if you do not specify **dfstab** and **devtab** entries for file systems that are ″crossed into″, the SMB server takes care of those file systems on its own. The only file systems that must have a **dfstab** and **devtab** entry are file systems that are the root of a share (that is, file systems that are represented as the first / in the **smbtab** Directory Path Name entry).

As a simple case, when dynamic export is used, it is possible to make the entire HFS tree (including automounted file systems) available to PC users by specifying / of the root file system as the shared directory in **smbtab** and the root file system in **dfstab** and **devtab**. (The root file system is also referred to as the version file system.) No other entries are required. Of course, PC users can only access data that they are authorized to. Also, they cannot write to file systems that are mounted read-only. Here is an example of what the **smbtab**, **dfstab**, and **devtab** entries might look like:

smbtab:

```
/dev/ufs4  hfsroot  ufs  "My share description"  r/w  100  /
```

dfstab:

```
/dev/ufs4  hfs4  ufs  103  0,,1721
```

devtab:

```
define_ufs 4
OMVS.ROOT
```

When dynamic export is used, it is still allowable for the administrator to specify **dfstab** and **devtab** entries. This allows you to use your current **dfstab** and **devtab** entries and still take advantage of the dynamic export function. When dynamic export assigns numbers for file systems, it uses large numbers such as:

- Minor device numbers start at 10000
- File system IDs start at 100000
- Fileset IDs start at 100000.

If you add **dfstab** and **devtab** entries, you should use numbers that are lower than these so as not to interfere with dynamically assigned numbers.

In addition, a **devtab** entry can be specified without a **dfstab** entry to set the translation option for a file system. This entry is used when the SMB server crosses into that file system and dynamically exports it.

Alternatively, the translation option can be inherited from the parent file system when there is no **devtab** entry. The **_IOE_INHERIT_TRANSLATION=ON** environment variable in the **dfskern** process controls whether the file system translation option can be inherited from the parent file system.

If a file system is not referenced for a period of time (and you are using dynamic export), the administrator has the ability to control whether the SMB server should dynamically unexport a file system if it has not been referenced for a period of time. Only file systems that are not the root of a share are dynamically unexported. The **_IOE_EXPORT_TIMEOUT** environment variable in the **dfskern** process is used to specify this. Unexporting an unreferenced file system frees resources in the SMB server. Later, if the file system is referenced again, it is dynamically exported again. There is a relationship between the SMB server export timeout value and the z/OS Automount Facility **delay** timeout value. Automount waits (at least) the automount **delay** timeout period after the file system has been dynamically unexported before attempting to unmount it again.

As the PC user **cd**'s down the tree, file systems of different file system types may be encountered. Some are supported by the SMB server and some are not. The file system types that are supported by the SMB server are HFS, ZFS, TFS, and AUTOMNT. If you are in a sysplex with Shared HFS, SMB support of ZFS is limited to ZFS compatibility mode file systems. The NFS and DFSC file system types are not supported.

If you are using DCE DFS and SMB (or DCE DFS alone), you cannot use the dynamic export capability. Dynamic export is disabled when **_IOE_PROTOCOL_RPC=ON**.

Working with automounted file systems and home directories:

It is common for an OMVS user's home directory to be automounted. This means that the user's home directory does not exist and the file system is not mounted until the home directory is referenced. Refer to the *z/OS: UNIX System Services Planning* document, GA22-7800, for information on automount. Using dynamic export, you can specify the root of the automount file system as the shared directory. For example, consider the following configuration files:

Automount Facility Master File:

```
/u          /etc/home.map
```

MapName Policy File: (/etc/home.map)

```
Name         *
Type         HFS
Filesystem   OMVS.HOME.<uc_name>
Mode         rdwr
Duration     60
Delay        0
```

smbidmap:

```
smith
cmsmith

jones
TSJONES
```

smbtab:

```
/dev/ufs10  homeshare  ufs  "Root of Home Directories"  r/w  100  /
```

dfstab:

```
/dev/ufs10  hfs10  ufs  10  0,,10
```

devtab:

```
define_ufs 10
"*AMD/u"  text
```

Note: Notice that the `cmsmith` z/OS user ID is in lowercase letters. The case of the z/OS user ID in the **smbidmap** file has implications on directories that are mount points for automounted file systems and used in **smbtab** entries with **&USERID**. Use the same case for the z/OS user ID in the

smbidmap file as that used in the user ID portion of the user's home directory. (The z/OS user ID will be folded to upper case when it is used to logon to z/OS.)

With this configuration, a user could connect to the shared directory named homeshare (**net use h: \\computer1\homeshare**), and she could then **cd** to her home directory (**cd h:\cmsmith**). This causes automount and dynamic export of the user's home file system to occur.

However, this has several possible usage problems:

- When a user connects to the shared directory, the shared directory is referenced (for example, /u on OMVS), but the actual home directory is not referenced (for example, **/u/cmsmith** on OMVS). Therefore, the cmsmith directory is not listed if a user does a **DIR** from an MS-DOS window or uses Windows Explorer to click on the shared directory drive letter (since the home directory does not exist yet). The home directory is not created by the Automount Facility until it is directly referenced by name. This requires the user to **cd** to the home directory in an MS-DOS window. (If the user tries to create a new folder with the home directory name, Windows first tries to create a folder by the name of **New Folder**. This causes the Automount Facility to try to mount a file system called **OMVS.HOME.NEW FOLDER** and this is an invalid name.)
- In addition, the name that the user needs to **cd** to is not the Windows user ID but rather the z/OS user ID (the z/OS user ID that is mapped by the **smbidmap** file; or the guest user ID). This name may not be obvious to the PC user.
- Even after the user issues **cd** to the correct directory, the file system may get unexported and unmounted after a while if there is no access to the file system for a period of time. This may put the user back into the situation where her home directory does not exist again.

To resolve this, you could create a separate shared directory for each user that points directly to the user's home directory. For example:

smbtab:

```
/dev/ufs10  smith  ufs  "Smith's Home Directory"  r/w  100  /cmsmith
/dev/ufs10  jones  ufs  "Jones' Home Directory"   r/w  100  /tsjones
             .
             .
             .
```

dfstab:

```
/dev/ufs10  hfs10  ufs  10  0,,10
```

devtab:

```
define_ufs 10
"*AMD/u"  text
```

This technique also has several disadvantages:

- You need a separate **smbtab** entry for each user. As users are added and deleted from the system, you must add or delete an **smbtab** entry.
- Since each shared directory resides in a corresponding automounted file system, none of those file systems ever get unexported (or unmounted) after they are referenced.
- Each PC user must net use to a different shared directory name.

Recommended technique for PC user access to automounted home directories

The SMB server defines a special keyword to be used in the **Directory path name** in an **smbtab** entry. The keyword is **&USERID**. It represents the PC user's z/OS user ID. This keyword allows a single shared directory (that is, a single **smbtab** entry) to mean a different directory based on the z/OS user ID of the PC user connecting to the shared directory name. When a user connects to this shared directory name, a new (special) share is be created. This share has a connection reference count that is incremented when the PC user connects to the shared directory name. The reference count is decremented when the PC user disconnects from the shared directory name by issuing a **net use h: /d**. When the reference count is zero, the user's home file system is eligible to be unexported by the SMB server and then unmounted by

the Automount Facility. The user would need to connect to the shared directory name again to access the home directory. In addition, if the user was inactive on the session for the **_IOE_SMB_IDLE_TIMEOUT** time period, the session would be disconnected. This would also decrement the reference count and if zero, an unexport and unmount would occur. In this case, when the user references the drive letter again, the mount and export would occur automatically. For example, with the following entries:

smbtab:

```
/dev/ufs10  myhomedir  ufs  "Each user's Home Directory"    r/w  100  /&USERID
/dev/ufs10  homeshare  ufs  "Root of All Home Directories"  r/w  100  /
```

dfstab:

```
/dev/ufs10  hfs10  ufs  10  0,,10
```

devtab:

```
define_ufs 10
"*AMD/u"  text
```

User smith could issue the following **net use** command:

```
net use h: \\computer1\myhomedir
```

This would cause the SMB server to reference the **/u/cmsmith** directory. This would cause the Automount Facility to create the cmsmith directory in the *AMD/u file system. Then the Automount Facility would mount smith's home directory file system on the **/u/cmsmith** directory. The SMB server would then export smith's home directory file system and then create a (special) share and increment the reference count. PC user smith would then be able to issue **DIR** in an MS-DOS window for the drive letter or use Windows Explorer to click on the drive letter to see the contents of the home directory.

If necessary, smith could still access jones' home directory (assuming proper authorization) by issuing **net use x: \\computer1\homeshare** and then **cd x:\TSJONES**.

Note: The z/OS user ID specified in the **smbidmap** file determines the directory name (with the exact case) that will be substituted for the **&USERID** keyword in the **smbtab**. If a **net use** to that shared directory in the **smbtab** causes the automount to occur, it will also cause the directory to be created in the *AMD/u file system with that exact case. You should specify the z/OS user ID in the **smbidmap** file exactly as the user ID in the user's home directory is specified. Otherwise, you may create a directory that does not match the user's home directory and that user will not be able to log on locally to OMVS.

File data translation for HFS

Distributed File Service SMB support provides basic support for character data translation. There are several mechanisms provided to allow an administrator to specify when file data should be translated. Character data translation can be:

- Based on the HFS file tag (if Enhanced ASCII is active in the SMB server, that is, the **_IOE_HFS_FILETAG** environment variable is set to **QUERY** or **SET** in **dfskern**),
- Based on the HFS file format attribute,
- A global specification based on the file name suffix (**hfsattr** file),
- Specified on a file system basis (**devtab** *hfs-data-set-name* **text** or **binary** option),
- Specified on a file system basis based on file's contents (**devtab** *hfs-data-set-name* **auto** option),
- Inherited from a parent file system's translation option (**_IOE_INHERIT_TRANSLATION** environment variable is set to **ON** in **dfskern**),
- A global specification to translate or not for all HFS file systems exported (**_IOE_HFS_TRANSLATION** environment variable set to **ON** or **OFF** in **dfskern**),
- A global specification based on file's contents for all HFS file systems exported (**_IOE_HFS_TRANSLATION** environment variable set to **AUTO** in **dfskern**).

The following sequence determines whether file data is translated:

1. If the file exists and Enhanced ASCII is active in the SMB server (**_IOE_HFS_FILETAG** is **QUERY** or **SET** in the **dfskern** process) and the file is tagged, then when the tag indicates binary, no translation is done. When the tag indicates text, translation is done based on the codepage in the tag.
2. If the file exists and has no file tag (or Enhanced ASCII is not active in the SMB server) and has a non-zero file format attribute, then a value of 1 (meaning binary) causes no translation to be done. A value greater than 1 causes translation to be done. Refer to the *z/OS: UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803, for information on the **stat** and the **chattr** callable services. These callable services can be used to query or change a file format attribute. The SMB server does not convert between different end of line delimiters.

   Note: The OMVS ISPF Shell (IShell) can be used to query or set the file format attribute for a file. Choose a file and then choose attributes. This shows the file format attribute (NA, Binary, NL, CR, LF, CRLF, LFCR, CRNL). Choose Edit and then File Format. You can then change the file format. The IShell uses choice numbers for the file format attributes that are one greater than those used for the actual file format value (that is, a file format of binary has a value of 1 but the Binary choice in the IShell is 2.
3. If the file does not exist or has a zero file format attribute, then, if an **hfsattr** file is specified (by the **dfskern _IOE_HFS_ATTRIBUTES_FILE** environment variable), and the file name suffix matches a directive in the **hfsattr** file, then that controls whether translation occurs. Refer to “hfsattr” on page 94.
4. The file system’s **devtab** translation option (**text**, **binary**, or **auto**) is used when no **hfsattr** file is specified, or if the file name suffix does not match any of the directives in the **hfsattr** file, or the file name has no suffix. Refer to “devtab” on page 87.
5. The file system’s inherited translation option is used if **_IOE_INHERIT_TRANSLATION** is **ON** in the **dfskern** process and no **devtab** translation option is specified for the file system. Refer to “Dynamic export for HFS” on page 38 and to the **_IOE_INHERIT_TRANSLATION** environment variable on page 121.
6. The **dfskern _IOE_HFS_TRANSLATION** environment variable is used if the file system does not inherit a translation option. Refer to the **_IOE_HFS_TRANSLATION** environment variable on page 120.

The file format attribute for HFS files is set if it is not already set, and either an **hfsattr** file was used to determine whether to translate or **auto** (on the file system or globally) was used to determine whether to translate. If the file is determined to be text, the file format attribute is set to 2 (NL). If the file is determined to be binary, the file format attribute is set to 1 (Binary). In addition, if the file is determined to be text, the file is being created and the **_IOE_HFS_FILETAG** environment variable in the **dfskern** process equals **SET**, the file tag is set to the codepage from the **MOUNT TAG** (if it exists) or the local codepage of the **dfskern** process.

Translation is done using ISO8859-1 for network data (or the codepage specified in the **_IOE_WIRE_CODEPAGE** environment variable of **dfskern**) and the local code page for the **dfskern** process is used for data in HFS.

You should use caution if you change an HFS **devtab** option from **binary** to **text** or from **text** to **binary**. If you have already stored a file under one option (for example, **text**), and then you change the option (to, for example, **binary**), the data has been translated from ASCII to EBCDIC when it was originally written to HFS, but is not translated back from EBCDIC to ASCII when it is read. This causes the data to appear garbled to the PC user. Also note that if you are using dynamic export and translation inheritance, the translation option is effectively changed if you first mount the file system (that has no **devtab** translation option) under a text file system and later under a binary file system

Authorization for HFS

When a PC user attempts to access a directory or file, the normal HFS file authorization mechanism is used (including Access Control Lists (ACLs)). The PC user’s SMB user ID is mapped to a local z/OS user ID and that z/OS user ID is used to determine if the user is authorized to the file or directory. Refer to Chapter 6, “Mapping SMB user IDs to z/OS user IDs” on page 29 for information on how SMB users are

mapped to z/OS users. A z/OS user ID's authorization to a directory or file is determined by the permission bits and the user and group IDs of the directory or file. Refer to the *z/OS: UNIX System Services User's Guide*, SA22-7801, for more information on HFS security.

The only authorization that a PC user can directly change is the write (w) permission of a file. It can be changed by modifying the read-only attribute of a file. Setting the read-only attribute on turns the write permission off, and vice versa. To change the read-only attribute, use the **attrib** command or right click on the file from Windows Explorer and choose Properties. The PC user must be the owner of the file (or must have a **UID=0**) to change the write permission. The read-only attribute for a directory does not stop a user from creating or deleting files in the directory. So, modifying the read-only attribute of a directory does not change the write permission of a directory and is in effect, ignored by the SMB server.

Note: When the read-only attribute is turned on (which turns off write permission), all the write permissions (for user, group and other) are turned off. When the read-only attribute is turned off (which turns on write permission), only those write permissions that intersect with the default create permissions are turned on. Default create permissions are specified in the **smbtab** entry for the shared directory or globally, on the **_IOE_SMB_DIR_PERMS** and the **_IOE_SMB_FILE_PERMS** environment variables.

From experimentation, it appears that when a read-only file is copied via drag and drop, the read-only attribute is maintained on the new file. However, when the MS-DOS **copy** command is used, the read-only attribute is not maintained.

When a PC user does not have read and execute permission to a directory, the contents of the directory is not listed. The user is allowed to **cd** to the directory but when a **dir** is issued, access is denied or the contents of the directory shows up as if it were empty. The user is not allowed to access files in that directory. When a PC user does not have write and execute permission to a directory, they are not able to create, erase, or rename a file (or directory) that is contained in that directory. When a PC user does not have read permission to a file, they are not able to read (or execute) the file. When a PC user does not have write permission to a file, they are not able to change the contents of the file.

Free space for HFS

The amount of free space that is reported for a drive letter that is mapped to a shared directory that resides in an HFS file system is based on the amount of free space in the file system that contains the shared directory. That is, if you cross into a lower file system by referencing subdirectories that reside in a lower file system, the free space does not reflect the amount of free space in the lower file system. Rather, the free space left in the file system that contains the shared directory is reported.

Sharing RFS files

This section discusses the RFS file system.

Note: The RFS file system supports reading and writing but it is more suitable for applications that only read files. (You can limit an RFS shared directory to read-only access by making it read-only in the **smbtab**.) Applications can write to RFS files, but there are many restrictions. An application that writes to or creates RFS files must abide by all the restrictions imposed by RFS. Refer to Appendix E, "Using data sets" on page 143 for a full description of RFS considerations.

Exporting and sharing RFS file systems

A Record File System (RFS) is a collection of data sets with a common data set name prefix. The data sets supported include sequential data sets (on DASD), partitioned data sets (PDS), partitioned data sets extended (PDSE) and Virtual Storage Access Method (VSAM) data sets. These data sets are then exported by the SMB server as a single "file system". An RFS file system is not locally mounted so it is not part of the HFS hierarchy. It can still be exported by the SMB server, even though it is not part of the HFS

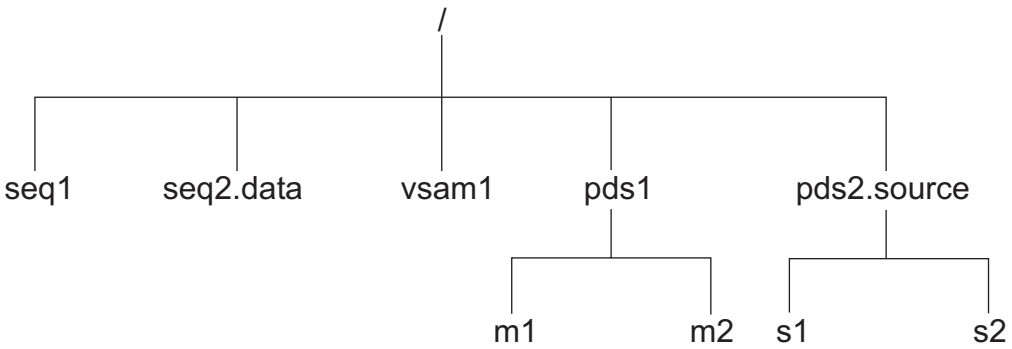
hierarchy. An RFS file system must be exported by the SMB server before a directory contained in it can be shared. Exporting is done on an RFS file system basis.

Sharing is done on a directory tree basis and allows PC clients to access data in an RFS file system. In order to allow a directory to be shared, the file system that the directory is contained in must be exported.

Suppose we had the following data sets on a z/OS system:

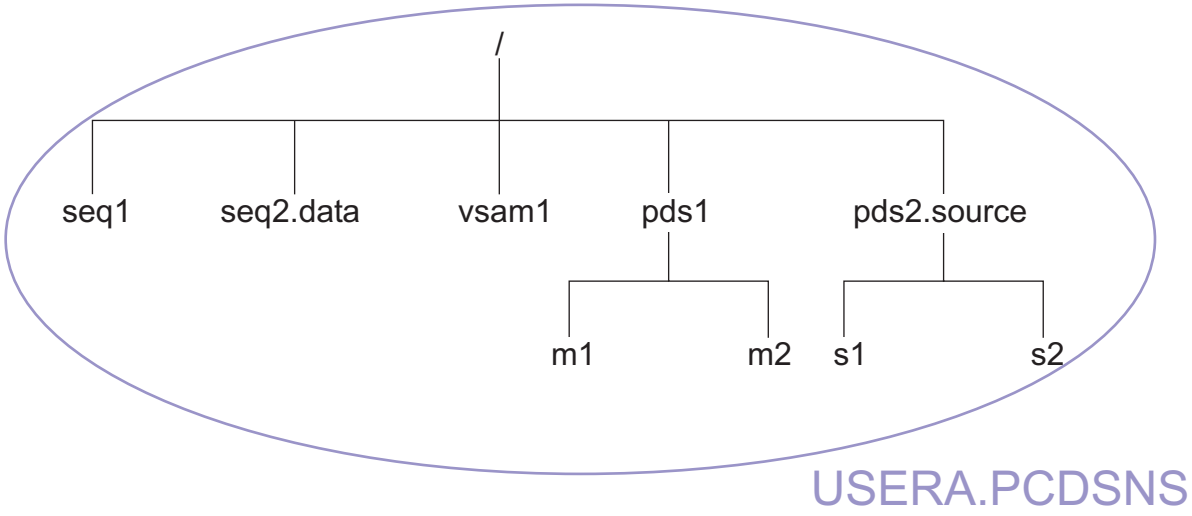
USERA.PCDSNS.SEQ1
USERA.PCDSNS.SEQ2.DATA
USERA.PCDSNS.VSAM1
USERA.PCDSNS.PDS1() with members M1 and M2
USERA.PCDSNS.PDS2.SOURCE() with members S1 and S2

These data sets would be represented by the following RFS hierarchy:



This hierarchy is made up of data sets that all begin with the same prefix (USERA.PCDSNS). Notice that the RFS file names do not include the prefix. The files are all directly below the root of the file system. PDSs and PDSEs appear as directories with their members as files in the directory. All file and directory names appear in lower case. (This is controlled by a setting in the **rfstab** file for this file system.)

The RFS file system is defined by the data set name prefix. The RFS file system includes all data sets that begin with that data set name prefix. Refer to the following representation of this:



smbtab, dfstab, and devtab entries for RFS

The files that the SMB server uses to relate the shared directory and its exported file system are the **smbtab**, the **dfstab**, and the **devtab**. (They are all located in **/opt/dfslocal/var/dfs**.) A common **minor**

device number is used to tie related entries in these three files together. The **smbtab** is used to define the shared directory. The **dfstab** and the **devtab** are used to define the file system to be exported.

In **smbtab**, the minor device number is specified using the format **/dev/ufs***n*, where *n* is a locally assigned unique minor device number that must refer to the file system data set where the root of the shared directory path name resides (that is the file system where the first / resides). This should always be the file system where the directory that you want to share resides.

In **dfstab**, the minor device number is specified using the format **/dev/ufs***n* to identify the assigned unique identifiers for the file system.

In **devtab**, the minor device number is specified using the format **define_ufs** *n* to identify the data set name prefix for the RFS file system.

If the shared directory is /, then the **smbtab**, **dfstab**, and **devtab** entries might be:

smbtab:

```
/dev/ufs10  myrfsshare  ufs  "My rfs share description"  r/w  100  /
```

dfstab:

```
/dev/ufs10  rfs10  ufs  110  0,,1730
```

devtab:

```
define_ufs 10 rfs
USERA.PCDSNS
```

Creating a shared directory for RFS

This section describes the steps that are involved in creating a shared directory for RFS data. Refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410, for information on how to use data sets. (RFS file systems are not locally mounted in the HFS hierarchy.)

To create a shared directory, perform the following steps:

1. Choose a set of data sets with a common data set name prefix that you want to share with PC clients (for example, **USERA.PCDSNS**).
2. Choose the RFS directory that you want to share. Usually, it is **/. /** is a virtual directory that is the root of the RFS file system and contains all the data sets that begin with the common data set name prefix.
3. Add an entry to the **devtab** (**/opt/dfslocal/var/dfs/devtab**) for this RFS file system. This entry maps a unique minor device number to the RFS file system you want to export and share. Choose a unique minor device number (for example, **10**) that is not in any other **define_ufs** statement and put it in the **define_ufs** statement for this RFS file system. Put the RFS file system data set name prefix (in this example, **USERA.PCDSNS**) on the next line. An example entry for an RFS file system might look like the following:

   ```
   * RFS devices
   define_ufs 10 rfs
   USERA.PCDSNS
   ```

4. You must add a corresponding entry in **dfstab** (**/opt/dfslocal/var/dfs/dfstab**) for this RFS file system. Use the same minor device number (in this example, **10**) in the device parameter (so in this example, it would be /dev/ufs**10**). Use a unique file system name (for example, **rfs10**), a file system type of ufs and a unique file system ID (for example, **110**). Finally, use a unique fileset ID (for example, **0,,1730**). An example **dfstab** entry might look like this:

   ```
   /dev/ufs10  rfs10  ufs  110  0,,1730
   ```

Note: If you are using only SMB protocols (and not DCE DFS protocols[5]), you can use the same number for all the numeric values in a **dfstab** entry as long as the number is unique. For example, the **dfstab** entry might look like this:

```
/dev/ufs10  rfs10  ufs  10  0,,10
```

5. Add an entry in **smbtab** (**/opt/dfslocal/var/dfs/smbtab**) for the directory you want to share (the directory you chose in Step 2 on page 46). Use the same minor device number (in this example, **10**) in the device parameter (so in this example, it would be /dev/ufs**10**). Choose a unique share name (for example, **myrfsshare**), a file system type of **ufs**, a description, share permission (for example, **r/w**), maximum users (for example, **100**) and the directory name (in this example, **/**). For example, the **smbtab** entry might look like this:

   ```
   /dev/ufs10  myrfsshare  ufs  "My rfs share description"  r/w  100  /
   ```

6. Issue the **dfsshare** command

   ```
   # dfsshare -share myrfsshare
   ```

At this point, a PC user can connect to this share.

Removing a shared directory for RFS

A shared directory may be made unavailable by issuing the **dfsshare** command with the **-detach** option. For example, if you want to stop the shared directory named **myrfsshare** from being available to PC clients, you would issue the following **dfsshare** command:

```
# dfsshare -share myrfsshare -detach
```

This command makes the shared directory unavailable until the SMB server is restarted or the **dfsshare** command is issued to make it available. Since the shared directory is still in the **smbtab**, the shared directory would be made available again to PC clients. In order to make the shared directory permanently unavailable, it must be removed from the **smbtab** file.

Making a shared directory unavailable does not affect whether the underlying file systems are exported (that is, they remain exported). There may be other shares that apply to those underlying file systems. The **dfsexport** command may be used to unexport file systems.

File data translation for RFS

Distributed File Service SMB support provides basic support for character data translation. Character data translation can be specified globally (that is, for all RFS file systems) or on a per RFS file system basis. Character data translation can be:

- Specified on an RFS file system basis (**devtab** *rfs_data_set_name_prefix* **text** or **binary** option),
- Specified in the attributes file (**rfstab**) on an RFS file system basis (**devtab** *rfs_data_set_name_prefix* **attrfile** option),
- Specified in a global attributes file (**rfstab**) (**_IOE_RFS_ATTRIBUTES_FILE** environment variable in the **dfskern** process),
- Specified globally (**_IOE_RFS_TRANSLATION** environment variable in the **dfskern** process).

The following sequence determines whether RFS file data is translated:

1. The RFS file system's **devtab** translation parameter (**text** or **binary**) is used to determine if translation is done.
2. If there is no translation parameter on the **devtab** for the RFS file system, then translation is controlled by the attributes file (**rfstab**) for the RFS file system's **text** or **binary** specification.
3. If there is no attributes file (**rfstab**) for the RFS file system, or if there is no **text** or **binary** specification in the **rfstab**, then translation is controlled by the global attributes file (**rfstab**). The global attributes file is specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable of the **dfskern** process.

---

5. If you are using SMB and DCE DFS protocols, the file system ID must be assigned by the **fts crfldbentry** command. Refer to Appendix C, "Using both SMB and DCE DFS" on page 139 for more information.

4. If there is no global attributes file (**rfstab**), then translation is controlled by the global translation specification for RFS file systems. The global translation specification for RFS file systems is specified in the **_IOE_RFS_TRANSLATION** environment variable of the **dfskern** process.

