z/OS

Distributed File Service SMB Administration

Version 2 Release 2

SC23-6886-01

Note

Before using this information and the product it supports, read the information in “Notices” on page 217.

This edition applies to Version 2 Release 2 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

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) support of the IBM® z/OS® Distributed File Service base element of z/OS. SMB is a protocol for remote file and print access used by Microsoft Windows clients. This protocol is also known as Common Internet File System (CIFS).

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).

Note: As of z/OS Version 1 Release 13, IBM has withdrawn support for the z/OS Distributed File Service support that utilizes the Distributed Computing Environment (DCE) architecture. IBM recommends the z/OS Network File System (NFS) implementation as the replacement. For more information about NFS, see *z/OS Network File System Guide and Reference*.

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 and each part is 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 91 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.
\	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. 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**).

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS V2R2 Information Roadmap*.

To find the complete z/OS library, go to IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

DFS information

For information about installing Distributed File Service components, refer to *z/OS Program Directory*.

Information concerning Distributed File Service-related messages are in *z/OS Distributed File Service Messages and Codes*.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R2 Distributed File Service SMB Administration
  SC23-6886-01
- The topic and page number that is related to your comment.
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS Support Portal (http://www-947.ibm.com/systems/support/z/zos/).

Summary of changes

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

Summary of changes for z/OS Version 2 Release 2 (V2R2)

The following changes are made in z/OS Version 2 Release 2 (V2R2).

New

- Information was added to "Using passthrough authentication" on page 61 as a result of APAR OA43395.
- With APAR OA46044, "z/OS SMB implementation restriction for SMB digital signing" on page 80 was added.
- "Keeping medium weight threads during pthread_exit processing" on page 188 was added as a result of APAR OA39909, which described changes by z/OS UNIX that enable applications to specify how medium weight threads handle pthread_exit processing, and APAR OA45175, which describes the SMB updates that were made as a result of the change in threading behavior.

Deleted

- Mentions of the _IOE_SMB_PROTOCOL_LEVEL environment variable were deleted. It had been used to allow either NTLMV2 or LanMan protocol, but because LanMan is no longer supported by Microsoft, the environment variable should no longer be used.
- Information about Windows XP was deleted because it is no longer supported.
- Because Windows Server 2003 is no longer supported, these references have been removed:
  - Microsoft Windows Server 2003, Standard Edition and Microsoft Windows Server 2003, Enterprise Edition
  - Microsoft Windows Terminal Server on Windows Server 2003

Summary of changes for z/OS Version 2 Release 1

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS V2R2 Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS V2R2 Introduction and Release Guide*

Part 1. SMB support administration guide

This section covers guidance information for system administrators and Personal Computer (PC) users

Chapter 1. An overview of SMB support

The Distributed File Service Server Message Block (SMB) support provides a server that makes Hierarchical File System (HFS) files and data sets available to SMB clients. At the same time, these files can be shared with local z/OS UNIX applications. In addition, Windows SMB clients can make remote print requests to z/OS printers that are connected to the Infoprint Server for z/OS.

Server Message Block (SMB) is a protocol for remote file/print access that is used by Windows clients. This protocol is also known as Common Internet File System (CIFS). The data sets supported include: sequential data sets on DASD (Direct Access Storage Device), partitioned data sets (PDS), partitioned data sets extended (PDSE) and Virtual Storage Access Method (VSAM) data sets. The data set support is typically referred to as Record File System (RFS) support. To access shared directory paths and shared printers, the SMB protocol is supported through the use of TCP/IP on z/OS. Personal Computer (PC) clients on the network use the file and print sharing functions that are included in their operating systems.

Note: Throughout this documentation, references are made to HFS. Unless otherwise stated, HFS is a generic reference that includes HFS, z/OS File System (zFS), TFS, and AUTOMNT file system data. If you are in a sysplex shared file system environment, SMB support of zFS is limited to zFS compatibility mode file systems that are non-sysplex aware.

Supported SMB clients

The supported SMB clients include the following clients:

- Microsoft Windows Server 2008
- Microsoft Windows Terminal Server on Windows Server 2008
- Microsoft Windows 7 Professional, Microsoft Windows 7 Enterprise, and Microsoft Windows 7 Ultimate Editions (32- and 64-bit clients)
- Microsoft Windows Vista Business and Windows Vista Enterprise
- SUSE LINUX with Samba (SUSE LINUX Enterprise Server 9 (s390) - Kernel 2.6.5-7.202.5-s390 with Samba 3.0.14a-0.4)
- Redhat Linux with Samba (Kernel 2.4.20-43.9.legacy with Samba 3.0.20-2).

Restrictions for Linux clients: Linux clients can use many file systems. Only Linux clients with kernels that support the Samba file system (smbfs) can connect to the z/OS SMB server. The smbfs file system is part of the Linux smbfs package. The choice to use smbfs is specified when mounting a file system. For example, the following Linux **mount** command specifies smbfs (the command itself is typically one long unbroken line but is not shown as one unbroken line due to space limitations):

```
mount -t smbfs //servername/sharename /mountdirectory
   -o username=myusername,password=mypassword
```

The Samba **smbmount** command automatically uses smbfs (the command itself is typically one long unbroken line but is not shown as one unbroken line due to space limitations):

```
smbmount //servername/sharename /mountdirectory
  -o username=myusername,password=mypassword
```

Consult your Linux client documentation for availability of the smbfs file system or use the Linux **man mount** command to determine whether the `-t smbfs` option is available.

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 one or more disks on the computer. Each disk is referred to by a different drive letter (for example, `A:`, `B:`, `C:`, or `D:`). 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 the printer to which to send the print request.

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 might be other functions that are not supported.

Using SMB support, PC users can access files that are on a z/OS system. That is, PC users can access files that are not located on their computer. Remote files are visible 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 called "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, Windows or Linux PC users are able to use remote printers that are attached to a z/OS system. Remote printers are visible as additional printers available to the PC user. Remote printers are installed on PCs using existing commands or install utilities.

SMB processes

SMB support provides a server process that makes file data and printers available to PC users. Administrators can 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

A PC accesses files located on a z/OS system through one or more shared directories 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. (See Chapter 10, “Accessing data,” on page 75 for more information about 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

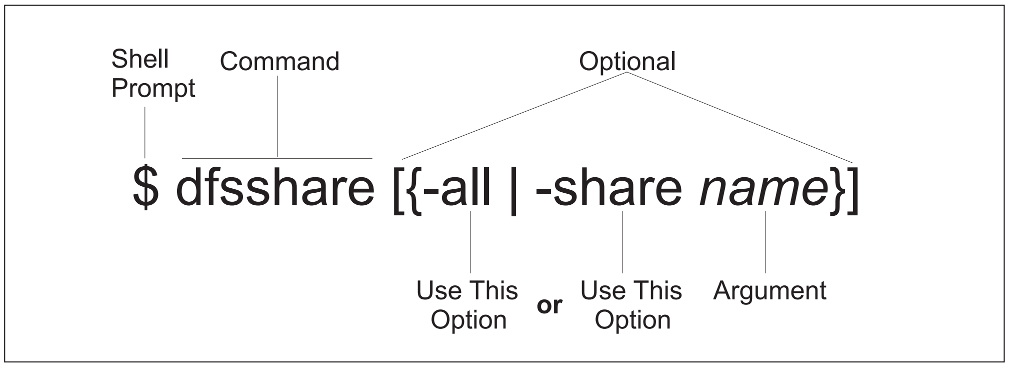
Distributed File Service administrators make z/OS printers available to Windows PCs by creating shared printers. A shared printer is given a share name and specifies a z/OS Infoprint Server printer definition name. To access a remote printer from a Windows or Linux PC, clients can install a remote printer 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 name}]
```

The following example illustrates and 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 iInformation
: 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 91.
- 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 `-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 a new release to use the SMB support, it is important to follow the instructions in *z/OS V2R2 Migration* and review the topics in this section.

New release considerations

When migrating to a new release of Distributed File Service, it is not necessary to copy and save your customized files before installing. The files you can customize are listed in Appendix C, "Customizable files," on page 189.

Some of the customizable files listed might 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.

Review the following list before installing a new release of Distributed File Service:

Distributed File Service zFS considerations

The SMB server cannot export zFS read-write file systems when zFS is running sysplex-aware on the same system where the SMB server is running or on the system that owns the zFS file system. If you want to export zFS file systems using the SMB server, ensure zFS is configured as non-sysplex aware (sysplex=off in the zFS IOEFSPRM configuration file). Beginning with z/OS V1R12, you also have the option of running zFS sysplex-aware on a file system basis (sysplex=filesys in the zFS IOEFSPRM configuration file). With this support, when the SMB server tries to export a file system, it checks the zFS file system configuration to determine whether zFS is running sysplex-aware or non-sysplex aware and only export non-sysplex aware file systems. For additional information, see:
- The topics on What's new or changed for zFS and Using the SMB server in *z/OS Distributed File Service zFS Administration*.
- The topic on Distributed File Service migration actions in *z/OS V2R2 Migration*.

DCE DFS considerations

As of z/OS Version 1 Release 13, IBM has withdrawn support for the z/OS Distributed File Service support that utilizes the Distributed Computing Environment (DCE) architecture. (Support for SMB is not affected by the withdrawal of DCE support.) If your installation used DCE DFS support, IBM recommends using the z/OS Network File System (NFS) implementation as a replacement. For more information, see "DCE migration considerations" on page 10.

Symbolic links considerations

The Distributed File Service customizable files are in the `/etc/dfs` path. 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 Appendix C, “Customizable files,” on page 189.

Installing Distributed File Service

If you have not already done so, install the current release of Distributed File Service using the instructions in *z/OS Program Directory*.

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 *z/OS Program Directory*.

Stopping the Distributed File Service server.

If the Distributed File Service server is running on the target image, stop the server now using the operator command **`stop dfs`**. See Chapter 4, “Managing SMB processes,” on page 25 for more information.

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 *z/OS Program Directory* 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. Socket option access control is provided with the SERVAUTH class resource EZB.SOCKOPT.*sysname*.*tcpname*.SO_BROADCAST. If a profile protecting this resource is defined, the SMB server user ID must be given at least READ authority to this profile.

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. See Chapter 6, “Mapping SMB user IDs to z/OS user IDs,” on page 37 for more details. You might also need to review the _IOE_MVS_DFSDFLT environment variable in the `/opt/dfslocal/home/dfskern/envar` file. See “Logon considerations” on page 59 for more details.

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.

Printers 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.

Running the /opt/dfsglobal/scripts/dfs_cpfiles program

Some of the customizable files listed in Appendix C, "Customizable files," on page 189, might 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 following the instructions in *z/OS Program Directory*, you should run it now against the target image `/etc/dfs` to ensure the new release has all the new configuration files.

More information about the **dfs_cpfiles** program can be found in Chapter 3, "SMB post installation processing," on page 13.

Note: The **dfs_cpfiles** program does not create all the files necessary for SMB support. See "SMB configuration file considerations" on page 10 for more information.

Updating existing 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.

Authorized programs considerations

SMBPW is the authorized program for the Distributed File Service. See *z/OS Program Directory* for a description of the PARMLIB member updates required for the member IKJTSOxx.

SMB file and print considerations

For 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.

Ensure 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` to find the Open Cryptographic Service Facility (OCSF) library. For example, the environment variable 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 are in the `/opt/dfslocal/var/dfs` directory. See Chapter 7, "Sharing files," on page 41 and Chapter 8, "Sharing printers," on page 67 for more information about 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** can 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 might need to be changed. Other **dfskern** environment variables that might 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, "SMB post installation processing," on page 13.

DCE migration considerations

As of z/OS Version 1 Release 13, the DFS client function has been removed. The DFS client (DFSCM) is a physical file system that was started during z/OS UNIX initialization, based on a FILESYSTYPE statement in the BPXPRMxx parmlib member. If your installation used the DFS client, you should remove the following statement from the BPXPRMxx parmlib member to prevent the client from initializing.

```
FILESYSTYPE TYPE(DFSC)
ENTRYPOINT(IOECMINI)
PARM('ENVAR("_EUV_HOME=/opt/dfslocal/home/dfscm") /
>DD:IOEDFSD 2>&1')
 ASNAME(DFSCM)
```

If you do not remove this statement before the first IPL of a z/OS V1R13 or later system, you receive error message IOEP12402E. z/OS UNIX successfully initializes, but you need to follow the guidance in the message to remove the entry and restart z/OS UNIX. See *z/OS Distributed File Service Messages and Codes* for more information.

Also, if you have not already done so, you should use the z/OS UNIX **pax** command to migrate any data in DCE DFS or Episode file systems to other file systems (for example, zFS). The recommended general procedure is as follows:

1. Set up a zFS file system to receive the data. See *z/OS Distributed File Service zFS Administration* for more information.
2. Copy your DCE DFS or Episode file system data to the zFS file system, using the z/OS UNIX **pax** command. For more information, see *z/OS UNIX System Services Command Reference*.
3. Set up a z/OS NFS server to allow data access from a remote z/OS UNIX system. See *z/OS Network File System Guide and Reference* for details.

Note: RACF DCE segments are still used and are required for SMB encrypted password support. For more information, see "RACF DCE segments for SMB encrypted password support" on page 64.

Chapter 3. SMB post installation processing

This topic contains information to assist you in completing the installation and configuration of the Distributed File Service SMB file and print server support. Before using the SMB support, you must install the z/OS release, Distributed File Service, and the other base elements of z/OS using the appropriate release documentation. During the installation of the z/OS release, you must perform actions to activate the SMB support. If you are migrating from a prior release of Distributed File Service and you want to use the SMB function, review Chapter 2, "Considerations for a new SMB release," on page 7. For supporting installation and migration information, go to the following URL:

`z/OS Install/Migration page (http://www.ibm.com/systems/z/os/zos/installation/)`

Installation verification

After you have installed SMB, as instructed by the *z/OS Program Directory*, you may want to verify that the installation completed successfully.

The installation verification procedure is typically used to start the DFS server for the first time during the installation step of a new release. But, if the DFS server is already running, be sure to stop the DFS server before following the installation verification sets described below. Note that the term "DFS server" is used to describe the Distributed File Service server when it provides SMB file/print serving support.

To ensure the installation completed successfully, start the DFS started task by issuing the following command.

```
S  DFS,PARM='-nodfs'
```

View the SYSLOG and look for the following message, which indicates that the DFS server started successfully. However, additional configuration is required before the server can be used as an SMB file/print server. If the DFS server started task does not initialize successfully, contact the IBM Support Center for help.

```
IOEP01103I DFS kernel initialization complete.
```

General configuration guidelines

Before you perform any post-installation configuration steps, you should review the following guidelines:

- You have the option of running SMB with either UID(0) or a unique nonzero UID. See "RACF definitions for SMB" on page 21.
- If your system has RACF program control enabled, some features might require you to place specific resources under program control. These features include:
  - Hardware encryption using the Open Cryptographic Service Facility (OCSF)
  - SMB encrypted password support configuration
  - Security environment active through the use of the _IOE_USE_PTHREAD_SECURITY environment variable

During initialization or program execution, messages can notify you when program control is not active. For example, message ICH420I identifies that CSNBCKI needs program control.

```
ICH420I PROGRAM CSNBCKI FROM LIBRARY SYS1.CSF.SCSFMOD0 CAUSED THE
ENVIRONMENT TO BECOME UNCONTROLLED.
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR SERVER (BPX.SERVER)
```

- To use the SMB File/Print support, you might 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 use 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.

  To use the SMB support, you must configure the support on the system using the following topics:
  - "Installing and configuring the SMB file and print server," which includes configuration tasks such as:
    - Activating the SMB file/print serving function
    - 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 or other operating systems that issue requests to file/print servers using the SMB protocol.
  - "Creating the default DFS configuration files" on page 19
  - "Defining SMB administrators" on page 18
  - "RACF definitions for SMB" on page 21

Note: During **dfskern** initialization, a small file (`opt/dfslocal/var/dfs/rfsfile`) is created (or updated) to keep track of RFS file identifiers. Do not erase or change this file.

Installing and configuring the SMB file and print server

Before you begin this process, you must ensure that the user ID the SMB Server is running under has authorization to the BPX.DAEMON and BPX.SERVER resources in the RACF FACILITY class. If you are configuring SMB for the first time, or for a new release, see "RACF definitions for SMB" on page 21.

To install, configure, and access the Distributed File Service server (**dfskern**) for SMB file and print server operation, perform the following steps:

1. Install and perform post-installation processing of the Distributed File Service, using the applicable instructions in *ServerPac: Installing Your Order* (for ServerPac users) and *z/OS Program Directory* (for CBPDO users).

The following list summarizes the information in those documents:

a. Ensure that the target and distribution libraries for the Distributed File Service are available.

b. Run the prefix.SIOESAMP(IOEISMKD) job from UID 0 to create the symbolic links that are used by the Distributed File Service. This job reads the member prefix.SIOESAMP(IOEMKDIR) to delete and create the symbolic links.

c. Ensure that the DDDEFS for the Distributed File Service are defined by running the prefix.SIOESAMP(IOEISDDD) job.

d. Install the Load Library for the Distributed File Service. The Load Library (hlq.SIEALNKE) must be APF-authorized and must be in link list.

e. Install the samples (hlq.SIOESAMP).

f. If you plan to use encrypted passwords (recommended) and optionally, you want to use OCSF and hardware encryption, you must ensure that the appropriate authorizations have been given to the DFS server user ID to use OCSF services. See the section on Cryptographic Services OCSF customization considerations" in *z/OS Program Directory* and the "Configuring and Getting Started" section in *z/OS Open Cryptographic Services Facility Application Programming* for information about this topic.

   If you are using Integrated Cryptographic Service Facility (ICSF), define the following service names to the CSFSERV resource class:

CSFCKI	CSFDEC	CSFENC	CSFKEX	CSFKGN
CSFKIM	CSFKPI	CSFKRW	CSFKRC	CSFKRD
CSFKRR	CSFMGN	CSFOWH	CSFRNG	

   You might also need to PERMIT the user ID DFS READ access to the profiles in the CSFSERV general resource class. For more information about the CSFSERV resource class, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

g. The SMB server process (DFSKERN) needs a relative dispatching priority set less than TCP/IP and UNIX System Services, but not too low. When the dispatch priority is too low, TCB and associated SRBs are not processed quickly enough, which can result in possible resource contention and hang conditions.

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

2. Stop the Distributed File Service server (**dfskern**), if it is already running, using the instructions in Chapter 4, "Managing SMB processes," on page 25.

3. Define administrators on the host system using the instructions in "Defining SMB administrators" on page 18.

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 and print server, are typically created before the Distributed File Service installation is verified by the `/opt/dfsglobal/scripts/dfs_cpfiles` shell script, as indicated in *z/OS*

Program Directory. See “Creating the default DFS configuration files” on page 19 for more information about dfs_cpfiles.

5. Modify the `/opt/dfslocal/home/dfskern/envar` file to activate SMB file and print servers by setting the environment variable _IOE_PROTOCOL_SMB=ON.

   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 *z/OS Program Directory*.

   If you are using the print capability of the SMB file and print server, ensure that the Infoprint Server is installed and customized using the applicable instructions in *z/OS Program Directory*. 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 *z/OS Program Directory*.

   For example, a LIBPATH that specifies both the OCSF DLL directory and the Infoprint Server DLL directory might be `LIBPATH=/usr/lib:/usr/lpp/Printsrv/lib`.

   There is a relationship between number of threads specified for the SMB server and the maximum number of threads that z/OS UNIX permits in a process. The following DFSKERN envars have an effect on the number of threads created for the SMB server:
   - _IOE_RFS_WORKER_THREADS
   - _IOE_SMB_CALLBACK_POOL
   - _IOE_SMB_MAIN_POOL
   - _IOE_TKMGLUE_SERVER_THREADS

   There are also a number of dynamically created DFSKERN threads (approximately 25). The total of the DFSKERN threads must be less than the z/OS UNIX MAXTHREADS specification in the BPXPRMxx. If this is not the case, DFSKERN can abend during thread creation. The number of z/OS UNIX MAXTHREADS can be increased using the SETOMVS MAXTHREADS=*nn* operator command. The number of z/OS UNIX MAXTHREADS can be displayed using the D OMVS,O operator command. See *z/OS MVS System Commands* for additional information about these operator commands.

6. Because the SMB file and print server runs as an APF-authorized server, you must ensure that any DLLs that are used by the SMB file and print server are APF-authorized. This can be accomplished by using the z/OS UNIX **`extattr +a`** command. If you are using the Infoprint Server or OCSF, see the “Infoprint Server Customization Considerations” section in *z/OS Program Directory* and the “Cryptographic Services OCSF Customization Considerations” section in *z/OS Open Cryptographic Services Facility Application Programming* for information about the location of the DLLs and setting the APF-authorized extended attribute. The DFS load library is called hlq.SIEALNKE.

7. SMB 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, 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 host name. See Chapter 5, “Networking considerations,” on page 35.

8. SMB communicates over several TCP/IP ports. Check your TCP/IP profile data set and verify that there are no reserves for ports 137,138, 139, and 445. See *z/OS V2R2.0 Communications Server: IP Configuration Reference* for information about TCP/IP configuration and reserving ports.

9. 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 using the instructions in Chapter 6, “Mapping SMB user IDs to z/OS user IDs,” on page 37.

   In addition, z/OS users should put the following line in their HFS .profile file in their home directory or in `/etc/profile`. This value is then set for all z/OS UNIX users.

```
export _EUV_AUTOLOG=NO
```

10. Determine whether you intend to use passthrough authentication. See “Using passthrough authentication” on page 61 for information about passthrough authentication. Users in the domain will be authenticated using a Windows Server acting as a 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.
11. Determine whether you intend to use password encryption. For more information, see the _IOE_SMB_CLEAR_PW environment variable and _IOE_SMB_CLEAR_PW. Before you enable password encryption, your PC users must store their SMB password into their RACF DCE segment. Otherwise, they are not able to log on except possibly as a guest user.
12. Determine whether you intend to permit 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 permitted 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 z/OS user ID can access. If guest users are permitted, users that specify an incorrect password or no password become the guest user ID. It is better to deny guest users until you are certain you need this capability and that it meets your security guidelines.
13. 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 permits the SMB server to support file systems mounted by using the z/OS Automount Facility. See *z/OS UNIX System Services Planning* for information about the automount facility. Dynamic export also permits the SMB server to dynamically “discover” mounted file systems without the need to provide **dfstab** and **devtab** entries for the file systems. See “Dynamic export for HFS” on page 48 for information about using the dynamic export capability of the SMB server.
14. Define shared directories if the SMB file and 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 file sets using the applicable instructions in Chapter 7, “Sharing files,” on page 41. For RFS, the DFS server user ID (typically DFS) must have RACF ALTER authority to the data sets that are made available to PC users. Alternatively, you can 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 because 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 has authority only to the required data sets. See *z/OS Security Server RACF Security Administrator's Guide* for information about the OPERATIONS attribute).

15. Define shared printers if the SMB file and 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`. See Chapter 8, "Sharing printers," on page 67 for more information.
16. SMB server performance can be significantly enhanced using the Language Environment® HEAPPOOLS(ON) parameter. See "ioepdcf" on page 114 on how to specify HEAPPOOLS for the SMB server. See *z/OS Language Environment Programming Guide* for information about HEAPPOOLS.
17. Start the Distributed File Service server (**dfskern**) by following the applicable instructions in Chapter 4, "Managing SMB processes," on page 25.
18. Configure PC client workstations to access the SMB file and print server using the instructions in Chapter 9, "Locating the SMB server," on page 69.

Rule: 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

The SMB Administrator must have root authority on the z/OS system. This means a z/OS user ID with a UID=0 or superuser authority assigned through the BPX.SUPERUSER resource in the RACF FACILITY class or the equivalent for another security-based product. This authority is required in order to issue certain commands and to define shared directories and shared printers. For additional details on assigning superuser authority, see the topic on using the BPX.SUPERUSER resource in the FACILITY class in *z/OS UNIX System Services Planning*.

Two types of users can start or stop the Distributed File Service server address space and the processes controlled by DFSCNTL (see Chapter 4, "Managing SMB processes," on page 25):

- 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.

The following environment variable should be specified in the SMB Administrator's z/OS UNIX environment (for example, in the SMB Administrator's `$HOME/.profile` file).

```
export _EUV_AUTOLOG=NO
```

Creating the 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. See "Symbolic links to /etc/dfs created during installation" on page 190 for more information about 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 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 system (**UID=0** or superuser authority).
2. Enter the following command in the shell environment to invoke the **dfs_cpfiles** program.

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

   Note: To ensure that any existing user customized data is not overlaid, any existing DFS configuration file is not replaced.

When you complete these steps, the **dfs_cpfiles** program creates customizable files for all aspects of the Distributed File Service. A subset of these files are applicable to the SMB file and print server. Figure 1 lists the customizable files that are applicable to the SMB file and print server.

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

Figure 1. Customizable files for the SMB file and print server

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. See Chapter 7, "Sharing files," on page 41 for more information.

- The **smbtab** file must be updated to define shared printers. See Chapter 8, "Sharing printers," on page 67 for more information.

- The **smbidmap** file must be updated to map PC user IDs to z/OS user IDs. See Chapter 6, "Mapping SMB user IDs to z/OS user IDs," on page 37 for more information.
- Optionally, the **hfsattr** and **rfstab** files can be updated to first define HFS data translation by file name extension attributes and next by RFS attributes for RFS files. See Chapter 7, "Sharing files," on page 41 for more information.

Figure 2 displays the possible output and messages produced after using **dfs_cpfiles** to create the customizable configuration files for the Distributed File Service.

- 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
```

```
***********************************************************************
**                 Distributed File Service                          **
**           Default Configuration Files Creation Program            **
***********************************************************************
    Attempt to Create envar Files....
       File /opt/dfslocal/home/daemonct/envar created
       File /opt/dfslocal/home/dfscntl/envar created
       File /opt/dfslocal/home/dfsexport/envar created
       File /opt/dfslocal/home/dfskern/envar created

     Attempt to Create Miscellaneous Configuration Files....
      File /opt/dfslocal/etc/ioepdcf 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/hfsattr created
      File /opt/dfslocal/home/dfskern/smbidmap created
```

Figure 2. Example output of dfs_cpfiles

Migration Tips:

1. If you are migrating to a new release of the Distributed File Service and **dfs_cpfiles** created a new set of customizable files, you might need to add your customization data to the newly created files.
2. If you are migrating to a new release of z/OS and new customizable configuration files were not created by **dfs_cpfiles**, you might want to update preexisting customizable files with new customization options available with this release of Distributed File Service. See Chapter 2, "Considerations for a new SMB release," on page 7 for more information about what is new in this release.
3. The SMB user identity mapping file is identified by the environment variable _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.

RACF definitions for SMB

You must define SMB clients and servers to the z/OS Security Server RACF (or equivalent security product) so they have the correct authorization to initialize and perform required system services. Access to z/OS file data is done using the authorization of the user accessing the data or the default authorization defined by your installation.

For more information about RACF administration tasks, see *z/OS Security Server RACF Security Administrator's Guide*. See *z/OS Security Server RACF Messages and Codes* for information about RACF messages.

Note: If the RACF FSACCESS class is used on your system to restrict access to z/OS UNIX file systems, the following authority levels are required:

1. The user ID under which SMB runs (typically, this is "DFS") must have update authority to any SMB-exported file system that is protected by FSACCESS.
2. Each MVS™ user ID that is mapped to an SMB user ID and requires access to an FSACCESS protected file system must also be given update authority to the file system.

For more information, see *z/OS UNIX System Services Planning*.

Setting up a new SMB installation to run under a UID(0) TSO user

Perform the following steps if you are setting up SMB for the first time and you want the TSO/E user for SMB to have UID(0) authority:

1. Define DFSGRP as a group. For example you could issue the following command:

```
ADDGROUP DFSGRP SUPGROUP(SYS1) OMVS(GID(2))
```

2. Create a RACF-defined UID(0) TSO/E user, under which the SMB server will run. For example, you could issue the following command. (Note that this command would be entered on one line.)

```
ADDUSER DFS OMVS(HOME('/opt/dfslocal/home/dfscntl') UID(0))
       DFLTGRP(DFSGRP) AUTHORITY(CREATE) UACC(NONE)
```

3. Define DFS and DFSKERN as started tasks. For example:

```
RDEFINE STARTED DFS.** STDATA(USER(DFS))
RDEFINE STARTED DFSKERN.** STDATA(USER(DFS))
SETROPTS RACLIST(STARTED) REFRESH
```

4. Create a profile in the FACILITY class for DFSKERN, and give the user ID that runs DFS, SMB, or both server functions UPDATE access to the profile. For RACF, use the following commands:

```
RDEFINE FACILITY DFSKERN.START.REQUEST UACC(NONE)
PERMIT DFSKERN.START.REQUEST CLASS(FACILITY) ID(DFS) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
```

5. If profiles have not been defined to protect the BPX.SERVER and BPX.DAEMON resources in the FACILITY class, define them in RACF. For example:

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

6. Give the TSO/E user authorization to the BPX.SERVER and BPX.DAEMON resources in the RACF FACILITY class. For example:

```
PERMIT BPX.SERVER CLASS(FACILITY) ACCESS(READ) ID(DFS)
PERMIT BPX.DAEMON CLASS(FACILITY) ACCESS(READ) ID(DFS)
SETROPTS RACLIST(FACILITY) REFRESH
```

7. If your system is set up for RACF program control protection, allow the SMB daemon control task (IOEPDCT) to run using your installation's procedures.

Setting up new SMB installation to run under a unique nonzero UID TSO/E user

Perform the following steps if you are setting up SMB for the first time and have a requirement that the TSO/E user that SMB runs under is a unique nonzero UID:

1. Define DFSGRP as a group. For example:

```
ADDGROUP DFSGRP SUPGROUP(SYS1) OMVS(GID(2))
```

2. Create a RACF-defined TSO/E user under which the SMB server will run. Do not use “DFS” as the TSO/E user ID for a unique nonzero UID setup because “DFS” is already in use for zFS. In this example, `NEWID` is new user ID. (Note that this command would be entered on one line.)

```
ADDUSER NEWID OMVS(HOME('/opt/dfslocal/home/dfscntl') UID(unique UID))
      DFLTGRP(DFSGRP) AUTHORITY(CREATE) UACC(NONE)
```

3. If a profile has not already been defined in the RACF FACILITY class protecting the resource BPX.SUPERUSER, define one and give the user ID you just defined authorization to it. For example:

```
SETROPTS CLASSACT(FACILITY)
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ACCESS(READ) ID(NEWID)
```

4. Define DFS and DFSKERN as started tasks. For example:

```
RDEFINE STARTED DFS.** STDATA(USER(NEWID))
RDEFINE STARTED DFSKERN.** STDATA(USER(NEWID))
SETROPTS RACLIST(STARTED) REFRESH
```

5. Create a profile in the FACILITY class for DFSKERN, and give the user ID that runs DFS, SMB, or both server functions UPDATE access to the profile. For RACF, use the following commands:

```
RDEFINE FACILITY DFSKERN.START.REQUEST UACC(NONE)
PERMIT DFSKERN.START.REQUEST CLASS(FACILITY) ID(NEWID) ACCESS(UPDATE)
```

6. If profiles are not defined protecting the BPX.SERVER and BPX.DAEMON resources in the RACF FACILITY class, define them. For example:

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

7. Permit the user ID to the BPX.SERVER and BPX.DAEMON RACF FACILITY class profiles. For example:

```
PERMIT BPX.SERVER CLASS(FACILITY) ACCESS(READ) ID(NEWID)
PERMIT BPX.DAEMON CLASS(FACILITY) ACCESS(READ) ID(NEWID)
SETROPTS RACLIST(FACILITY) REFRESH
```

8. If your system is set up for RACF program control protection, allow the SMB daemon control task (IOEPDCT) to run using your installation's procedures.

Modifying an existing SMB installation to run with a unique nonzero UID

Perform the following steps if you are modifying an existing SMB setup that runs under a TSO/E user with UID(0) authority to run under a TSO user without UID(0) authority. In this procedure, you create a new unique nonzero UID user ID that SMB will run under, establish the user ID's relationship with the existing RACF defined group, and give the new user ID the appropriate RACF authorizations.

Do not use DFS as the TSO/E user for a unique nonzero UID setup because DFS is already in use for zFS.

1. Create a RACF-defined TSO/E user under which the SMB server will run. The following procedure uses `NEWID` as the example new TSO user ID. (Note that this command would be entered on one line.)

```
ADDUSER NEWID OMVS(HOME('/opt/dfslocal/home/dfscntl') UID(unique UID))
        DFLTGRP(DFSGRP) AUTHORITY(CREATE) UACC(NONE)
```

2. Create a profile in the RACF FACILITY class protecting the resource BPX.SUPERUSER and give the new user ID access to it.

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ACCESS(READ) ID(NEWID)
```

3. Give the new user ID UPDATE access to the resource DFSKERN.START.REQUEST in the FACILITY class.

```
PERMIT DFSKERN.START.REQUEST CLASS(FACILITY) ID(NEWID) ACCESS(UPDATE)
```

4. Update the current generic started task definition for the unique nonzero UID user, `NEWID`, that the SMB server will run under by issuing the following commands.