Translation is done using ISO8859-1 for network data (or the codepage specified in the **_IOE_WIRE_CODEPAGE** environment variable of **dfskern**) and the local code page for the **dfskern** process is used for data in RFS.

When translation of RFS data occurs, the end of line character used when the data set records are converted to byte stream data is controlled by the **attributes** file (**rfstab lf** or **crlf** entry) that is controlling the RFS file system.

You should use caution if you change a translation option from **binary** to **text** or from **text** to **binary**. If you have already stored data under one option (for example, **text**), and then you change the option (to, for example, **binary**), the data has been translated from ASCII to EBCDIC when it was stored in RFS, but is not translated back from EBCDIC to ASCII when it is read. This causes the data to appear garbled to the PC user.

Authorization for RFS

When a PC user attempts to access a directory or file, the normal data set authorization mechanism is used. The PC user's SMB user ID is mapped to a local z/OS user ID and that z/OS user ID is used to determine if the user is authorized to the data set. Refer to Chapter 6, "Mapping SMB user IDs to z/OS user IDs" on page 29 for information on how SMB users are mapped to z/OS users. A z/OS user ID's authorization to a data set is determined by the local security subsystem (for example, RACF).

A PC user cannot directly change any attributes of an RFS file. Attempting to change an RFS file attribute using the **attrib** command or right clicking on the file from Windows Explorer and choosing Properties appears successful but is ignored by the SMB server.

When a PC user does not have authority to a PDS, the contents of the directory is not listed. The user is allowed to **cd** to the directory but when a **dir** is issued, access is denied or the contents of the directory shows up as if it were empty. The user is not allowed to access files in that directory. When a PC user does not have update authority to a PDS, they are not able to create new members or delete or update existing members. When a PC user does not have read authority to an RFS file, they are not able to read the file. When a PC user does not have update authority to an RFS file, they are not able to change the contents of the file.

Free space for RFS

The amount of free space that is reported for a drive letter that is mapped to a shared directory that resides in an RFS file system is a fabricated number. This is due to the fact that there really is no RFS file system, but only a set of data sets with the same data set name prefix. The SMB server always reports a capacity of 122,880,000 bytes with half (61,440,000 bytes) available. Free space for RFS is actually controlled by various aspects of the system such as z/OS DFSMS System Managed Storage, free space on a volume, primary and secondary allocations, etc.

Logon considerations

In order for you to create a session and access resources (files or printers) from a PC, the SMB server must be able to identify you in terms of a local z/OS user ID. **This session creation occurs on the first command or communication with the SMB server**. The SMB server can identify you either by authenticating you by a user ID and password that you supply, or if that fails, by allowing you to run unauthenticated with a guest z/OS user identification. If both of these fail, your session is denied.

1. In order to authenticate you, the SMB server must receive an SMB user ID and password. The SMB user ID that is received is normally the user ID you specified when you logged on to your PC. You can, however, specify a user ID on a Windows NT/2000 **net use** command or the Windows NT Explorer

Tools pull down, Map Network Drive selection, Connect As field. The SMB user ID that is received must be mapped to a z/OS user ID. This mapping is accomplished through the **smbidmap** file. Refer to Chapter 6, “Mapping SMB user IDs to z/OS user IDs” on page 29 for information on the **smbidmap** file.

The password is either a clear text password or it is an encrypted password. The type of password expected by the SMB server depends on whether passthrough authentication is enabled (refer to the **_IOE_SMB_AUTH_SERVER dfskern** environment variable) and if not, the setting of the **_IOE_SMB_CLEAR_PW** environment variable for the **dfskern** process.

a. When passthrough authentication is enabled, this means that when the SMB server receives a logon request, the SMB server forwards the logon request to a Windows NT or 2000 Domain Controller for authentication. In this case, the user ID and password must be the user’s Windows Domain user ID and password. If the Domain Controller successfully authenticates the user, then the SMB server maps the SMB user ID to a local z/OS user ID using the **smbidmap** file.

b. If **_IOE_SMB_CLEAR_PW** is set to **REQUIRED** (or it is unspecified), then the SMB server tells your PC to send a clear password for authentication. This password needs to be your z/OS password. Normally, your PC sends the password that you specified when you logged on to your PC and therefore, it must match your z/OS password.

   Some levels of Windows do not send a clear password unless a Registry entry is added. Refer to Appendix B, “Additional information using the SMB server” on page 133 for more information on this topic.

   If, however, you specify a password on a **net use** command, then that is sent to the SMB server and that password must match your z/OS password. In this case, your PC logon password and your z/OS password do not need to be the same.

   On Windows NT/2000, if the password is not specified on **net use**, you may be prompted for a password (so Windows NT can obtain a clear text password) and that password is sent to the SMB server. Some commands do not prompt (for example, **net view**) and therefore fails to create a session. You should do a **net use** or Find Computer as the first communication in order to be prompted for the password and create a session.

c. If **_IOE_SMB_CLEAR_PW** is set to **NOTALLOWED**, then the SMB server tells your PC to send an encrypted form of the password[6]. Your PC encrypts the password that you specified when you logged on to your PC. Because the SMB server needs to determine your actual SMB password for authentication, your PC logon password must be stored into your RACF DCE segment[7] by using the OMVS **smbpw** command prior to creating a session with the SMB server. If you change your PC logon password, you must also change the SMB password in your RACF DCE segment.

   If, however, you specify a password on a **net use** command, then that is used for authentication and must match the SMB password in your RACF DCE segment. In this case, your PC logon password and your SMB password do not need to be the same.

2. If authentication fails, then you may be allowed to run as a guest z/OS user ID. This is determined by the **_IOE_MVS_DFSDFLT** environment variable for the **dfskern** process. If **_IOE_MVS_DFSDFLT** is set to a valid z/OS user ID, then the SMB server allows you to run with this user ID. Otherwise, your session is denied.

Using passthrough authentication

Passthrough authentication allows you to use your Windows Domain Controller to authenticate PC users in your Windows domain for access to data through the SMB server. This has the advantage that when PC users change their password on the Domain Controller, there is no other password that needs to be changed to access data through the SMB server. You must have a Domain Controller and your PC users must be enrolled in it. Also, PC users must be mapped to a local z/OS user ID.

---

6. The PC does not actually send an encrypted password over the network. An algorithm call “challenge/response authentication” is used that does not actually send any passwords over the network.

7. Using the RACF DCE segment does not imply that DCE needs to be active.

Passthrough authentication is enabled by using the following environment variables in the **/opt/dfslocal/home/dfskern/envar** file:

_IOE_SMB_AUTH_SERVER
Used to set the IP address of the Primary Domain Controller to authenticate PC users.

_IOE_SMB_BACKUP_AUTH_SERVER
Used to set the IP address of the Backup Domain Controller (if it exists).

_IOE_SMB_AUTH_SERVER_COMPUTER_NAME
Used to set the computer name of the Primary Domain Controller.

_IOE_SMB_BACKUP_AUTH_SERVER_COMPUTER_NAME
Used to set the computer name of the Backup Domain Controller (if it exists).

_IOE_SMB_AUTH_DOMAIN_NAME
Used to set the domain name of the Windows domain.

If authentication at the Domain Controller fails, and the domain of the client is not the same as the domain of the Domain Controller, the login is attempted locally (encrypted or clear based on the method chosen by the Domain Controller), otherwise, the authentication fails. If the authentication fails (including the case where a local authentication failed), you may be able to run as a guest z/OS user ID as described in item 2 on page 49.

Note: Passthrough authentication will not be used if the Domain Controller allows guests. In this case, local authentication is used.

RACF DCE segments for SMB encrypted password support

Note: This section assumes that you are using the IBM RACF support. For further information on using the RACF commands that are referenced in the following instructions, refer to the *z/OS: Security Server RACF Security Administrator's Guide*, SA22-7687. If you are using an equivalent security support product, refer to the appropriate documentation to perform the equivalent functions.

To allow the SMB server to use encrypted passwords, each z/OS user (that a PC user is mapped to) must be set up for SMB encrypted password support. Setting up an z/OS user for SMB encrypted password support requires defining a RACF DCE segment and storing the SMB password for the z/OS user into the user's RACF DCE segment. The RACF commands are issued from TSO.

An outline of the steps required to set up the z/OS system to have the SMB server use encrypted passwords processing is as follows:

- Activate class KEYSMSTR

  ```
  SETROPTS CLASSACT(KEYSMSTR)
  ```

- Activate class DCEUUIDS

  ```
  SETROPTS CLASSACT(DCEUUIDS)
  ```

- Define entry DCE.PASSWORD.KEY in class KEYSMSTR being sure to supply a 16 position KEYMASKED value

  ```
  RDEFINE KEYSMSTR DCE.PASSWORD.KEY SSIGNON(KEYMASKED(nnnnnnnnnnnnnnnn))
  ```

  A complete example of this command is:

  ```
  RDEFINE KEYSMSTR DCE.PASSWORD.KEY SSIGNON(KEYMASKED(0034639986ACCFDE))
  ```

- Define a RACF DCE segment for each z/OS **user_id** that requires the SMB encrypted password capability using the command:

  ```
  ALTUSER user_id DCE
  ```

  A complete example of this command is:

```
ALTUSER g1dfst2 DCE
```

- You can display the RACF DCE segment for a user by using the command:

  ```
  LISTUSER user_id NORACF DCE
  ```

  A complete example of this command is:

  ```
  LISTUSER g1dfst2 NORACF DCE
  ```

- To allow PC users to connect to the SMB server once encrypted passwords have been enabled, each user must issue the following command from OMVS:

  ```
  $ smbpw smb_login_password smb_login_password
  ```

  where *smb_login_password* is the PC user's SMB password (specified twice).

- To enable the SMB server to use encrypted passwords, the **_IOE_SMB_CLEAR_PW dfskern** environment variable must be set to **_IOE_SMB_CLEAR_PW=NOTALLOWED** and the SMB server must be restarted.

Chapter 8. Sharing printers

Before a PC client can print to a z/OS Infoprint Server printer through the SMB server, a shared printer must be created. A shared printer represents a z/OS printer controlled by the Infoprint Server. Once the PC client is connected to the shared printer, PC users can print to that z/OS printer as though it were a local printer.

Steps for creating a shared printer

This section describes the steps that are involved in creating a shared printer for an Infoprint Server printer.

Before you begin: The printer must be defined on the z/OS system. Refer to the *z/OS: Infoprint Server Operation and Administration* document, S544-5745, for information on how to define printers on z/OS. The SMB server can be running during this procedure.

Perform the following steps to create a shared printer:

1. Choose the Infoprint Server printer that you want to share (for example, `printname1`). You can determine the printer definitions that are available on the system by using the OMVS **lpstat -p** command. Refer to the *z/OS: Infoprint Server User's Guide*, S544-5746, for information on the **lpstat** command.
2. Add an entry in **smbtab** for the printer you want to share.
   a. Choose a unique minor device number (for example, **1**) in the device parameter (in this example, it would be `/dev/prt`**1**).
   b. Choose a unique share name (for example, `myprt`), a file system type of `prt`, a description, and put the printer definition name in the entry (for example, `printname1`).
   c. In addition, put the printer type information in the entry (for example, "`Generic / Text Only`"). You should choose a printer type that is compatible with the printer that you chose in step 1. Refer to the *z/OS: Infoprint Server Operation and Administration* document, S544-5745, for information on z/OS printers and their supported printer types.

      Note: The **dfstab** and **devtab** files do not apply to shared printers.

   Example: An **smbtab** entry follows:

   ```
   /dev/prt1  myprt  prt  "Department printer"  printname1  "Generic / Text Only"
   ```
3. If the SMB server is running, issue the **dfsshare** command to cause the new shared printer to be made available to PC users:

   ```
   # dfsshare -share myprt
   ```
4. If the SMB server is not running, start the SMB server with the following z/OS system command.

   ```
   start dfs
   ```

You know you have created a shared printer when you can connect to this share or you can run the Windows Add Print Wizard to connect to this shared printer.

Steps for removing a shared printer

A shared printer may be made unavailable by issuing the **dfsshare** command with the **-detach** option. For example, if you want to stop the shared printer, named **myprt**, from being available to PC clients, you would issue the following **dfsshare** command:

```
# dfsshare -share myprt -detach
```

This command makes the shared printer unavailable until the SMB server is restarted or the **dfsshare** command is issued to make it available. Since the shared printer is still in the **smbtab**, the shared printer would be made available again to PC clients. In order to make the shared printer permanently unavailable, it must be removed from the **smbtab** file.

Print data translation

When a PC user prints a file, no data translation is done by the SMB server. Any data translation is provided by the Infoprint Server. The Infoprint Server provides data transforms (for example, PDF to AFP™, PostScript to AFP, and PCL to AFP). It also provides conversion from one code page to another (such as, ASCII to EBCDIC). For print requests that come through the SMB server, the SMB server specifies ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process) in the **document-codepage** job attribute. For more information, refer to the *z/OS: Infoprint Server Operation and Administration* document, S544-5745.

Authorization

When a PC user prints a file, the PC user's SMB user ID is mapped to a local z/OS user ID. The z/OS user ID shows the owner of the print request on the SDSF view of the JES output queue. Refer to Chapter 6, "Mapping SMB user IDs to z/OS user IDs" on page 29 for information on how SMB users are mapped to z/OS users. If you are displaying a printer queue from a PC, refer to Chapter 11, "Accessing printers" on page 67.

Chapter 9. Locating the SMB server

The SMB server allows PC clients to access z/OS files and printers.

Selecting and configuring the PC client connection to the SMB server allows you to ensure that network clients can use the server properly. Proper configuration allows all PC clients on the network to locate the server and use shared directories and printers.

Set up the SMB server

This section discusses setting up the SMB server for use from the following PC clients:
- Windows 98
- Windows NT or Windows 2000.

PC clients may use the following methods of connecting to the SMB server on a TCP/IP network:
- User Datagram Protocol (UDP) Broadcast
- Domain Name Service (DNS)
- Windows Internet Naming Service (WINS)
- LMHOSTS static configuration files.

Setting up a PC client running Windows 98, Windows NT, or Windows 2000 allows you to use the SMB server. You can also use the Windows Network Neighborhood or Find Computer to find the SMB server and to access shared resources from your Windows client. This allows you to easily access all shared objects on the SMB server from your PC client.

Note: If the SMB server and your Windows client are in the same workgroup (domain) and the same subnet (network segment), then no additional set up on the client is necessary because the Windows client finds the SMB server by using UDP broadcast. If you are using Dynamic Host Configuration Protocol (DHCP) to assign client machine IP addresses, the steps to assign DNS addresses and WINS addresses on your clients may be unnecessary.

Steps for Windows 98 using DNS, WINS, or LMHOSTS file

Before you begin: You must ensure that clients can locate the SMB server on the network when you set up your Windows 98 client to use the SMB server.

DNS:

Perform the following steps to configure your Windows 98 client using DNS:

1. Click the **Start** button.
2. Point to **Settings** and click the **Control Panel**.
3. Double-click the **Network Control Panel**.
4. Click the **Configuration** tab.
5. Click on **TCP/IP** protocol.
6. Click on **Properties**.
7. Click the **DNS Configuration** tab.
8. Make sure the **Enable DNS** button is marked. If it is not marked, click on the button.
9. Enter the **Host** and **Domain**.
10. Click the **Add** button in the **DNS Server Search Order** area to enter your **TCP/IP DNS Server**.
11. Click the **Add** button in the **Domain Suffix Search Order** area to enter your **TCP/IP Domain Suffix**.

    Note: The previous information may already be supplied for you.
12. Click the **OK** button.

13. Click the **OK** button on the **Network Control Panel**.
14. Determine whether or not you want to restart your computer, click **Yes** or **No**.

You should have now located the SMB server on your Windows 98 client after using DNS.

WINS:
Perform the following steps to configure your Windows 98 client using WINS:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network Control Panel**.
4. Click the **Configuration** tab.
5. Click on **TCP/IP** protocol.
6. Click on **Properties**.
7. Click the **WINS Configuration** tab.
8. Make sure the **Enable WINS Resolution** button is marked. If it is not marked, click on the button.
9. Enter the **WINS Server Search Order** IP Address.
10. Click the **OK** button.
11. Click the **OK** button on the **Network Control Panel**.

You should have now located the SMB server on your Windows 98 client after using WINS.

LMHOSTS file:
If you are using the LMHOSTS file, configure the LMHOSTS file with the IP address and the computer name of the SMB server.

Steps for Windows NT using DNS, WINS, or LMHOSTS file
Before you begin: You must ensure that clients can locate the SMB server on the network when you set up your Windows NT client to use the SMB server.

DNS:
Perform the following steps to configure your Windows NT client using DNS:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network**.
4. Click the **Protocols** tab.
5. Click on **TCP/IP Protocol**.
6. Click on **Properties**.
7. Click the **DNS** tab.
8. Enter the **Host Name** and **Domain**.
9. Click the **Add** button in the **DNS Service Search Order** area to enter your **TCP/IP DNS Server**.
10. Click the **Add** button in the **Domain Suffix Search Order** area to enter your **TCP/IP Domain Suffix**.

    Note: The previous information may already be supplied for you.
11. Click the **OK** button.
12. Click the **OK** button on the Network panel.

You should have now located the SMB server on your Windows NT client after using DNS.

WINS:

Perform the following steps to configure your Windows NT client using WINS:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network**.
4. Click the **Protocols** tab.
5. Click on **TCP/IP Protocol**.
6. Click on **Properties**.
7. Click the **WINS Address** tab.
8. Select the **Adapter** that applies to the SMB File/Print Server.
9. Enter the **Primary WINS Server** and, if applicable, the **Secondary WINS Server** IP address in the **Windows Internet Name Services (WINS)** area.
10. Make sure the **Enable DNS for Windows Resolution** box is checked. If it is not checked, click on the box.
11. Click the **OK** button.
12. Click the **OK** (or **Close**) button on the Network panel.

You should have now located the SMB server on your Windows NT client after using WINS.

LMHOSTS file:

If you are using the LMHOSTS file, configure the LMHOSTS file with the IP address and the computer name of the SMB server.

Perform the following steps to configure the Windows NT client using the LMHOSTS file:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network**.
4. Click the **Protocols** tab.
5. Click on **Properties**.
6. Click the **WINS Address** tab.
7. Select the **Adapter** that applies to the SMB File/Print Server.
8. Enter the **Primary WINS Server** and, if applicable, the **Secondary WINS Server** IP address in the **Windows Internet Name Services (WINS)** area.
9. Make sure the **Enable LMHOSTS Lookup** box is checked. If it is not checked, click on the box.
10. Click the **OK** button.
11. Click the **OK** (or **Close**) button on the Network panel.

You should have now located the SMB server on your Windows NT client after using the LMHOSTS file.

Steps for Windows 2000 using DSN, WINS, or LMHOSTS file

Before you begin: You must ensure that clients can locate the SMB server on the network when you set up your Windows 2000 client to use the SMB server.

DNS:

Perform the following steps to configure your Windows 2000 client using DNS:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network and Dial-up Connections**.
4. Right-click the **Local Area Connections** icon and choose **Properties**.

5. Click on **Internet Protocol (TCP/IP)**.
6. Click on **Properties**.
7. Click the **Advanced** button.
8. Click the **DNS** tab.
9. Click the **Add** button in the **DNS Server Addresses: in order of use** area to enter the IP address(es) of one or more of your **TCP/IP DNS Servers**.
10. Click the radio button for either **Append primary and connection specific DNS suffixes** or **Append these DNS suffixes (in order)** and click the **Add** button after entering one or more **TCP/IP Domain Suffixes**.

    Note: The previous information may already be supplied for you.
11. Click the **OK** button.
12. Click the **OK** button on the **General** tab of the **Internet Protocol (TCP/IP) Properties** window.
13. Click the **OK** button of the **General** tab of the **Local Area Connection Properties** window.

You should have now located the SMB server on your Windows 2000 client after using DNS.

WINS:

Perform the following steps to configure your Windows 2000 client using WINS:

1. Click the **Start** button.
2. Point to Settings and click **Control Panel**.
3. Double-click the **Network and Dial-up Connections**.
4. Right click the **Local Area Connections** icon and choose **Properties**.
5. Click on **Internet Protocol (TCP/IP)**.
6. Click on **Properties**.
7. Click the **Advanced** button.
8. Click the **WINS** tab.
9. Enter one or more IP addresses of WINS server machines in the **WINS addresses: in order of use** area for each WINS IP address and click **Add** for each.
10. Make sure the **Enable NETBIOS over TCP/IP** button is selected. If it is not selected, click on the radio button.
11. Click the **OK** button.
12. Click **OK** on the **General** tab of the **Internet Protocol (TCP/IP) Properties** window.
13. Click **OK** on the **General** tab of the **Local Area Connection Properties** window.

You should have now located the SMB server on your Windows 2000 client after using WINS.

LMHOSTS file:

If you are using the LMHOSTS file, configure the LMHOSTS file with the IP address and the computer name of the SMB server.

Perform the following steps to configure your Windows 2000 client using the LMHOSTS file:

1. Click the **Start** button.
2. Point to **Settings** and click **Control Panel**.
3. Double-click the **Network and Dial-up Connections**.
4. Right click the **Local Area Connections** icon and choose **Properties**.
5. Click on **Internet Protocol (TCP/IP)**.
6. Click on **Properties**.
7. Click the **Advanced** button.

8. Click the **WINS** tab.
9. Make sure the **Enable LMHOSTS Lookup** box is checked. If it is not checked, click on the box.
10. Click the **OK** button.
11. Click the **OK** button on the **General** tab of the **Internet Protocol (TCP/IP) Properties** window.
12. Click **OK** on the **General** tab of the **Local Area Connection Properties** window.

You should have now located the SMB server on your Windows 2000 client after using the LMHOSTS file.

Find the SMB server

This section discusses how to find the SMB server and access shared resources from the following Windows PC clients:
- Windows 98 or Windows NT
- Windows 2000.

The Windows PC clients can use one of the following methods for finding the SMB server:
- Network Neighborhood
- Find Computer
- Search Computer.

Steps for using Network Neighborhood

Before you begin: If the SMB server and your PC client are in the same workgroup (domain) and in the same subnet (network segment), perform the following steps using Network Neighborhood to find the server:

1. Click on the **Network Neighborhood**.
2. Select *zOSS1* (where *zOSS1* is the computer name of the SMB server).

You should have now located the server after using Network Neighborhood.

Steps for using Find Computer

Before you begin: If you are using Windows 98 or Windows NT, perform the following steps by using Find Computer to locate the SMB server on your network:

1. Click the **Start** button.
2. Point to **Find**, click **Computer**.
3. Enter the **Computer Name** for the SMB server.
4. Click the **Find Now** button.
5. The screen shows the results of your search.

You should have now located the server after using Find Computer.

Steps for using Search Computer

Before you begin: If you are using Windows 2000, perform the following steps by using Search Computer to locate the SMB server on your network:

1. Click the **Start** button.
2. Select **Programs**.
3. Select **Accessories**.
4. Select **Windows Explorer** and release the button.
5. Click on the **Search** button on the status bar.
6. Click **Computers** in the **Search for other items** area.
7. Enter the **Computer Name** for the SMB server.

8. Click the **Search Now** button.
9. The screen shows the results of your search.

You should have now located the server after using Search Computer.

Chapter 10. Accessing data

When your PC client has located the SMB server, it can then access shared directories that have been created. This allows the PC client to read and write file data on the z/OS system.

Using SMB server shared directories

This section discusses how you can access shared directories on the SMB server from the following clients:

- Windows 98
- Windows NT or Windows 2000.

Note: If you are using clear text passwords in the SMB server (**_IOE_SMB_CLEAR_PW=REQUIRED**), you may need to update your Windows registry. Refer to “Client does not communicate” on page 133.

Windows 98 or Windows NT clients

You can use a Microsoft Windows 98 or Windows NT PC client to access shared directories on the SMB server by using one of the following methods:

- Map shared directories to logical drives
- Universal Naming Convention (UNC) mapping.

You may find it easier, however, to work with logical drive letters as opposed to UNC mapping.

Mapping shared directories to logical drives:

You can map an SMB server shared directory to a logical drive by performing the following steps:

1. Click on the **Start** button.
2. Choose **Programs**.
3. Click on **Windows Explorer** (or **Windows NT Explorer**).
4. Click the **Tools** pull-down menu on the Windows Explorer and click **Map Network Drive**.
5. Choose the letter of a free drive for the shared directory (it may already be filled in).
6. Enter the **Path** name of an SMB shared directory. For example, you enter the following:

   `\\zOSS1\myshare`

   where *zOSS1* is the computer name and *myshare* is the shared directory name.
7. Click the **OK** button.

However, if your z/OS password (when using clear passwords) or your SMB password in your RACF DCE segment (when using encrypted passwords) is not the same as your Windows password, you should use the **net use** command with your z/OS or SMB password from the command prompt. This avoids you being logged in as DFSDFLT (that is, a guest user). For example,

`net use x: \\zOSS1\myshare mypassword`

where *x* is an available drive letter, *zOSS1* is the computer name, *myshare* is the shared directory name, and *mypassword* is your password.

Universal Naming Convention (UNC) mapping:

You can also use Universal Naming Convention (UNC) mapping to access SMB server shared directories by performing the following steps:

1. Enter the following command:

   `net use \\zOSS1\myshare`

where *zOSS1* is the computer name and *myshare* is the shared directory name.

2. Enter the following command to display the computer name and the shared directory.

   ```
   net use
   ```

   The following output appears:

```
Status                          Local name                      Remote name
-------------------------------------------------------------------------------
OK                                                              \\zOSS1\myshare
```

3. You can enter the following to delete the shared directory:

   ```
   net use \\zOSS1\myshare /d
   ```

Windows 2000 clients

You can use a Microsoft Windows 2000 PC client to access shared directories on the SMB server by using one of the following methods:

- Map shared directories to logical drives
- Universal Naming Convention (UNC) mapping.

You may find it easier, however, to work with logical drive letters as opposed to UNC mapping.

Mapping shared directories to logical drives:

You can map an z/OS SMB server shared directory to a logical drive by performing the following steps:

1. Click on the **Start** button.
2. Choose **Programs**.
3. Choose **Accessories**.
4. Click on **Windows Explorer**.
5. Click the **Tools** pull-down menu on the Windows Explorer and click **Map Network Drive**.
6. Choose the letter of a free drive for the shared directory (it may already be filled in).
7. Enter the **Path** name of an SMB shared directory. For example, you enter the following:

   ```
   \\zOSS1\myshare
   ```

   where *zOSS1* is the computer name and *myshare* is the shared directory name.
8. Click the **Finish** button.

However, if your z/OS password (when using clear passwords) or your SMB password in your RACF DCE segment (when using encrypted passwords) is not the same as your Windows password, you should use the **net use** command with your z/OS or SMB password from the command prompt. This avoids you being logged in as DFSDFLT (that is, a guest user). For example,

```
net use x: \\zOSS1\myshare mypassword
```

where *x* is an available drive letter, *zOSS1* is the computer name, *myshare* is the shared directory name, and *mypassword* is your password.

Universal Naming Convention (UNC) mapping:

You can also use Universal Naming Convention (UNC) mapping to access SMB server shared directories by performing the following steps:

1. Enter the following command:

   ```
   net use \\zOSS1\myshare
   ```

   where *zOSS1* is the computer name and *myshare* is the shared directory name.
2. Enter the following command to display the computer name and the shared directory.