```
RALTER STARTED DFS.** STDATA(USER(NEWID))
RALTER STARTED DFSKERN.** STDATA(USER(NEWID))
SETR RACLIST(STARTED) REFRESH
```

5. Give `NEWID` READ access to the BPX.SERVER and BPX.DAEMON resources in the RACF FACILITY class.

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
PERMIT BPX.SERVER CLASS(FACILITY) ACCESS(READ) ID(NEWID)
PERMIT BPX.DAEMON CLASS(FACILITY) ACCESS(READ) ID(NEWID)
SETROPTS RACLIST(FACILITY) REFRESH
```

6. In some cases, you might have to modify the **dfscntl** and **dfskern** environment variable files to ensure they have the correct permissions (that is, file permissions set to 644). The files originally have owner, group, and other permissions set to 644 (owner read/write, group read, other read); but, over time, the file permission can change. If the nonzero UID created in the preceding steps is not able to read the **dfscntl** and **dfskern** files, you can receive a RACF ICH408I message (or unauthorized request message from equivalent security products). Check the following files to ensure the file permissions are set correctly:
   - `/opt/dfslocal/home/dfscntl/envar`
   - `/opt/dfslocal/home/dfskern/envar`

   For example, the correct permissions display in this format.

```
-rw-r--r--   1 DFSID    DFSGRP       267 Dec  3 08:56 envar
```

   If they are not set correctly, you can change the permissions using the **`chmod`** command.

```
chmod 644 envar
```

Chapter 4. Managing SMB processes

The SMB daemons (DFSCNTL, DFSKERN, and EXPORT) run as separate processes in the DFS address space, as shown in Figure 3. DFCNTL controls the starting of the DFSKERN and EXPORT daemons.

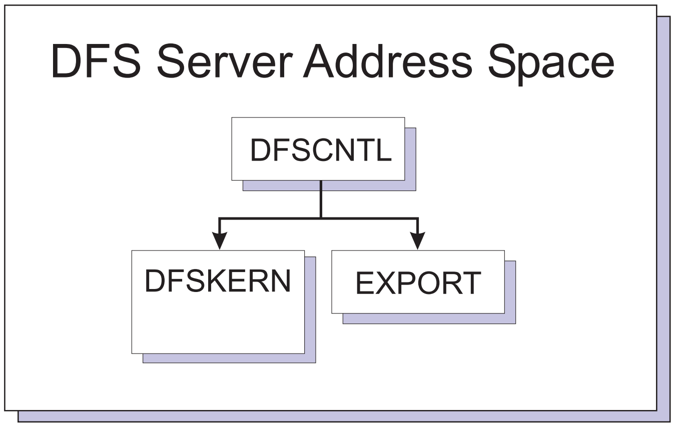


Figure 3. Address space for the DFS server

If the SMB server daemons are not started as instructed, they fail with the following return codes. The return codes for the IOEDFSCL and IOEPDCT daemons are shown as follows:

4	The module is not APF-authorized.
8	The module was not started independently through JCL.
12	The module is not running as a started task; instead, it is running as EXEC PGM=*xxxxxxxx* started from JCL.

The codes for the IOEDFSKN and IOEDFSXP daemons are shown as follows:

4	The module is not APF-authorized.
8	The module was started independently through JCL.

Figure 3 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 the _IOE_DAEMONS_IN_AS 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 its own address space (called the dfskern address space) as shown in Figure 4 on page 26.

Running dfskern in its own address space might 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.

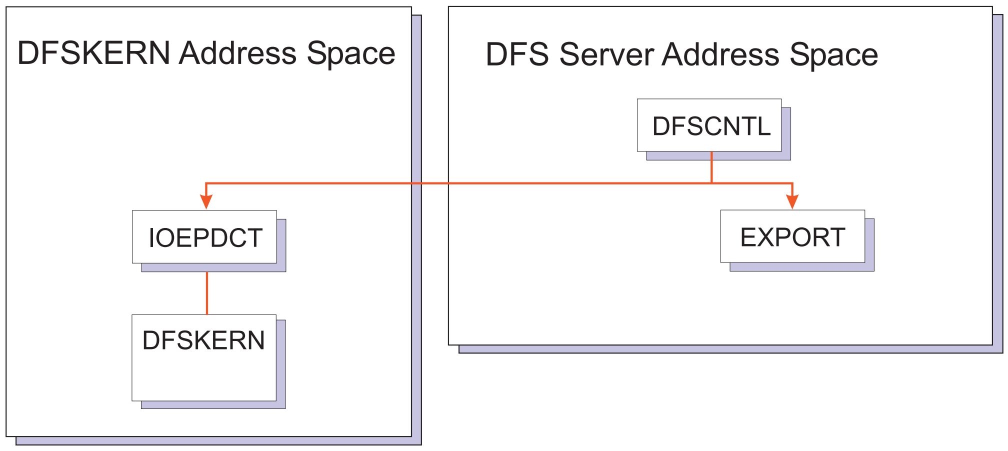


Figure 4. DFSKERN in a separate address space

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 about 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, see "Daemon configuration file" on page 31.

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 32.

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

dfskern
The load module name for the dfskern daemon. 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?

Two types of users 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 more information, see *z/OS Program Directory*.

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 can be started automatically during system IPL. To start the DFS server address space, use the z/OS **START** command as shown in the following example.

```
start dfs
```

Ideally, the daemons run continuously in the background and do not need to be started or stopped again. However, you might start the DFS server daemons 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 briefly in the following sections. For complete details, see Chapter 12, “z/OS system commands,” on page 93.

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]
```

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 and 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 (**dfskern**, **export**, **dfscntl,** and **unexport**).

options Values that are passed to the daemons.

Using MODIFY DFS to start DFS server daemons

With the MODIFY DFS operator command, you can use a single command to start either an individual daemon or all the daemons.

To start the dfskern daemon, issue the following command:

```
modify dfs,start dfskern
```

To start all daemons, issue the following command:

```
modify dfs,start all
```

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 can start out of order or stopped erroneously. This can 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. For complete details on MODIFY, see Chapter 12, “z/OS system commands,” on page 93.

Order of starting DFS server daemons

When individual DFS server daemons are started manually, the successful startup of some daemons depends on the availability of the services provided by other daemons. Therefore, the DFS server daemons must be started in the following sequence:
1. **dfskern**
2. **export**

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.

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

DFS server daemons are stopped in any of the following ways:
- Using the **STOP** operator command.
- Using the **MODIFY DFS** operator command

Using the STOP command to stop DFS server daemons

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 command:

```
stop dfs
```

To stop the DFS server daemons, but not the DFS server address space, use the STOP ALL command. The STOP ALL command causes **dfscntl** to stop all daemons that it controls. For example:

```
modify dfs,stop all
```

Using MODIFY DFS to stop DFS server daemons

With the MODIFY DFS operator command, you can either stop an individual daemon or all the daemons that are configured on the host using a single command.

To stop the dfskern daemon, issue the following command:

```
modify dfs,stop dfskern
```

To stop all daemons, issue the following command:

```
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 command.

```
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, as shown in the following examples.

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

Figure 5 is an example of a query command to **dfscntl**. The output is sent to the z/OS console.

```
modify dfs,query dfskern

IOEP00022I DFS daemon DFSKERN status is READY and process id is 781.
```

Figure 5. Example output from query command to dfscntl

Figure 6 on page 31 shows an example of the output produced by a **modify dfs,query all** command.

```
IOEP01101I DFS daemon DFSCNTL status is READY and process ID is 67108898.
IOEP01101I DFS daemon DFSKERN status is READY and process ID is 16777252.
IOEP01101I DFS daemon EXPORT status is DOWN and process ID is 0.
IOEP01101I DFS daemon UNEXPORT status is DOWN and process ID is 0.
```

Figure 6. Example output from query all command

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 gives you the ability 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 can be started automatically during system IPL. To start the DFS server address space, use the z/OS **START** command, as shown in the following 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 Language Environment 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 7 on page 32 shows the typical contents of the Daemon Configuration file.

```
DFSKERN  CONFIGURED=Y LMD=DFSKERN ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern')/
>DD:DFSKERN  2>&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:UNEXPORT 2>&1"    RESTART=300 TIMEOUT=300
```

Figure 7. Daemon configuration file

In the **ARG** (argument list) field of this example, **ENVAR** indicates the Language Environment 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. See "envar" on page 112 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.

The Daemon Configuration File is optional. When it is omitted, the defaults are as listed in Figure 7. The Daemon Configuration File can only be modified when the DFS server address space is not running.

Important notice to users: Under typical circumstances, you do not need to edit the Daemon Configuration File. Although there might be certain situations when you modify the Daemon Configuration File, it is recommended that it is done under the supervision of an IBM service representative.

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 can 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, the command would have the following format.

```
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. See "Dynamic export for HFS" on page 48 for information about 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

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 that is 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 might 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.

Note: The SMB server does not support Internet Protocol version 6 (IPv6). Use Internet Protocol version 4 (IPv4) instead.

If you want PC clients to access the SMB server by using DNS, then you must ensure that the host name 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) host name 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**.)

The supported Windows SMB clients make SMB calls directly over TCP or through NetBIOS over TCP/IP. When connecting directly over TCP, the client makes calls using a TCP/IP connection to server port 445. When using NetBIOS over TCP/IP, the client makes calls to ports 137, 138 and 139. When connecting to the server, these clients will attempt to connect to ports 445 and 139 to establish a session. The _IOE_SMB_TRANSPORTS environment variable specifies the enable modes as NetBIOS (port 139), DIRECT (port 445), or BOTH (ports 445 and 139). The default is NetBIOS. Many SMB clients have configuration options to enable NetBIOS over TCP/IP, SMB directly over TCP/IP, or both. The server responds on the enabled ports; the client software can choose to attempt one protocol prior to the other or both in parallel.

Microsoft Windows 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 that is 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 network directory 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 directory of PCs that contain the WINS server IP address. The SMB server announces itself to the Browser by using subnet broadcast on User Datagram Protocol (UDP) port 138. It does this at 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 and Sharing Center 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 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.

To enable the Network and Sharing Center directory function, there must be a Master Browser on the same network segment as the SMB server. Typically, Windows clients and servers are configured to act as Master Browsers by default; therefore, a Windows client or server should already meet this requirement.

Windows 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, a sequential data set or a member of a PDS, 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 can be used to map SMB user IDs to z/OS user IDs; each of these procedures are described in the following sections:

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

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.

Restriction: To limit the number of calls to the security manager, the default behavior of the SMB server is to cache user credential information, including the Security Server RACF group information, when it is started. The cached information is not refreshed. To ensure the SMB server recognizes user credentials, you must stop and restart the SMB server. To change the default behavior to dynamically recognize changes to user credential information, set the **_IOE_USE_PTHREAD_SECURITY** environment variable to ON.

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 can be used (for example: **`oedit`** or **`vi`**). 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 suggested for readability). The following list 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 can be a simple SMB user ID (when the domain of the SMB user ID is unimportant) 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.

  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. 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 correct case so that the directory created by automount has the correct name. (For example, it might 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 uppercase when used to logon to z/OS.

For information about other entries in **smbidmap**, see "smbidmap" on page 116.

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 fold the PC user ID to uppercase before sending it to the SMB server. In this case, if the directory created by automount needs to be lowercase or mixed case, the administrator needs to add an entry to the **smbidmap** file to map the uppercase PC user ID to the correct case z/OS user ID.

Each SMB user can only have one mapping to a z/OS user. Conversely, you can map different SMB users to the same z/OS user.

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
```

If the **smbidmap** file is contained in data set HLQ.DFSKERN, member name SMBIDMAP, this variable is set by the following entry in the envar file.

```
_IOE_SMB_IDMAP="//'HLQ.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 must either restart the SMB server or reload the **smbidmap** file by using the **modify dfs,send dfskern,reload,smbmap** command. See Chapter 12, “z/OS system commands,” on page 93 for more information about 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 when you enter the **modify dfs,send dfskern,reload,smbmap** 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 yes, 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 yes, 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 the user specifies when logging on to Windows. 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). See “Logon considerations” on page 59 for information about 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 might be prompted for the correct password or denied.

On Windows clients, 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 gives you the ability to use a different SMB user ID than the one used to logon to Windows.

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**).

Considerations for exporting file systems

Before a shared directory can be created, the file system that contains 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 file system 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.

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 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 might try to export (and move the ownership of) the same file system and one of them will fail.

When the SMB server tries to export a file system, it checks the zFS file system configuration to determine whether zFS is running sysplex-aware or non-sysplex aware and only exports a non-sysplex aware file system. To check whether zFS is running sysplex-aware, use the **`zfsadm configquery -sysplex_state`** command. If it returns 2, it is running sysplex-aware. To check whether zFS is running sysplex-aware on a read/write file system basis, use the **`zfsadm configquery -syslevel`** command or the **`modify zfs,query,level`** operator command. If it returns with the field `sysplex(filesys,norwshare) interface(3)` or `sysplex(filesys,rwshare) interface(3)`, zFS is running sysplex-aware on a file system basis.

Sharing HFS files

This section discusses interactions with 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).

PC clients access data on HFS through sharing that is done on a directory tree basis. For a directory to be shared, the file system that the directory is contained in must be exported. Suppose we had the (purposely simplified) entire file hierarchy on a z/OS system shown in Figure 8.

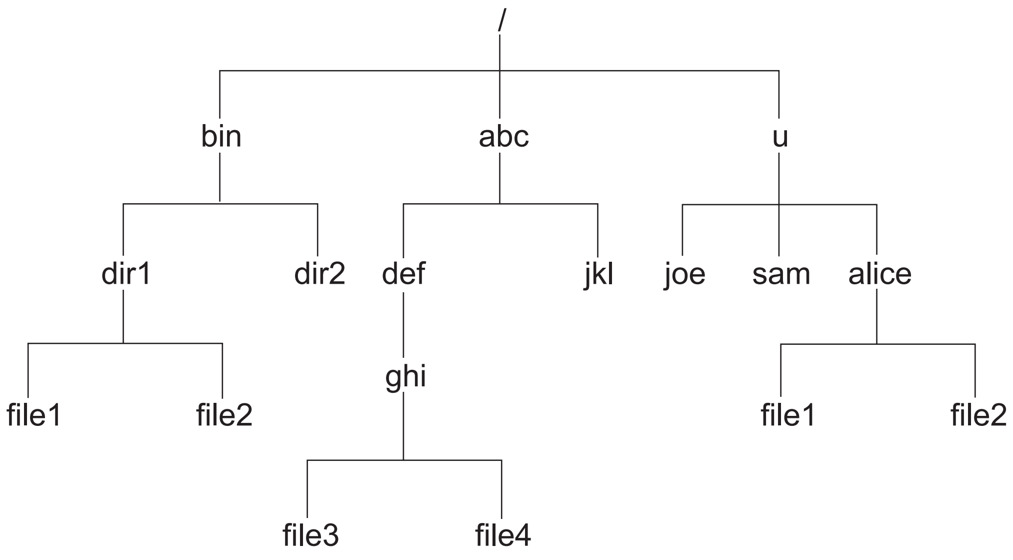


Figure 8. Example of simplified file hierarchy on a z/OS system

This file hierarchy is made up of separate file systems mounted together, as Figure 9 on page 43 shows.

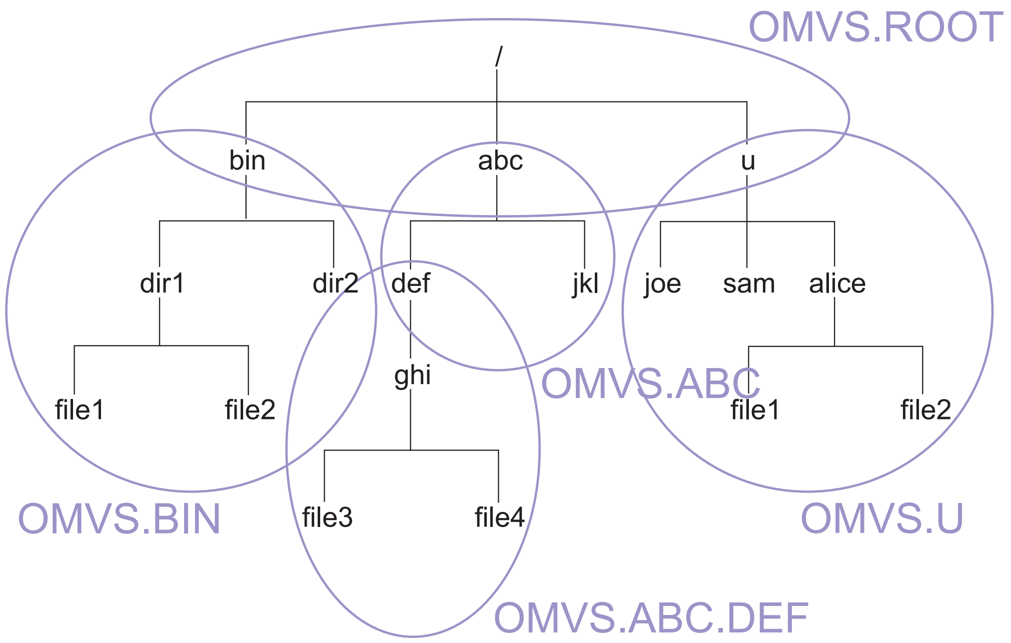


Figure 9. Example of separate file systems that are mounted together

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. Figure 10 on page 44 examines the **/abc/def/ghi** path more closely.

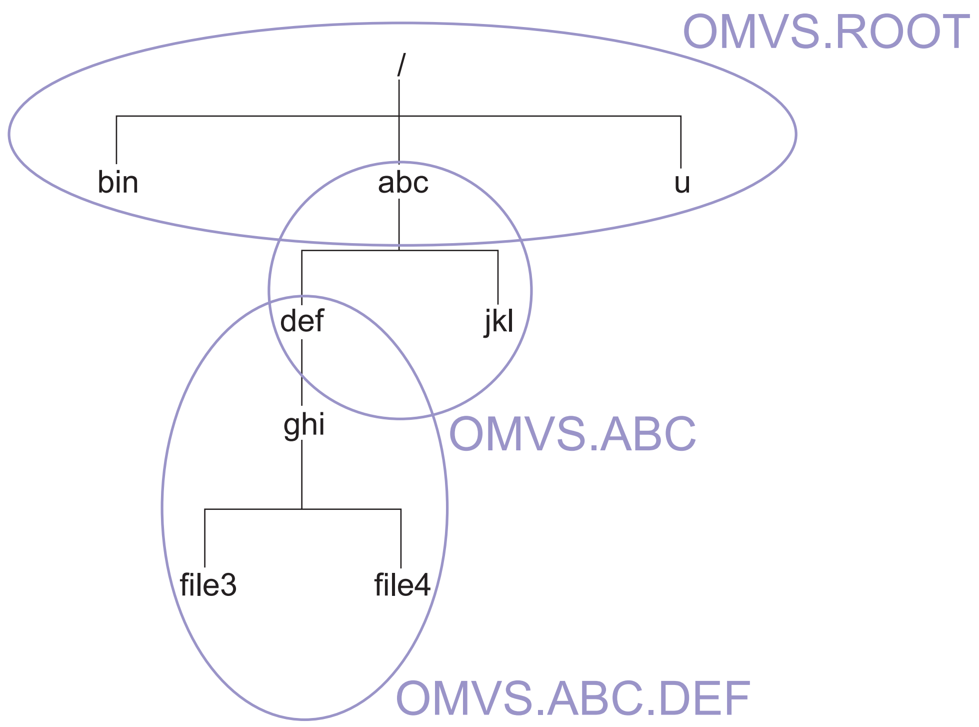


Figure 10. Example of three file systems

The **/abc/def/ghi** path consists of three file systems mounted, as shown in Figure 11. 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).

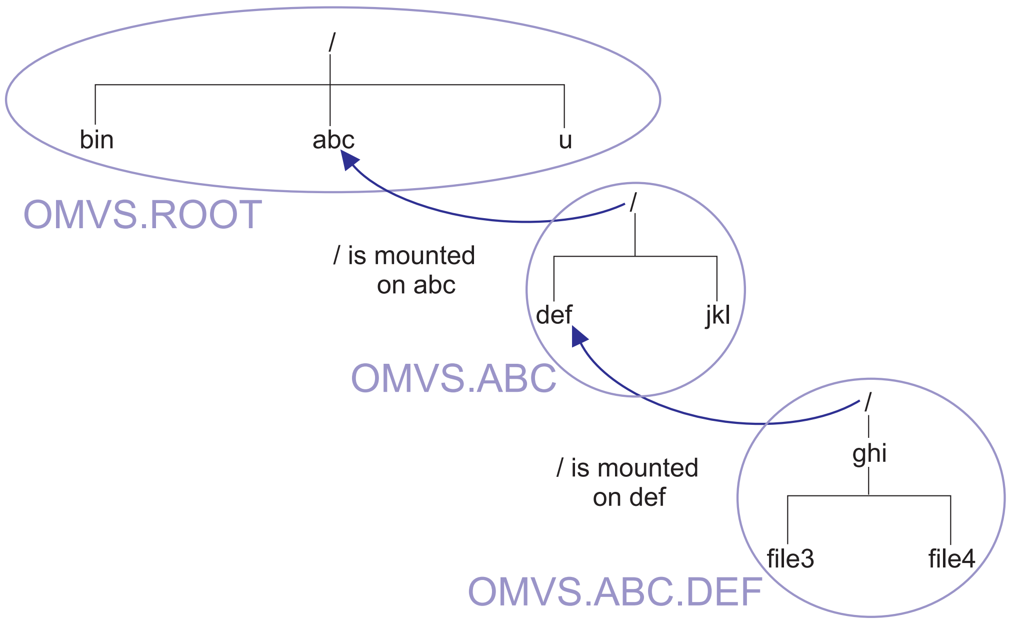


Figure 11. Example of where file systems are mounted

In addition, if there are other file systems below the file system containing the shared directory, you might want to also export these 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 (see “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 ; this number 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 that you want to export, specify them in the **dfstab** and **devtab**. PC users might want to access these directories and files. Alternatively, you can use the dynamic export capability. Refer to "Dynamic export for HFS" on page 48.

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, see "dfsexport" on page 121 and "dfsshare" on page 126.)

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. See *z/OS UNIX System Services Planning* for information about creating and mounting an HFS file system. See *z/OS Distributed File Service zFS Administration*, for information about creating and mounting 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* for information about 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 that might 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
   ```

   You can use the same number for all the numeric values in an individual **dfstab** entry, as long as the number used is unique between dfstab entries. For example, the **dfstab** entry might look like this:

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

   ```
   # dfsshare -share myshare
   ```

If there were additional file systems below the shared directory, you can export those too so that PC users can 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” on page 48.

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 can 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. Because 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 might be other shares that apply to those underlying file systems. Use the **`dfsexport`** command to unexport file systems.

Dynamic export for HFS

Previously, in order to make HFS file systems available to PC users through the SMB server, the administrator had to create a **dfstab** and **devtab** entry for each HFS file system for 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 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. (See *z/OS UNIX System Services Planning* for information about automounted file systems.)

The SMB server attempts to move the ownership of a shared file system to the system that the SMB server is running on when the shared 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 SMB 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, using 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 might 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 if 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. Table 1 shows an example of **smbtab**, **dfstab**, and **devtab** entries.

Table 1. Example of making the entire HFS tree available to PC users

Directory path name	Entry
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. You can 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, they might encounter file systems of different file system types. Some are supported by the SMB server and some are not. SMB servers supports HFS, zFS, TFS, and AUTOMNT file systems. If you are in a sysplex with shared file system, SMB support of zFS is limited to zFS compatibility mode file systems. SMB does not support sysplex-aware zFS (on V1R11 and above) or NFS and DFSC file system types.

Working with automounted file systems and home directories:

It is common for a z/OS UNIX user's home directory to be automounted. The user's home directory does not exist and the file system is not mounted until the home directory is referenced. Refer to *z/OS UNIX System Services Planning* for information about 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`

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 uppercase when it is used to log on to z/OS.)

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

However, some usage problems are possible.

- 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 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 named New Folder. This causes the Automount Facility to try to mount a file system called OMVS.HOME.NEW FOLDER, which 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 might not be obvious to the PC user.

- Even after the user issues **cd** to the correct directory, the file system can get unexported and unmounted after a while if there is no access to the file system for a period of time. This can put the user back into the situation where the home directory no longer exists.

To resolve this, you can 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, user **smith** could issue the following **net use** command.

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

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
```

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 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**.

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 might 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. Some mechanisms give you the ability 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 if 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 code page 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 nonzero 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. See *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for information about 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.

   The z/OS UNIX 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 113.
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 106.
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.
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.

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 code page from the MOUNT TAG (if it exists) or the local code page of the **dfskern** process.

Translation is done using ISO8859-1 for network data (or the code page that is 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. See "ioepdcf" on page 114 for information about how to change the local code page for the **dfskern** process.

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 37 for information about 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 V2R2.0 UNIX System Services User's Guide* for more information about 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 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 that 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. 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 "RFS restrictions" on page 79 for a full description of RFS restrictions.

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. The supported VSAM data set types are key sequential (KSDS), relative record (RRDS) and entry sequence (ESDS). 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. To allow a directory to be shared, the file system that the directory is contained in must be exported.

For example, suppose the following data sets are 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
```

As Figure 12 shows, these data sets would be represented by the following RFS hierarchy.

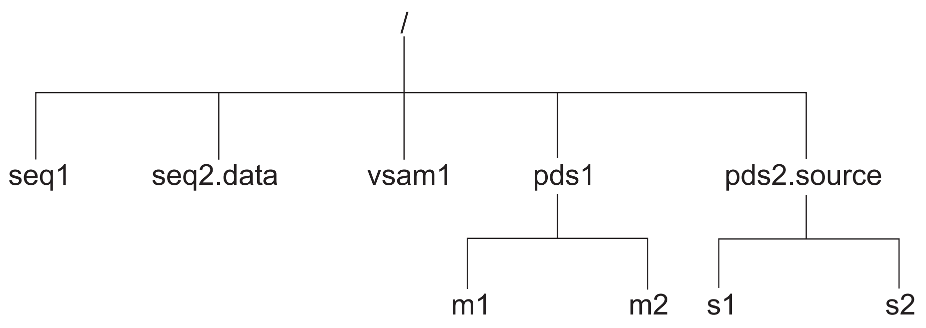


Figure 12. RFS file system hierarchy

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 lowercase. (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, as shown in Figure 13 on page 56.

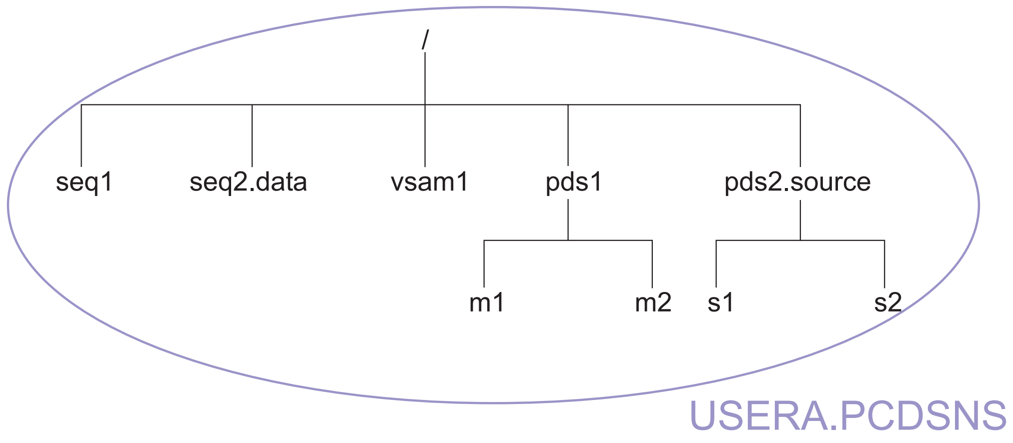


Figure 13. Representation of the RFS file system

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-ssigned 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 *z/OS DFSMS Using Data Sets* for information about 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 example.

   ```
   * 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 the following example.

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

   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 the following example.

   ```
   /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). 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 the following example.

   ```
   /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 can 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 be other shares that apply to those underlying file systems. The **dfsexport** command 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 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 if 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.
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 code page 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. See "ioepdcf" on page 114 for information about how to change the local code page for the dfskern process.

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 37 for information about 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. To authenticate you, the SMB server must receive an SMB user ID and password. The SMB user ID that is received is usually the user ID you specified when you logged on to your PC. You can, however, specify a user ID on a Windows `net use` command. 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 37 for information about 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 (see 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 Server acting as a 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. Typically, your PC sends the password that you specified when you logged on to your PC; therefore, it must match your z/OS password. The SMB server supports mixed case passwords for your z/OS password.

      Some levels of Windows do not send a clear password unless a Registry entry is added. See Appendix B, "Additional information about using the SMB server," on page 183 for more information about 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.

      If the password is not specified on **net use**, you will be prompted for a password; that password is then sent to the SMB server. Some commands do not prompt (for example, **net view**) and, therefore, fail to create a session. In this case, first issue the **net use** command; then you will be prompted for the password and can 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. (The PC does not actually send an encrypted password over the network. An algorithm called *challenge/response authentication* is used that does not actually send any passwords over the network.) 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 by using the z/OS UNIX **smbpw** command before 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 Windows Terminal Server as a client to the z/OS SMB server

When a PC connects to the z/OS SMB server, each PC usually has its own communications session to the z/OS SMB server. See Figure 14 on page 61 for an example.

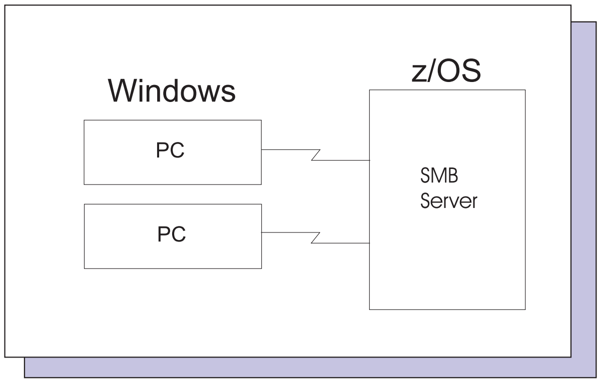


Figure 14. SMB server with a single user session on each communications session

When the Windows Terminal Server is used by a PC, multiple PC users connect to the Windows Terminal Server. They see another entire Windows system screen in a window of their local PC. This window represents applications running on the Windows Terminal Server machine. If the PC user issues a net use command inside that window, that net use command will actually be executed on the Windows Terminal Server machine. The Windows Terminal Server machine can create multiple user sessions (one for each Windows Terminal Server client) on a single communication session to the z/OS SMB server. See Figure 15 for an example.

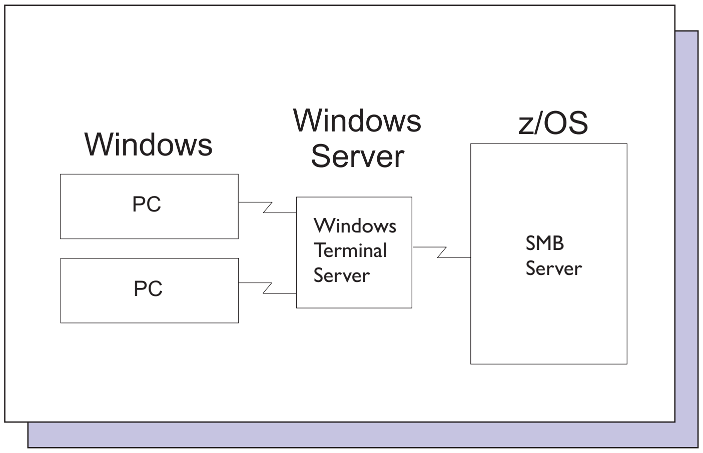


Figure 15. SMB server with multiple, concurrent users on a single communications session

Using passthrough authentication

Passthrough authentication gives you the ability to use your Windows Server acting as a domain controller to authenticate PC users in your domain for access to data through the SMB server. 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.

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 that is chosen by the domain controller), otherwise, the authentication fails. If the authentication fails (including the case where a local authentication failed), you be able to run as a guest z/OS user ID, as described in "Logon considerations" on page 59.

When passthrough authentication is enabled, the SMB server forwards the logon request to a Windows Server acting as a domain controller to perform the authentication of the user. The domain controller determines what security protocol (for example; Kerberos, NTLM, or NTLMv2) will be used during authentication. The domain controller replies back to the SMB server with the method that it requires for the authentication. The SMB server implementation supports the NTLM and NTLMv2 authentication protocol. If the authentication method does not allow for the use of the NTLM or NTLMv2 protocol, the authentication to the domain controller will fail and local authentication might be attempted. During the authentication to the domain controller, audit records are produced in the Windows security event log of successful and unsuccessful authentication attempts. Reviewing this log be used to determine the reason for the failed passthrough authentication attempt. For details on setting the authentication levels for Windows clients, see "Adjusting the authentication level" on page 63.

Passthrough authentication does not allow multiple sessions when digital signing is active. An example of this would be an SMB client concurrently running multiple Windows processes to the SMB server, such as applications running as scheduled tasks. Therefore, multiple sessions are only supported with passthrough authentication when digital signing is not required.

To properly authenticate using passthrough authentication, the SMB server must establish a communications session with the domain controller. This communications session must be maintained in order for the SMB server to handle new user authentications. If the communications session that the SMB server has with the domain controller drops, new users coming to the SMB server (and existing users attempting to reauthenticate due to idle timeout of the PC communications session) will not be able to be authenticated until a domain controller communications session can be reestablished. In addition, new users on an existing PC communications session will not be able to authenticate until that PC communications session has been reestablished (that is, resynchronized with the domain controller communications session). The SMB server will attempt to reestablish a domain controller communications session and to resynchronize existing PC communication sessions as soon as possible.