   ```
   net use
   ```

The following output appears:

```
Status                          Local name                      Remote name
-------------------------------------------------------------------------------
OK                                                              \\zOSS1\myshare
```

3. You can enter the following to delete the shared directory:

   ```
   net use \\zOSS1\myshare/d
   ```

Accessing HFS data

This section discusses accessing HFS data.

HFS directory and file name case sensitivity considerations

The SMB server makes HFS file system data available to PC clients. The HFS file system is case-sensitive. PC clients are case-insensitive (that is, they treat uppercase letters and lower case letters as the same when you are searching for a file).

The case of file names is significant in case-sensitive file systems and can consist of both upper-case and lower-case characters. For example, the HFS file system could have three files in it with the following names:

```
DFSSMB.DAT
DFSsmb.dat
dfssmb.dat
```

These three files have technically different names (because the HFS file system is case-sensitive) and they represent three distinct, separate objects on z/OS.

All the PC clients that the SMB server supports (Windows 98, Windows NT, and Windows 2000) are case-insensitive. This means that the case of file names is insignificant. For example, from the three example files that are listed above, only one would be recognized. Which one would depend on how the user specified the name and whether the PC client code folded the name to upper case.

The following algorithm is used by the SMB server when searching for a file or directory name in HFS:

The name received over the network is searched as is. If found, that name is returned. If not, the name is folded to lowercase letters. Then the HFS directory is searched for that name. If found, that name is returned. If not, then the name received in the SMB (still folded to lowercase) is compared against each name in the HFS directory folded to lowercase letters. The first match found is returned. Otherwise, the name is not found.

Note that many PC clients may fold a name to uppercase letters before sending the name over the network.

When a new file or directory is created, the name received over the network is used as the name of the object. When a file or directory is renamed, the object is located as described above and then, if located, its name is replaced with the new name as received over the network.

HFS symbolic links

This section discusses how symbolic links are treated on the SMB server. Refer to the *z/OS: UNIX System Services User's Guide*, SA22-7801, for information on symbolic links.

A symbolic link is a special "file" that contains another path name. It can be created by z/OS UNIX commands (for example, **ln -s old new**) and applications. It is used to redirect access to another file or directory in the file system hierarchy. The closest analogy to a symbolic link in Windows terminology is a shortcut. However, they are not the same thing. A Windows application cannot create a symbolic link. However, if a symbolic link already exists, a Windows application can access it through the SMB server

with some restrictions. The SMB server supports relative symbolic links (that is, symbolic links that do not begin with a */*). In addition, the SMB server supports absolute symbolic links (that is, symbolic links that begin with a */*) when the **_IOE_SMB_ABS_SYMLINK** environment variable is set to **ON** in the **/opt/dfslocal/home/dfskern/envar** file. This allows Windows applications to reference a directory/file that is outside the shared directory tree. Directories and files can only be accessed if the PC user is authorized to the data.

In general, symbolic links can be referenced by Windows applications if they are accessing them in a read-only manner. However, this access fails if the symbolic link contains one of the following:

- A path name that begins with / (that is, it is an absolute path name) unless the **_IOE_SMB_ABS_SYMLINK** environment variable is set to **ON** in the **/opt/dfslocal/home/dfskern/envar** file
- Is circular (it traverses more than 50 symbolic links)
- A path to an object that does not exist
- A path to an object that goes out of the shared directory tree unless the **_IOE_SMB_ABS_SYMLINK** environment variable is set to **ON** in the **/opt/dfslocal/home/dfskern/envar** file.

For the cases listed above, the path name does not display as a file of zero length (for example, if listed with a **dir** command).

Other than these restrictions, symbolic links can be used during path name resolution. For example, if you **cd** to a path name that contains a symbolic link as part of the name resolution, this is successful (and follows the symbolic link) as long as none of the restrictions above apply and the user is authorized to all the components in the path name. Reading a file whose name is really a symbolic link that points to another file that exists is also successful. Note that, in general, a Windows user is not able to tell when the path name being used is a symbolic link or traverses one or more symbolic links.

Writing to a file whose path name is a symbolic link is allowed in some cases. If the file exists (that is, if the symbolic link and the linked file both exist), then opening the file for update, append, or truncate succeeds. An open for create fails (regardless of whether the linked file exists or not) because the symbolic link exists.

The symbolic link itself can be renamed if it remains in the same directory. Otherwise, it is denied. A symbolic link cannot be removed by a PC user.

Accessing RFS data

This section discusses accessing RFS (Record File System) data.

RFS directory and file name considerations

The SMB server makes RFS data available to PC clients. The data sets supported include sequential data sets (on DASD), partitioned data sets (PDS), partitioned data sets extended (PDSE) and Virtual Storage Access Method (VSAM) data sets.

Data set names are always upper case.

RFS has many restrictions due to the fact that RFS files are not really hierarchical, byte stream files. Since they are really z/OS data sets that store record oriented data, they must follow data set rules. Among the restrictions are the following:

- The data set name prefix plus the file name is limited to 44 characters. (Note that the data set name prefix is not included in file names displayed at the PC.)
- A data set name segment (characters between the dots) cannot be longer than 8 characters.
- The characters allowed in a data set name are more limited than file names in HFS (e.g, ~ is not allowed). Refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410, for more information of data set naming rules.

- Use lower case file names at the PC. Data set names are mapped to upper case and are displayed as lower case if **maplower** is specified in **rfstab** (this is the default). Mixed case file names should not be used.
- A PDS or PDSE cannot be created under another PDS or PDSE.
- PDS and PDSE member names are limited to 8 characters.
- Only one PDS member can be open for write at a time.
- PDS members cannot be moved between PDSs. Copy and erase must be used.
- PDS member aliases are not supported and are not displayed.
- If an attempt is made to write a record that exceeds the maximum record size, the write fails.
- Editors or commands (e.g., Wordpad or copy) that truncate the file to zero length before rewriting a file usually work. Editors or commands that try to replace without truncate to zero attempts to replace each record and this requires that each record be exactly the same size it was before. This is not likely and therefore the writes probably fail.
- SMB access is limited to RFS data sets that are no larger than 4 GB.

In general, data sets that contain variable length text data work correctly as far as record processing is concerned (although VSAM files do not support zero length records).

Refer to Appendix E, “Using data sets” on page 143 for more information on accessing RFS files.

Chapter 11. Accessing printers

Once your PC client can locate the SMB server, it can then access shared printers that have been created. This allows the PC client to print data on z/OS printers. This chapter explains how to access the SMB server shared printers and how to add a printer from the following clients:

- Windows 98 (hereafter referred to as Windows 98)
- Windows NT
- Windows 2000.

You can map an SMB server shared printer to a logical printer on the Windows PC clients by performing the following steps:

1. Open an **MS-DOS** window.
2. Enter the following command at the DOS prompt to get a list of SMB server shared printers:

   ```
   net view \\zOSS1
   ```

   where *zOSS1* is the computer name of the SMB server.
3. Choose a shared printer name from the list provided.
4. Enter the following command:

   ```
   net use lpt2: \\zOSS1\myprt /persistent:yes
   ```

   where *lpt2* is the name of the logical printer that you want to map the shared printer to, *zOSS1* is the computer name of the SMB server, *myprt* is the name of the desired shared printer, and `/persistent:yes` is an optional parameter specifying that the shared printer connection should be restarted by the client after reboot.

Accessing shared printers

This section shows you how to access shared printers using the Windows 98, Windows NT, and Windows 2000 clients.

Windows 98 client

You can access shared printers on the SMB server using a Windows 98 client by performing the following steps:

1. Click the **Start** button.
2. Point to **Find** and click on **Computer**.
3. In the Find: Computer dialog box, enter the **Computer Name** for the SMB server.
4. Click the **Find Now** button.
5. Double-click on the found computer name.
6. Double-click on a shared printer.
7. If the printer you selected has been defined to the PC client, then you get a list of jobs that are queued to print for that printer.
8. If the printer you selected has not been defined to the PC client, click **Yes** to set up the printer on your computer. If you are prompted further, select the appropriate printer characteristics to get the drivers set up.
9. Click the **OK** button.

Windows NT client

You can access shared printers on the SMB server using a Windows NT client by performing the following steps:

1. Click the **Start** button.

2. Point to **Find** and click on **Computer**.
3. Enter the **Computer Name** for the SMB server.
4. Click the **Find now** button.
5. Double-click on the found computer name.
6. Double-click on a shared printer.
7. If the printer you selected has been defined to the PC client, then you get a list of jobs that are queued to print for that printer.
8. If the printer you selected has not been defined to the PC client, click **Yes** to set up the printer on your computer. If you are prompted further, select the appropriate printer characteristics to get the drivers set up.
9. Click the **OK** button.

Windows 2000 client

You can access shared printers on the SMB server using a Windows 2000 client by performing the following steps:

1. Click the **Start** button.
2. Select **Programs**.
3. Select **Accessories**.
4. Select **Windows Explorer** and release the button.
5. Click on the **Search** button on the status bar.
6. Click **Computers** in the **Search for other items** area.
7. Enter the **Computer Name** of the SMB server.
8. Click the **Search Now** button.
9. Double-click on the found computer name.
10. Double-click on a shared printer.
11. If the printer you selected has been defined to the PC client, then you get a list of jobs that are queued to print for that printer.
12. If the printer you selected has not been defined to the PC client, click **Yes** to set up the printer on your computer. If you are prompted further, select the appropriate printer characteristics to get the drivers set up.
13. Click the **Finish** button.

Adding a printer

This section shows you how to add a printer using Windows 98, Windows NT, and Windows 2000 clients.

Note: You may get prompted to specify a print driver if the **Printer type** specified on the **smbtab** entry for this shared printer does not exist on your PC.

Windows 98 client

You can add a printer from a Microsoft Windows 98 client by performing the following steps:

1. Click the **Start** button.
2. Point to **Settings** and click on **Printers**.
3. Double-click on **Add Printer**.
4. Click **Next** on the Add Printer Wizard panel.
5. Choose **Network printer** (if it is not already selected) and click the **Next** button.
6. Enter the **Network path** or **queue name** of the SMB server shared printer. For example, you could enter the following network path:

```
\\zOSS1\myprt
```

where *zOSS1* is the computer name of the SMB server and *myprt* is the name of the shared printer.

7. Decide whether or not you want to print from MS-DOS based programs by clicking **Yes** or **No**, then click the **Next** button.
8. Decide whether or not you want this printer as your default by clicking **Yes** or **No**, then click the **Next** button.
9. Click the **Finish** button.

Windows NT client

You can add a printer from a Windows NT PC client by performing the following steps:

1. Click the **Start** button.
2. Point to **Settings** and click on **Printers**.
3. Double-click on **Add Printer**.
4. Choose **Network printer server** (if it is not already selected) and click the **Next** button.
5. Enter the network path of the SMB server shared printer. For example, you could enter the following network path:

   ```
   \\zOSS1\myprt
   ```

   where *zOSS1* is the computer name of the SMB server and *myprt* is the name of the shared printer.
6. Decide whether or not you want this printer as your default by clicking **Yes** or **No**, then click the **Next** button.
7. Click the **Finish** button.

Windows 2000 client

You can add a printer from a Windows 2000 PC client by performing the following steps:

1. Click the **Start** button.
2. Select **Settings**.
3. Select **Printers**.
4. Click on **Add Printer**.
5. Click **Next**.
6. Choose the **Network Printer** radio button.
7. Select **Type the printer name, or click Next to browse for a printer**.
8. Choose **Network printer server** (if it is not already selected) radio button.
9. Enter the network path of the SMB server shared printer and click the **Next** button. For example, you could enter the following network path:

   ```
   \\zOSS1\myprt
   ```

   where *zOSS1* is the computer name of the SMB server and *myprt* is the name of the shared printer.
10. Install the driver, if necessary.
11. Click the **Finish** button.

Displaying a printer queue

Windows PC clients have the ability to display a printer queue. Perform the following steps to display a printer queue:

1. Click the **Start** button.
2. Point to **Settings** and click **Printers**.
3. Determine which printer you want to display and double-click on it.

For print requests that come through the SMB server, the SMB server specifies the SMB user ID of the submitter in the **name-text** job attribute (refer to the *z/OS: Infoprint Server Operation and Administration* for more information on the **name-text** job attribute). When the printer is displayed, it shows one line for each print request for that printer that has not printed yet. When a print request contains a **name-text** job attribute, the following fields are displayed:

Document Name
The JES Jobname followed by the JES Jobid (as shown on the SDSF output queue panel). For a print request submitted through the SMB server, the JES Jobname is the z/OS user ID for the SMB user that submitted the print request. Refer to Chapter 6, "Mapping SMB user IDs to z/OS user IDs" on page 29 for information on how SMB user IDs are mapped. The JES Jobid is normally the characters PS followed by a number assigned by the Infoprint Server. Print jobs submitted by using the SMB server always has JES Jobids less than 65536.

Owner
The **name-text** job attribute. For a print request submitted using the SMB server, this is the SMB user ID of the submitter.

For print requests that do not have a **name-text** job attribute, the **Document Name** normally contains the file name of the print request and the **Owner** contains the z/OS user ID of the submitter.

Note: It is best to avoid leaving a window for a remote printer open for long periods of time. While the remote printer window is open, the server is continually queried in order to update the window with the latest printer status. This may lead to unnecessary network contention.

Using PC client print drivers with SMB server shared printers

The SMB server acts as a print server that makes the services of the Infoprint Server available to PC clients. This allows clients with the proper print drivers to submit print jobs to the Infoprint Server through the SMB server. The available print types include the following:

- Postscript
- PCL
- text
- Advanced Function Printing™ (AFP).

You can access print drivers for supported Windows PC clients in either of these two ways:

- If the printer type specified on the **smbtab** (for example, **Generic / Text Only**) is available on the Windows system, Windows automatically uses that printer type (and print driver) when the printer is added by the Add Print Wizard.
- Print drivers are available for free downloading from the IBM Printing Systems Company World Wide Web (WWW) site. It is located at **http://www.printers.ibm.com**.

Part 2. SMB support reference

This part of the document discusses the reference information and is organized into the following chapters:

- Chapter 12, “z/OS system commands” on page 73
- Chapter 13, “Distributed File Service SMB files” on page 79
- Chapter 14, “Distributed File Service SMB commands” on page 105.

Each chapter begins with a short introduction and is followed by information about the processes, the files and the commands, which are ordered alphabetically.

Chapter 12. z/OS system commands

This chapter introduces you to the following commands:

- **MODIFY**, a system command which enables you to start, stop, and query the status of the SMB daemons.
- **START** and **STOP**, system commands that enable you to start and stop the DFS Control Task (**DFS**).

These commands may be invoked from the operator console or from a Spool Display and Search Facility (SDSF) screen.

modify dfs processes

Purpose

Starts, stops, or queries the status of DFS Control Task daemons. This command can also be used to send a command string to the **dfskern** daemon.

Format

You can use any of the following formats for this command.

modify *procname*,**{start** | **stop** | **query}** *daemon*

modify *procname*,**send dfskern,reload,{print** | **smbmap}**

Parameters

procname The name of the DFS Control Task. On DFS, the default procname is **DFS**.

command The action that is performed on the DFS daemon or daemons. This parameter can have one of the following values:

start Starts either a single DFS daemon or all the DFS daemons.

stop Stops either a single DFS daemon or all the DFS daemons.

query Displays the status and process identifier (PID) of the DFS daemon or DFS daemons.

send Sends a command string to the **dfskern** daemon.

daemon The name of the DFS Control Task daemon for which the action is being requested. This parameter can have one of the following values:

dfskern The **dfskern** daemon. The **dfskern** server daemon is the DFS File Exporter process.

The **dfskern** daemon parameter of the send command can be further modified through the following parameters:

reload,print Causes the Infoprint Server DLL to be reloaded and reinitialized. Refer to the *z/OS: Program Directory*, GI10-0669, for more information on the Infoprint Server DLL.

reload,smbmap
Ensures that new SMB identity mappings are used by first eliminating any cached SMB identity mappings and then reloading the updated **smbidmap** file.

export The **export** daemon. The **export** server daemon allows exporting and sharing of HFS File Systems.

unexport The **unexport** daemon. The **unexport** server daemon allows unexporting and unsharing of HFS File Systems.

all All of the DFS daemons.

Usage

The **modify dfs** *daemon* command is used to manually start or stop one or more DFS daemons, or to view the status of the daemons. It is especially useful in situations where a daemon has stopped abnormally. On DFS, the DFS daemons are contained in the **dfs** address space (with the possible exception of **dfskern**).

Starting and Stopping z/OS DFS Daemons: Using the **MODIFY** system command, you can start or stop either a single daemon or all the daemons configured on the host.

Viewing the Status of z/OS DFS Daemons: Using the **MODIFY** system command, you can view the status of the DFS daemons from the operator console using the query option.

Sending a Parameter to the dfskern Daemon: Using the **MODIFY** system command, you can send a parameter to the **dfskern** daemon.

The **reload,print** parameter causes the print DLL used by the SMB File/Print Server to communicate with the Infoprint Server to be reloaded and reinitialized. Refer to the *z/OS: Program Directory*, GI10-0669, for more information on the Infoprint Server DLL. This allows the Infoprint Server to be enabled or updated without the need to restart **dfskern**. (You may need to update **smbtab** and you may need to issue the **dfsshare** command.) Print jobs in the process of being submitted may need to be resubmitted. An open window for a remote printer may need to be refreshed.

The **reload,smbmap** parameter eliminates any cached identity mappings and reloads a new **smbidmap** file to update identity mappings between SMB user IDs and z/OS user IDs. The new identity mappings take effect for new connections.

Privilege Required

This command is a z/OS system command.

Examples

The following example starts the DFS **dfskern** process:

```
modify dfs,start dfskern
```

The following example starts all the DFS daemons:

```
modify dfs,start all
```

The following example views the status of all the DFS Control Task daemons that are currently active on the host.

```
modify dfs,query all
```

In the following example, the **dfskern** daemon is instructed to eliminate existing identity mappings and to reload an updated **smbidmap** file that includes recently added or deleted user identity mapping information:

```
modify dfs,send dfskern,reload,smbmap
```

Related Information

Files:

ioepdcf
smbidmap

start dfs

Purpose

Starts the DFS Control Task.

Format

start *procname[,start_options]*

Parameters

procname The name of the DFS Control Task. On DFS, the default procname is **DFS**.

start_options The value passed to the **START** command. On DFS, *start_options* has only one value, *parm=' -nodfs'*. You can use the *parm=' -nodfs'* option to start the DFS Control Task without starting the DFS Control Task daemons. (This is usually used during installation verification.)

Usage

The **START** command is used to initiate the DFS Control Task.

You can use the *parm=' -nodfs'* option to start the DFS Control Task without starting the DFS Control Task daemons.

Note: In DFS, *start_options* values **must** be enclosed in single quotes. This is because the **START** command converts all user-supplied *start_options* values to uppercase characters unless they are enclosed in single quotes.

Privilege Required

This command is a z/OS system command.

Examples

The following command starts the DFS Control Task and all daemons on DFS:

```
start dfs
```

The following command starts the DFS Control Task without starting the DFS daemons:

```
start dfs,parm='-nodfs'
```

Related Information

File:

ioepdcf

stop dfs

Purpose

Stops the DFS Control Task processes and the DFS Control Task.

Format

stop *procname*

Parameters

procname The name of the DFS Control Task. On DFS, the default procname is **DFS**.

Usage

The STOP command stops the DFS Control Task (**dfscntl**) and the processes controlled by the **dfscntl** process. These processes are:

dfskern The **dfskern** daemon. The **dfskern** server daemon is the SMB File Server process.

export The **export** daemon. The **export** daemon allows exporting of HFS file systems.

Privilege Required

This command is a z/OS system command.

Examples

The following command stops the DFS Control Task and all daemons on DFS:

```
stop dfs
```

Related Information

File:

ioepdcf

stop dfs

Chapter 13. Distributed File Service SMB files

This chapter introduces you to the Distributed File Service SMB files. These files contain configuration parameters for server processes and define HFS files and Infoprint Server printers for sharing with Windows clients. These are EBCDIC files and each line is usually delimited by newline (X'15'). They can also be delimited by carriage return/line feed (X'0D15') as would be the case if they were edited from a Windows application.

This chapter provides an alphabetical listing of all relevant SMB files.

Attributes file (rfstab)

Purpose

Contains tables describing the attributes used to manipulate RFS files in the SMB server. It also contains descriptions for:

- Data set creation attributes
- Processing attributes
- Site attributes.

The PC user can also modify data set creation attributes that provide information about a data set to the SMB server, such as the type of data set, or how the data set is allocated (for example, blocks, cylinders, or tracks). Processing attributes and site attributes can only be modified by the system administrator.

Note: The attributes file is sometimes referred to as the **rfstab** file.

Specifying the Location of the Attributes File (rfstab)

The attributes file is in either an HFS file, a fixed-block partitioned data set, or a fixed-block sequential data set with a record length of 80. The file's location is specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable for the **dfskern** process. For example, **_IOE_RFS_ATTRIBUTES_FILE=/opt/dfslocal/var/dfs/rfstab** is the default location.

The SMB server can use the same attributes file as the DFSMS/MVS® NFS server. For example, **_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'**, would cause the SMB server to use the member NFSSATT in PDS NFSADMIN.NFSS for the RFS attributes. (The NFS server attributes file is specified in the NFSATTR DD statement of the NFS server startup procedure.) NFS server attributes that are not supported by the SMB server are ignored. Refer to "Unsupported Attributes" on page 85 for a list of attributes that are not supported.

An attributes file can also be specified on an RFS file system basis. Its location can be specified in the **devtab** entry for the RFS file system on the same line as the data set prefix name: **attrfile** *attributes_file* (where *attributes_file* is the path name of the attributes file that controls the data set creation, processing and site attributes for this RFS file system). An example **devtab** entry might look like this:

```
* RFS devices
define_ufs 10 rfs
USERA.PCDSNS attrfile /opt/dfslocal/var/dfs/rfstab2
```

If no **attrfile** parameter is specified in the devtab entry for the RFS file system, the attributes are taken from the global attributes file specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable in the **dfskern** process.

If no **_IOE_RFS_ATTRIBUTES_FILE** environment variable is specified, then the SMB server system defaults are used for RFS attributes.

Using Multipliers

Instead of entering numeric values for the attributes, you can use the multipliers K (1024), M (1024 x 1024), or G (1024 x 1024 x 1024) for specifying sizes. For example, lrecl(8192) is the same as lrecl(8K).

Data Set Creation Attributes

The data set creation attributes are used to define the structure of data sets when creating a file. These attributes correspond to the data control block (DCB) or the job control language (JCL) parameters used to define a data set when it is created. Refer to the *z/OS: MVS JCL Reference*, SA22-7597, for more detailed information about data set creation attributes.

The data set creation attributes are described in the following table. Defaults are underlined.

You can override these attributes by specifying an attributes file in the devtab entry for the RFS file system or by using a file creation command. For PDS and PDSE, members have the same attributes as the data set attributes, so the file creation attributes for members are ignored.

Table 1. Data set creation attributes

Data Set Creation Attribute	Description
blks	Specifies that disk space (refer to the **space** attribute in this table) is allocated by blocks, except for VSAM data sets.
cyls	Specifies that disk space (refer to the **space** attribute in this table) is allocated by blocks, except for VSAM data sets.
recs	Specifies that disk space is allocated by records for VSAM data sets. **blks** and **recs** are identical for VSAM data sets
trks	Specifies that disk space is allocated by tracks.
blksize(0 | *quan***)**	Specifies the maximum length, in bytes, of a physical block on disk. *quan* is a number **0** (the default) to 32,760. If **blksize**(0) is specified, the system determines an optimal block size to use.
dataclas(*class_name***)**	Specifies the data class associated with the file creation. The class_*name* must be defined to DFSMS/MVS before it can be used by the client. The system-managed storage automatic class selection routine must also assign a storage class to the file being created. For more information on data classes, refer to the *z/OS: DFSMdfp Storage Administration Reference*, SC26-7402.
dir(27 | *quan***)**	Specifies the number of 256-byte records needed in the directory of a PDS. Use it with the **mkdir** command when you are creating a PDS. *quan* is a number from 1 to 16,777,215 (the default is **27**). The maximum number of PDS members is 14,562.
dsntype(library | pds)	Specifies whether a PDSE or a PDS is to be created when the **mkdir** client command is used. **library** is for PDSE. **pds** is for PDS. You cannot create a PDS (or PDSE) within another PDS (or PDSE). If you need help deciding whether to create a PDS or a PDSE, refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410.
dsorg(*org***)**	Specifies the organization of a data set. *org* can be a physical sequential (**ps**) data set, direct access (**DA**) data set, VSAM KSDS (**indexed**), VSAM RRDS (**numbered**), or VSAM ESDS (**nonindexed**). This attribute is ignored for directory-oriented client commands. If you are using VSAM data sets in binary mode, then **nonindexed** is recommended.
keys(*len,off***)**	Specifies the length and offset of the keys for VSAM KSDS data sets. Keys can only be specified when using **dsorg(indexed)**. *len* and *off* are specified in bytes. *len* is between 1 and 255 (the default is **64**). *off* is between 0 and 32,760 (the default is **0**). When you create a VSAM KSDS data set, the records you are loading into it must be keyed-sequenced or the write fails. Each write of the data set is treated like a first load, and requires that the records being loaded are in ascending key sequence.
lrecl(8196 | *quan***)**	Specifies: • The length, in bytes, for fixed-length records. • The maximum length, in bytes, for variable-length records. If the **blksize** attribute is specified, the value must be at least 4 bytes less than the **blksize** quantity. *quan* is a number from 1 to 32,760 (the default is **8196**).
mgmtclas(*mgmt_class_name***)**	Specifies the management class associated with the file creation. The *mgmt_class_name* must be defined to DFSMS/MVS before it can be used by the client. The system managed storage automatic class selection (ACS) routine must also assign a storage class to the file being created. For more information on management classes, refer to the *z/OS: DFSMdfp Storage Administration Reference*, SC26-7402.