Passthrough authentication is not used if the domain controller allows guests. In this case, local authentication is used.

Restriction: When using Windows Server 2008 as a domain controller for passthrough authentication, digital signing is not supported. Digital signing is a local security configuration option on both Windows clients and Windows servers. You must ensure the digital signing options match what is shown below. To access the local security policy:

1. From a Windows command, prompt type `secpol.msc`.
2. Select Local Policies > Security Options.

   On the Windows clients ensure that the fields are set as follows:

   Microsoft network client: Digitally sign communications (always) Disabled

   Microsoft network client: Digitally sign communications (if server agrees) Either Enabled/Disabled

   Note: If the client is being used to support multiple sessions, the setting must be set to disabled. An example of this is an SMB client that is concurrently running multiple Windows processes to the SMB server, such as applications running as scheduled tasks.
3. On the Windows 2008 server, ensure that the fields are set as follows:

   Microsoft network server: Digitally sign communications (always) Disabled

   Microsoft network server: Digitally sign communications (if client agrees) Disabled

   When using passthrough authentication, any request that attempts to enforce digital signing will result in access being denied. In these cases, the domain controller returns an error status code of x'C00000022' (STATUS_ACCESS_DENIED). To determine if you are encountering this problem, obtain an SMB trace and find the failing session/negotiate setup as follows:

   a. Issue the modify command to reset the SMB trace:
      ```
      f dfs,send dfskern,trace,reset
      ```
   b. Attempt the access from the client.
   c. Issue the modify command to print the SMB trace:
      ```
      f dfs,send dfskern,trace,print
      ```
   d. Browse the DFSKERN job log and locate the trace statement that corresponds to the SMB negotiate (x73) command of the client:
      ```
      >>>-SMB-sess=0DD29CE8 refct=00000002 csp=0DC87218 com=x73
      issue_aio_request: rb=7E6C8D60 s=10 cmd=134 pools=00000001 rv=0
      --SMB Session Setup & X LAN username=<Administrator>
      ...
      ...
       --SMB session setup & X PT got DC resp, com=x73 err32=220000C0
      The err32 code 220000C0, endian decoded to x'C00000022'.
      ```

Adjusting the authentication level

Perform the following procedure to adjust the authentication level on clients that run supported Windows platforms (for the complete list, see “Supported SMB clients” on page 3).

1. Click **Start**.
2. Select **Control Panel**, and then **Administrative Tools**.
3. Select **Local Security Policy** > **Local Policies** > **Security Options** >**Network security: LAN Manager authentication level**.
4. Using Table 2 on page 64, select the correct setting for your environment.

Table 2 shows the possible level settings and descriptions for the LAN Manager authentication.

Table 2. LAN Manager authentication settings and descriptions

Value	Settings	Description
0	Send LM & NTLM responses	Clients use LM and NTLM authentication and never use NTLMv2 session security. Domain controllers accept LM, NTLM, and NTLMv2 authentication.
1	Send LM & NTLM - use NTLMv2 session security if negotiated	Clients use LM and NTLM authentication, and use NTLMv2 session security if the server supports it. Domain controllers accept LM, NTLM, and NTLMv2 authentication.
2	Send NTLM response only	Clients use NTLM authentication only and use NTLMv2 session security if the server supports it. Domain controllers accept LM, NTLM, and NTLMv2 authentication.
3	Send NTLMv2 response only	Clients use NTLMv2 authentication only and use NTLMv2 session security if the server supports it. Domain controllers accept LM, NTLM, and NTLMv2 authentication.
4	Send NTLMv2 response only/refuse LM	Clients use NTLMv2 authentication only and use NTLMv2 session security if the server supports it. Domain controllers refuse LM and accept only NTLM and NTLMv2 authentication).
5	Send NTLMv2 response only/refuse LM & NTLM	Clients use NTLMv2 authentication only and use NTLMv2 session security if the server supports it. Domain controllers refuse LM and NTLM (they accept only NTLMv2 authentication).

RACF DCE segments for SMB encrypted password support

This section assumes that you are using the IBM Security Server RACF support. For more information about the RACF commands that are referenced in the following instructions, see *z/OS Security Server RACF Security Administrator's Guide*. If you are using an equivalent security product, see the appropriate documentation to perform the equivalent functions.

To permit 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, as shown in the following example.
  ```
  SETROPTS CLASSACT(KEYSMSTR)
  ```
- Activate class DCEUUIDS, as shown in the following example.
  ```
  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 follows.

```
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 ALTUSER command.

  ```
  ALTUSER user_id DCE
  ```

  A complete example of this command follows.

  ```
  ALTUSER g1dfst2 DCE
  ```
- You can display the RACF DCE segment for a user by using the LISTUSER command.

  ```
  LISTUSER user_id NORACF DCE
  ```

  A complete example of this command follows.

  ```
  LISTUSER g1dfst2 NORACF DCE
  ```
- To enable the SMB server to use encrypted passwords, set the **_IOE_SMB_CLEAR_PW dfskern** environment variable to **_IOE_SMB_CLEAR_PW=NOTALLOWED** and restart the SMB server.
- After encrypted passwords have been enabled, each user must enter the following command from OMVS to connect to the SMB server; note that the PC user's SMB password must be specified twice.

  ```
  $ smbpw smb_login_password smb_login_password
  ```

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 *z/OS Infoprint Server Operation and Administration* for information about 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 *z/OS Infoprint Server User's Guide* for information about 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`"). Choose a printer type that is compatible with the printer that you chose in step 1. See *z/OS Infoprint Server Operation and Administration* for information about z/OS printers and their supported printer types.

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

   An example of 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 *z/OS Infoprint Server Operation and Administration*.

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 37 for information about 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 83.

Chapter 9. Locating the SMB server

You can use the SMB server with your PC and Linux clients to access z/OS files and printers. Selecting and configuring the PC client connection to the SMB server ensures that network clients can use the server properly.

Setting up the SMB server

This section discusses setting up the SMB server for use with Windows and Linux clients. For the list of supported clients, see "Supported SMB clients" on page 3.

PC clients can 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.

Using UDP Broadcast

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 might be unnecessary.

Using DNS, WINS, or LMHOSTS file on Windows clients

The following sections describe different methods to locate the SMB server; the methods vary, depending on the level of Windows that is running on the PC client. Before you begin the steps provided for each client, you must ensure that your Windows client can locate the SMB server on the network.

Note: On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Using DNS for Windows 7 clients

Perform the following steps to configure your Windows 7 Professional/Enterprise/ Ultimate client using DNS.

1. Click **Start**.
2. Click **Control Panel**.
3. Access the **Network and Sharing Center** option; the steps to do this vary, depending on how you are viewing this window:
   a. If you are viewing by **Category**: select **Network and Internet** and then select **Network and Sharing Center**.
   b. If viewing by the **Large** or **Small** icon: select **Network and Sharing Center**.
4. Select **Change Adapter Settings**.

5. Right-click **Local Area Connection** for the connection used to connect with the z/OS SMB Server.
6. Select **Properties**.
7. Click **Internet Protocol Version 4**.
8. Click **Properties**.
9. Click **Advanced**.
10. Click the **DNS** tab.
11. Click **Add** in the **DNS Server Addresses: in order of use** area to enter the IP address of one or more of your **TCP/IP DNS Servers**.
12. Click either **Append primary and connection specific DNS suffixes** or **Append these DNS suffixes (in order)** and click **Add** after entering one or more **TCP/IP Domain Suffixes**.

    Note: The previous information may already be supplied for you.
13. Click **OK**.
14. Click **Close** on the **Networking** tab of the **Local Area Connection Properties** window.

Using DNS for Windows Vista clients

Perform the following steps to configure the Windows Vista Business/Enterprise client using DNS.

1. Click **Start**.
2. Point to **Settings** and click **Control Panel**.
3. Windows Vista Business/Enterprise: Click **Network and Sharing Center** and then select **Manage network connections** from the tasks pane.
4. Right-click the **Local Area Connections** icon and choose **Properties**.
5. Click **Internet Protocol (TCP/IP)**.
6. Click **Properties**.
7. Click **Advanced**.
8. Click the **DNS** tab.
9. Click **Add** in the **DNS Server Addresses: in order of use** area to enter the IP address of one or more of your TCP/IP DNS Servers.
10. Click either **Append primary and connection specific DNS suffixes** or **Append these DNS suffixes (in order)** and click **Add** after entering one or more **TCP/IP Domain Suffixes**.

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

Using WINS for Windows clients

Perform the following steps to configure Windows Vista Business/Enterprise or Window 7 Professional/Enterprise/Ultimate client using WINS.

1. Click **Start**.
2. Click **Control Panel**.
3. Access network connection information. The process varies, depending on the Windows client you are using.

- Windows Vista Business/Enterprise: Click **Network and Sharing Center** and then select **Manage network connections** from the tasks pane.
- Windows 7 clients: the steps to access the **Network and Sharing Center** vary, depending on how you are viewing this window:
  a. If you are viewing by **Category**: select **Network and Internet** and then select **Network and Sharing Center**.
  b. If viewing by the **Large** or **Small** icon: select **Network and Sharing Center**.

  Next, select **Manage network connections**.
4. Vista Business/Enterprise: Right click the **Local Area Connections** icon and choose **Properties**.

   Windows 7 Professional/Enterprise: Click **Change adapter settings** from left pane.
5. Vista Business/Enterprise: Click **Internet Protocol (TCP/IP)**.

   Windows 7 Professional/Enterprise: Click **Internet Protocol Version 4**.
6. Click **Properties**.
7. Click **Advanced**.
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 **Enable NetBIOS over TCP/IP** is selected. If it is not selected, click it.
11. Click **OK**.
12. Click **OK** on the **General** tab of the **Internet Protocol (TCP/IP) Properties** window.
13. Click **Close** on the **General** tab of the **Local Area Connection Properties** window.

Using the 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. To configure your Windows Vista Business/Enterprise or Windows 7 Professional/Enterprise/Ultimate client using the LMHOSTS file, perform the following steps.

1. Click **Start**.
2. Click **Control Panel**.
3. Access network connection information. The process varies, depending on the Windows client you are using.
   - Windows Vista Business/Enterprise: Click **Network and Sharing Center** and then select **Manage network connections** from the tasks pane.
   - On Windows 7 clients, the steps to access the **Network and Sharing Center** vary, depending on how you are viewing this window:
     a. If you are viewing by Category: select **Network and Internet** and then select **Network and Sharing Center**.
     b. If you are viewing by the Large or Small icon: select **Network and Sharing Center**.

   Next, select **Manage network connections**.
4. Right-click the Local Area Connection icon and choose **Properties**.
5. Click **Internet Protocol (TCP/IP)**.

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

Finding the SMB server

This section discusses how to find the SMB server and access shared resources from the supported SMB clients (see "Supported SMB clients" on page 3). The process varies, depending on the platform (Linux or Windows) and the level of the operating system. For example, Windows PC clients can use one of the following methods for finding the SMB server:
- Network Folder
- Network and Sharing Center
- Search Computer.

Note:
1. To enable the Network and Sharing Center directory function, there must be a Master Browser on the same network segment as the SMB server. Typically, Windows clients and servers are configured to act as Master Browsers by default; therefore, a Windows client or server should already meet this requirement.
2. On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Using Linux

Linux clients can use the fully-qualified name in the **mount** command; for examples, see "Linux Samba clients" on page 76.

Using the Network Folder

If the SMB server and your Windows 7 PC client are in the same workgroup (domain), perform the following steps using the **Network Folder** to find the server:
1. Click **Start** and then **Computer**.
2. Select **Network** in the panel to view the computers.
3. The computer name of the SMB server, for example `\\ZOSS1`, should be visible.
4. If the SMB server is not visible, ensure that the **Network Discovery** feature is enabled.
5. If the SMB server is still not visible, right-click on **Network** and follow the instructions in "Mapping shared directories to logical drives" on page 75.

Using the Network and Sharing Center

If the SMB server and your Windows Vista PC client are in the same workgroup (domain), perform the following steps using Network and Sharing Center to find the server:

1. Click **Start** and then **Computer**.
2. Select **Network** in the panel to view the computers.
3. Select **Network and Sharing Center** from the toolbar.
4. Click **View computers and devices** from the task pane.
5. The computer name of the SMB server, for example `\\ZOSS1`, should be visible.
6. If the SMB server is not visible, turn on network discovery by selecting it from the toolbar.
7. If the SMB server is still not visible, right-click on Network and follow the instructions for mapping shared directories to logical drives in "Mapping shared directories to logical drives" on page 75.

Searching to find the server

To locate the SMB server from a Windows 7 Professional/Enterprise/Ultimate client, perform the following steps.

1. Click **Start**.
2. In the Search programs and files bar, enter the z/OS SMB server's computer name or domain name. For example, you could enter `\\ZOSS1`.

Troubleshooting

If you cannot locate the SMB after you have tried the appropriate method described in the previous sections for your Windows client, you can use one of the following alternate methods.

- If you know its computer name, you can contact the SMB server by using the Map Network Drive function.
- Windows clients can contact the SMB server by using its DNS domain name in the **`net use`** command. Some Windows systems may also require an LMHOSTS file.

For more information, refer to "Using SMB server shared directories" on page 75.

Chapter 10. Accessing data

When your PC client has located the SMB server, it can then access the shared directories to read and write file data on the z/OS system. You can use shared directories and access different types of data.

Using SMB server shared directories

This section discusses how you can access shared directories on the SMB server from the supported SMB clients (Windows and Linux). For the complete list, refer to "Supported SMB clients" on page 3.

Note that if you are using clear text passwords in the SMB server (**_IOE_SMB_CLEAR_PW=REQUIRED**), you might need to update your Windows registry. See "Checking whether the client can communicate" on page 184.

Note: On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Windows 7 and Windows Vista clients

For Windows 7 Professional/Enterprise/Ultimate or Windows Vista Business/Enterprise clients, use one of the following methods to access shared directories on the SMB server. You might find it easier, however, to work with logical drive letters, as opposed to UNC mapping.

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

Mapping shared directories to logical drives

You can map an SMB server shared directory to a logical drive. Follow these steps:

1. Click **Start**.
2. Click **All Programs** on Windows 7 or **Programs** on Windows Vista.
3. Click **Accessories**.
4. Click **Windows Explorer**.
5. Right click **Computer** and then click **Map Network Drive**.
6. Choose the letter of a free drive for the shared directory (it might already be completed).
7. Enter the **Path** name of an SMB shared directory.

   For example, you could enter the following information, where *zOSS1* is the computer name and *myshare* is the shared directory name.

```
\\zOSS1\myshare
```

8. Click **Finish**.

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 the possibility of being logged in as DFSDFLT (that is, a guest user) if you specify DFSDFLT on the _IOE_MVS_DFSDFLT environment variable.

For example, you could enter the following command, where *x* is an available drive letter, *zOSS1* is the computer name, *myshare* is the shared directory name, and *mypassword* is your password.

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

You might also need to specify your domain name on the **net use** command, as shown in the following example:

```
net use x: \\ZOSS1\myshare mypassword /user:smbdomain\smbuser
```

Using Universal Naming Convention (UNC) mapping on Windows clients

On supported Windows clients, you can also use Universal Naming Convention (UNC) mapping to access SMB server shared directories. To do so, perform the following steps.

1. Enter the following command, where *zOSS1* is the computer name and *myshare* is the shared directory name.

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

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

   `net use`

   The following output is displayed:

```
Status                  Local                   Remote          Network
-----------------------------------------------------------------------------------------------------
OK                                              \\zOSS1\myshare    Microsoft Windows Network
```

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

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

Linux Samba clients

You can use a Linux Samba client to access shared directories on the SMB server by mapping shared directories to mount points. To do so, use the Linux **mount** command or the Samba **smbmount** command to mount shared file systems to mount points. Figure 16 on page 77 shows the use of the Linux Samba command line to mount and unmount SMB fileshares.

```
mount -t smbfs //smbserver.com/fileshare /mtpt -o username=user,password=userpw password=pw,workgroup=subdomainname

umount /mtpt

smbmount //smbserver.com/fileshare /mtpt -o username=user,password=userpw

umount /mtpt
```

Figure 16. Linux mount and unmount commands issued from a Linux UID 0 user

Accessing HFS data

The following sections discuss how to access 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. Windows PC clients are not case-sensitive (that is, they treat uppercase letters and lowercase 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 uppercase and lowercase characters.

For example, the HFS file system might contain three files with the names shown in Figure 17. Because the HFS file system is case-sensitive, these files have technically different names; as such, they represent three distinct, separate objects on z/OS.

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

Figure 17. Example of case-sensitive file names

All of the Windows PC clients that the SMB server supports are not case-sensitive. This means that the case of file names is insignificant. For example, although Figure 17 shows three files, only one is recognized. Which one depends on how the user specified the name and whether the PC client code folded the name to uppercase.

The SMB server uses the following algorithm to search for a file or directory name in HFS. (Note that many PC clients can fold a name to uppercase letters before sending the name over the network.)

- 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.

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; if it is found, 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. See *z/OS V2R2.0 UNIX System Services User's Guide* for information about 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. In Windows terminology, the closest analogy to a symbolic link is called 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. Windows applications can then refer to 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 these cases, 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, it is successful (and follows the symbolic link) if 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.

SMB restrictions

The SMB server does not support the use of the MS-DOS **move** command to move directories across z/OS UNIX files systems. However, you can use any of the following alternative methods to move directories and files across z/OS UNIX file systems.

- Issue the **copy** command to copy the source directory to the target file system; then, issue the **erase** command to delete the directory from the source file system.
- Use Windows Explorer to move directories across z/OS UNIX file systems. (Each file system must be independently mapped to a separate network drive; they must also be statically mounted and shared.) To do so, you can use either of the following methods:
  - **Copy** the directory from the source file system, then **paste** it to the target file system.
  - Drag the directory and files from the mapped source drive and then drop them to the mapped target drive. The drag and drop option will work across z/OS UNIX file systems if the target directory exists and has the same name as the source directory. If you use this method, you must specify the copy option.

  In either case, you should delete the source directory after you have copied it over to the new target directory.

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 supported data sets include:

- sequential data sets (on DASD)
- partitioned data sets (PDS)
- partitioned data sets extended (PDSE)
- Virtual Storage Access Method (VSAM) data sets, which include:
  - key sequential (KSDS)
  - relative record (RRDS)
  - entry sequence (ESDS)

RFS restrictions

RFS has many restrictions because RFS files are not hierarchical, byte stream files. Rather, they are z/OS data sets that store record-oriented data that must follow data set rules. Among the restrictions are the following (see Appendix D, "Using data sets," on page 193 for more information about accessing RFS files):

- Data set names are always uppercase.
- 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 in a data set name are more limited than file names in HFS. See *z/OS DFSMS Using Data Sets* for more information of data set naming rules.

- Use lowercase file names at the PC. Data set names are mapped to uppercase and are displayed as lowercase 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 and data set 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 (for example, Wordpad or copy) that truncate the file to zero length before rewriting a file typically 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.
- Some editors (for example, Notepad and WordPad) do not automatically put an end of line character at the end of the last line. To RFS, this is an incomplete record. When RFS closes the data set, the incomplete record will be discarded. To avoid this problem, hit enter at the end of the last line before saving the file or use an editor that automatically places an end of line character at the end of the last line (for example, Visual SlickEdit).
- SMB access is limited to RFS data sets that are no larger than 4 GB.
- The SMB server does not support DFSMS large format data sets.

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).

z/OS SMB implementation restriction for SMB digital signing

The z/OS Distributed File Service SMB server does not support server-side SMB digital signing. The determination of whether to use and enforce digital signing is performed during the initial negotiation and session setup of SMB transactions between the supported clients, the z/OS DFS/SMB server, and the Microsoft Domain controllers if pass-through authentication is configured.

If digital signing is used, attempts to connect to sessions or to access exported shares will fail with access denied or unexpected network error messages.

Refer to Microsoft documentation for information about setting the value for the digital signing option. These options can be set with the group policy editor `secpol.msc` or the `regedit` Microsoft utility.

How to determine the current settings

Use the Group Policy settings and its corresponding values to determine the settings for client-side and server-side digital signing.

Group Policy setting for client-side siding using secpol.misc:
  Microsoft network client: Digitally sign communications (if server agrees)
  Microsoft network client: Digitally sign communications (always)

Corresponding registry value for client-side signing using regedit:

```
HKEY_LOCAL_MACHINE SYSTEM CurrentControlSet Services
Lanmanworkstation Parameters Enablesecuritysignature
HKEY_LOCAL_MACHINE SYSTEM CurrentControlSet Services
Lanmanworkstation Parameters Requiresecuritysignature
```

Group Policy setting for server-side signing using secpol.msc:

Microsoft network server: Digitally sign communications (if server agrees)

Microsoft network server: Digitally sign communications (always)

Corresponding registry value for server-side signing using regedit:

```
HKEY_LOCAL_MACHINE SYSTEM CurrentControlSet Services
Lanmanserver Parameters Enablesecuritysignature
HKEY_LOCAL_MACHINE SYSTEM CurrentControlSet Services
Lanmanserver Parameters Requiresecuritysignature
```

Recommended configuration for the z/OS SMB implementation

On the Windows clients, ensure that the fields are set as follows:

Microsoft network client: Digitally sign communications (always) Disabled

Microsoft network client: Digitally sign communications (if server agrees) Either Enabled/Disabled

On the Windows 2008 server, ensure that the fields are set as follows:

Microsoft network client: Digitally sign communications (always) Disabled

Microsoft network client: Digitally sign communications (if client agrees) Disabled

Chapter 11. Accessing printers

Once your supported 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. For a summary of the supported clients, see “Supported SMB clients” on page 3. This section explains how to access the SMB server shared printers and how to add a printer from the supported SMB clients.

Mapping shared printers to logical printers on Windows clients

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

Note: On Windows clients, these steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

1. Open an **MS-DOS** window.
2. Enter the following command at the DOS prompt to get a list of printers shared by SMB servers; *zOSS1* is the computer name of the SMB server:

```
net view \\zOSS1
```

3. Choose a shared printer name from the list provided.
4. Enter the following command, 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.

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

   If no other shared file systems are attached for this Windows client, you must also specify a password.

```
net use lpt2: \\server\printshare pw /u:user /persistent:yes
```

   Windows 7 and Windows Vista clients must also specify a valid Domain name on the `/u:` option if no other shared file systems are attached.

```
net use lpt2: \\server\printshare pw /u:domain\user /persistent:yes
```

   Note also that Windows 7 clients cannot specify port LPT1 on the **net use** command.

Mapping shared printers to logical printers on Linux clients

You can map an SMB server shared printer to a logical printer on Linux clients through interactive or command-line access.

Use the following steps for **interactive access**:

1. Open a Linux command-line window.
2. Enter the following command to obtain a list of SMB server shared printers, where *zOSS1* is the computer name of the SMB server, *username* is the name of a Linux user mapped to a TSO/E user, and *password* is the CLEAR or encrypted password for the mapped user.

```
smbclient -L //zOSS1 -U username%password
```

3. Enter the following command to gain interactive access to a shared printer, where *shared-printer* is the name of the SMB server's shared printer.

```
smbclient //zOSS1/shared-printer -U username%password
```

4. Enter the following command to print a local Linux file whose fully-qualified path name is `/root/.profile`. You can use -W option to specify the workgroup, where *subdomainname* is the name of a valid subdomain.

```
print /root/.profile -W subdomainname
```

5. Enter `queue` to view the queue of print job requests.
6. Enter `quit` to exit the interactive session.

Use the following steps for **command-line** access:

1. Open a Linux command-line window.
2. Enter the following command to obtain a list of SMB server shared printers, where *zOSS1* is the computer name of the SMB server.

```
smbclient -L //zOSS1 -U username%password
```

3. Enter the following command to print a local Linux file whose fully-qualified path name is `/root/.profile` and *prt1* is the name of the SMB server's shared printer.

```
smbclient //server/prt1 -U user%pw -c "print /root/.profile"
```

4. Enter the following command to view the queue of print job requests made through the preceding print command:

```
smbclient //zOSS1/prt1 -U user%pw -c "queue"
```

To can cancel a print job, you can issue the following command.

```
smbclient //zOSS1/prt1 -U user%pw -c "cancel jobid"
```

Accessing shared printers

This section shows you how to access shared printers using the various supported Windows clients. See "Supported SMB clients" on page 3 for the complete list of these clients.

Note: On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Windows 7 clients

To access shared printers on the SMB server using a Windows 7 Professional/Enterprise/Ultimate client, perform the following steps.

1. Click the **Start** button.
2. Enter the computer name of the z/OS SMB server in the search bar.

   For example, you may enter the following command.

```
\\dcefvt8.pdl.pok.ibm.com
```

3. Double-click the found computer name.
4. You might need to enter a Windows 7 Professional/Enterprise/Ultimate user ID and password that is associated with that user's credentials.
5. If the printer you selected has been defined to the PC client, you get a list of jobs that are queued to print for that printer.
6. 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. At this step, ensure you choose **local printer** and not network print.
7. Click the **Finish** button.

Alternatively, you can issue a **net use** command from a MS-DOS command prompt window (note that port LPT1 cannot be used with Windows 7 clients). The following command shows an example of how to do so.

```
net use lpt2: \\dcefvt8.pdl.pok.ibm.com\ddecolor dfs111smb /u:SMBFVTDOM\client1
```

Windows Vista

You can access shared printers on the SMB server using a Windows Vista Business/Enterprise by performing the following steps.

1. Click **Start**.
2. Click **Search**.
3. Click **Computers or people**.
4. Click **A computer on the network**.
5. Enter the **Computer Name** of the SMB server (for example, `\\ZOSS1`).

6. Click **Search**.
7. Double-click the found computer name.
8. You might need to enter a Windows Vista Business/Enterprise user ID and password that is associated with that user's credentials.
9. Double-click a shared printer.
10. 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.
11. 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.
12. Click **Finish**.

Adding a printer

This section explains how to add a printer using a supported Windows client. You might 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.

Authentication occurs during the Add a printer wizard process. For successful authentication to complete, you might be prompted to enter a Windows user ID and password. If authentication fails, Windows issues a message indicating it could not connect to the printer and that access is denied.

Note: On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Windows 7 clients

You can add a printer from a Windows 7 Professional/Enterprise/Ultimate client by performing the following steps:

1. Click the **Start** (the "Flag" symbol in Windows 7)
2. Select **Devices and Printers**.
3. Click **Add a printer**, then select **Add a Local Printer**.
4. In the **Choose a printer port** window, select the **Create a new port** radio button (if you have not already created a new local port name for this shared printer).
   a. From the drop-down box, select **Local Port** and click the **Next** button.
   b. In the **Port Name** dialogue box that opens, enter the SMB server host name and shared printer name in the following format:

```
SMB host name/shared printer name
```

5. In the **Install Printer Driver** dialogue box, choose a brand and model that matches (or most closely matches) the type of printer mentioned in the **smbtab** file for the SMB shared printer you are trying to access. The printer information will be in the remarks (comments within double quotes) on the record.
6. Enter a new printer name in the next dialogue box.
7. Select **No** for shared printing.
8. Click the **Finish** button.

Windows Vista clients

You can add a printer from Windows Vista Business/Enterprise client by performing the following steps:

1. Click**Start**.
2. Select **Control Panel**.
3. Select **Printers and Faxes**.
4. Click **Add a printer**.
5. Click **Next**.
6. Choose the **Network Printer** or a printer attached to another computer radio button, and click **Next**.
7. Click the correct radio button to **Browse for a printer** or **Connect to this printer** (and enter the SMB printer name or the network path), and click **Next**.

   For example, you could enter the following network path, where *zOSS1* is the computer name of the SMB server and *myprt* is the name of the shared printer.

```
\\zOSS1\myprt
```

8. Select the correct radio button to determine if this will be your default printer.
9. Install the driver, if necessary.
10. Click **Finish**.

Displaying a printer queue

The following sections describe how you can display a printer queue on Windows PC and Linux clients.

Note: On Windows clients, the following steps may vary slightly, depending on how your workstation is set up (for example, whether you use the Start Menu or the Classic view).

Displaying a printer queue on Windows clients

Perform the following steps to display a Windows PC printer queue. Some of the steps vary, depending on the level of Windows running on the client.

1. Click **Start**.
2. On a Windows 7 Professional/Enterprise/Ultimate client, open the Control Panel and click **Devices and Printers**.

   For Windows 7 clients, Devices and Printers can also be accessed by clicking the Windows start “Flag” and choosing **Devices and Printers**. Then, double-click the shared printer icon.

   On Windows Vista Business/Enterprise: open Control Panel and click **Printers**.
3. Determine which printer you want to display and double-click 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 (see *z/OS Infoprint Server Operation and Administration* for more information). 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 Job ID (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. (See Chapter 6, “Mapping SMB user IDs to z/OS user IDs,” on page 37 for information about how SMB user IDs are mapped.) The JES Job ID is usually comprised of the characters “PS”, followed by a number assigned by the Infoprint Server. Print jobs submitted by using the SMB server always have JES Job IDs 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 typically contains the file name of the print request and the Owner contains the z/OS user ID of the submitter.

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 to update the window with the latest printer status. This continual query might lead to unnecessary network contention.

Displaying a printer queue on Linux clients

To display a printer queue on Linux clients, you can use either the interactive or command-line mode.

Perform the following steps to display a Linux printer queue in **interactive** mode:

1. Open a Linux command-line window.
2. Enter the following command to get a list of SMB server shared printers, where *zOSS1* is the computer name of the SMB server.

```
smbclient -L //zOSS1 -U username%password
```

3. Enter the following command to gain interactive access to a shared printer (*shared-printer* is the name of the SMB server's shared printer, *username* is the name of a Linux user mapped to a TSO user, and *password* is the CLEAR or encrypted password for the mapped user).

```
smbclient //zOSS1/shared-printer -U username%password
```

4. Enter the following command to view the queue of print job requested through the above command.

```
queue
```

5. Enter the following command to exit the interactive session.

```
quit
```

Perform the following steps to display a Linux printer queue in **command-line** mode:

1. Open a Linux command-line window.
2. Enter the following command to get a list of SMB server shared printers, where *zOSS1* is the computer name of the SMB server. The `-W` option to specifies the workgroup, where *subdomainname* is the name of a valid subdomain.

```
smbclient -L //zOSS1 -U username%password -W subdomainname
```

3. Enter the following command to view the queue of print job requested through the above command (*prt1* is the name of an SMB server shared printer, *user* is the name of a Linux user mapped to a TSO/E user, and *pw* is the CLEAR or encrypted password for the mapped user).

```
smbclient //server/prt1 -U user%pw -c "queue"
```

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 to download from Ricoh Infoprint Solutions, at z/OS Print Management Software (http://www.ibm.com/systems/z/os/zos/printsoftware/).

Part 2. SMB support reference

This section contains reference material.

- Chapter 12, “z/OS system commands,” on page 93
- Chapter 13, “Distributed File Service SMB files,” on page 99
- Chapter 14, “SMB commands,” on page 121
- Chapter 15, “Tuning and debugging guidelines,” on page 133.

Chapter 12. z/OS system commands

This section covers 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**).

You can use these commands from the operator console or from a System Display and Search Facility (SDSF) screen.

modify dfs processes

Purpose

The **modify dfs** command can start, stop, or query the status of DFS control task daemons or send a command string to the **dfskern** daemon. For additional **modify dfs** command options, see Chapter 15, “Tuning and debugging guidelines,” on page 133.

Format

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

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

modify procname,send dfskern,[reload,{print | smbmap}]
[trace,{reset | print | tsize,nnM | qtab}]
```

Parameters

procname
: The name of the DFS control task. On DFS, the default procname is **DFS**.

command
: The action that is performed on 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. See *z/OS Program Directory* for more information.
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 parameters:

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

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

reload Reload or reinitialization based on the following options:

> **print** Causes reload or reinitialization of the Infoprint Server DLL. See *z/OS Program Directory* for more information.
>
> **smbmap**
> Ensures that new SMB identity mappings are used by first eliminating any cached SMB identity mappings and then reloading the updated **smbidmap** file.

trace Reset, print, resize, or query the internal SMB trace table based on the following options:

> **reset** Resets the internal (wrap around) trace table to empty.
>
> **print** Format and print the trace records.
>
> **tsize,***nnM*
> Sets the size specification for the trace table. The variable *nn* represents a numeric multiplier and *M* represents the multipliers KB (1024 bytes), or MB (1024 x 1024 = 1 048 576 bytes).
>
> **qtab** Display the current size of the **dfskern** internal trace table.

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

unexport
The **unexport** daemon. The **unexport** server daemon permits 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**).

Using the **MODIFY** system command, you can perform the following tasks:
- Start or stop one daemon or all the daemons configured on the host.
- View the status of the DFS daemons from the operator console, using the query option
- 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. See *z/OS Program Directory*for more information about the Infoprint Server DLL. This enables or updates the Infoprint Server 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
```

The following example resets the internal trace table to empty.

```
modify dfs,send dfskern,trace,reset
```

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.)

  The *start_options* values **must** be within single quotation marks. The **START** command converts all user-supplied *start_options* values to uppercase characters, unless they are within single quotation marks.

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.

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 permits 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

Chapter 13. Distributed File Service SMB files

This section introduces the Distributed File Service SMB files, which are listed in alphabetic order. 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 typically delimited by a newline (X'15'). They can also be delimited by carriage return/line feed (X'0D15'), as is the case if they are edited from a Windows application.