Table 1. Data set creation attributes (continued)

Data Set Creation Attribute	Description
model(*dsname***)**	The name of the cataloged VSAM data set from which to copy data set creation attributes when creating a new VSAM data set. *dsname* is a fully qualified MVS™ data set name without quotation marks. The **model** attribute must be used with one of the **dsorg** attributes which imply a VSAM organization. You can do this by specifying the **dsorg** attributed for a VSAM in the command. Refer to the **dsorg** entry in this table.

recfm(*cccc***)**	Specifies the format and characteristics of the records in the data set. *cccc* can be 1 to 4 characters, in one of the following combinations:
	`f | fb | fs | fbs`
	`u`
	`v | vb | vs | vbs`
	
	Valid record format characters:
	b Blocked
	f Fixed-length records
	s Spanned for variable records, standard format for fixed records
	u Undefined-length records
	v Variable-length records
	
	In **recfm**, codes **v**, **f**, and **u** are mutually exclusive. The **s** code is not allowed for a PDS or PDSE

recordsize(*avg,max***)**	The average and maximum record size for VSAM data sets. *avg* and *max* are specified in bytes. They can each range from 1 to 32,760 (the defaults are **512** and **4096**, respectively). These values must be equal for VSAM RRDS.
rlse	Specifies that unused space should be released from the data set the first time a new data set is closed. For slow clients with long pauses between writes, the **rlse** attribute causes space to be released from the primary extent prematurely. Further writes cause secondary space to be allocated.
norlse	Specifies that unused space should not be released from the data set.
shareoptions(*xreg,xsys***)**	Specifies the cross-region and cross-system share options for a VSAM data set. *xreg* is a number from 1 to 4; *xsys* is either 3 or 4. The defaults are **1** and **3**, respectively.
	This applies to VSAM data sets only. For spanned records of non-VSAM detests, refer to the entry for **recfm** in this table.
spanned	Specifies that VSAM KSDS or ESDS data sets can contain records that span control intervals (spanned records).
nonspanned	Specifies that data sets do not have spanned records.
space(*prim[,aux]***)**	Specifies the amount of primary and auxiliary space allocated for a new data set on a direct access volume. *prim* is the number (from 0 to 16,777,215) of primary tracks, cylinders, or data blocks in the data set. *aux* (optional) is the number (from 0 to 16,777,215) of additional tracks, cylinders, or blocks allocated if more space is needed. If this attribute is not specified, the default is used. The defaults are **100** and **10**, respectively.
storclas(*class_name***)**	Specifies the storage class associated with the file creation. The *class_name* must be defined to the DFSMS/MVS before it can be used by the client. For more information on storage classes, refer to the *z/OS: DFSMdfp Storage Administration Reference*, SC26-7402.

Table 1. Data set creation attributes (continued)

Data Set Creation Attribute	Description
unit(*unit_name***)**	Specifies the unit on which to create a data set. *unit_name* is a generic or symbolic name of a group of DASD devices. The *unit_name* must be specified as 3390 for extended format data sets. **Note:** You cannot create or access tape data sets on a z/OS host using the SMB Server. You cannot create extended format data sets with the SMB Server, except by ACS routines.
vol(volser**)**	Specifies the name of the DASD volume to be used to store the created data set. **vol** is the keyword and *volser* represents the volume name. If a data set is to be system-managed, as is determined by the DFSMS/MVS automatic class selection (ACS) routines, you can omit this attribute.

Processing Attributes

Processing attributes are used to control how files are accessed by clients. The processing attributes are described in the following table. Defaults are underlined. The administrator can override the default processing attributes in the **devtab** entry for the fileset. The client user cannot override processing attributes.

Table 2. Processing attributes

Processing Attribute	Description
binary	Indicates that the data is processed between the client and server using binary format and no data conversion occurs between ASCII and EBCDIC formats.
text	Converts the contents in the data set between EBCDIC and ASCII formats. Use this format to share text between clients and z/OS applications. In text mode, the following attributes apply: • **blankstrip** and **noblankstrip**. Refer to entry for **blankstrip** in this table. • End-of-line specifiers (**lf, cr, lfcr, crlf,** or **noeol**) are used to indicate the logical record boundary. Refer to the entry for **lf** in this table.
blankstrip	With text mode, strips trailing blanks at the end of each record of a fixed-length text file when the file is read. Pads the end of each file or record with blanks when a text file is written.
noblankstrip	Does not strip trailing blanks at the end of fixed-length records when a fixed-length text file is read. Does not pad records when writing a text file. The file must be of the correct size or an I/O error is reported to the client. For information on the **text** mode, refer to the **text** option of “devtab” on page 87.
	With text mode, use one of the following end-of-line specifiers:
lf	Line Feed is the end-of-line terminator (standard AIX® or z/OS UNIX).
cr	Carriage Return is the end-of-line terminator.
lfcr	Line Feed followed by Carriage Return is the end-of-line terminator.
crlf	Carriage Return followed by Line Feed is the end-of-line terminator (standard DOS).
noeol	No end-of-line terminator. For information on the **text** mode, refer to the **text** option of “devtab” on page 87.
executebiton	Turns on the execute bits in user, group, and other (as reported with the **ls** (list) AIX or z/OS UNIX command) for files. Use when storing executables or shell scripts on the z/OS system.

Table 2. Processing attributes (continued)

Processing Attribute	Description
executebitoff	Turns off the execute bits in user, group, and other for the mount point's files.
fastfilesize	Causes the SMB server to approximate the file size. For more information, refer to "Handling of the file size value" on page 153.
nofastfilesize	For Direct Access data sets (PDSs and PDSEs) and non-system managed data sets, this specifies to read the entire file or member to get the file size. Using this attribute might cause a noticeable delay when first accessing very large data sets. For more information, refer to "Using fastfilesize to avoid read-for-size:" on page 155.
mapleaddot	Turns on mapping of a single leading "." from a client file name to a leading "$" on z/OS. This option would normally be enabled for access by AIX and z/OS UNIX clients.
nomapleaddot	Turns off mapping of a single leading "." from a client to a leading "$" on z/OS.
maplower	Turns on mapping of lower case file names to upper case when accessing files on z/OS, and back when sending to the network. This option would normally be enabled for access by PC, AIX, or z/OS UNIX clients.
nomaplower	Turns off mapping of lower case file names to upper case and back when using files on z/OS.
	The SMB server uses DFSMShsm™ to recall or delete migrated files. The action that the server takes against the migrated files depends on which of the **retrieve** or **notrieve** attributes is active.
retrieve(nowait)	When the **retrieve(nowait)** attribute is active, the server does not wait for the recall to finish, and immediately returns a "device not available" message. You can try accessing the file later when the recall has completed.
noretrieve	When the **noretrieve** attribute is active, the server does not recall the file, and can return "device not available" upon a lookup, an rdwr, or a create request for a file. For more information, refer to "Retrieve Attributes".
setownerroot	Sets the user ID in a file's attributes to `root`.
setownernobody	Sets the user ID in a file's attributes to `nobody`.

Retrieve Attributes

The server deletes the migrated file upon a remove request for a file, regardless of whether the **retrieve** or the **noretrieve** attribute is active. Typically, a remove request is preceded by a lookup request. If the data set was migrated with DFSMS/MVS 1.2 or below, retrieve attribute causes a recall because lookup processing needs to open the data set and read for size. If the data set was migrated under DFSMS/MVS 1.3 and DFSMShsm 1.3 or later, and is SMS managed, its attributes were saved on DASD; therefore it is not always necessary to recall the data set to read for size and the data set may be deleted without recall. If the **noretrieve** attribute is active, the lookup can return a "device not available" message. If the client code decides to ignore the error and go with the remove, the migrated file is then deleted.

The z/OS UNIX command **ls mvsusera** does not issue requests for individual files under the `mvsusera` directory. Migrated files under the `mvsusera` directory are displayed, but are not recalled. However, the z/OS UNIX command **ls -l mvsusera** issues lookup requests for individual files under the `mvsusera` directory.

Site Attributes

The site attributes are used to control SMB server resources. These attributes are described in the following table. Site attributes can be overridden in the **devtab** entry for the fileset.

Table 3. Site attributes

Site Attribute	Description
bufhigh(*n*)	Specifies the maximum size (in bytes) of allocated buffers before buffer reclamation (refer to the **percentsteal** attribute in this table) is initiated. *n* is an integer from 1MB to 128MB (the default is **2MB**). If the combined total specified in the **bufhigh** and **logicalcache** attributes is greater than the available storage in the extended private area (implied by the REGION parameter in your procedure) at startup, the server shuts down immediately. A higher number means more caching and potentially better read performance.
filetimeout(*n*)	Specifies the amount of time, in seconds, before a data set is closed and data is written to DASD. The minimum specification is 30. The default is **30**.
percentsteal(*n*)	Specifies the percent of the buffers reclaimed for use when the **bufhigh(*n*)** limit has been reached. A higher value means a reclaim operation is performed less often, but the cached data is significantly trimmed on each reclaim. This can result in poor read performance because readahead buffers might be stolen. Lower values result in more frequent reclaim operations, but the cached data normal water mark is higher, meaning possibly better performance by reading out of cached data. n is an integer from 1 to 99 (the default is **20**).

Unsupported Attributes

The following NFS Server attributes are not supported by the SMB server and are ignored.

- **attrtimeout**
- **cachewindow**
- **logicalcache**
- **maxrdforszleft**
- **noattrtimeout**
- **noreadtimeout**
- **nowritetimeout**
- **readaheadmax**
- **readtimeout**
- **retrieve**
- **retrieve(wait)**
- **writetimeout.**

Examples

The following is an example of an attributes (**rfstab**) file:

```
blksize(6160)
dir(250)
dsntype(pds)
dsorg(ps)
keys(64,0)
lrecl(80)
recfm(fb)
recordsize(512,4K)
nonspanned
space(100,10),blks
blankstrip
lf
executebiton
nofastfilesize
filetimeout(30)
mapleaddot
maplower
noretrieve
setownerroot
bufhigh(2M)
percentsteal(20)
```

Attributes file (rfstab)

Note: An example attributes file can be found in **/opt/dfsglobal/examples**.

Implementation Specifics

The attributes file is in either an HFS file, a fixed-block partitioned data set, or a fixed-block sequential data set with a record length of 80. The file's location is specified by the **_IOE_RFS_ATTRIBUTES_FILE** environment variable for the **dfskern** process. For example, **_IOE_RFS_ATTRIBUTES_FILE=/opt/dfslocal/var/dfs/rfstab** is the default location.

The SMB server can use the same attributes file as the DFSMS/MVS NFS server. For example, **_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'**, would cause the SMB server to use the member NFSSATT in PDS NFSADMIN.NFSS for the RFS attributes. (The NFS server attributes file is specified in the NFSATTR DD statement of the NFS server startup procedure.) NFS server attributes that are not supported by the SMB server are ignored. Refer to "Unsupported Attributes" on page 85 for a list of attributes that are not supported.

An attributes file can also be specified on an RFS file system basis. Its location can be specified in the **devtab** entry for the RFS file system on the same line as the data set prefix name: **attrfile** *attributes_file* (where *attributes_file* is the path name of the attributes file that controls the data set creation, processing and site attributes for this RFS file system). An example **devtab** entry might look like this:

```
* RFS devices
define_ufs 10 rfs
USERA.PCDSNS attrfile /opt/dfslocal/var/dfs/rfstab2
```

If no **attrfile** parameter is specified in the devtab entry for the RFS file system, the attributes are taken from the global attributes file specified in the **_IOE_RFS_ATTRIBUTES_FILE** environment variable in the **dfskern** process.

If no **_IOE_RFS_ATTRIBUTES_FILE** environment variable is specified, then the SMB server system defaults are used for RFS attributes.

Related Information

Commands:

- **dfsexport**
- **dfsshare**

Files:

- **devtab**
- **dfstab**
- **hfsattr**
- **smbtab**

devtab

Purpose

Stores identifying information for all HFS and RFS file systems to be exported (and shared). (Unless otherwise noted, HFS includes file systems of type HFS, ZFS, TFS, and AUTOMNT. If you are in a sysplex with Shared HFS, SMB support of ZFS is limited to ZFS compatibility mode file systems.)

Format

The **devtab** file contains the following lines:

```
*  comment
define_ufs n [hfs | rfs]
{hfs-file-system-name [text | binary | auto] |
rfs-data-set-prefix [text | binary] [attrfile attributes-file] |
rfs-data-set-name [text | binary] [attrfile attributes-file]}
```

Options

*** comment** Specifies a comment line.

define_ufs *n* **[hfs | rfs]**
Defines an HFS file system or RFS file system, *n* specifies a minor device number which is a unique identifier that can be any number greater than zero. Specifying **hfs** (Hierarchical File System) or **rfs** (Record File System) determines the type of file system. **hfs** is the default.

Note: **hfs** includes file systems of type HFS, ZFS, TFS, and AUTOMNT. If you are in a sysplex with Shared HFS, SMB support of ZFS is limited to ZFS compatibility mode file systems.

hfs-file-system-name
Identifies the file system name of the HFS, ZFS, TFS, or AUTOMNT file system that you want exported. If you are in a sysplex with Shared HFS, SMB support of ZFS is limited to ZFS compatibility mode file systems.

Note: If you do not want the *hfs-file-system-name* to be folded to upper case, put the name in double quotes. For example, "/tmp". This is usually appropriate for file systems of type TFS and AUTOMNT. It may be appropriate for HFS or ZFS if the HFS file system was mounted with lowercase characters in the file system name (using an environment that does not fold the file system name to upper case, for example, ishell). To determine if a mounted file system's name includes lowercase characters, use the OMVS **df** command.

rfs-data-set-prefix
Identifies the prefix of the record data sets that you want exported.

rfs-data-set-name
Identifies the data set name of a Partitioned Data Set (PDS) or Partitioned Data Set Extended (PDSE) that you want exported.

attrfile *attributes-file*
Identifies the name of the attributes file (**rfstab**) to be used for this RFS file system.

text | binary Specifies whether the data needs to be translated (**text**) or not (**binary**). The default is controlled by the **_IOE_HFS_TRANSLATION** environment variable setting in the **dfskern** process (for HFS file systems) and by the **_IOE_RFS_TRANSLATION** environment variable setting in the **dfskern** process (for RFS file systems).

auto Specifies that the decision to translate HFS data is based on whether the first 255 bytes of the file are deemed to be valid characters.

Usage

The **devtab** file is used to define HFS file systems and RFS file systems to be exported. The **devtab** file resides in the directory named **/opt/dfslocal/var/dfs**. HFS and RFS file systems must be exported in order to be accessible by PC users. The file system containing the shared directory is automatically exported when the **dfsshare** command is issued (assuming it has proper **dfstab**, **devtab**, and **smbtab** entries). HFS file systems below the shared directory must be explicitly exported by using the **dfsexport** command if they are being made available to PC users (unless you are using dynamic export).

The **devtab** file is an EBCDIC file that can be edited with a text editor. You must have write (**w**) and execute (**x**, sometimes called search) permissions on the **/opt/dfslocal/var/dfs** directory to create the file. You must have write permission on the file to edit it.

To export an HFS or RFS file system, it must be defined to SMB. It is defined by creating an entry in the **devtab** file. Entering **define_ufs** *n* in the **devtab** file defines the type of file system as **ufs**, an HFS file system, and maps a unique minor device number, *n*, to the HFS file system you want to export. You can also enter **define_ufs** *n* **rfs** to the **devtab**, where **rfs** indicates that the file system is an RFS file system. The minor device number is a unique identifier that can be any number greater than zero. This number becomes part of the name of the device name (in the **dfstab** entry). Each HFS or RFS file system being exported must have a unique device number defined. The file system name of the HFS file system or the data set name prefix for RFS is entered in the **devtab** file on the next line after the device number definition. Before exporting an HFS file system, the HFS file system **must** be locally mounted. For information about allocating and mounting ZFS file systems, refer to the *z/OS: Distributed File Service zSeries File System Administration* document. For information about allocating and mounting HFS file systems, refer to the *z/OS: UNIX System Services Planning* document, GA22-7800. RFS file systems are not locally mounted.

You can also specify an optional character data translation parameter on the same line after the file system name of the HFS File System or the data set name prefix of the RFS file system. The possible values for the translation control parameter are the following:

binary Do not translate the data.

text Translate the data with the default translation tables. The default for local data is the local code page for the **dfskern** process. The code page for network data is ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process). Refer to “Examples” on page 89 for an example using the **text** parameter.

An additional translation control parameter value (**auto**) for HFS is available for HFS file systems and an additional translation configuration file (**hfsattr**) is available for HFS.

The additional translation control parameter value for HFS is:

auto Determine whether to translate the data based on the contents of the data. The algorithm is as follows:

- Outgoing data (client read) - if the first 255 bytes of data are valid EBCDIC characters then translate to ASCII
- Incoming data (client write) - if the first 255 bytes of data are valid ASCII characters then translate to EBCDIC.

Valid characters include POSIX C printable characters plus carriage-return, newline, and tab. In EBCDIC, the POSIX C printable characters include X'40', X'4B'-X'50', X'54'-X'61', X'6B'-X'6F', X'79'-X'7F', X'81'-X'89', X'91'-X'99', X'A1'-X'A9', X'AD', X'BD', X'C0'-X'C9', X'D0'-X'D9', X'E0', X'E2'-X'E9', X'F0'-X'F9'. Tab is X'05', carriage-return is X'0D', and

newline is X'15'. In ASCII, the POSIX C printable characters include X'20'-X'7E'. Tab is X'09', carriage-return is X'0D', and newline is X'0A'.

If the translation control parameter is omitted, the default is controlled by the **_IOE_HFS_TRANSLATION** environment variable setting in the **dfskern** process (for HFS file systems) and by the **_IOE_RFS_TRANSLATION** environment variable setting in the **dfskern** process (for RFS file systems).

In the case of an RFS file system, you can also optionally specify the name of an attributes file (**rfstab**) on the same line after the *rfs-data-set-prefix* or *rfs-data-set-name* by using the **attrfile** keyword. Refer to "Attributes file (rfstab)" on page 80 for more information on the attributes file.

Examples

The following examples show **devtab** entries for HFS and RFS file systems. All entries beginning with an asterisk (*) are comment lines.

The following example shows a **devtab** entry for an HFS file system. The second line, **define_ufs 2**, defines the type of file system as **ufs**, an HFS file system. A unique minor device number of **2** is assigned. The line following the definition of the HFS file system minor device number, **omvs.user.abc**, identifies the name of the HFS file system being exported and specifies a translation control parameter of **text**. A translation control parameter of text means data is translated. The HFS file system device name is **/dev/ufs2** with **2** representing the device's minor number.

```
* HFS devices
define_ufs 2
omvs.user.abc text
```

Examples of other file system types:

```
* ZFS devices
define_ufs 6
omvs.prv.compat.aggr001 text
* TFS devices
define_ufs 4
"/dev" text
* AUTOMNT devices
define_ufs 5
"*AMD/home" text
```

The following example shows a **devtab** entry for an RFS file system. The second line, **define_ufs 3 rfs**, defines the type of file system as **ufs** and the **rfs** parameter qualifies it as an RFS file system. A unique minor device number of **3** is assigned. The next line, **USERA.PCDSNS**, is the prefix of the record data sets that you want to export as a single RFS file system. The RFS file system device name is **/dev/ufs3** with **3** representing the device's minor number.

```
* RFS devices
define_ufs 3 rfs
USERA.PCDSNS text
```

Note: An example **devtab** file can be found in **/opt/dfsglobal/examples**.

In summary, the **define_ufs 2** entry in the **devtab** corresponds to the **/dev/ufs2** entry in the **dfstab** and **smbtab**. The **define_ufs 3 rfs** entry in the **devtab** corresponds to the **/dev/ufs3** entry in the **dfstab** and **smbtab**.

Implementation Specifics

The **devtab** file is stored as an EBCDIC file in HFS.

devtab

Related Information

Commands:

dfsexport
dfsshare

Files:

dfstab
hfsattr
rfstab
smbtab

dfstab

Purpose

Specifies HFS and RFS file systems that can be exported.

Usage

The **dfstab** file includes information about each file system that can be exported from the local disk and then shared with SMB clients. The file is read by the **dfsexport** command, which exports specified HFS and RFS file systems. (It is also read by the **dfsshare** command, which initializes SMB shares.) The **dfstab** file must reside in the directory named **/opt/dfslocal/var/dfs**. The **dfsexport** command looks in that directory for the file; if the file is not there, no file systems can be exported.

The **dfstab** file is an EBCDIC file that can be edited with a text editor. You must have write (**w**) and execute (**x**, sometimes called search) permissions on the **/opt/dfslocal/var/dfs** directory to create the file. You must have write permission on the file to edit it.

The file contains a one-line entry for each HFS or RFS File System available for exporting and sharing. Each entry in the file must appear on its own line.

The fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab.

Device name The device name of the HFS or RFS file system being exported; for example, **/dev/ufs2**.

File System name
The name associated with the HFS or RFS file system being exported. A file system name can contain any characters, but it can be no longer than 31 characters. It must be different from any other file system name in the **dfstab** file. File system names cannot be abbreviated, so you should choose a short, descriptive name; for example, **hfs2**.

File System type
The identifier for the type of file system. For HFS and RFS file systems, it must be **ufs**. Enter the identifier in all lowercase letters.

File System ID
A positive integer different from any other file system ID in the **dfstab** file.

Fileset ID The unique fileset ID number associated with the HFS or RFS fileset. Fileset ID numbers are represented as two positive integers separated by a pair of commas. For example, the fileset ID number of the first fileset is **0,,1**. When specifying a new fileset, increment the fileset ID. When the integer after the commas becomes greater than 2^{32}, the integer before the commas becomes 1 and the integer after the commas returns to 0 (zero) (that is, **1,,0**). This number must be different from any other fileset ID in the **dfstab**.

Note: The file system parameters are referred to as aggregate parameters in the *z/OS: Distributed File Service DFS Administration* document. If DCE DFS protocols are used along with the SMB protocols, the Fileset ID parameter must be assigned by the **flserver** and stored in the FLDB (Fileset Location Data Base) by using the **fts crfldbentry** command (refer to Appendix C, “Using both SMB and DCE DFS” on page 139 for information on using the SMB protocol with the DCE DFS protocol). If DCE DFS protocols are not being used, the Fileset ID needs only to be unique from other Fileset IDs.

When the **dfsexport** command is executed, it reads the **dfstab** file to verify that each HFS and RFS file system being exported is listed in the file. A file system must have an entry in the **dfstab** file if it is being exported unless you are using dynamic export. To ensure that it does not export a file system that is currently exported, the **dfsexport** command refers to a list (in memory) of all currently exported file systems.

Examples

The following **dfstab** file specifies that an HFS File System and an RFS File System (**/dev/ufs2** and **/dev/ufs3**) can be exported:

```
/dev/ufs2   hfs2   ufs  101   0,,1715
/dev/ufs3   rfs3   ufs  102   0,,1718
```

There is nothing in a **dfstab** entry that distinguishes between an HFS file system and an RFS file system. It is the corresponding entry in the **devtab** (**define_ufs 3 rfs**) that indicates to the SMB server that the file system is an RFS file system.

Note: You can put comments in the **dfstab** file. Comments have a # in column 1. An example **dfstab** file can be found in **/opt/dfsglobal/examples**.

Implementation Specifics

The **dfstab** file is stored as an EBCDIC file in HFS.

Related Information

Commands:

dfsexport
dfsshare

Files:

devtab
smbtab

envar

Purpose

Specifies the environment variables for a process.

Usage

The **envar** file contains the environment variables for each Distributed File Service process. There is one **envar** file for each process. Each **envar** file is located in the corresponding home directory of each process. (For example, the environment variables for the **dfskern** process are contained in the **/opt/dfslocal/home/dfskern/envar** file.) Refer to Chapter 3, “Post installation processing” on page 11 for information on process home directories. The **envar** file is read during process initialization and each environment variable specified in the **envar** file is set for the process. Environment variables are specified in the **envar** file in the following format:

variable-name=value

Note: There should be no space between the *variable-name* or the *value* and the equal sign that separates them.

An environment variable:
- cannot be longer than 4096 characters
- must have an =
- can be continued by terminating the line with \

A line that begins with # is treated as a comment.

The **envar** file is an EBCDIC file that can be edited with a text editor. You must have write and execute permissions on the process home directory to create the file. You must have write permission on the file to edit it. Refer to Table 4 on page 117, for information on all the environment variables supported for SMB processing.

Examples

The following **envar** file entry (located in the **/opt/dfslocal/home/dfskern/envar**) specifies that SMB processing should be on:

```
_IOE_PROTOCOL_SMB=ON
```

Note: Example **envar** files can be found in **/opt/dfsglobal/examples**.

Implementation Specifics

The **envar** file is stored as an EBCDIC file in HFS.

hfsattr

Purpose

Contains directives that map a file name extension (suffix) to an indication whether the SMB File Server (**dfskern**) should translate the file data from ASCII to EBCDIC and vice versa (encoding).

Usage

The **hfsattr** file is a text file that is stored in HFS. The File Server locates the **hfsattr** file during startup (or restart) by examining the **_IOE_HFS_ATTRIBUTES_FILE** environment variable for the **dfskern** process. The **hfsattr** file has the following format:

AddType *.suffix representation encoding* **[***quality***]**

The **hfsattr** AddType directive has the same format as the WebSphere® Application Server uses in its configuration file (httpd.conf). You can point the File Server to this file. All other directives are ignored. The File Server only examines the first, second, and fourth fields. The others are ignored. Comments can be created by using the # character. All fields are case sensitive.

AddType A keyword that indicates a directive to map a suffix to an encoding.

.suffix The file name suffix. Wildcard characters are not allowed.

representation The MIME type and subtype you want to bind to files that match the corresponding suffix. This field is ignored by DFS.

encoding The type of data the file contains. The only value that the File Server looks for is **ebcdic**. This means that incoming data should be translated from ASCII (ISO8859-1 or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process) to EBCDIC (IBM-1047 or the current code page for the **dfskern** process). Outgoing data should be translated from EBCDIC to ASCII. All other values for encoding are ignored and the data is not translated. Before data is translated, it is checked for valid characters to avoid translating data that is already in the correct format.

quality This is an optional indicator of the relative importance (on a scale of 0.0 to 1.0) for the content type. This field is ignored by DFS.

If the file name suffix is not found in the **hfsattr** file, or the file name has no suffix, then translation is determined by the **devtab** translation control parameter or the **dfskern _IOE_HFS_TRANSLATION** environment variable. For more information on **devtab** refer to “devtab” on page 87, and for the **_IOE_HFS_TRANSLATION** environment variable, refer to page 120.

Examples

The following is an example of an **hfsattr** file:

```
# Map suffixes to the encoding
AddType  .bin  application/octet-stream  binary  1.0
AddType  .ps   application/postscript    ebcdic  0.8 # PostScript
AddType  .PS   application/postscript    ebcdic  0.8 # PostScript
AddType  .c    text/plain                ebcdic  0.5 # C source
AddType  .html text/html                 ebcdic  1.0 # HTML
AddType  .htm  text/html                 ebcdic  1.0 # HTML on PCs
AddType  .gif  image/gif                 binary  1.0 # GIF
```

Implementation Specifics

The **hfsattr** file is stored as an EBCDIC file in HFS.

Related Information

File:

devtab

ioepdcf

Purpose

Specifies the processes to be started by the DFS Control Task.

Usage

The **ioepdcf** file, also referred to as the Daemon Configuration File, is an EBCDIC text file used by the DFS Control Task to determine which daemons can be started during the initialization of DFS. The file is located in the **/opt/dfslocal/etc** directory. The information contained in the **ioepdcf** file includes the following:

Process Name

The name of the process to be entered in the **ioepdcf** file. Valid processes associated with SMB are:

dfskern The **dfskern** daemon. The **dfskern** server daemon is the SMB File/Print Server process.

export The **export** daemon. The **export** server daemon allows exporting of HFS file systems.

Configuration Type

The configuration type for each server. Available types for z/OS are:

CONFIGURED=Y

Specifies that the process starts during initialization. If the process abends or fails for any reason, it is automatically restarted by the DFS Control Task.

Note: Do not use this configuration type for the **export** process. This process executes once and then stops. The **export** process must not be automatically restarted once it has run. Use **CONFIGURED=I** or **CONFIGURED=M**.

CONFIGURED=N

Specifies that the process is not started by the DFS Control Task nor can the process be started manually.