Attributes file (rfstab)

Purpose

The attributes file (which is sometimes referred to as the **rfstab** file) contains tables that describe 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.

Examples

The following is an example of an attributes (**rfstab**) file; an example attributes file can be found in `/opt/dfsglobal/examples`.

```
startfileattributes:
file: b.c, b.c.d
binary, space(50,0), dsorg(ps)
file: x.y
text
endfileattributes: 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)
```

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 NFS server. For example, `_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'`, causes 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. See "Unsupported attributes" on page 106 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). For example, a **devtab** entry might display follows.

```
* 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.

A DFS attributes file can specify file attributes (creation and processing) on an individual file basis. These file attributes are specified at the beginning of the attributes file , delimited by the **startfileattributes:** and **endfileattributes:** keyword pair. Between these two keywords there can be multiple **file:** keywords. Each **file:** keyword is followed by one or more file names. The file name specified is the z/OS data set name minus the exported prefix. The attributes for this file or files are specified on subsequent lines and delimited by another **file:** keyword or the **endfileattributes:** keyword. If a file being accessed is not found in the specific file attributes section of the attributes file, its attributes are taken from the general section of the attributes file.

Related information

Commands:
  `dfsexport`
  `dfsshare`

Files:
  devtab
  dfstab
  hfsattr
  smbtab

Specifying the location of the attributes file (rfstab)

The attributes file is in 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 NFS server. For example, `_IOE_RFS_ATTRIBUTES_FILE=//'NFSADMIN.NFSS(NFSSATT)'`, causes 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. See "Unsupported attributes" on page 106 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 display as in the following example:

```
* 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.

Specifying attributes for a specific file

A DFS attributes file can specify file attributes (creation and processing) on an individual file basis. These file attributes are specified at the beginning of the attributes file, delimited by the **startfileattributes:** and **endfileattributes:** keyword pair. Between these two keywords there can be multiple **file:** keywords. Each **file:** keyword is followed by one or more file names. The file name specified is the z/OS data set name minus the exported prefix. The attributes for this file or files are specified on subsequent lines and delimited by another **file:** keyword or the **endfileattributes:** keyword. If a file being accessed is not found in the specific file attributes section of the attributes file, its attributes are taken from the general section of the attributes file.

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 that are used to define a data set when it is created. See *z/OS MVS JCL Reference* for more detailed information about data set creation attributes.

The data set creation attributes are described in Table 3; 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, therefore the file creation attributes for members are ignored.

Table 3. Data set creation attributes

Data set creation attribute	Description
<u>**blks**</u>	Specifies that disk space (see the **space** attribute in this table) is allocated by blocks, except for VSAM data sets.
cyls	Specifies that disk space (see 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(<u>**0**</u> | *quan***)**	Specifies the maximum length, in bytes, of a physical block on disk. *quan* is a number <u>**0**</u> (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 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 about data classes, see *z/OS DFSMSdfp Storage Administration*.
dir(<u>**27**</u> | *quan***)**	Specifies the number of 256-byte records that are 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 <u>**27**</u>). The maximum number of PDS members is 14,562.
dsntype(library | <u>**pds**</u>**)**	Specifies whether a PDSE or a PDS is to be created when the **`mkdir`** 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, see *z/OS DFSMS Using Data Sets*.
dsorg(*org***)**	Specifies the organization of a data set. *org* can be a physical sequential (<u>**ps**</u>) 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 in the range of 1 - 255 (the default is <u>**64**</u>); *off* is in the range of 0 - 32,760 (the default is <u>**0**</u>). 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 as a first load, and requires that the records being loaded are in ascending key sequence.
lrecl(<u>**8196**</u> | *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 <u>**8196**</u>).

Table 3. Data set creation attributes (continued)

Data set creation attribute	Description
mgmtclass(*mgmt_class_name***)**	Specifies the management class associated with the file creation. The *mgmt_class_name* must be defined to DFSMS 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 about management classes, see *z/OS DFSMSdfp Storage Administration*.
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. The *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. See 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 |` <u>**`vb`**</u> `| 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 - 32,760 (the defaults are <u>**512**</u> and <u>**4096**</u>, 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.
<u>**norlse**</u>	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 <u>**1**</u> and **3**, respectively.
space(*prim[,aux]***)**	Specifies the amount of primary and auxiliary space that is 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 <u>**100**</u> and <u>**10**</u>, respectively.
spanned	Applies to VSAM data sets only. For spanned records of non-VSAM data sets, see the entry for **recfm**, in this table. Specifies that VSAM KSDS or ESDS data sets can contain records that span control intervals (spanned records),
<u>**nonspanned**</u>	Applies to VSAM data sets only. For spanned records of non-VSAM data sets, see the entry for **recfm**, in this table. Specifies that data sets do not have spanned records.

Table 3. Data set creation attributes (continued)

Data set creation attribute	Description
storclas(*class_name***)**	Specifies the storage class associated with the file creation. The *class_name* must be defined to the DFSMS before it can be used by the client. For more information about storage classes, seer *z/OS DFSMSdfp Storage Administration*.
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. You cannot create or access tape data sets on a z/OS host using the SMB server. You also 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 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 Table 4; defaults are underlined. The administrator can override the default processing attributes in the **devtab** entry for the file set. The client user cannot override processing attributes.

Table 4. 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**. • End-of-line specifiers (**lf, cr, lfcr, crlf,** or **noeol**) are used to indicate the logical record boundary.
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 about the text mode, see the **text** option of "devtab" on page 106.
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. **Note:** This is ignored for Windows clients.
executebitoff	Turns off the execute bits in user, group, and other for the mount point's files. **Note:** This is ignored for Windows clients.
fastfilesize	Causes the SMB server to approximate the file size. For more information, see "Handling of the file size value" on page 208.
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, see "Using fastfilesize to avoid read-for-size" on page 210.

Table 4. Processing attributes (continued)

Processing attribute	Description
mapleaddot	Turns on mapping of a single leading **.** from a client file name to a leading $ on z/OS systems. This option is typically 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 systems.
maplower	Turns on mapping of lowercase file names to uppercase when accessing files on z/OS systems, and back when sending to the network. This option is typically enabled for access by PC, AIX, or z/OS UNIX clients.
nomaplower	Turns off mapping of lowercase file names to uppercase and back when using files on z/OS systems.
setownerroot	Sets the user ID in a file's attributes to `root`.
setownernobody	Sets the user ID in a file's attributes to `nobody`.

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 about the text mode, see the **text** option of "devtab" on page 106.

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.

- 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.
- When the **noretrieve** attribute is active, the server does not recall the file, and can return a message that device is not available after a lookup, an rdwr, or a create request for a file. For more information, see "Retrieve attributes."

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 can 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 Table 5. Site attributes can be overridden in the **devtab** entry for the file set.

Table 5. Site attributes

Site attribute	Description
bufhigh(*n***)**	Specifies the maximum size (in bytes) of allocated buffers before buffer reclamation (see 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 typical 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.**

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 file system, 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 number 1 - 65535. Specifying `hfs` (Hierarchical File System) or `rfs` (Record File System) determines the type of file system. `hfs` is the default.

  `hfs` includes file systems of type HFS, zFS, TFS, and AUTOMNT. If you are in a sysplex with shared file system, 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 file system, SMB support of zFS is limited to zFS compatibility mode file systems.

  If you do not want the *hfs-file-system-name* to be folded to uppercase, put the name within double quotation marks; for example, `"/tmp"`. This is typically appropriate for file systems of type TFS and AUTOMNT. It might 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 uppercase, for example, ishell). To determine if a mounted file system's name includes lowercase characters, use the **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.

  Note: If you export a PDS or PDSE rather than a data set name prefix, the data set will remain allocated for the entire time it is 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 the appropriate **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, see *z/OS Distributed File Service zFS Administration*. For information about allocating and mounting HFS file systems, see *z/OS UNIX System Services Planning*. 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). See "Examples" on page 109 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 as follows:

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'5A'-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. See "Attributes file (rfstab)" on page 99 for more information about 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 first 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
```

The next examples show 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.

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 be 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 display 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, therefore you should use 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.

File set ID
The unique file set ID number associated with the HFS or RFS file set. File set ID numbers are represented as two positive integers separated by a pair of commas. For example, the file set ID number of the first file set is **0,,1**. When specifying a new file set, increment the file set 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 file set ID in the **dfstab**.

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
```

The **dfstab** entry does not distinguish 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.) See Chapter 3, "SMB post installation processing," on page 13 for information about process home directories. The **envar** file is read during process initialization; each environment variable specified in the 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=value* statement or the *value* and the equal sign that separates them.

An environment variable:
- Cannot be longer than 4096 characters
- Must have an =
- Must begin in column 1
- 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. See Table 9 on page 163, for information about all the environment variables supported for SMB processing.

The **envar** file cannot contain any blank lines between environment variables. If blank lines are found when the **envar** file is parsed, ABEND U4093 will occur and the initialization for that daemon will fail. If you want to have a line between entries, you can use the # symbol (comment) at the beginning of a line and then enter an environment variable on the next line.

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 permitted.

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 about **devtab**, see "devtab" on page 106, and for the **_IOE_HFS_TRANSLATION** environment variable, .

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** server daemon is the SMB file and print server process.

export The **export** server daemon permits 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.

Do not use this configuration type for the **export** process. This process executes one time and then stops. The **export** process must not be automatically restarted after 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 can 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 can be manually started by using the **MODIFY** system command. 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*.SIEALNKE (where *xxx* is installation dependent) data set created during installation (for more information, see *z/OS Program Directory*. 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 runtime options). This is also called the argument list. Any runtime overrides such as storage specifications and redirection of output can 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 and print server. There are no options that need to be overridden for this process for SMB file and print serving.

export The **export** daemon permits exporting of HFS file systems. See *z/OS Program Directory* for information about options available for this process.

Additional special parameters that control restart and timeout intervals might 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. Note that this should be entered on one line, even though multiple lines are used in this example.

```
dfskern CONFIGURED=Y LMD=IOEDFSKN ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern',
'LANG=En_US.IBM-1047'),HEAPP(ON)/-mainprocs 7
>DD:dfskern 2>&1" RESTART=300 TIMEOUT=300
```

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. The home directory is identified as **_EUV_HOME=/opt/dfslocal/home/dfskern**and the local code page is specified as LANG=En_US.IBM-1047 (which is the default). An Language Environment 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.

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.

```
DFSKERN  CONFIGURED=Y LMD=DFSKERN ARG="ENVAR('_EUV_HOME=/opt/dfslocal/home/dfskern')/
 >DD:DFSKERN  2>&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:UNEXPORT 2>&1"    RESTART=300 TIMEOUT=300
```

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 file system 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 can be created as a z/OS UNIX file, a sequential data set, or a member of a partitioned data set. 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 can 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.

One of the following entries can also be in **smbidmap**:

Entry	Details
* or Domain/* =	If = is used in the second line, the z/OS user ID is used exactly as is.
* or Domain/* <	If < is used in the second line, the z/OS user ID is used in lowercase.
* or Domain/* >	If > is used in the second line, the z/OS user ID is used in uppercase.

Note: If a Domain/* entry is used (the Domain from the SMB matches, but no Domain/SMB-user-ID was found in the smbidmap file), any SMB-user-ID from that Domain is used as the z/OS user ID.

If no z/OS user ID can be determined (because the SMB user ID cannot be found in the **smbidmap** file) and the SMB user ID (in the SMB) is eight characters or less, the SMB user ID (passed in the SMB) is used as the z/OS user ID. If you want, this can be the only entry in **smbidmap**. This is also relevant for use with &USERID in the smbtab.

The order of search in the **smbidmap** file for a matching domain and SMB-user-ID is:
- Domain/user ID
- Domain/*
- User ID
- *

See “Working with automounted file systems and home directories:” on page 50 and “Recommended technique for PC user access to automounted home directories” on page 51for information about &USERID.

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
```

You can put comments in **smbidmap** file. Use a semicolon (;) to begin the comment. (The semicolon is not valid as an SMB-user-ID nor as a z/OS-user-ID.) Comments can begin in any column. After the semicolon, all data on the line is ignored. The following statements are examples of comment usage.

```
; this whole line is a comment
usera ; the userid is usera and the rest of the line is a comment
abc#def; the userid is abc#def and the rest of the line is a comment
```

Implementation specifics

The **smbidmap** file is stored as an EBCDIC file in HFS, or a sequential data set or a member of a PDS.

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 within quotation marks; 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 permitted. 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 in the range of 0 - 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 be 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 (see "dfstab" on page 110). The directory path name can be up to 1024 characters. It must be within quotation marks if it contains embedded blanks. For HFS, you might want to export HFS file systems below this directory in order to permit all path names below this directory to be accessible by this share or you might want to use dynamic export. See "Dynamic export for HFS" on page 48 for information about using the dynamic export capability of the SMB server.

If you are using dynamic export, you can specify a special keyword in the **Directory path name** field. The keyword is **&USERID**. It represents the PC user's z/OS user ID. It permits a single **smbtab** entry to mean a different directory depending on which user connects to the shared directory. See "Recommended technique for PC user access to automounted home directories" on page 51 for more information about 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 within quotation marks. 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. See *z/OS Infoprint Server Operation and Administration*.

Printer type
For shared printers, specifies the type of printer. This can be the name of a printer type supplied by Windows. See *z/OS Infoprint Server Operation and Administration*.

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 that 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"
```

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

Files:
devtab
dfstab

Chapter 14. SMB commands

This section provides an alphabetical listing of all relevant SMB commands.

These commands are issued from the z/OS UNIX environment. z/OS UNIX users should have the following line in their `.profile` file in their home directory.

```
export _EUV_AUTOLOG=NO
```

Alternatively, the preceding line can be placed in the `/etc/profile` file. This causes it to take effect for all z/OS UNIX users.

dfsexport

Purpose

A z/OS UNIX command that exports (or unexports) HFS and RFS file systems. This action makes underlying file systems available so that directories contained in them can be shared (see *z/OS Program Directory*).

Format

```
dfsexport [{-all | -aggregate name | -filesystem name}] [-detach]
 [-verbose] [-force] [-type] [-level] [-help]
```

Options

-all
Specifies that all HFS and RFS file systems that are 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.

-aggregate *name* | **-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.

If you enter this command in TSO/E, the name parameter must be within quotation marks ("").

-detach
Used with the `-all` option, specifies that the file systems indicated with the command's other options are to be detached (no longer exported), which makes the corresponding shares unavailable to Windows clients. Use `-all` or `-filesystem` with this option to indicate that 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.

-force This option is no longer supported; if specified, it is ignored.

-level Prints the level of the **dfsexport** command. This information is useful when you are diagnosing a problem. All other valid options that are specified with this option are ignored.

-type Specifies that only aggregates or file systems whose file system types match the type that is specified with this option are to be exported. The SMB server only supports `-typeufs`. Use this option only with the `-all` option; it is ignored if it is specified without the `-all` option.

-verbose
Directs the command to report on its actions as it executes.

-help Prints the online help for this command. All other valid options that are specified with this option are ignored.

Usage

The **dfsexport** command exports underlying HFS and RFS file systems so that directories can 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 that are based on the values that are 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 that is 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 exported.

Issuing the **dfsexport** command with no options lists the file systems that are 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, see “ioepdcf” on page 114 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.

Before 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 if character data translation should occur when the data is read or written by PC clients. For more information, see "devtab" on page 106.
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 or BPX.SUPERUSER authority.

Cautions

Before using the `-detach` option with this command, make sure that 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 that is 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 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
```

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
```

The following examples illustrates the steps that are needed to detach a file system named OMVS.PRV.DCESVSR.HOME. You must first find the file system ID value followed by its device name.

In this example, OMVS.PRV.DCESVSR.HOME was exported during the server initialization. To find the file system ID value, issue the following MODIFY command. The file system ID is listed in the Aggr # column of the output:

```
F DFS,SEND DFSKERN,QUERY,FILES
```

Output:

```
Fileset Name            Aggr #  Flg     SMBs       Local
--------------------    ------  ---  ---------- ----------
PLEX.LOCAL.ROOT            101  AME           6          0
OMVS.PRV.DCESVSR.HOME      110  AME           1          0
```

The file system ID value is 110. Next, determine the device name that matches this value. Use the output from the **dfsexport** command to find the device name. For example:

```
/SYSTEM/home/susvsr/> dfsexport
```

Output:

```
IOEX18114I Dfsexport: /dev/ufs1, ufs, 101, 0,1700
IOEX18114I Dfsexport: /dev/ufs6, ufs, 110, 0,1728
```

Using the file system ID value of 110, locate the line from the **dfsexport** command output. The device name is the value following the Dfseport: **keyword**. In this case, the device name is `/dev/ufs6`.

To detach the file system OMVS.PRV.DCESVSR.HOME, reissue the **dfsexport** command with this device name along with the detach option.

```
dfsexport -filesystem /dev/ufs6 -detach
```

It might be possible that the file system to be detached has been dynamically exported. Dynamically exported file systems do not have entries in the `dfstab`, `devtab` and `smbtab` files. The file system ID and device name are assigned as part of dynamic export process. Typically the file system ID values start at 100000 and are incremented by 1.