CONFIGURED=I

Specifies that the process is started during initialization or when the **MODIFY** command, **START ALL** is issued. The process may be started manually. The process does not restart if it abends or ends for any reason.

CONFIGURED=M

The process is not started by the DFS Control Task but may be manually started by using the z/OS system command **MODIFY**. The specified process does not restart if it ends for any reason.

Load Module Name

The name of the load module (**LMD**) in a partitioned data set. The load module refers to the name of the member in the *xxx*.SIOELMOD (where *xxx* is installation dependent) data set created during installation (for further information, refer to the *z/OS: Program Directory*, GI10-0669). The following are the PDS member names for the processes started by the DFS Control Task:

dfskern The **dfskern** daemon load module name. The name, **dfskern**, is an alias for the load library entry, **IOEDFSKN**.

In addition, the **export** process is started by the DFS Control Task and executes the load library entry, **IOEDFSXP**. The **export** process has no alias.

Special Parameters

Parameters that are passed to the load module when a daemon is started (including Language Environment/370 (LE/370) runtime options). This is also called the argument list. Any runtime overrides such as storage specifications and redirection of output may also be added. The first argument is the home directory for each process. The home directory points the process to the directory holding the environment variable (**envar**) file for the process. Program parameters for the DFS process are preceded with a slash, **/**, in the argument list.

The **ioepdcf** file can be used to override the default parameter options for the following daemons:

dfskern
The **dfskern** server daemon is the SMB File/Print Server. There are no options that need to be overridden for this process for SMB File/Print serving.

export
The **export** daemon allows exporting of HFS file systems. Refer to “dfsexport” on page 106 for information on options available for this process.

Additional special parameters that control restart and timeout intervals may also be entered in the **ioepdcf** file. These parameters are:

Restart
The interval defined for the DFS Control Task to attempt a restart of the process. Restart values are entered in seconds.

Timeout
The maximum interval that the Control Task waits for the process to complete initialization. Timeout values are entered in seconds. If this interval is exceeded with no confirmation of successful completion received by the Control Task, the status of the process is set to **UNKNOWN**.

Examples

The following example is an **ioepdcf** file entry for the **dfskern** process. The configuration type is **Y**, specifying that the process start during DFS initialization. The load module, **LMD**, is identified as **IOEDFSKN**. In the argument list, **ARG, ENVAR** indicates the environment variables to be used. In the example, the home directory is identified as **_EUV_HOME=/opt/dfslocal/home/dfskern**. An LE runtime option is specified: **HEAPP(ON)**. Parameters for the **dfskern** process follow the **/**. The > symbol is a redirection character which indicates that the output is redirected to the DD name that follows. In this example, the redirection of the **STDERR** to **STDOUT** of the process (**dfskern**) is specified by: **>DD:dfskern 2>&1. RESTART** and **TIMEOUT** values are both set at 300 seconds.

```
dfskern CONFIGURED=Y LMD=IOEDFSKN ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern'),HEAPP(ON)/
-mainprocs 7 -admingroup subsys/dce/dfs-admin >DD:dfskern 2>&1" RESTART=300 TIMEOUT=300
```

Note: The previous example should be entered on one line even though multiple lines are used in this example.

Implementation Specifics

The **ioepdcf** file is stored as an EBCDIC file in HFS.

This file is optional. If it does not exist, the following default values are used:

ioepdcf

```
DFSKERN  CONFIGURED=Y LMD=DFSKERN  ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern')/-admingroup
subsys/dce/dfs-admin>DD:DFSKERN2>&1" RESTART=300 TIMEOUT=300
EXPORT CONFIGURED=I LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-all -verbose       >DD:EXPORT
2>&1" RESTART=300 TIMEOUT=300
UNEXPORT CONFIGURED=M LMD=IOEDFSXP ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfsexport')/-detach -all -ver
>DD:UNEXPORT2>&1" RESTART=300 TIMEOUT=300
BOSERVER CONFIGURED=M LMD=BOSERVER ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/boserver')/ >DD:BOSERVER 2>&1"
RESTART=300 TIMEOUT=300
BUTC01 CONFIGURED=M LMD=BUTC01 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc01')/ -tcid 0 >DD:BUTC01 2>&1"
RESTART=300 TIMEOUT=300
BUTC02 CONFIGURED=M LMD=BUTC02 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc02')/ -tcid 1 >DD:BUTC02 2>&1"
RESTART=300 TIMEOUT=300
BUTC03 CONFIGURED=M LMD=BUTC03 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc03')/ -tcid 2 >DD:BUTC03 2>&1"
RESTART=300 TIMEOUT=300
BUTC04 CONFIGURED=M LMD=BUTC04 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc04')/ -tcid 3 >DD:BUTC04 2>&1"
RESTART=300 TIMEOUT=300
BUTC05 CONFIGURED=M LMD=BUTC05 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc05')/ -tcid 4 >DD:BUTC05 2>&1"
RESTART=300 TIMEOUT=300
BUTC06 CONFIGURED=M LMD=BUTC06 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc06')/ -tcid 5 >DD:BUTC06 2>&1"
RESTART=300 TIMEOUT=300
BUTC07 CONFIGURED=M LMD=BUTC07 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc07')/ -tcid 6 >DD:BUTC07 2>&1"
RESTART=300 TIMEOUT=300
BUTC08 CONFIGURED=M LMD=BUTC08 ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/butc08')/ -tcid 7 >DD:BUTC08 2>&1"
RESTART=300 TIMEOUT=300
```

rfstab

Purpose

Refer to “Attributes file (rfstab)” on page 80.

smbidmap

Purpose

Maps an SMB user ID to a z/OS user ID. The mapping is used to determine the SMB user's corresponding z/OS user ID. This determines access permissions for shared HFS and RFS directories and files and owners for print requests sent to shared printers. If no **smbidmap** file is specified or it does not exist, then the SMB user ID is mapped to the default user ID (specified in the **_IOE_MVS_DFSDFLT** environment variable of **dfskern**). If no default user ID is specified, the request is denied.

Usage

The **smbidmap** file is a text file that the administrator creates and maintains. This must be created as an HFS file. The location of this file is specified in the **_IOE_SMB_IDMAP** environment variable of **dfskern**.

The **smbidmap** file contains one or more identity mapping declarations and has the following general format:

```
SMB-user-ID1

z/OS-user-ID1
...
```

This format illustrates an SMB user ID mapping entry. Each entry has two elements: SMB user ID and z/OS user ID. A blank line is required between entries. The following explains each element in an SMB user ID mapping entry:

SMB-user-ID or Domain/SMB-user-ID or Workgroup/SMB-user-ID
: Specifies the client's SMB identity. This may either be a simple SMB user ID (when you do not care what the domain of the SMB user ID is) or a fully qualified name (for clients within and outside the domain/workgroup). The SMB user ID can be up to 20 characters in length. A Domain/Workgroup name can be up to 15 characters in length.
  - *SMB-user-ID* is assumed to be in any domain.
  - *Domain/SMB-user-ID* is assumed to be in the specified domain.
  - *Workgroup/SMB-user-ID* is assumed to be in the specified workgroup.

z/OS-user-ID
: Is the z/OS user ID of the client. All potential SMB clients must have z/OS user IDs on the system where the DFS server is running.

Another entry that is allowed in **smbidmap** is:

```
*
=
```

This means that if no z/OS user ID can be determined, the SMB user ID should be used as the z/OS user ID. This only occurs if the SMB user ID is eight characters or less. This entry can be the only entry in the **smbidmap** file, if desired.

Each SMB user can only have one mapping to a z/OS user ID. However, different SMB users can be mapped to the same z/OS user ID, if desired.

Examples

In the following example, the SMB user ID **smith** in **domain1** is mapped to the z/OS user ID **CMSMITH** and SMB user ID **jones** (in any domain) is mapped to the z/OS user ID **TSJONES**.

```
domain1/smith
CMSMITH

jones
TSJONES
```

Implementation Specifics

The **smbidmap** file is stored as an EBCDIC file in HFS.

smbtab

Purpose

Specifies HFS and RFS shared directories and shared printers being made available to PC clients.

Usage

The **smbtab** file includes information about each shared directory and each shared printer that can be made available to PC clients. It resides in the directory **/opt/dfslocal/var/dfs**.

The file contains a one-line entry for each shared directory or shared printer. Each entry in the file must appear on its own line.

The shared directory fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab.

Device Name For shared directories, the device name of the HFS or RFS file system that contains the root of the directory path name being shared; for example, **/dev/ufs2**. This must match the device name of the file system in the **dfstab**.

Share name The name to be associated with the HFS or RFS directory being shared. A share name can contain numbers (0-9), letters (A-Z), and the following special characters: $ % ' _ @ ~ ` ! ( ) ^ # &. To ensure that a client can connect to an SMB share, you should limit yourself to these characters. A share name can be up to 12 characters.

Device type The device type identifier for the type of device housing the share. For HFS and RFS directories this must be **ufs**. Enter the identifier in all lowercase letters.

Share description
The text description of the share. It can be up to 40 characters and is surrounded by double quotes; for example, **"Department HFS files"**.

Shared directories permissions
For shared directories, specifies whether the share is limited to read-only access or whether read-write access is allowed. Read-only access is specified as **r/o**. Read-write access is specified as **r/w**.

In addition, permissions used during a create of a file or directory can be specified. They are specified directly after **r/w**. For example, you might specify **r/w,f=700,d=755**. These permissions override the global create permissions (**_IOE_SMB_DIR_PERMS** and **_IOE_SMB_FILE_PERMS** environment variables in **dfskern**) for this shared directory. Create permissions cannot be specified for **r/o** access.

Maximum users
For shared directories, the maximum number of users that can be connected to a share name. It can be a number between 0 and 4294967295. 0 means that there is no limit to the number of users.

Directory path name
For shared directories, specifies the path name of the HFS or RFS directory (relative to the root of the file system referred to by the **Device name**) that this share represents. For HFS, the directory must reside in a locally mounted HFS file system. For RFS, the directory must be the root of the RFS file system (/) or a directory (that is, PDS or PDSE) within the RFS file system. The file system must be exported (refer to "dfstab" on page 91). The directory path name can be up to 1024 characters. It must be surrounded by double quotes if it contains embedded blanks. For HFS, you may want to export HFS file systems below this directory in order to allow all path names below this directory to be

accessible by this share or you may want to use dynamic export. Refer to “Dynamic export for HFS” on page 38 for information on using the dynamic export capability of the SMB server.

If you are using dynamic export, a special keyword may be specified in the **Directory path name** field. The keyword is **&USERID**. It represents the PC user’s z/OS user ID. It allows a single **smbtab** entry to mean a different directory depending on which user connects to the shared directory. Refer to “Recommended technique for PC user access to automounted home directories” on page 41 for more information on the usage of the **&USERID** keyword.

The shared printer fields in the following list must appear for each entry; they must appear in the order listed, and each field must be separated by at least one space or tab.

Device name For shared printers, the device name of the printer being shared; for example, **/dev/prt1**.

Share name The name to be associated with the printer queue being shared. A share name can contain numbers (0-9), letters (A-Z), and the following special characters: $ % ’ _ @ ~ ` ! ( ) ^ # &. To ensure that a client can connect to an SMB share, you should limit yourself to these characters. A share name can be up to 12 characters.

Device Type The device type identifier for the type of device housing the share. For printers this must be **prt**. Enter the identifier in all lowercase letters.

Share description
The text description of the share. It can be up to 40 characters and is surrounded by double quotes; for example, **“Department printer”**.

Printer definition name
For shared printers, specifies the name of the printer definition. The printer definition name is created during the definition of the printer. Refer to the *z/OS: Infoprint Server Operation and Administration* document.

Printer type For shared printers, specifies the type of printer. This can be the name of a printer type supplied by Windows. Refer to the *z/OS: Infoprint Server Operation and Administration* document.

When the **dfsshare** command is executed, it reads the **smbtab** file to verify that each directory or printer to be shared is listed in the file. A directory or printer must have an entry in the **smbtab** file if it is being shared. If a directory or printer is currently shared, it is not shared again. The **smbtab** file is also read and shared directories and shared printers are created during server initialization.

Examples

The following **smbtab** file specifies an HFS directory, an RFS directory, and one printer should be shared.

```
/dev/ufs2  myshare    ufs  "My share description"    r/w  100  /ghi
/dev/ufs3  rfsshare1  ufs  "USERA.PCDSNS z/OS data sets"  r/o   50  /
/dev/prt1  myprt      prt  "Department printer"    printname1  "Generic / Text Only"
```

Note: You can put comments in **smbtab** file. Comments start with # in column 1.

Implementation Specifics

The **smbtab** file is stored as an EBCDIC file in HFS. An example **smbtab** file can be found in **/opt/dfsglobal/examples**.

Related Information

Commands:

dfsexport
dfsshare

smbtab

Files:

devtab
dfstab

Chapter 14. Distributed File Service SMB commands

This chapter provides an alphabetical listing of all relevant SMB commands.

These commands are issued from the OMVS environment. OMVS users that are not using DCE should have the following line in their HFS **.profile** file in their home directory:

```
export _EUV_AUTOLOG=NO
```

Alternatively, if no OMVS users are using DCE, then the above line may be placed in the **/etc/profile** file. This causes it to take effect for all OMVS users.

dfsexport

Purpose

An OMVS command that exports (or unexports) HFS and RFS file systems. This makes underlying file systems available so that directories contained in them may be shared (refer to “dfsshare” on page 109).

Format

dfsexport [{-all | -filesystem *name***}] [-detach] [-verbose] [-help]**

Options

-all Specifies that all HFS and RFS file systems listed in the **/opt/dfslocal/var/dfs/dfstab** file are being exported. Use this option or use **-filesystem**; omit both options to list all file systems currently exported.

-filesystem *name*
Specifies the file system name of the HFS or RFS file system to be exported. This name is specified in the first or second field of the entry for the HFS or RFS file system in the **dfstab** file. Use this option or use **-all**; omit both options to list all HFS file systems currently exported.

Note: If you enter this command in TSO/E, the name parameter **must** be surrounded by double quotes (*""*). Also, the **-filesystem** option can also be specified as **-aggregate** as shown in the *z/OS: Distributed File Service DFS Administration*.

-detach Used with the **-all** option, specifies that the file system(s) indicated with the command’s other options are to be detached (no longer exported), making the corresponding shares unavailable to Windows clients. Use **-all** or **-filesystem** with this option indicating the exports are to be detached.

Use the **-detach** option only when no users are accessing data on the HFS or RFS file systems to be detached or when a serious emergency warrants its use. When the **-detach** option is used, the command breaks all oplocks for data on a corresponding share before the file system is detached.

To permanently detach a file system, it must also be removed from the **dfstab** file. Otherwise, the **dfsexport** command exports file systems the next time it is run.

-verbose Directs the command to report on its actions as it executes.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Note: There is a **-force** and a **-type** option on the **dfsexport** command that only apply to DCE DFS protocols.

Usage

The **dfsexport** command exports underlying HFS and RFS file systems so that directories may be shared from z/OS to PC clients. Directories and printers that are shared are still available to other z/OS users. Issue this command with no options to list the file systems already exported. The binary file for the **dfsexport** command resides in **/opt/dfsglobal/bin/dfsexport**.

The **dfsexport** command exports HFS and RFS file systems based on the values provided with its options. If the **-all** option is provided, the command shares all file systems listed in the **/opt/dfslocal/var/dfs/dfstab** file. If the **-filesystem** option is provided, it exports only the file system whose name is specified with the option. The specified name must be listed in the **dfstab** file.

When **dfsexport** executes, it reads the **dfstab** file on the local disk of the machine to determine the file systems available to be exported. A file system must have an entry in the **dfstab** file if it is being exported. Because this command reads the **dfstab** file, information supplied with its options must match exactly the information for a file system specified in that file.

The **dfsexport** command reads a list of all currently exported file systems that is maintained in the SMB server of the local machine. The command does not export a file system that is currently exported.

Issuing the **dfsexport** command with no options lists the file systems currently exported from the local file server.

The **dfsexport** command is normally executed automatically when the SMB server is started. This is controlled by the **export** entry of the **ioepdcf** file, refer to “ioepdcf” on page 96 for more information. When export is enabled, all indicated HFS and RFS file systems listed in the **dfstab** file are exported and all HFS and RFS directories listed in **smbtab** that are contained in those exported file systems are shared.

Prior to using this command to export a file system for the first time, perform the following:

1. For HFS file systems, ensure that the file system is mounted locally; it can contain data or it can be empty. The SMB server must be running on the z/OS system that owns the HFS file system. (RFS file systems cannot and need not be locally mounted.)
2. Create an entry for the file system in the **devtab** file on the z/OS system on which the file system resides. This entry maps the minor device number to the HFS or RFS file system data set you wish to export. It also allows you to (optionally) specify whether character data translation should occur when the data is read or written by PC clients. For further information, refer to “devtab” on page 87.
3. Create an entry for the file system in the **dfstab** on the machine on which the partition resides. Use a unique file system name and ID number and a unique fileset ID number in the appropriate fields of the entry for the file system.

Before exporting a file system, also make sure that no local users have files open on the file system. The SMB server cannot effectively synchronize file access between users who opened files from a file system before the file system was exported and users who open files from the file system after the file system is exported because only the latter have tokens.

Privilege Required

The issuer must be logged in as **root** on the local machine. On z/OS, **root** refers to a user with a **UID = 0**.

Cautions

Before using the **-detach** option with this command, make sure no users are currently accessing data from file systems to be detached. The command does not verify that a device is not in use before removing it from the namespace. A user who is accessing data housed on a file system when it is detached is not able to save the data back to the device. Any attempt to perform an action that involves a detached file system elicits a message reporting that the device is Unknown.

Examples

On SMB, the **dfsexport** process, by default, is started automatically by the DFS control task program, **dfscntl. dfscntl** and its child processes are, in turn, controlled in SMB by the z/OS system command, **MODIFY**.

You can also use the **MODIFY** system command to export and unexport file systems SMB.

The following command causes the **dfsexport** program to run with the parameters specified in the **ioepdcf** configuration file. By default, this command exports all of the file systems that have entries in the machine’s **dfstab** file:

```
modify dfs,start export
```

dfsexport

The following command causes the **dfsexport** program to run with the parameters specified in the **ioepdcf** configuration file. In this example, the command unexports all of the file systems that have entries in the **dfstab** file:

```
modify dfs,start unexport
```

The following command exports the file system whose device name (as it appears in the **dfstab** file) is **/dev/ufs1**:

```
# dfsexport /dev/ufs1
```

Implementation Specifics

The **dfsexport** process on SMB, by default, is started automatically by the DFS control task program, **dfscntl**. The **dfsexport** process on DFS runs under the control of the DFS control task, **dfscntl. dfscntl** and its child processes are controlled on DFS by the **MODIFY** system command. For further information, refer to “modify dfs processes” on page 74.

After **dfsexport** exports the file system(s) specified, it creates any shared directories defined in **smbtab** that refer to those file systems. No shared printers are created. When **-detach** is specified, **dfsexport** unshares any shared directories that refer to file systems being unexported before unexporting the file systems.

This command may be issued from TSO/E or a z/OS shell.

Note: In order to export an HFS file system, that file system must be owned by the system that the SMB server is running on. When export is attempted on a (sysplex) shared file system that is owned by a system other than the one that the SMB server is running on, the SMB server attempts to move the ownership of a shared HFS file system to the system that the SMB server is running on (when the **_IOE_MOVE_SHARED_FILESYSTEM** environment variable is **ON** in the **dfskern** process). If this is unsuccessful (when, for example, the file system is being exported by another SMB server on the other system), the SMB server will not be able to export that file system and it will not be available to PC clients.

Related Information

Files:

- **devtab**
- **dfstab**
- **smbtab**

dfsshare

Purpose

An OMVS command that shares (or unshares) HFS and RFS directories or printers with SMB clients.

Format

dfsshare [{-all | -share *name*}] **[-type** *name*] **[-detach] [-verbose] [-help]**

Options

-all Specifies that all HFS and RFS directories and printers listed in the **/opt/dfslocal/var/dfs/smbtab** file are shared with SMB clients. Use the **-type** option with this option to export only HFS and RFS files or only Infoprint Server printers. Use this option or use **-share**; omit both options to list all shared files and shared printers currently shared with SMB clients.

-share *name* Specifies the share name of the HFS or RFS shared directory or shared printer. This name is specified in the second field of the entry for the HFS directory or printer in the **smbtab** file. Use this option or use -**all**; omit both options to list all HFS directories and printers currently shared with SMB clients.

Note: Some share names cannot be specified on OMVS commands or must be enclosed in single quotes. A share name that begins with dollar ($) must be enclosed in single quotes.

-type *name* Specifies that only shared directories or shared printers whose type matches the type specified with this option are shared. The type can be specified as **ufs** to share only HFS and RFS directories, or it can be specified as **prt** to share only printers. The type of each share appears in the third field of the entry for the device in the **smbtab** file.

Use this option only with the **-all** option; it is ignored if it is used without the **-all** option. If it is omitted and **-all** is used, the command shares both **ufs** and **prt** types.

-detach Used with the **-all** option, specifies that the shared directories and shared printers indicated with the command's other options are detached (no longer shared), making them unavailable to SMB clients. Use **-all** or **-share** with this option to indicate the shares to be detached; use the **-type** option with **-all** to detach only one type of share.

Use the **-detach** option only when no users are accessing data on the HFS or RFS shared directories to be detached or when a serious emergency warrants its use.

To permanently detach a share, it must also be removed from the **smbtab** file. Otherwise, the **dfsshare** command shares the directories and printers the next time it is run (provided the directories or printers are included in the specification for the types to be shared).

-verbose Directs the command to report on its actions as it executes.

-help Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

The **dfsshare** command shares HFS and RFS directories and printers from z/OS to Windows clients. Directories and printers that are shared are still available to other z/OS users. Issue this command with no options to list the directories and printers already shared. The binary file for the **dfsshare** command resides in **/opt/dfsglobal/bin/dfsshare**.

The **dfsshare** command shares HFS and RFS directories, printers, or both based on the values provided with its options. If the -all option is provided, the command shares all directories and printers listed in the

/opt/dfslocal/var/dfs/smbtab file. If the **-share** option is provided, it shares only the directory or printer whose share name is specified with the option. The specified name must be listed in the **smbtab** file.

The **-type** option can be used with the **-all** option to indicate that only directories or only printers are shared. If **ufs** is provided with the **-type** option, the command exports only directories; if **prt** is provided with the **-type** option, it exports only printers. If the **-type** option is used, the **-all** option must also be included; otherwise, the **-type** option is ignored.

When **dfsshare** executes, it reads the **smbtab** file to determine the directories and printers available to be shared. A directory or printer must have an entry in the **smbtab** file if it is shared. This command also invokes **dfsexport** to ensure that the underlying HFS or RFS file system that contains the shared directory is exported. Therefore, the corresponding HFS or RFS file system information must exist in **dfstab** and **devtab** for the shared directories that are being created. If there are additional file systems mounted below the shared directory that you want to allow PC users to access, those file systems must be exported also. The **dfsexport** command can be used for this purpose.

The **dfsshare** command reads a list of all currently shared directories and printers that is maintained in the SMB Server of the local machine. The command does not share a directory or printer that is currently shared. If you want to change the parameters for a printer or an HFS or RFS directory that is already shared, you must first detach (by using the **dfsshare -share** *name* **-detach** command) and then reshare (by using the **dfsshare -share** *name* command) for the new parameters to take effect.

Issuing the **dfsshare** command with no options lists the directories and printers currently shared to SMB clients.

The **dfsshare** command is automatically executed when the SMB server is started (with SMB processing enabled). This is controlled by the **export** entry of the **ioepdcf** file, refer to “ioepdcf” on page 96 for more information. When export is enabled, all indicated HFS and RFS directories listed in the **smbtab** file are shared and all HFS and RFS file systems listed in **dfstab** and **devtab** are exported. In addition, all printers listed in the **smbtab** are shared (if the Infoprint Server is enabled) regardless of the export entry in the **ioepdcf** file.

Privilege Required

The issuer must be logged in as **root** on the local machine. On z/OS, **root** refers to a user with a **UID = 0**.

Implementation Specifics

The **dfsshare** command exports the file system that is referred to by the shared directory (if it is not already exported) before creating the shared directory. When **-detach** is specified, the shared directory is unshared but no file systems are unexported.

The **dfsshare** command can only be issued from the z/OS shell. It is not supported as a TSO/E command.

Note: In order to export an HFS file system, that file system must be owned by the system that the SMB server is running on. When export is attempted on a (sysplex) shared file system that is owned by a system other than the one that the SMB server is running on, the SMB server attempts to move the ownership of a shared HFS file system to the system that the SMB server is running on (when the **_IOE_MOVE_SHARED_FILESYSTEM** environment variable is **ON** in the **dfskern** process). If this is unsuccessful (when, for example, the file system is being exported by another SMB server on the other system), the SMB server will not be able to export that file system and it will not be available to PC clients.

Related Information

Files:

devtab
dfstab
smbtab

Command:

dfsexport

smbpw

Purpose

An OMVS command that stores a z/OS user's SMB password in the RACF database for use with SMB encrypted password support. Encrypted password support is used when the SMB server is configured to support encrypted passwords by using the **_IOE_SMB_CLEAR_PW** environment variable in the **dfskern** process.

Format

smbpw [-pw1 *newpassword* **-pw2** *newpassword***] [-help]**

Options

-pw1 *newpassword*
: Specifies the SMB password to be stored in the current z/OS user's RACF DCE segment. The SMB password can be up to 14 characters. If the user issues the **smbpw** command without providing the password, **smbpw** prompts the user for the SMB password.

-pw2 *newpassword*
: Specifies the same password again. This is to verify that the password was entered correctly.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored.

Usage

During SMB login processing, when encrypted password processing is configured, the SMB server retrieves the user's SMB password from the user's RACF DCE segment. The **smbpw** command is used when a user needs to initially store or change the SMB password in their RACF DCE segment. The password entered is case sensitive. That is, it is stored in the RACF DCE segment exactly as the user typed it. In general, the password should be entered in lower case. It is folded to upper case by the SMB server, if necessary. This command requires that the changed password is entered twice.

The user may provide none or both occurrences of the password on the command line. If none is supplied, the system prompts for the changed password and then prompts for the changed password again. These passwords are verified to match.

The **smbpw** command does not verify that the password specified is the same as your Windows password. If you enter an incorrect password twice, the password is saved. If you determine that you have stored an incorrect password, use **smbpw** again, supplying the correct password.

If you are using SMB encrypted passwords and DCE single sign-on, your SMB password and your DCE password must be the same since both of these come from the RACF DCE segment. However, if the DCE Security Server is running on the same system as the DCE user, then the DCE single sign-on processing does not require the DCE password to be stored in RACF. So, in this case, the SMB password and the DCE password can be different.

Privilege Required

No privileges are required.

Examples

In this example, **mynewpw** is the SMB password to be stored in the RACF DCE segment:

```
$ smbpw mynewpw mynewpw
```

The following example is the same as above except that the user is prompted for the new password:

```
TEST1:/home/test1/> smbpw
IOEW16057I Enter Password: mynewpw
IOEW16058I reenter Password: mynewpw
IOEW16055I Your password has been updated.
```

Note: The passwords would not actually be displayed when they are entered at the prompt.

Implementation Specifics

The **smbpw** command can only be issued from the z/OS shell. It is not supported as a TSO/E command. The user issuing the **smbpw** command must have a RACF DCE segment.

z/OS users that do not use DCE should put the following line in their **.profile** file in their home directory:

```
export _EUV_AUTOLOG=NO
```

Alternatively, if no z/OS users are using DCE, the above line may be placed in the **/etc/profile** file.

smbpw

Part 3. Appendixes

Appendix A. Environment variables in SMB

This appendix lists the environment variables that affect the behavior of the SMB components. The environment variables that begin with **_IOE** are interpreted by Distributed File Service processes. The other environment variables are interpreted by other z/OS components (for example, z/OS Language Environment).

Note: In the following table all the examples should be entered on one line, with no blanks, even though some of them appear on multiple lines.

Table 4. Environment variables in SMB

Name	Description
_EUV_AUTOLOG	This environment variable controls whether DCE autologin processing is attempted. For the SMB server processes (**dfscntl**, **dfskern**, and **dfsexport**), this environment variable should always be set to **NO**. SMB Administrators who are not using DCE facilities should also specify this environment variable as **NO** in their OMVS environment. SMB users that are storing their SMB password by using the **smbpw** command should also specify this environment variable as **NO** in their OMVS environment. The valid value is: **NO** Do not enable single sign-on processing. Any other value causes DCE autologin processing to be attempted.
DCE_START_SOCKET_NAME	The path name used as the well-known socket name by the SMB server processes during initialization. **Default Value** /opt/dfslocal/home/dfskern/ioepk.soc **Expected Value** Character string. **Example** `DCE_START_SOCKET_NAME=/opt/dfslocal/home/dfscntl/ioepk.soc` **Where Variable is Used** All programs. If the default is not being used, this environment variable must be coded in the **envar** file for each process.
LIBPATH	The path names used to find DLLs. **Default Value** None **Expected Value** Character string. **Example** `LIBPATH=/usr/lib:/usr/lpp/Printsrv/lib` **Where Variable is Used** DFSKERN
NLSPATH	A POSIX environment variable used by DCE that sets the search path for message catalogs. **Default Value** /usr/lib/nls/msg/En_US.IBM-1047/%N: /usr/lib/nls/msg/%L/%N: /usr/lib/nls/msg/prime/%N **Where Variable is Used** All SMB server processes

Table 4. Environment variables in SMB (continued)

Name	Description
TZ	Sets the time zone used by a process. **Default Value** localtime **Where Variable is Used** All SMB server processes
_IOE_DAEMONS_IN_AS	This environment variable controls whether the DFSKERN process runs in its own address space or in the DFS Server Address Space. **Default Value** The default is ''. If '' is specified or the environment variable is not specified, **dfskern** runs in the DFS Server Address Space. **Expected Value** DFSKERN or '' **Example** `_IOE_DAEMONS_IN_AS=DFSKERN` In this case, DFSKERN runs in its own address space **Where Variable is Used** DFSCNTL
_IOE_DFS_MODIFY_PATH	The path used as the well-known socket name by the SMB processes when registering with DFSCNTL to receive modify commands (F DFS,QUERY DFSKERN). **Default Value** /opt/dfslocal/home/dfscntl/modify.rendezvous **Expected Value** Character string. **Example** `_IOE_DFS_MODIFY_PATH=/opt/dfslocal/home/dfscntl/test.rendezvous` **Where Variable is Used** All programs. If the default is not being used, this environment variable must be coded in the **envar** file for each process.
_IOE_DIRECTORY_CACHE_SIZE	The number of 512 byte blocks used to cache HFS directory entries. **Default Value** 2048 **Expected Value** The value specified must be a numeric value greater than or equal to 768. **Example** `_IOE_DIRECTORY_CACHE_SIZE=4096` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_DYNAMIC_EXPORT	If ON, when a client crosses a local mount point into a file system that is not known to the SMB server, the file system is dynamically assigned values and exported. **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_DYNAMIC_EXPORT=ON` **Where Variable is Used** DFSKERN **Notes** This environment variable is turned off when DCE DFS processing is enabled (when _IOE_PROTOCOL_RPC=ON).
_IOE_EXPORT_TIMEOUT	The number of minutes that a file system must be idle before it is dynamically unexported. The root of a share is never dynamically unexported. **Default Value** 15 **Expected Value** OFF or a number of minutes greater than 0 and less than 480 (8 hours). **Example** `_IOE_EXPORT_TIMEOUT=OFF` **Where Variable is Used** DFSKERN
_IOE_HFS_ATTRIBUTES_FILE	Specifies the path name of the **hfsattr** file that contains the definition of file name suffixes that controls whether the data should be translated by the SMB server. **Default Value** None **Expected Value** Character string, 132 characters or less. **Example** `_IOE_HFS_ATTRIBUTES_FILE=/etc/httpd.conf` **Where Variable is Used** DFSKERN **Notes** This file contains AddType statements in the same format as the IBM HTTP Server's httpd.conf file. All statements other than AddType are ignored. IBM supplies an example file in **/opt/dfsglobal/examples/cmattr**. This file can be copied and modified appropriately. It is recommended that the modified file reside in the **/opt/dfslocal/var/dfs** directory so that it is with the other customizable files.

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_HFS_FILETAG	The valid values are: **IGNORE** Do not query or set file tags. **QUERY** Honor tags when reading or writing but do not set tags when a file is created. **SET** QUERY and also set tags when files are created. **Default Value** IGNORE **Expected Value** IGNORE, QUERY, or SET. **Example** `_IOE_HFS_FILETAG=QUERY` **Where Variable is Used** DFSKERN
_IOE_HFS_TRANSLATION	This environment variable controls the conversion of HFS data to the appropriate data format. Converts incoming data from ASCII ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process) to the local code page. Converts outgoing data from the local code page to ASCII ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process). Refer to “File data translation for HFS” on page 42, for information on data translation. **Default Value** OFF **Expected Value** ON, OFF, or AUTO **ON** means that all data is translated. **OFF** means that no data is translated. **AUTO** means that data that is received by (read) or sent to (written) the SMB server is examined to determine if the data is text or not. When data is received by (read) the SMB server, the first 255 bytes of data are compared against a table of valid text (EBCDIC) characters. If all 255 characters are deemed to be valid characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. When data is sent from (written) the SMB server, the first 255 bytes of data are compared against a table of valid ASCII text characters. If all 255 characters are deemed to be valid characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. Refer to “devtab” on page 87(**auto** parameter) for a list of valid code points. **Example** `_IOE_HFS_TRANSLATION=ON` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_MOVE_SHARED_FILESYSTEM	Specifies whether the SMB server should attempt to move the ownership of (Sysplex) Shared File Systems when it exports them. **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_MOVE_SHARED_FILESYSTEM=ON` **Where Variable is Used** DFSKERN
_IOE_INHERIT_TRANSLATION	When ON, for a dynamically exported file system, the translation option of the parent file system is inherited. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_INHERIT_TRANSLATION=OFF` **Where Variable is Used** DFSKERN
_IOE_MVS_DFSDFLT	The name of a RACF-defined user that is associated with unauthenticated users attempting to access shared directories or shared printers. This ID must be RACF-defined with a z/OS UNIX segment. **Default Value** None **Expected Value** A character string, eight characters or less. **Example** `_IOE_MVS_DFSDFLT=DFSDFLT` **Where Variable is Used** DFSKERN
_IOE_PROTOCOL_RPC	An environment variable that controls whether the DCE DFS protocol is supported (using DCE RPC). **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_PROTOCOL_RPC=OFF` **Where Variable is Used** DFSKERN **Notes** This environment variable also affects DCE applications and commands. It should not be used in any other processes including shell user processes.
_IOE_PROTOCOL_SMB	An environment variable that controls whether the SMB protocol is supported (using TCP/IP). **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_PROTOCOL_SMB=ON` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_RFS_ALLOC_TIMEOUT	Specifies the time period (in seconds) that an RFS data set remains allocated after there has been no access to the data set through the DFS/SMB server. After this time period, the data set is deallocated and is available to other applications such as ISPF. **Default Value** 300 (5 minutes) **Expected Value** A number greater than or equal to 30. **Example** `_IOE_RFS_ALLOC_TIMEOUT=600` **Where Variable is Used** DFSKERN
_IOE_RFS_ATTRIBUTES_FILE	Specifies the name of the (**rfstab**) file that contains the table for describing the attributes used to manipulate RFS files. The value can be the name of an HFS file, a fixed block partitioned data set, or a fixed-block sequential data set with a record length of 80. **Default Value** /opt/dfslocal/var/dfs/rfstab **Expected Value** Character string describing the attributes file being used (maximum 255 characters). **Example** `_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'` **Where Variable is Used** DFSKERN **Notes** This environment variable can be overridden for specific file systems in the **devtab** file. Refer to “devtab” on page 87.
_IOE_RFS_STATUS_REFRESH_TIME	Specifies the time interval (in seconds) that the SMB server uses to refresh its cache of exported record data set names and attributes. **Default Value** 600 (10 minutes) **Expected Value** A number greater than zero. **Example** `_IOE_RFS_STATUS_REFRESH_TIME=360` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_RFS_TRANSLATION	Specifies whether RFS data should be translated. If conversion is on, incoming data is translated from ASCII ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process) to the local code page. Outgoing data is converted from the local code page to ISO8859-1 (or the code page specified in the **_IOE_WIRE_CODEPAGE** environment variable of the **dfskern** process). **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_RFS_TRANSLATION=ON` **Where Variable is Used** DFSKERN **Notes** This environment variable can be overridden for specific file systems in the `devtab` file. Refer to “devtab” on page 87.
_IOE_RFS_WORKER_THREADS	Specifies the number of threads to be started in **dfskern** to service open and close requests for RFS files. **Default Value** 1 **Expected Value** A number greater than zero. **Example** `_IOE_RFS_WORKER_THREADS=3` **Where Variable is Used** DFSKERN
_IOE_SMB_ABS_SYMLINK	Specifies whether the SMB server allows absolute symbolic links. **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_SMB_ABS_SYMLINK=OFF` **Where Variable is Used** DFSKERN
_IOE_SMB_AUTH_DOMAIN_NAME	Specifies the domain name of the Domain Controller. **Default Value** None **Expected Value** Domain name, 15 characters or less. **Example** `_IOE_SMB_AUTH_DOMAIN_NAME=DOMAIN1` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_AUTH_SERVER	Specifies the IP address of the Domain Controller that will be used to authenticate PC users. **Default Value** None **Expected Value** An IP address (*n.n.n.n*, where *n* is a number between 0 and 255). **Example** `_IOE_SMB_AUTH_SERVER=9.200.150.99` **Where Variable is Used** DFSKERN
_IOE_SMB_AUTH_SERVER_ COMPUTER_NAME	Specifies the computer name of the Domain Controller that will be used to authenticate PC users. **Default Value** None **Expected Value** Computer name, 15 characters or less. **Example** `_IOE_SMB_AUTH_SERVER_COMPUTER_NAME=MYCOMPUTER` **Where Variable is Used** DFSKERN
_IOE_SMB_BACKUP_AUTH_SERVER	Specifies the IP address of the Domain Controller that will be the backup for PC user authentication. **Default Value** None **Expected Value** An IP address (*n.n.n.n*, where *n* is a number from 0 to 255). **Example** `_IOE_SMB_BACKUP_AUTH_SERVER=9.200.150.98` **Where Variable is Used** DFSKERN
_IOE_SMB_BACKUP_AUTH_SERVER_ COMPUTER_NAME	Specifies the computer name of the Backup Domain Controller that will be used to authenticate PC users. **Default Value** None **Expected Value** Computer name, 15 characters or less **Example** `_IOE_SMB_BACKUP_AUTH_SERVER_COMPUTER_NAME= MYOTHERCOMPUTER` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_BLOCKSIZE	Specifies the blocksize for the Physical File System being accessed that is returned to the SMB client. This does not affect functionality and may result in the client sending fewer requests when the client thinks that the blocksize is larger. It has been determined that a value of 32K improves performance. **Default Value** 32K **Expected Value** 0K, 8K, 16K, or 32K (the 'K' is requried). '0K' means that the SMB server returns what the Physical File System says the blocksize is. **Example** `_IOE_SMB_BLOCKSIZE=16K` **Where Variable is Used** DFSKERN
_IOE_SMB_BROWSE_INTERVAL	Specifies the maximum Browser announcement interval (in milliseconds). **Default Value** 720000 (12 minutes) **Expected Value** A number between 600000 (10 minutes) and 720000 (12 minutes). **Example** `_IOE_SMB_BROWSE_INTERVAL=600000` **Where Variable is Used** DFSKERN
_IOE_SMB_CALLBACK_POOL	Specifies the number of secondary pool threads for processing SMB callback requests. **Default Value** 2 **Expected Value** A number greater than 0. **Example** `_IOE_SMB_CALLBACK_POOL=3` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_CLEAR_PW	Specifies the SMB server policy for flowing the SMB password in the clear. **REQUIRED** Specifies that passwords in the clear are required. **ALLOWED** Specifies that passwords in the clear are allowed. Authentication is attempted using encrypted passwords if the client supports it. Otherwise, authentication is attempted assuming an unencrypted (clear text) password was used. **Note:** ALLOWED was originally provided for clients that had no support for encrypted passwords. We have subsequently determined that all supported clients include encrypted password support, so the ALLOWED setting is not useful. We suggest that you use REQUIRED for clear passwords and NOTALLOWED for encrypted passwords. **NOTALLOWED** Specifies that passwords in the clear are not allowed. Authentication is attempted using encrypted passwords only. **Default Value** REQUIRED **Expected Value** REQUIRED, ALLOWED, NOTALLOWED **Example** `_IOE_SMB_CLEAR_PW=NOTALLOWED` **Where Variable is Used** DFSKERN
_IOE_SMB_COMPUTER_NAME	Specifies the name being used by SMB redirectors (that is, clients) to contact this server. If a Windows Internet Naming Service (WINS) server is available (refer to the **_IOE_SMB_PRIMARY_WINS** environment variable on page 129), the SMB computer name is used to identify this server to the WINS server. **Default Value** The TCP/IP hostname of this system. **Expected Value** Character string, 15 characters or less **Example** `_IOE_SMB_COMPUTER_NAME=OS390DATA1` **Where Variable is Used** DFSKERN **Notes** If you are using Windows 2000 clients, this environment variable must specify (or be defaulted to) the TCP/IP hostname to allow Search Computername functionality to work.
_IOE_SMB_CROSS_MOUNTS	When ON, SMB clients cross local mount points (as in prior releases). When OFF, SMB clients go under local mount points. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_SMB_CROSS_MOUNTS=OFF` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_DESCRIPTION	Specifies the description of this server that appears on the PC. **Default Value** DFS/MVS CIFS Server **Expected Value** Character string, 50 characters or less **Example** `_IOE_SMB_DESCRIPTION=z/OS File Server` **Note:** In this example there are embedded blanks. **Where Variable is Used** DFSKERN
_IOE_SMB_DIR_PERMS	The permissions for a directory created by the SMB server at the request of an SMB client. (This can also be specified on a shared directory basis. Refer to “smbtab” on page 102.) **Default Value** 777 **Expected Value** The value specified must be a three digit numeric value where each digit is greater than or equal to 0 and less than or equal to 7. **Example** `_IOE_SMB_DIR_PERMS=755` **Where Variable is Used** DFSKERN
_IOE_SMB_DOMAIN_NAME	Specifies the name being used as the domain name for this server. If possible, specify the same domain or workgroup as your client PCs. **Default Value** None **Expected Value** Character string, 15 characters or less **Example** `_IOE_SMB_DOMAIN_NAME=OS/390DOMAIN1` **Where Variable is Used** DFSKERN
_IOE_SMB_FILE_PERMS	The permissions for a file created by the SMB server at the request of an SMB client. (This can also be specified on a shared directory basis. Refer to “smbtab” on page 102.) It also limits which write permission bits can be turned on by an SMB client. **Default Value** 777 **Expected Value** The value specified must be a three digit numeric value where each digit is greater than or equal to 0 and less than or equal to 7. **Example** `_IOE_SMB_FILE_PERMS=750` **Where Variable is Used** DFSKERN **Notes** If Read-only is set, the write permissions (222) are turned off. In this case, a file with permissions of 750 would become 550. If Read-only is cleared, the write permissions that intersect with this specification are turned on. In the example, the intersection of 750 and 222 would cause 200 to be or’d into the permissions of the file.

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_IDMAP	Specifies the location of the **smbidmap** file. The **smbidmap** file contains the mapping of SMB user IDs to z/OS user IDs. **Default Value** None **Expected Value** Character string specifying the path name of the **smbidmap** file. **Example** `_IOE_SMB_IDMAP=/opt/dfslocal/home/dfskern/smbidmap` **Where Variable is Used** DFSKERN
_IOE_SMB_IDLE_TIMEOUT	Specifies how long (in seconds) an SMB session can remain inactive before it is terminated. **Default Value** 400 **Expected Value** A number between 0 and 4294967295. **Example** `_IOE_SMB_IDLE_TIMEOUT=4000` **Where Variable is Used** DFSKERN
_IOE_SMB_MAIN_POOL	Specifies the number of primary pool threads for processing SMB requests. This is the number of SMB requests that can be handled by the SMB server at the same time without queuing them. If you have a large number of concurrently active PC clients, consider increasing this number. Note that we are referring to concurrently active users, not just logged on users. Concurrently active users are sending SMB requests to the SMB server at the same time. This number should be set to your best estimate of the maximum number of concurrent requests that might be received at the SMB server at the same time. **Default Value** 14 **Expected Value** A number greater than 0. **Example** `_IOE_SMB_MAIN_POOL=20` **Where Variable is Used** DFSKERN
_IOE_SMB_MAXXMT	Specifies the maximum server buffer size that is returned on the SMB Server Negotiate response. **Default Value** 65535 bytes **Expected Value** A number between 1024 and 65535 **Example** `_IOE_SMB_MAXXMT=8192` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_NT_SMBS	Specifies whether new SMBs in the NT protocol dialect are to be accepted from PC clients. There may be some workloads that perform better with this set to OFF. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_SMB_NT_SMBS=OFF` **Where Variable is Used** DFSKERN
_IOE_SMB_OCSF	Specifies whether OCSF is used for encrypted passwords. In order to use encryption hardware, OCSF (and therefore ON) is required. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_SMB_OCSF=OFF` **Where Variable is Used** DFSKERN
_IOE_SMB_OPLOCK_TIMEOUT	Specifies the Opportunistic Lock Timeout period (in seconds). **Default Value** 35 **Expected Value** A number between 1 and 4294967295 **Example** `_IOE_SMB_OPLOCK_TIMEOUT=60` **Where Variable is Used** DFSKERN
_IOE_SMB_OPLOCKS	Specifies whether the server allows clients to use Opportunistic Locks on files. This allows some SMB clients to cache data at the client. This can improve performance. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_SMB_OPLOCKS=OFF` **Where Variable is Used** DFSKERN
_IOE_SMB_PRIMARY_WINS	Specifies the IP address of the Windows Internet Naming Service (WINS) server that this server announces itself to and forwards proxy WINS requests to. **Default Value** None **Expected Value** An IP address (*n.n.n.n*, where *n* is a number between 0 and 255). **Example** `_IOE_SMB_PRIMARY_WINS=9.120.44.55` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_PROTOCOL_LEVEL	Specifies the highest level of SMB protocol dialect that is being negotiated with the PC client. NT is the highest dialect that the SMB server supports. LANMAN is the next lower dialect supported by the SMB server. Normally, this value should remain NT. **Default Value** NT **Expected Value** NT or LANMAN **Example** `_IOE_SMB_PROTOCOL_LEVEL=LANMAN` **Where Variable is Used** DFSKERN
_IOE_SMB_RAW	Specifies whether raw mode is supported in the SMB Server Negotiate response. Raw mode is used for large data transfers. **Default Value** ON **Expected Value** ON or OFF **Example** `_IOE_SMB_RAW=OFF` **Where Variable is Used** DFSKERN
_IOE_SMB_SCOPE	Specifies the Scope ID for the Windows Internet Naming Service (WINS) server. The Scope ID defines a group of computers that recognize a registered NetBIOS name. (This should normally be omitted.) **Default Value** None **Expected Value** Character string, 224 characters or less **Example** `_IOE_SMB_SCOPE=MYDEPARTMENTSCOPE` **Where Variable is Used** DFSKERN
_IOE_SMB_SECONDARY_WINS	Specifies the IP address of the Windows Internet Naming Service (WINS) server that this server announces itself to and forwards proxy WINS requests to if the Primary WINS server does not respond. **Default Value** None **Expected Value** An IP address (*n.n.n.n*, where *n* is a number between 0 and 255). **Example** `_IOE_SMB_SECONDARY_WINS=9.120.66.77` **Where Variable is Used** DFSKERN
_IOE_SMB_TOKEN_FILE_MAX	Specifies the maximum number of files that the SMB token cache should keep tokens for. **Default Value** 4096 **Expected Value** A number greater than or equal to 4096. **Example** `_IOE_SMB_TOKEN_FILE_MAX=6144` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_WINS_PROXY	Specifies whether this server can act as a Windows Internet Naming Service (WINS) server proxy. WINS requests sent to this server are forwarded to a WINS server on another computer (refer to the **_IOE_SMB_PRIMARY_WINS** environment variable on page 129). **Default Value** OFF **Expected Value** ON or OFF **Example** `_IOE_SMB_WINS_PROXY=ON` **Where Variable is Used** DFSKERN
_IOE_TKCGLUE_CACHE_SIZE	Specifies the number of file tokens for local HFS access that can be cached. **Default Value** 4096 **Expected Value** A number greater than or equal to 4096. **Example** `_IOE_TKCGLUE_CACHE_SIZE=8192` **Where Variable is Used** DFSKERN
_IOE_TKM_MAX_TOKENS	Specifies the maximum number of tokens that can be held in the SMB server memory. **Default Value** 1024 **Expected Value** A positive number. The minimum value is 10240. **Example** `_IOE_TKM_MAX_TOKENS=20480` **Where Variable is Used** DFSKERN
_IOE_TKMGLUE_SERVER_THREADS	The number of threads to be started in DFSKERN to service token requests from the glue layer. The glue layer makes token requests during local HFS access. **Default Value** 5 **Expected Value** The value specified must be a numeric value greater than or equal to 5. **Example** `_IOE_TKMGLUE_SERVER_THREADS=8` **Where Variable is Used** DFSKERN

Table 4. Environment variables in SMB (continued)

Name	Description
_IOE_VM_CACHE_SIZE	Specifies the size, in bytes, of the virtual memory used by DFSKERN for all file systems that are exported. **Default Value** 1M **Expected Value** A positive number. The value may be a whole positive integer followed by a 'K' (denoting kilobytes), or a whole positive integer followed by an 'M' (denoting megabytes). **Example** `_IOE_VM_CACHE_SIZE=2M` **Where Variable is Used** DFSKERN **Notes** The virtual memory is used for data in all file systems that are exported.
_IOE_VM_MAX_FILES	Specifies the maximum number of files that can be contained in the virtual memory used by DFSKERN for all file systems that are exported. **Default Value** 4096 **Expected Value** A positive number. The minimum value is 4096. **Example** `_IOE_VM_MAX_FILES=6144` **Where Variable is Used** DFSKERN
_IOE_VNODE_CACHE_SIZE	The size of the HFS vnode cache, that is, the number of vnodes. **Default Value** 4096 **Expected Value** The value specified must be a numeric value greater than or equal to 2048. **Example** `_IOE_VNODE_CACHE_SIZE=6144` **Where Variable is Used** DFSKERN
_IOE_WIRE_CODEPAGE	Specifies the codepage to be used for text data on the wire for clients. **Default Value** ISO8859-1 **Expected Value** ISO8859-1 or *codepage* **Example** `_IOE_WIRE_CODEPAGE=ISO8859-1` **Where Variable is Used** DFSKERN

Appendix B. Additional information using the SMB server

Important Note to Users

This appendix contains information that is outside the scope of the z/OS Distributed File Service SMB server. However, it is included here since it may be helpful to the SMB administrator or the PC user.

Client does not communicate

If the client does not communicate with the SMB server, it may be because a service pack has been applied or is present on the client that requires the challenge/response type of logon. For the service pack noted below (or a later service pack), the registry needs to be updated to allow unencrypted passwords.

Attention: Using Registry Editor incorrectly can cause serious, system-wide problems that may require you to reinstall Windows to correct them. Microsoft cannot guarantee that any problems resulting from the use of Registry Editor can be solved. Use this tool at your own risk.

Note: If you are using the encrypted password support and your users have entered their SMB password in RACF using the OMVS **smbpw** command, this registry change is not necessary. Users that have made this change to the registry do not need to remove it to use encrypted passwords.

Windows NT 4.0

This section shows you how to update the registry for Windows NT 4.0 with Service Pack 3 (or later).

Updating the registry:

The following steps show you how to update the registry:

1. From an MS-DOS command prompt window, type **regedt32**.
2. Double-click on **SYSTEM** (below HKEY_LOCAL_MACHINE).
3. Double-click on **CurrentControlSet**.
4. Double-click on **Services**.
5. Double-click on **Rdr** (you may need to scroll down to find it).
6. Click on **Parameters**.
7. Click the **Edit** pull-down menu on the Registry Editor and click **Add Value**.
8. Type a Value Name of EnablePlainTextPassword.
9. Choose a Data Type of REG_DWORD.
10. Click the **OK** button.
11. Type a Data value of 1.
12. Ensure that the Radix is Hex.
13. Click the **OK** button.

Your registry should now be updated. If you make this change on one machine, you can export this change to a **reg** file. If you can then make that file available to the client machines that need it, users can apply the change by double clicking the **reg** file.

Creating a reg file that contains the registry update:

After making the registry change, use the following steps to create the reg file:

1. From an MS-DOS command prompt window, type **regedit**.
2. Click the plus (+) next to HKEY_LOCAL_MACHINE to expand the sub-entries.
3. Click the plus (+) next to SYSTEM to expand the sub-entries.
4. Click the plus (+) next to CurrentControlSet to expand the sub-entries.

5. Click the plus (+) next to Services to expand the sub-entries.
6. Click the plus (+) next to Rdr to expand the sub-entries.
7. Click on **Parameters** to show the values in the right pane.
8. Click the **Registry** pull-down menu on the Registry Editor and click **Export Registry File...**.
9. Choose the directory that you want to save the **reg** file into.
10. Type a file name for the **reg** file (the .reg suffix is added for you).
11. Ensure that the Export Range indicates Selected Branch.
12. Click the **Save** button.

The **reg** file is created. If you right-click on the **reg** file and choose Edit, the contents of the **reg** file looks like the following:

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters]
"EnablePlainTextPassword"=dword:00000001
```

Windows 2000

This section shows you how to update the registry for Windows 2000 Professional.

1. Click **Start**, point to **Settings**, click on **Control Panel**.
2. Double-click on **Administration Tools**.
3. Double-click on **Local Security Policy**.
4. On the left pane, click on the plus (+) to expand Local Policies (under Security Settings).
5. On the left pane, click on **Security Options**.
6. Double-click on Send unencrypted passwords to connect to third-party SMB servers.
7. Click the Enabled radio button, and then click **OK**.

Windows 98

This section shows you how to update the registry for Windows 98.

Updating the registry:

The following steps shows you how to update the registry:

1. From an MS-DOS command prompt window, type **regedit**.
2. Click the plus (+) next to HKEY_LOCAL_MACHINE to expand the sub-entries.
3. Click the plus (+) next to SYSTEM to expand the sub-entries.
4. Click the plus (+) next to CurrentControlSet to expand the sub-entries.
5. Click the plus (+) next to Services to expand the sub-entries.
6. Click the plus (+) next to VxD to expand the sub-entries.
7. Click on **VNETSUP** to show the values in the right pane.
8. Click the **Edit** pull-down menu on the Registry Editor and move your mouse to New.
9. Click **DWORD Value**.
10. Type a New Value of EnablePlainTextPassword and press **Enter**.
11. Right-click the **EnablePlainTextPassword** entry and click on **Modify**.
12. Type a Value Data of 1.
13. Ensure that the Base is Hexadecimal.
14. Click the **OK** button.

Your registry should now be updated. If you make this change on one machine, you can export this change to a **reg** file. If you can then make that file available to the client machines that need it, users can apply the change by double clicking the reg file.

Creating a reg file that contains the registry update:

After making the registry change, use the following steps to create the reg file:

1. From an MS-DOS command prompt window, type **regedit**.
2. Click the plus (+) next to HKEY_LOCAL_MACHINE to expand the sub-entries.
3. Click the plus (+) next to SYSTEM to expand the sub-entries.
4. Click the plus (+) next to CurrentControlSet to expand the sub-entries.
5. Click the plus (+) next to Services to expand the sub-entries.
6. Click the plus (+) next to VxD to expand the sub-entries.
7. Click on **VNETSUP** to show the values in the right pane.
8. Click the **Registry** pull-down menu on the Registry Editor and click **Export Registry File...**.
9. Choose the directory that you want to save the **reg** file into.
10. Type a File name for the **reg** file (the .reg suffix is added for you).
11. Ensure that the Export Range indicates Selected Branch.
12. Click the **Save** button.

The **reg** file is created. If you right-click on the **reg** file and choose Edit, the contents of the **reg** file looks like the following:

```
REGEDIT4

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001
```

Windows NT client does not allow net view command

If a Windows NT client does not allow the **net view** command, it may be because you have not connected to the SMB server yet. The registry entry modification in 'Client Does Not Communicate' above, does not allow the **net view** command on Windows NT if you have not yet connected to the SMB server. The solution is to issue **net use** to the SMB server before you issue a **net view** to the SMB server. Another possible solution is to issue a **net use** to shared directory **IPC$** with your password.

Editor or word processor changes the owner/permissions of the HFS file

Many Editors and Word Processors, when saving a file, do not simply update the file. Rather, in order to avoid losing data if the system should fail during the save operation, they first create a new file with a temporary name and save the data there, then they erase the original file, and then they rename the file with the temporary name to the original file name. Since a new file was created, it is owned by the user that created it. This may be different than the original file's owner. Also, the file permissions are changed to the default permissions which may be different from the original permissions.

Editor or word processor cannot save a file to HFS

Many Editors and Word Processors, when saving a file, do not simply update the file. Rather, in order to avoid losing data if the system should fail during the save operation, they first create a new file with a temporary name and save the data there, then they erase the original file, and then they rename the file with the temporary name to the original file name. Since a new file is being created, the user must have write and execute permission to the directory that the file is contained in and read and write permission to the file. If the user does not have the required permissions on both the directory and the file, the save is denied.

Different end of line characters in text files

The PC environment and the z/OS UNIX environment normally use different end of line characters in text files. The end of line characters are placed into the file as the text lines are written by the application (for example, an editor) based on the environment the application is running on (Windows versus z/OS UNIX). If you are sharing text files between the PC environment and the z/OS UNIX environment, one of the environments sees end of line characters at the end of each line of text that it does not normally expect. Some applications may not process a text file if the wrong end of line characters are in the text file. For example, the C/C++ compiler does not compile source code with PC end of line characters. You should try to use applications on the PC that support z/OS UNIX end of line characters (see note below). For example, Microsoft WordPad correctly displays text files that have z/OS UNIX end of line characters. It also maintains z/OS UNIX end of line characters when such a file is saved. There are also PC applications that optionally create a text file with z/OS UNIX end of line characters. An example of an application that creates a text file with z/OS UNIX end of line characters is MicroEdge Visual SlickEdit. Many PC applications tolerate z/OS UNIX end of line characters.

If you want to determine which end of line characters a text file contains, you can use the following OMVS command to display the hexadecimal values of the characters in a file:

```
od -cx textfile.txt
```

where *textfile.txt* is the name of the text file you want to display.

When a text file has PC end of line characters, you can create another file with the same data but with z/OS UNIX end of line characters with the following OMVS command:

```
tr -d '\r' <textfile.txt >newfile.txt
```

where *textfile.txt* is the text file with PC end of line characters and **newfile.txt** is a new text file with z/OS UNIX end of line characters.

Note: In general, z/OS UNIX text files contain a newline character at the end of each line. In ASCII, newline is X'0A'. In EBCDIC, newline is X'15'. (For example, ASCII code page ISO8859-1 and EBCDIC code page IBM-1047 translate back and forth between these characters.) Windows programs normally use a carriage return followed by a line feed character at the end of each line of a text file. In ASCII, carriage return/line feed is X'0D'/X'0A'. In EBCDIC, carriage return/line feed is X'0D'/X'15'. The **tr** command above simply deletes all of the carriage return characters. (Line feed and newline characters have the same hexadecimal value.) The SMB server can translate end of line characters from ASCII to EBCDIC and back but it does not change the type of delimiter (PC vs. z/OS UNIX) nor the number of characters in the file.

PC clients disconnect during high DASD I/O activity

When there is high DASD I/O activity on an HFS file system, there may be an undue delay for HFS requests. This may cause the PC clients to disconnect from the SMB server because they think the SMB server is down. A possible bypass to this situation is to mount the HFS file system with a sync interval that is less than 60 seconds. For example,

```
MOUNT FILESYSTEM('OMVS.ABC.DEF') MOUNTPOINT('/abc/def') TYPE(HFS) MODE(RDWR) PARM('SYNC(30)')
```

sets the sync interval to 30 seconds for the HFS file system specified. Refer to the TSO/E MOUNT command in the *z/OS: UNIX System Services Command Reference*, SA22-7802, for information on the sync interval parameter.

Note: This does not apply to ZFS file systems.

SMB server does not show up in Network Neighborhood

Note, that the Network Neighborhood function is not required in order to use the SMB server. The SMB server can be contacted if you know its computer name by using the Find Computer function. It can also be contacted from Windows NT and Windows 2000 clients by using the DNS domain name of the SMB server in the **net use** command or the Map Network Drive function. Other Windows systems may require an LMHOSTS file.

If you want to use the Network Neighborhood function, you must meet the following networking requirements:

- There must be a Windows NT or Windows 2000 Domain Controller on the same subnet as the SMB server.
- The Windows NT or Windows 2000 Domain Controller must be acting as a Domain Master Browser.

  You can determine the roles that a Windows machine is running by using the **browstat view** command. (**browstat** is supplied with the Windows NT Server Resource Kit.)

Appendix C. Using both SMB and DCE DFS

This appendix contains information that must be considered if you plan to use the Distributed File Service DCE DFS protocols (enabled by using the **_IOE_PROTOCOL_RPC=ON** environment variable) along with the SMB protocols (enabled by using the **_IOE_PROTOCOL_SMB=ON** environment variable).

Note: Dynamic export is not supported when **_IOE_PROTOCOL_RPC=ON**.

Fileset IDs in dfstab

The **dfstab** and **devtab** files specify HFS and RFS file systems that are being exported. Those files are used for the SMB protocol and the DCE DFS protocol.

When you use the DFS protocol, fileset IDs must be assigned by the **flserver**. (This is required whether SMB protocols are being used or not.) For an HFS fileset, a fileset ID is assigned by the **flserver** by using the **fts crfldbentry** command. The administrator enters the assigned fileset ID in the fileset's **dfstab** entry.

If you have been using DCE DFS protocols and you are now adding SMB protocols, the fileset IDs for any exported HFS or RFS filesets have already been assigned by the **flserver** and are already specified in the **dfstab**. The Distributed File Service SMB Server exports these same filesets for the SMB protocol. You can then create your **smbtab** entries to specify which shared directories should be available to PC clients. If you need to export any additional HFS or RFS filesets, the fileset IDs must be assigned by the **flserver** (even if they are only going to be accessed by SMB protocols).

Or, if you have been using SMB protocols and you are now adding DCE DFS protocols, the HFS and RFS fileset IDs specified in the **dfstab** must be changed to numbers that have been assigned by the **flserver**. This means that you must enable and start the Distributed File Service Server for DCE DFS protocols, assign the HFS and RFS fileset IDs by using the **fts crfldbentry** command, update your **dfstab** entries for the HFS and RFS fileset IDs, and then export the filesets (use the **modify dfs,start export** system command or the **dfsexport** command). If you need to add any HFS or RFS filesets, their fileset IDs must also be assigned by the **flserver**.

Crossing local mount points

When the SMB protocol is used (without the DCE DFS protocol), PC users cross local mount points. That is, when a PC user changes the current directory (for example, dirA) to a subdirectory (for example, dirB) by using the **cd** command and that subdirectory (dirB) is a mount point, the PC user "crosses" into the root directory of the file system that is mounted on dirB. A **dir** command against dirB shows the files and directories contained in the root directory of the file system mounted on dirB (as opposed to showing the files and directories that might actually be contained in dirB). This assumes that the file system has been exported and that the user is authorized. In fact, any files or subdirectories that are actually contained in dirB are inaccessible. This is normal behavior for a z/OS UNIX file system.

When the DCE DFS protocol is used (regardless of whether the SMB protocol is used), clients do not cross mount points. Rather, clients see the files and directories that are actually contained in the mount point directory. This is how DFS clients expect the server to perform, and is how the server has performed in previous releases.

It is possible for SMB clients to cross local mount points and for DCE DFS clients to not cross local mount points. In fact, this is the default behavior. Whether SMB clients cross local mount points is controlled by the **_IOE_SMB_CROSS_MOUNTS** environment variable of **dfskern**.

SMB encrypted passwords and DCE single sign-on

If you are using SMB encrypted passwords and DCE single sign-on, your SMB password and your DCE password must be the same since both of these come from the RACF DCE segment. However, if the DCE Security Server is running on the same system as the DCE user, then the DCE single sign-on processing does not require the DCE password to be stored in RACF. So, in this case, the SMB password can be stored in the RACF DCE segment and the RACF DCE segment is not used for DCE single sign-on.

Appendix D. Customizable files

Notes:

1. The symbolic link **/opt/dfsglobal** refers to the directory **/usr/lpp/dfs/global**.
2. The symbolic link **/opt/dfslocal** refers to the directory **/etc/dfs**.

The following table summarizes the Distributed File Service example files that are supplied by IBM and where they are placed so that you can customize them to meet your local DCE DFS or SMB support requirements. The table indicates whether the file applies to (DCE) DFS, SMB, or both. Files that apply to support not being used are created but do not need to be modified.

Most of the IBM supplied files are copied from the **/opt/dfsglobal/examples** directory to the target subdirectory in **/opt/dfslocal (/etc/dfs)** by the **/opt/dfsglobal/scripts/dfs_cpfiles** program run as part of the Distributed File Service post installation processing described in Chapter 3, “Post installation processing” on page 11.

The **dfs_cpfiles** program does not replace a file in the target directory if it already exists. When installing a new Distributed File Service release, you can compare the contents of the IBM supplied files with your current customized files to determine if there are any new optional parameters that you may want to consider specifying.

The customizable files used for a previous release can be used for a new release if the same IP address, RACF user definitions and exported file definitions apply to the target image for the new release. If any of these definitions are not the same for the target image, the customizable files should be reviewed and modified as required. Refer to the *z/OS: Distributed File Service DFS Customization* document for more information.

Table 5. DFS customizable files

IBM Supplied File	Customizable File	Applies To
/opt/dfs/global/examples/bakserver.envar	/opt/dfslocal/home/bakserver/envar	DFS
/opt/dfs/global/examples/boserver.envar	/opt/dfslocal/home/boserver/envar	DFS
/opt/dfs/global/examples/butc01.envar	/opt/dfslocal/home/butc01/envar	DFS
/opt/dfs/global/examples/butc02.envar	/opt/dfslocal/home/butc02/envar	DFS
/opt/dfs/global/examples/butc03.envar	/opt/dfslocal/home/butc03/envar	DFS
/opt/dfs/global/examples/butc04.envar	/opt/dfslocal/home/butc04/envar	DFS
/opt/dfs/global/examples/butc05.envar	/opt/dfslocal/home/butc05/envar	DFS
/opt/dfs/global/examples/butc06.envar	/opt/dfslocal/home/butc06/envar	DFS
/opt/dfs/global/examples/butc07.envar	/opt/dfslocal/home/butc07/envar	DFS
/opt/dfs/global/examples/butc08.envar	/opt/dfslocal/home/butc08/envar	DFS
/opt/dfs/global/examples/cmattr (applies to both cmattr and hfsattr)	CMATTR (not created by dfs_cpfiles; cmattr file defined by envar _IOE_CM_ATTRIBUTES_FILE)	DFS
/opt/dfs/global/examples/daemonct.envar	/opt/dfslocal/home/daemonct/envar	both
/opt/dfs/global/examples/devtab	/opt/dfslocal/var/dfs/devtab	both
/opt/dfs/global/examples/dfs.ioepdcf	/opt/dfslocal/etc/ioepdcf	both
/opt/dfs/global/examples/dfscm.CacheInfo	/opt/dfslocal/etc/CacheInfo	DFS
/opt/dfs/global/examples/dfscm.envar	/opt/dfslocal/home/dfscm/envar	DFS
/opt/dfs/global/examples/dfscntl.envar	/opt/dfslocal/home/dfscntl/envar	both
/opt/dfs/global/examples/dfsexport.envar	/opt/dfslocal/home/dfsexport/envar	both

Table 5. DFS customizable files (continued)

IBM Supplied File	Customizable File	Applies To
(None)	DFSIDMAP (not created by dfs_cpfiles; dfsidmap file defined by envar _IOE_MVS_IDMAP)	DFS
/opt/dfs/global/examples/dfskern.envar	/opt/dfslocal/home/dfskern/envar	both
/opt/dfs/global/examples/dfstab	/opt/dfslocal/var/dfs/dfstab	both
/opt/dfs/global/examples/flserver.envar	/opt/dfslocal/home/flserver/envar	DFS
/opt/dfs/global/examples/ftserver.envar	/opt/dfslocal/home/ftserver/envar	DFS
/opt/dfs/global/examples/growaggr.envar	/opt/dfslocal/home/growaggr/envar	DFS
/opt/dfs/global/examples/cmattr (applies to both cmattr and hfsattr)	HFSATTR (not created by dfs_cpfiles; hfsattr file defined by envar _IOE_HFS_ATTRIBUTES_FILE)	both
/opt/dfs/global/examples/newaggr.envar	/opt/dfslocal/home/newaggr/envar	DFS
/opt/dfs/global/examples/repserver.envar	/opt/dfslocal/home/repserver/envar	DFS
/opt/dfs/global/examples/rfstab	/opt/dfslocal/var/dfs/rfstab	both
/opt/dfs/global/examples/salvage.envar	/opt/dfslocal/home/salvage/envar	DFS
(None)	SMBIDMAP (not created by dfs_cpfiles; smbidmap file defined by envar _IOE_SMB_IDMAP)	SMB
/opt/dfs/global/examples/smbtab	/opt/dfslocal/var/dfs/smbtab	SMB
/opt/dfs/global/examples/upclient.envar	/opt/dfslocal/home/upclient/envar	DFS
/opt/dfs/global/examples/upserver.envar	/opt/dfslocal/home/upserver/envar	DFS

Appendix E. Using data sets

This appendix explains what you need to know when you use data sets from a PC client. The SMB server can export record files (sequential, PDS, PDSE, and Virtual Storage Access Method (VSAM)). This makes record data available to PC client applications. The data is presented as byte stream data.

Note: Generation Data Groups (GDG) are not supported.

Mapping between the PC client's view and record data

In z/OS, a file is called a data set. These two terms are used interchangeably.

The files for a computer system are organized into a file system. The PC client environments use a byte file system which is a hierarchy of directories that can contain other directories or files. Record data, however, uses a nonhierarchical file system in which groups of data sets are referred to by specifying a high-level qualifier.

The high-level qualifier can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the high-level qualifier for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the high-level qualifier of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

Note: Only Integrated Catalog Facility (ICF) cataloged data sets are supported by the SMB server. Tape data sets are not supported.

The SMB server exports a set of files with a specified prefix as a single file system. All files with the specified prefix are shown as one level of hierarchy. If the data sets specified include partitioned data sets, a second level of hierarchy is shown.

Since RACF requires a data set profile that covers the prefix specified on the **devtab**, you must specify two levels of prefix to have it covered by a profile. For example, prefix USERA.PCDSNS is covered by data set profile 'USERA.**'. Otherwise, you must give the user the OPERATIONS attribute.

Mapping data sets onto an RFS file system

If the following data sets exist:

```
USERA.PCDSNS.B()      with members M1 and M2
USERA.PCDSNS.B.C
USERA.PCDSNS.B.C.D
USERA.PCDSNS.X.Y()    with members M1 and M2
USERA.PCDSNS.X.Y.Z
```

The following is an example of SMB server commands and operations to export an RFS file system that contains data sets that begin with the prefix USERA.PCDSNS.

1. Add an **rfs** file system to the **devtab** file.

   ```
   * RFS devices
   define_ufs 3 rfs
   USERA.PCDSNS
   ```

2. Add the **rfs** file system to the **dfstab** file using the same minor device number (in this example, **3**) in the device parameter (so in this example, it would be /dev/ufs**3**). Use a unique file system name (for example, **rfs3**), a file system type of **ufs** and a unique file system ID (for example, **102**). Finally, use a unique fileset ID (for example, **0,,1718**). An example **dfstab** entry might look like this:

   ```
   /dev/ufs3  rfs3  ufs  102  0,,1718
   ```

Note: If you are using only SMB protocols (and not DCE DFS protocols[8]), you can use the same number for all the numeric values in a **dfstab** entry as long as the number used is unique. For example, the **dfstab** entry might look like this:

```
/dev/ufs3  rfs3  ufs    3  0,,3
```

3. Add an entry in **smbtab** for the directory you want to share. Use the same minor device number (in this example, **3**) in the device name parameter (so in this example, it would be /dev/ufs**3**). Choose a unique share name (for example, **mvsusera**), a file system type of **ufs**, a description, share permission (for example, **r/w**), maximum users (for example, **100**) and the directory name (in this example */*). The directory name is relative to the root of the file system that the device name refers to.

   ```
   dev/ufs3  mvsusera  ufs  "mvsusera data sets"  r/w  100 /
   ```

   Note: RFS has many restrictions when writing to or creating files. If it is appropriate, you may want to specify read-only access to avoid these restrictions by using **r/o** in the share permission field of the **smbtab** entry.

4. Issue the following **dfsshare** command to cause the new share to be made available to PC users.

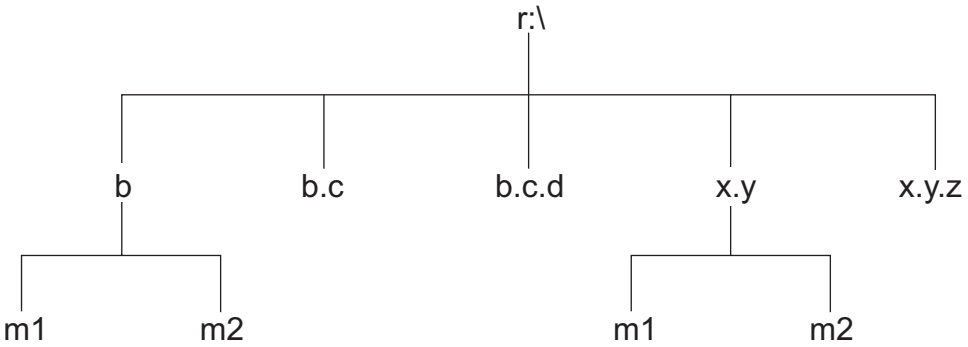
   ```
   # dfsshare -share mvsusera
   ```

   At this point, a PC user can connect to this share.

5. The PC user can now connect to the shared directory using the following **net use** command in Windows NT.

   ```
   C:\>net use r: \\computer1.abc.com\mvsusera password
   ```

   The following figure represents a view of the data set hierarchy.



This allows you to define one level of directory under the shared directory (at **r:**). Thus, you can issue **mkdir** to create a directory (stored as a PDS or PDSE) and then create files (stored as PDS or PDSE members) within that directory.

Alternatively, if the prefix specified in the **devtab** is an actual data set, the SMB server attempts to export the data set. If the data set is a PDS (or PDSE), it is handled as a directory and the members are exported. However, in this case, sharing with local z/OS record applications is not supported. Refer to “Sharing data” on page 145 for more information. If it is any other type of data set, the export for that data set fails. (The name that is specified for export must be able to be handled as a directory.)

Reading, writing, and creating data sets

You can use the SMB server to read data stored in a data set from your PC client system. You can also write data to an data set (stored on DASD) from your PC client system. The data sets appear as files on the drive letter on your PC client system.

8. If you are using both SMB protocols and DCE DFS protocols, the file system ID must be assigned by the **fts crfldbentry** command. Refer to Appendix C, “Using both SMB and DCE DFS” on page 139 for more information.

You can create data sets from a PC client using the SMB server. The default record data set creation attributes specified by the system administrator are used to create data sets, unless the user overrides them. These attributes determine how the data set is structured and where it is stored. You can override the data set creation attributes at file creation time.

Sharing data

The sharing semantics that local z/OS applications accessing record data expect and the semantics that PC clients expect are very different. PC clients can potentially allow multiple users to be able to open a file for write. The file integrity is protected through the use of byte range locks. Local z/OS applications, in general, do not expect other users to be writing to the file (data set) while their application is writing.

Also, SMB clients request opportunistic locks when opening a file. Possession of an exclusive opportunistic lock means that the client can act on the file without communicating to the SMB server. The SMB server keeps track of opportunistic locks it has given out. When a PC client requests an opportunistic lock that conflicts with another opportunistic lock that the SMB server has given out, the SMB server issues a callback (requests that it be given back to the server) for that opportunistic lock. PC clients that receive callbacks give back the opportunistic lock as soon as they can. PC clients expect that all requesters are properly requesting and returning opportunistic locks. Local z/OS applications using record data sets do not request opportunistic locks.

As a result, the SMB server must provide one set of semantics and opportunistic lock management to the PC clients and another set of semantics with no opportunistic lock management to record data applications. In order to provide this, the SMB server must construct a ″wall″ between the PC clients and the record data applications that are attempting to access the same record file. The SMB server only allows either PC clients or record applications to write the data, but not both. In some cases, it does allow them both to read the data.

Note: None of the following techniques for freeing the data set are supported when an actual data set (as opposed to a data set prefix) is specified in the **devtab**. This means that the RFS file system must be unexported to allow a batch job that needs access to the data set to continue.

The general technique that the SMB server uses to accomplish this is to do a dynamic allocate on any record data set that a PC client is attempting to access.

1. If the PC client is attempting to read record data, and the data set is not a PDS, the SMB server attempts a DISP=SHR allocation for the data set.
   a. If the data set is free, then the dynamic allocate issued by the SMB server is successful and the SMB server provides the record data set to its PC clients for reading.

      A batch job that subsequently attempts to access that same record data set with an allocation of DISP=SHR successfully accesses the data set.

      A batch job that subsequently attempts to access that same record data set with an allocation of DISP=OLD waits for the unallocate to occur. The SMB server is notified that a batch job is waiting for the unallocate. (The SMB server uses the Event Notification Facility (ENF) and an event supported by Global Resource Serialization (GRS) for resource contention. Refer to the *z/OS: MVS Programming: Authorized Assembler Services Guide*, SA22-7608, for information on the Event Notification Facility.) In this case, the SMB server frees up the record data set as soon as it can.
   b. If the data set is not free, the SMB server denies access to the data set. It appears to the PC clients that the data set is unavailable. (PC clients cannot wait an indefinite amount of time for the batch job to complete.)
2. If the PC client is attempting to write record data, or the data set is a PDS, the SMB server attempts a DISP=OLD allocation for the data set.
   a. If the data set is free, then the dynamic allocate issued by the SMB server is successful and the SMB server provides the record data set to its PC clients for writing.

A batch job that subsequently attempts to access that same record data set with an allocation of DISP=SHR or DISP=OLD waits for the unallocate to occur. The SMB server is notified that a batch job is waiting for the unallocate. In this case, the SMB server frees up the record data set as soon as it can.

b. If the data set is not free, the SMB server denies access to the data set. It appears to the PC clients that the data set is unavailable. (PC clients cannot wait an indefinite amount of time for the batch job to complete.)

ISPF users attempting to access a record data set that has been accessed by PC clients may be denied access because the SMB server may still have the data set allocated. ISPF (and TSO users) use dynamic allocation and this does not cause resource contention (and therefore no resource contention event occurs). The allocation request is simply denied. The SMB server does not recognize that the data set is being requested for use. However, the SMB server deallocates data sets that have had no activity for a period of time. This makes the data sets available to other applications such as ISPF. The time period is controlled by the **_IOE_RFS_ALLOC_TIMEOUT** environment variable. The default time period is 5 minutes. Refer to page 122 for more information on this environment variable.

In order to free up a data set right away, the user can submit a simple batch job that allocates the data set. This causes resource contention and therefore causes the SMB server to free up the data set as soon as it can. The batch job can be as simple as:

```
//JOBNAME JOB
//STEP1   EXEC   PGM=IEFBR14
//DD1     DD     DSN=USERA.PCDSNS.B.C,DISP=OLD
```

Note: If you are using JES3 to protect the data sets that are being shared with PC clients, resource contention does not occur (and therefore no resource contention event occurs). The SMB server does not recognize that the data set is being requested for use. In this case you could request that JES3 bypass these data sets (by using the DYNALDSN statement). Refer to the *z/OS: JES3 Initialization and Tuning Reference*, SA22-7550, for information on the DYNALDSN statement. Alternatively, you can use one of the other techniques that follow to free up the data set.

Another technique can be used by PC client users with the proper authority to force the SMB server to free up a data set. A user with RACF ALTER authority to the data set can issue the **mkdir** ″***name*,release**″ command from a PC client machine. For example, **mkdir** ″**b.c,release**″ would cause the SMB server to free up the **USERA.PCDSNS.B.C** data set as soon as it can. (If *name* is a member of a PDS or PDSE data set, the entire PDS or PDSE data set is released.)

When you issue the **mkdir** release command, a **mkdir** error message is expected. If the release was successful, the message indicates that the system attempted to access the file but the system could not because the object already exists. For example,

```
E:\>mkdir "r:\b.c,release"
A subdirectory or file r:\b.c,release already exists.
```

This is because the PC client sends this request as a make directory request. Since we are not really making a directory at the server (rather, we are doing release processing against a file or set of files), we must send an error code indicating that the object already exists back to the PC client when the release processing is completed successfully to keep the PC client from continuing with its create directory processing.

If the user does not have RACF ALTER authority to the file(s), the SMB server sends back EACCES (the user does not have the required permission) to the PC client.

An operator force command (**modify dfs**) is provided to allow the operator to force the SMB server to free up a data set if an important batch job has been waiting too long.

Forcing a data set to be freed by SMB

The following operator command may be issued to force the SMB server to free a data set for access by a batch job:

```
modify dfs,send dfskern,release,data-set-name
```

where *data-set-name* is the name of the data set that the SMB server should make available.

Refreshing RFS file names

The SMB server uses the Event Notification Facility to determine when a batch job is attempting to allocate a data set that is currently being accessed by PC clients. When the SMB server detects this, it issues callbacks for all opportunistic locks and unallocates the data set (thus making it available to the batch job).

Since there is no event to indicate when data sets are created, deleted, or renamed by a batch job, the SMB server refreshes its cache of exported data set names and attributes at a regular interval. This interval is controlled by the **_IOE_RFS_STATUS_REFRESH_TIME** environment variable in the **dfskern** process. The time is specified in seconds and the default is 600 (ten minutes). In this case, a PC client listing file names when positioned at an **rfs** root directory may not see a new file created by a z/OS job for up to ten minutes after it is created.

Note: This only affects commands that list file names and attributes. A newly created file can be directly referenced immediately after it is created.

Special considerations for record data

In addition to mapping between the PC client's view and z/OS file systems, you should be aware of the other ways in which the record data might differ from hierarchical byte data. These differences include:

- Selecting a data storage format
- File size determination and time stamps
- Client caching
- Record file names.

Selecting a data storage format for record data

PC clients can access data sets. These data sets are record oriented and can be sequential, direct, VSAM, partitioned, and so forth, and also can contain variable or fixed-length records. PC client environments, however, are byte-oriented and may write or read at certain byte offsets in the file.

The SMB server can map PC client requests to most types of data set organizations. However, how the time stamps and file size value are handled depends on the type of data set used, and the file size processing can affect performance.

Direct reads with the data set attributes **recfm(fbs)** or **recfm(f)** can be fast because in some cases, the SMB server can determine the physical block addresses from the record offsets. The z/OS sequential file organization with **recfm(f)** or **recfm(fbs)** on DASD allows for efficient updating or reading at any offset in the file. Other supported z/OS access methods (for example, VSAM) may not perform as efficiently.

File size determination and time stamps

The SMB server determines how to handle the file size value and time stamps depending on the type of data set used and the attributes used to access the data set. Refer to "Handling of the file size value" on page 153 for more information of file size determination and to "Handling of the time stamps" on page 155 for more information on handling time stamps.

PC client caching

Record file data is cached at PC clients in the normal manner. PC clients request and return opportunistic locks as usual. However, if the SMB server fails and restarts, and you are currently positioned within an RFS file system, then you need to change directory (**cd**) back to the root of that RFS file system before you can access files within that RFS file system.

Record file names

As explained in “Mapping between the PC client’s view and record data” on page 143, record file names consist of segments separated by a dot with a maximum length of 44 characters. Each segment can be from one to eight alphanumeric characters. (Refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410, for information on data set naming.) Each record file appears as a byte file to PC clients. A PDS appears as a directory with the members appearing as files within the directory. PDS member names are limited to eight characters.

Data set names are always in upper case characters. This means that file names would be displayed to PC client users in upper case and would need to be entered in upper case. There is, however, a processing attribute in the attributes file (refer to “Attributes file (rfstab)” on page 80 called **maplower**. **maplower** is the default. When this attribute is specified or defaulted in the attributes file, users can enter lower case letters for file names and they are mapped to upper case by the SMB server. Also, when the (upper case) file names are displayed to the user, they are mapped to lower case by the SMB server.

You are cautioned, however, that when the **maplower** processing attribute is in effect, you should only use lower case letters for file names. If you create a file with the name `AbCd`, it becomes a data set named `ABCD`. When the file name is displayed to the (PC client) user, it appears as `abcd`. The file name would be displayed as a different name than the user entered when it was created.

Also, many byte file systems typically allow file names to begin with a dot. This is an invalid name for a data set. However, a processing attribute in the attributes file called **mapleaddot** causes a leading dot in a file name to be mapped to a dollar sign in the data set name and back to a dot for the PC client. **mapleaddot** is the default.

Creating z/OS files

This section describes how to create the various types of data sets (files) supported by the SMB server.

The examples shown are for an MS-DOS session. Any examples for other platforms have been indicated.

Overriding data set creating attributes

When you create a z/OS file, default file creation attributes are applied, unless you override them. The attributes are passed to the z/OS host.

Data set creation attributes are controlled in the following ways, in increasing order of priority:

- Default server data set creation attributes (refer to “Attributes file (rfstab)” on page 80
- Default installation data set creation attributes, specified by the system administrator in the attributes file (refer to “Attributes file (rfstab)” on page 80)
- DFSMS data class attributes
- Data set creation attributes specified for the fileset in the **devtab** file (refer to “devtab” on page 87)
- Data set creation attributes specified at file creation, for example, in the **mkdir**, **notepad** (edit), or **copy** commands (highest priority).

Note: These data class attributes are not supported by the SMB server:

- Retention period/Expiration date
- Number of volumes

- VSAM imbed index option
- VSAM replicate index option
- CI size of data component
- Percentage of CI or CA free space.

Preparing to create a z/OS file

When you create a z/OS file, you may want to specify what type of file to create.

These following types of files are supported by the SMB server:
- Physical sequential (PS)
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS
- Sequential Access Method (SAM) extended format data sets.

Note: Keyed access to files is not supported by PC clients.

Naming z/OS files:

When naming conventional z/OS files, you must follow the file naming conventions, as described in the *z/OS: DFSMS: Using Data Sets* document, SC26-7410.

A file name (or data set name) can consist of one or several simple names joined so that each represents a level of qualification. For example, the file name `DEPT58.SMITH.DATA3` is composed of three qualifiers.

The following characteristics apply to the file name:
- Each qualifier consists of 1 to 8 alphanumeric characters, national characters (@, #, $), or a hyphen (-).
- Each qualifier must start with an alphabetical or national character.
- The period (.) separates simple names from each other.
- Including all simple names and periods, the length of the file name must not exceed 44 characters. Note that the high-level qualifier that was exported (in the previous example, USERA.PCDSNS) is added as a prefix to the file name. So, if you created file **r:\b.c.d.e**, the SMB server would create data set **USERA.PCDSNS.B.C.D.E**.
- PDS and PDSE member names can be up to 8 characters long.

Restrictions using alias names for z/OS files: For PDSs and PDSEs, alias names (for member names) are not supported. They are not displayed when you list the names in a directory (that is really a PDS or PDSE). You cannot access a file (member) by its alias name. If you try to create a file in the directory that has the same name as an alias, it is denied.

Creating physical sequential files

Note that the following examples assume that an **rfs** fileset is mapped to the **r:** drive and the data set prefix in **devtab** is SMITH. It is also assumed that these files are translated between ASCII and EBCDIC characters by an administrator specification of:
- **_IOE_RFS_TRANSLATION=ON**, or
- Text in the **devtab** entry for **rfs** fileset, or
- Text in the **attributes** file for the **rfs** fileset.

When creating a physical sequential (PS) file, specify the **dsorg(ps)** attribute (if it is not the default already) with a file creation command, such as the **notepad** command.

```
C:\>notepad "r:\new,dsorg(ps)"
```

When you save the file using **notepad**, you have just created a new PS file named SMITH.NEW.

When reading or changing data in a Physical Sequential file with fixed-length records in text mode, the **blankstrip** processing attribute in the attributes file controls how trailing blanks are handled. If **blankstrip** is specified in the attributes file, trailing blanks are removed from the end of each record when it is read, and blanks are padded to the end of each record when it is written. **blankstrip** is the default. If **noblankstrip** is specified in the attributes file, trailing blanks are not removed from the end of each record when it is read, and blanks are not padded to the end of each record when it is written. In this case, each record written must be the correct size or an I/O error is reported.

Note: When copying a file to an RFS file system, if you are overriding the data set creation attributes, you must specify the target file name in the **copy** command. For example:

```
C:\>copy file1 "r:\newfile1,space(2,5),cyls"
```

The previous example is correct.

```
C:\>copy file1 "r:\,space(2,5),cyls"
```

The previous example is incorrect.

Creating direct access files

When creating a direct access (DA) file, specify the **dsorg(da)** attribute (if it is not the default already) with a file creation command, such as the **notepad** command.

```
C:\>notepad "r:\new,dsorg(da)"
```

You have just created a new DA file named SMITH.NEW.

Creating PDSs and PDSEs

Partitioned data sets (PDSs) and partitioned data sets extended (PDSEs) can be used as directories, and their members are files within those directories. An illustration of the use of PDSs to act as directories is shown on page 144. For general information on PDSs and PDSEs, refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410.

You cannot create new directories within a PDS or PDSE, due to the nature of these data structures.

Creating a PDS or PDSE -- mkdir dsntype(pds), dsntype(library):

To create a PDS or PDSE, perform the following steps:

1. If creating a PDSE, use the **mkdir** (make directory) command specifying the **dsntype(library)** attribute to create a PDSE named smith.datalib:

   ```
   C:\>mkdir "r:\datalib,dsntype(library)"
   ```

   If creating a PDS, use the **mkdir** (make directory) command specifying the **dsntype(pds)** attribute as follows:

   ```
   C:\>mkdir "r:\datalib,dsntype(pds),dir(20)"
   ```

   Note: You can omit specifying the **dsntype(pds)** attribute, if **pds** has been specified for the **dsntype** attribute in the attributes file on page 81.

2. You can use the **notepad** command to create a PDS or PDSE member named smith.datalib(member1):

   ```
   C:\>notepad "r:\datalib\member1"
   ```

   Input your text, save it, and quit.

You have now created a PDS or PDSE member. You can use the **type** command to view the contents of your PDS or PDSE member.

Note: The SMB server supports a maximum of 14,562 members in a PDS or PDSE data set. When a PC read-directory request on a PDS or PDSE is processed, the SMB server returns up to 14,562 member names. Other requests, such as read and write, to individual members are not affected.

Removing a PDS or PDSE -- erase, rmdir:

To remove a PDS or PDSE, first make sure that the PDS or PDSE is empty. You can delete all members under the directory using the **erase** command. Then use the **rmdir** (remove directory) command. This example removes the `datalib` directory, and confirms its removal by a failed try to query it (**dir** is the list files command):

```
C:\>dir r:\DATALIB
 Volume in drive R has no label.
 Volume Serial Number is 0000-0000

 Directory of r:\datalib

06/29/00  09:04p        <DIR>          .
06/29/00  10:51a        <DIR>          ..
08/01/96  12:19p                   298 butcvsmp
02/17/95  02:13p                 1,019 copyunld
09/20/96  09:20a                   550 su2aloc
09/18/96  07:13a                   548 test
07/02/99  06:49a                     0 testtext
07/02/99  07:56a                    38 testtxt3
09/20/96  07:50a                   547 textaloc
               9 File(s)          3,000 bytes
                             61,440,000 bytes free
C:\>erase r:\datalib\*
C:\>rmdir r:\datalib
C:\>dir r:\datalib
File Not Found
```

Accessing PDS or PDSE members:

There is more than one way to access PDS and PDSE members. For example, you could display the existing PDS member `smith.source(bigblue)` by entering either of these command sequences:

```
C:\>type r:\source\bigblue
```

or

```
C:\>cd r:\source
C:\>type r:bigblue
```

These two approaches are equivalent.

Updating or extending a PDS or PDSE member

The SMB server does not generally support updating or extending a PDS or PDSE member directly. To update or extend a PDS or PDSE member, a client program must follow these steps:

1. Copy the file to the client machine
2. Update or extend the copied version on the local system
3. Truncate the original file to zero size by sending a SETATTR request with zero file size
4. Copy the updated version on the local host to z/OS by writing request.

Some client editors follow the above steps, for example, the AIX and z/OS UNIX **vi** editor. Other editors do not follow the above steps, for example, the z/OS OEDIT editor. In the latter case the user must save the updated version into a new file.

When reading or changing data in a PDS member or PDSE member with fixed-length records in text mode, the **blankstrip** processing attribute in the attributes file controls how trailing blanks are handled. If **blankstrip** is specified in the attributes file, trailing blanks are removed from the end of each record when it is read, and blanks are padded to the end of each record when it is written. **blankstrip** is the default. If **noblankstrip** is specified in the attributes file, trailing blanks are not removed from the end of each record

when it is read, and blanks are not padded to the end of each record when it is written. In this case, each record written must be the correct size or an I/O error is reported.

Renaming or moving a PDS or PDSE member:
Record file system (RFS) files and directories can only be renamed or moved in a way that does not cause them to be moved from one directory to another.

If you are writing to a PDS or PDSE member and a timeout occurs, the timeout causes the member to close. The remaining write requests appear to append to a PDS or PDSE member. This operation is not supported and causes an I/O error. To avoid timing out, increase the timeout setting.

Wildcard copy to a PDS or PDSE:
To ensure that a wildcard copy, `copy c:\mydir\* r:\source`, to a PDS or PDSE can be completed successfully, a prior PDS member is closed and dequeued (if necessary) to allow the creation of a new member.

Limitations of writing to a PDS:
The PDS support in the SMB server adheres to the conventions used in z/OS. For example, you cannot have more than one member of a PDS open for writing at a time. If you try to write to a member of a PDS while another member is open for write by a different user, you get a ″*** Permission denied″ message.

A PDS member stays open for the timeout period specified in the appropriate timeout processing attribute, **filetimeout**, or until you try to create or write to another member.

Concurrent writes to a PDSE:
The SMB server supports concurrent writes to a PDSE. If you are writing to one member of a PDSE, another client can write to any other member in the same PDSE.

Creating VSAM files
The SMB server supports three types of VSAM files:
- key-sequenced (KSDS)
- entry-sequenced (ESDS)
- relative record (RRDS).

However, keyed access and relative-number access to the files are not supported.

If you plan to update a VSAM data set (for example, with the **notepad** editor or with the **copy** command), the data set must have been defined with the REUSE option. Trying to write back a VSAM data set that was not defined as reusable results in an ″I/O error″, ″failure to open″, or similar error message. When you create a VSAM file through the SMB server, the REUSE option is always specified by the server.

For more information on VSAM files, refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410.

In the following example, the attributes indicate that:
- Spanned records are allowed
- Organization is key-sequenced
- Keys are 8 bytes long and start in position 0 of each record
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
C:\>copy r:\ksds.old "\r:ksds.new2,spanned,dsorg(indexed),keys(8,0),
  recordsize(1k,4k),space(50,10),shareoptions(1,3),
  vol(d80cat)"
```

In the following example for creating a VSAM ESDS file, the attributes indicate that:

- Spanned records are allowed
- Organization is entry-sequenced
- Average record size is 1024
- Maximum record size is 4096
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
C:\>copy r:esds.old "r:esds.new3,spanned,dsorg(nonindexed),
  recordsize(1k,4k),space(50,10),shareoptions(1,3),
  vol(d80cat)"
```

In the following example, the attributes indicate that:
- Spanned records are not allowed
- Organization is relative record, numbered in ascending order
- Average record size is 1024
- Maximum record size is 1024
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT.

```
C:\>copy r:rrds.old "r:rrds.new4,nonspanned,dsorg(numbered),
  recordsize(1k,1k),space(50,10),shareoptions(1,3),
  vol(d80cat)"
```

Specifying attributes multiple times

Specifying an attribute several times on a line does not cause an error. The line is read from left to right, and the last of any duplicate attribute is used. For example:

```
C:\>notepad "r:file,recfm(vb),recfm(fb)"
```

This results in a file created with a fixed-blocked format.

Exploiting SAM striped files

With SAM striping, data I/O is done in parallel to improve performance. For a file with 16 stripes, data is processed on the first track of the allocated space on the first volume (that is, the first stripe), then on the first track of the second volume, and so on for all 16 volumes. Then, processing continues with the second track of all the volumes, then the third, and so on.

The SMB server can support data set striping through the use of data class and storage class attributes that define extended format data sets. The SMB server can exploit the performance of extended format data sets by reading multiple blocks at a time when reading ahead.

For more information on striped files, refer to the *z/OS: DFSMS: Using Data Sets* document, SC26-7410.

Handling of the file size value

Many file system commands (such as: **dir**) require the file size to be returned. This appendix explains some performance and accuracy considerations in obtaining the file size value.

The meaning of the file size value returned by the SMB server and how fast the file size is returned depends on:
- Whether you use **text** or **binary** processing mode
- The type of data set being accessed
- If the data set is system-managed
- Whether you use **fastfilesize** or **nofastfilesize** processing.

Storage of the file size value

Whether or not the file size value is already stored on your z/OS system, affects how quickly files are accessed and depends on the type of z/OS data set used.

System-managed PS, VSAM, and PDSE data sets:

For system-managed data sets, text and binary file size are saved on non-volatile storage (DASD) and maintained by the SMB server for the following data set types:

- Physical sequential (including striped)
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDSE members.

When the SMB server accesses a data set for the first time, it performs a read-for-size to get the text or binary file size and stores this value on DASD. Subsequent file size requests from clients do not cause the server to read for size, thus improving performance. However, when the data set is modified outside the server by a non-PC application (for example, by the TSO editor), the stored file size could be incorrect. When the data set is accessed again by the server, read-for size is done to determine the correct file size.

Migrated system-managed data sets under DFSMS/MVS V1R3:

DFSMS/MVS Version 1 Release 3 allows data set attribute accessibility for SMS managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3. Supported SMS managed data set types:

- PS
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDS
- PDSE.

The SMB server is able to obtain the attributes of a supported SMS managed migrated data set without recalling the data set. Attributes such as the record format and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a non-PC application (for example, by the TSO editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall is done to determine the correct file size.

Non-system managed, PDS, and DA data sets:

The file size value for non-system managed data sets, PDS members, and DA data sets is cached in virtual storage until released but not written to DASD. Therefore, for these types of data sets, the file size value is regenerated after the file is released or after the server is restarted.

How the file size is generated

When a file is first accessed (for example with **dir**), usually the entire file is read to determine its size, except for when specifying the **recfm(f)** or **recfm(fbs)** attributes where the binary size can be computed without reading the file. If the file is a system-managed PS, VSAM, or PDSE member, both binary and text file sizes are stored on DASD, so that subsequent file size requests do not require the file to be read.

Binary file size can be quickly generated by using **recfm(f)** or **recfm(fbs)** to specify a fixed-length record format for the data set. With this format type, the server pads the last logical record with binary zeros in **binary** mode processing, because z/OS always expects complete logical records. If the application tolerates these zeros, using **recfm(f)** or **recfm(fbs)** allows the binary size to be computed quickly because the number of bytes can be computed from the number of blocks, which is stored by z/OS.

If you need the exact file size and are using **binary** mode processing, map it to a variable-format, sequential data set on DASD so that the SMB server does not need to pad a partially filled last logical record to a record boundary.

For reading small files or the beginning of files, the read-for-size might not add any processing time. As the file is being read for size, the beginning of the file is stored in the buffers set aside by the **maxrdforszleft** site attribute, until the buffers are full. When the application reads the beginning of the file, this read is fast because it reads directly from the buffer.

z/OS stores the number of blocks (rather than the number of bytes) in a z/OS file. For most files, therefore, without reading the entire file, the SMB server can only give an estimate of the number of bytes in the file, not the exact number of bytes in the file.

Using fastfilesize to avoid read-for-size:

If you can use an approximate file size for a PDS, PDSE, DA, or non-system managed data set, you can specify the **fastfilesize** attribute to improve performance. With this attribute, the server estimates the size without opening and reading the entire file.

PDS members	For PDS and PDSE members, the **fastfilesize** attribute gets the file size from ISPF statistics if they exist; otherwise, the size of the entire PDS or PDSE data set is returned as the member size.
PS or DA data sets	For PS or DA data sets, an approximate file size is calculated based on the device characteristics, the number of disk tracks in use, and the block size of the data set.
VSAM	For non-system-managed VSAM data sets, the estimated size using **fastfilesize** is the size of the data set.

The **fastfilesize** attribute speeds up data set access by calculating approximate file sizes during data set access. Use this only when you are displaying file names and sizes (using the **dir** Windows command, for example) because many commands (such as **copy** and editors) and many applications might not work correctly if **fastfilesize** is set. When modifying or copying a data set, the **nofastfilesize** attribute should be used to ensure accurate results.

nofastfilesize:

When you use the default, **nofastfilesize** attribute, the SMB server reads the entire file or member to get the file size. It stores the file size value in cache until release. Using this attribute might cause a delay when first accessing very large data sets.

Handling of the time stamps

z/OS UNIX file attributes define the following time stamps:

atime The last time the file was accessed (read).

mtime The last time the file was modified (write).

ctime The last time the file status was changed (chmod).

The SMB server handles time stamps differently for these types of data sets:

- System-managed PS data sets and system-managed VSAM data sets
- Direct Access data sets and non-system managed PS data sets
- Non-system managed VSAM data sets
- PDS and PDSE members.

Time stamps for system-managed VSAM and PS data sets

For system-managed PS data sets and system-managed VSAM data sets, *atime* and *mtime* are fully maintained, and the *ctime* is set to the mtime.

Time stamps for non-system managed PS and DS data sets

For non-system managed PS and DA data sets, consider the following:

- How time stamps are stored
- The requirements of your workstation programs
- The type of data set used to store the file.

Storing time stamps:

For non-system managed PS and DA data sets, the SMB server temporarily stores the time stamps in virtual storage, but not on DASD. These cached attributes are purged when the file is released or when the server is restarted. When the file is accessed again, the time stamps are regenerated.

Client program requirements:

Some workstation-based utilities (such as **make**) rely on date and time stamps. For example, **make** checks the update time of the object file with the source file and recompiles if the source has been updated. Before storing these types of files using the server, examine them before moving them to ensure that these attributes are unimportant. In an environment which relies on such utilities, use HFS.

Generating time stamps:

This is how the SMB server generates *atime* and *mtime* for non-system managed PS and DA data sets from the dates:

atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment

time_increment is either the server local time or 23:59 hours. If *reference_date* or *creation_date* is equal to the server local date, the server local time is added. Otherwise, a fixed value of 23 hours and 59 minutes is added.

If *reference_date* = 0 (that is, the file has not yet been referenced), *atime* and *mtime* are set equal to *ctime*.

Time stamps for non-system managed VSAM data sets

The time stamps for these types of data sets are set to the current time.

Time stamps for PDSs and PDSEs

A PDS data set can act as a directory. Members of the PDS are files within the directory. When the directory is accessed by the client, the z/OS UNIX times are expected for each file.

Ordinarily, z/OS does not maintain time stamps for members of a PDS. The z/OS UNIX time stamps here are generated from the z/OS creation and reference dates of the PDS data set containing the members. This is how the time stamps for PDS members are generated:

atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment

ISPF is a z/OS base element that does maintain some additional statistics for each member. They include the creation date and the last modification date and time.

If the ISPF time stamps are present for a PDS member, this is how the server generates the time stamps and initializes the z/OS UNIX times:

atime = mtime = modification_date + modification_time
ctime = ISPF_creation_date + time_increment

time_increment is either server local time or 23:59 hours as described for non-system managed PS and DA data sets.

The server also creates new ISPF statistics for PDS members created by the clients. The ISPF statistics are created even if existing members do not have statistics.

The time stamp information is saved in the PDS directory according to ISPF conventions. If STATS=ON was specified when the member was created, the server uses them to get more accurate attributes. Even if STATS=ON was not specified originally, the server writes back new time stamp information if the member is modified from the workstation.

Time stamp generation for a PDSE member is identical to that of a PDS with one exception. Accurate *mtime* of a PDSE member is returned to a client as the result of a file attribute request for that PDSE member.

Setting time stamps

PC clients can issue commands that result in SETATTR requests (such as, `Windows Explorer, right-click on file, and Choose Properties`) to set the *atime* and *mtime* for a system-managed PS or VSAM data set. For PDSE members, setting *mtime* is allowed, but setting *atime* is not supported. PDSE member *mtime* is also maintained by PDSE access methods, so it is modified when a TSO user modifies the PDSE member.

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS: TSO/E Primer*, SA22-7787, *z/OS: TSO/E User's Guide*, SA22-7794, and *z/OS: ISPF User's Guide Volume I*, SC34-4822, for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Appendix G. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your company or send inquires, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes are incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300

2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

Advanced Function Printing	AFP	AIX
BookManager	DFSMS/MVS	DFSMShsm
IBM	IBMLink	Infoprint
Language Environment	Library Reader	MVS
OS/390	RACF	Resource Link
WebSphere	z/OS	z/OS.e

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

This bibliography provides a list of z/OS publications that are useful when using the Distributed File Service SMB support. The complete title, order number, and a brief description is given for each publication.

Distributed File Service publications

- *z/OS: Distributed File Service DFS Customization*, SC24-5916

  This document helps system and network administrators configure the Distributed File Service.

- *z/OS: Distributed File Service DFS Administration*, SC24-5915

  This document introduces the Distributed File Service concepts to system and network administrators and provides an in-depth understanding of the Distributed File Service, its uses and benefits. This document also provides reference information for the commands and files used by system and network administrators to work with the Distributed File Service.

- *z/OS: Distributed File Service zSeries File System Administration*, SC24-5989

  This document provides guidance and reference information for system and network administrators to use when they work with the zSeries File System Administration support of Distributed File Service.

- *z/OS: Distributed File Service Messages and Codes*, SC24-5917

  This document provides detailed explanations and recovery actions for the messages, status codes, and exception codes issued by the Distributed File Service.

Infoprint Server publications

- *z/OS: Infoprint Server Operation and Administration*, S544-5745

  This document describes how to operate Infoprint Server and all of its components, and how to set up and maintain the new consolidated Printer Inventory, which contains all printer information required by Infoprint Server and its components.

- *z/OS: Infoprint Server User's Guide*, S544-5746

  Describes for the end user how to submit print jobs to Infoprint Server using JCL, TCP/IP commands, the new AOPRINT JCL procedure, z/OS UNIX commands (lp, lpstat, cancel), the Printer Port Monitor for Windows, and Internet Printing (IPP).

UNIX System Services publications

- *z/OS: UNIX System Services Command Reference*, SA22-7802

  Describes the UNIX shells, utilities, debugger, and UNIX TSO/E commands in UNIX System Services.

- *z/OS: UNIX System Services Planning*, GA22-7800

  Contains information needed to plan for the use of UNIX System Services in a z/OS environment.

- *z/OS: UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803

  Describes UNIX System Services callable services.

- *z/OS: UNIX System Services User's Guide*, SA22-7801

  Describes how to use the UNIX shells and file system.

Security Server publications

- *z/OS: Security Server RACF Security Administrator's Guide*, SA22-7683

  Explains RACF concepts and describes how to plan for and implement RACF.

- *z/OS: Security Server RACF Command Language Reference*, SA22-7687

  Contains the functions and syntax of all the RACF commands.

Index

Special characters

A

E

F

G

S

T

U

V

W

Z

Program Number: 5694-A01

Printed in U.S.A.

SC24-5918-04

Spine information:

z/OS V1R4.0 Distributed File Service SMB Administration