Query of a dynamic exported file system:

```
F DFS,SEND DFSKERN,QUERY,FILES
```

Output:

```
Fileset Name           Aggr #  Flg     SMBs       Local
--------------------   ------  ---  ---------- ----------
PLEX.LOCAL.ROOT           101  AME         14          0
OMVS.PRV.DCESVSR.HOME  100000  AME          5          0
```

From the preceding output, the file system ID value is 100000. Next, determine the device name that matches this value.

```
/SYSTEM/home/susvsr/> dfsexport
IOEX18114I Dfsexport: /dev/ufs1, ufs, 101, 0,1700
IOEX18114I Dfsexport: /dev/ufs10000, ufs, 100000, 0,100000
```

For this dynamic exported file system, the device name is `/dev/ufs10000`.

To detach this file system issue:

```
dfsexport -filesystem /dev/ufs10000 -detach
```

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 **dfscntl** DFS control task. **dfscntl** and its child processes are controlled on DFS by the MODIFY system command. For more information, see "modify dfs processes" on page 93.

After **dfsexport** exports the file systems 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 can be issued from TSO/E or a z/OS shell.

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 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.

If you want to export zFS file systems in a shared file system environment, the SMB server can check the zFS file system configuration to determine whether zFS is running sysplex-aware or non-sysplex aware and only export non-sysplex aware file systems. To check whether zFS is running sysplex-aware, use the **zfsadm configquery -sysplex_state** command. For more details, see Chapter 7, "Sharing files," on page 41.

Related information

Files:
  `devtab`
  `dfstab`
  `smbtab`

dfsshare

Purpose

A z/OS UNIX 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. Some share names cannot be specified on z/OS UNIX commands or must be enclosed in single quotation marks. A share name that begins with dollar ($) must be enclosed in single quotation marks.

-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, see “ioepdcf” on page 114 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 or BPX.SUPERUSER authority.

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.

To export an HFS or zFS 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`

dfssyntax

Purpose

The **dfssyntax** command is an SMB command that checks the syntax of an environment variable file.

Format

```
dfssyntax [filename]
```

Options

filename
The name of the environment variable file that will be checked for syntax errors.

Usage

The **dfssyntax** command performs automatic syntax checking on an environment variable. If no other environment variable file is specified, it checks the default file */opt/dfslocal/home/dfskern/envar*. The **dfssyntax** command writes messages to the console explaining the line number where the syntax mistake is located and any other problems it finds. Examples of syntax errors include blank lines, wrong environment variable name, wrong value for environment variable, trailing blanks, and others.

Privilege required

No privileges are required.

Related information

Files:
  envar

Command:
  dfssyntax

smbpw

Purpose

A z/OS UNIX 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 lowercase. It is folded to uppercase 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 cancel out of the **smbpw** command and are left in hidden mode, you can reset your terminal to make input invisible by usage of the **stty -sane** command. See *z/OS UNIX System Services Command Reference* for additional information about the **stty** command.

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 that the passwords are not displayed when they are entered at the prompt. If you use a shell metacharacter (for example, $) in your password and you specify it on the command line (as opposed to allowing **smbpw** to prompt for the password), you must use the escape character to keep the shell from interpreting it. For example, a password of `p$w` must be specified: `smbpw p\$w p\$w` For more information about shell metacharacters, see *z/OS V2R2.0 UNIX System Services User's Guide*.

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 should put the following line in their **.profile** file in their home directory: alternatively, the line may be placed in the `/etc/profile` file.

```
export _EUV_AUTOLOG=NO
```

Chapter 15. Tuning and debugging guidelines

This information is provided for administrators of the z/OS Distributed File Service SMB server; it describes the SMB tuning and debugging guidelines. The examples presented here are for illustrative purposes only; it is normal for the output of some reports to wrap.

As Figure 18 shows, like most file system servers, SMB is dependent on many factors. If problems arise, SMB provides diagnosis information to help determine the sources of bottlenecks or the effect of a hardware or software change on SMB performance. SMB also provides environment variables and the output of QUERY commands. The default settings are sufficient for most installations.

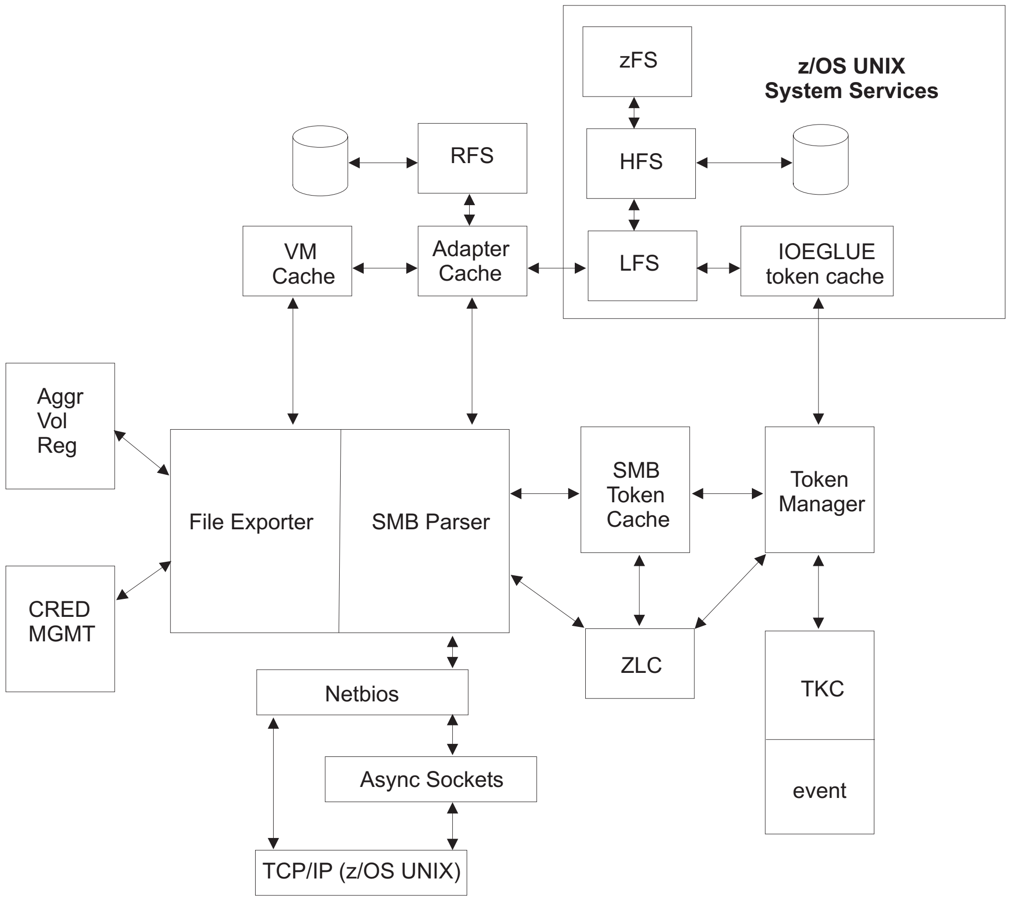


Figure 18. SMB server components

Tuning considerations and recommendations

z/OS Distributed File Service SMB server provides a **QUERY** command that provides information that is useful for gauging SMB performance and determining the effect of a change to a tuning option. In this document, tuning of the SMB server means:

- Determining the size of thread pools responsible for handling work
- Determining the amount of storage to reserve for various types of caches

Many of the SMB default settings or values work for most installations, but there are some defaults that might not be sufficient for larger installations. The most important tuning options are described in the following sections.

Threading

If the SMB server has a unit of work that needs to be dispatched on a processing thread and there are no available threads, that work needs to be placed on a queue to wait for an available service thread. The queue wait time increases server response time, therefore ensuring that the SMB server has enough defined processing threads. Distributed File Service default values for the number of processing threads is sufficient for smaller installations, but might be inadequate for larger production use. Each processing thread requires a certain amount of storage, therefore increasing the number of threads increases the storage requirements for the SMB server.

The following environment variables provide tuning options that control threading.

_IOE_SMB_MAIN_POOL
This **dfskern** environment variable controls the number of threads available to process incoming SMBs for the file server. This is an important tuning option for customers running SMB workloads.

_IOE_RFS_WORKER_THREADS
This **dfskern** environment variable controls the number of threads available to process RFS file system data set open or close requests. This is important for customers using RFS on a larger scale.

Caching

The following environment variable provides tuning options for disk caching.

_IOE_VM_CACHE_SIZE
This environment variable indicates how much file data to cache for SMB servers using disk caching. This is applicable to all server workloads, SMB, and all file systems such as zFS and HFS.

bufhigh
This rfstab (attribute file) site attribute indicates how much data to store in the RFS data set cache for RFS workloads. High RFS file read/write activity might require an increase in the size of this cache. See Table 5 on page 106.

I/O balancing

There are no server tuning options that can be used to balance I/O among disks or networks. That is a function of the physical placement of the hardware and where file systems reside. However, the SMB **QUERY** commands can possibly point out a bottleneck in a particular disk or the overall network. I/O response times reported by SMB might show the effect of a bottleneck or the effect of the removal of a bottleneck.

QUERY command

The SMB **QUERY** command is useful for debugging problems and capturing performance information. The output of the **QUERY** command goes to the system log and the job output.

The syntax of the command follows; also, see "modify dfs processes" on page 93 for related information.

```
modify dfs,send dfskern,query,<report>
```

Reports available for SMB

The following SMB query reports are listed in alphabetical order. References to z/OS UNIX physical file system (PFS) in this section include zFS, HFS, and TFS.

ADAPTERS
Shows the storage usage and performance of the SMB file server file status, directory contents, file name lookup, and file security cache used by the PFS and RFS. It also shows the number of calls made by the SMB protocol to the PFS and RFS and the average response time of each call. "Diagnostics for file system cache" on page 152 shows sample output.

ALL Shows every report described in this section.

FILESETS
Queries the fileset information. As shown in the following example, it includes the fileset name (file system), aggregate ID, flag field (A = attached, M = mounted, E = exported), number of SMB's, and number of local accesses.

```
modify dfs,send dfskern,query,filesets

Fileset Name                                  Aggr #  Flg     SMBs       Local
--------------------------------------------  ------  ---  ----------  ----------
SUIMGKO.PRIVATE.FVT2.TESTSHR3                    103  AME        6184          46

SUIMGKO.PRIVATE.FVT2.TESTSHR4                    104  AME        1837          12

OMVS.PRV.DCEIMGKO.HOME                            55  AME           1           0

SUIMGKO.PRIVATE.FVT.TESTSHR1                     101  AME           1           0

SUIMGKO.PRIVATE.FVT2                              67  AME           1           0

SUIMGKO.PRIVATE.SMBC1                            151  AME           1           0

SUIMGKO.PRIVATE.FVT                               66  AME           1           0

SUIMGKO.PRIVATE.FVT.TESTSHR2                     102  AME           1           0

SUIMGKO                                          110  A             1           0

G1SVT1                                           112  A             1           0
```

LEVEL
Queries the SMB service level. As the following example shows, it includes the module prefix, z/OS version and release identifier, service level, and FMID.

```
modify dfs,send dfskern,query,level

IOEN00195I DFSKERN: z/OS    Distributed File Service
Version 00.00.00 Service Level 0000000 0000000
Created on DAY DATE TT:TT:TT TIME ZONE YEAR.
IOEN00119I DFSKERN: SEND command - QUERY,LEVEL completed successfully.
```

LOCKING
Shows the SMB lock facility statistics. It includes the number of lock waits, the average lock wait time, and the time threads sleep waiting for certain specific events. Figure 27 on page 159 shows an example.

RFS Shows the access method counts and response times for the RFS physical file system. See "Diagnostics for RFS" on page 156 for an example.

SERVTHREADS
Queries the running service threads. As shown in the following example, it includes the thread ID (TCB), thread start time, and thread run time.

```
modify dfs,send dfskern,query,servthreads

                  SMB Service Threads
                  -------------------

Coordinate Universal Time (UTC) is Mon Aug 07, 2006 13:09:54.452857

TCB           Thread Start Time (UTC)      Run Time (sec)
--------  -------------------------------- -------------------
007C9968  Mon Aug 07, 2006 13:09:54.215317            0.237540
007C9E88  Mon Aug 07, 2006 13:08:55.643607           58.809250

IOEN00119I DFSKERN: SEND command - QUERY,SERVTHREADS completed successfully.
```

SESSFS
Queries the SMB file systems per session. It includes the computer name, number of open files, socket number, IP address, SMB user ID, username, number of SMB requests, and file system names with open files for each session.

```
f dfs,send dfskern,query,sessfs

                      DFS/SMB Filesystems Per Session
                      -------------------------------
Number of active SMB clients = 4

Computer Name   Open Files Socket   IP Address      Uid  Username        Requests  Filesystem
--------------- ---------- -------- --------------- ---- --------------- --------- -----------------------------
SMBCLIENT2               0       24   127.0.0.1     001E TTT2                    1
1234SMBCLI4              1       26   127.0.0.1     0014 TTT1                  517
                                                                                    ABCDEFG.PRIVATE.IAM.TESTSHR1
1234SMBCLI4              4       10   127.0.0.1     0001 JUS1               398228
                                                                                    ABCDEFG.PRIVATE.IAM.TESTSHR1
                                                                                    ABCDEFG.PRIVATE.IAM.TESTSHR2
SMBCCLIENT1              0       23   127.0.0.1     0018 JUS2                  333
                                                                                    ABCDEFG.PRIVATE.IAM.TESTSHR1
IOEN00119I DFSKERN: SEND command - QUERY,SESSFS completed successfully.
```

SMBCOMM
Shows the TCP/IP asynchronous socket calls made by the file server for SMB client communications and the amount of queuing for incoming SMB requests. See Figure 23 on page 145 for an example.

SMBINT

For the SMB file server, this report shows the number and type of remote Windows client SMB file requests made to the server, the average file server response and CPU times to process those requests, and the TCP/IP send and receive times that can be used to determine the network overhead for file server response times. See Figure 21 on page 143 for an example.

SMBMAP

Queries the SMB map table and displays the current mappings of SMB user to z/OS user.

SMBPRT

Shows the number and type of calls made to the Infoprint Server and the average response time and CPU time of the Infoprint Servers per call. This is used to determine the performance of the Infoprint Server. See Figure 22 on page 144 for an example.

SMBSESS

Queries the SMB sessions. As shown in the following example, it includes the computer name, number of open files, socket number, IP address, SMB user ID, username, and number of SMB requests for each session.

```
f dfs,send dfskern,query,smbsess

                       DFS/SMB Session Information
                       ---------------------------

Number of active SMB clients = 4

Computer Name   Open Files Socket   IP Address      Uid  Username        Requests
--------------- ---------- -------- --------------- ---- --------------- ---------
SMBCCLIENT1ABCD          2       10   127.0.0.1     00CD TTT1                   65
1234SMBCLI4              2       22   127.0.0.1     005E TTT5                 2191
1234SMBCLI4              0       20   127.0.0.1     0186 STU1                25766
JUSER                    1       23   127.0.0.1     01BF TSU2                   73

IOEN00119I DFSKERN: SEND command - QUERY,SMBSESS completed successfully.
```

SMBUSERIDSHARES

Queries the automounted file systems. It includes the auto-mounted file system name, the z/OS user ID that was used to create the directory, and the number of users accessing the file system.

```
f dfs,send dfskern,query,smbuseridshares

        SMB Dynamic UserID Shares

Name/UserID               Users
--------------------  ----------
MYHOME/suimgko                 1

IOEN00119I DFSKERN: SEND command - QUERY,SMBUSERIDSHARES completed successfully.
```

SMBTKC

Shows the storage usage and performance of the SMB token cache used to obtain tokens for remote SMB clients. It also shows SMB opportunistic lock callbacks and average callback response time. See Figure 25 on page 148 for an example.

STORAGE
Shows the total storage used by the DFS file server. It includes storage obtained from the Language Environment heap, storage obtained from MVS subpools, and TCB owned storage. See Figure 28 on page 160 for an example.

THREADS
Queries currently running threads. It includes the thread ID (TCB), the stack the thread is running on, the stack size, the routine that is running, the state of the thread, and the offset within the routine. Figure 19 on page 140 shows an example.

TKM Shows the token manager statistics for the SMB file server. It includes the size of the token manager cache and the token manager request rates. Figure 24 on page 147 shows an example.

VM Shows the storage usage and performance of the DFS file data cache that is stored in virtual memory. It is used to reduce disk I/O rates and physical file system calls. See Figure 26 on page 150 for an example.

VNODE
Queries the vnodes. It includes the maximum number of vnodes, number of vnodes in use, number of root vnodes, and number of directory vnodes. These fields are not mutually exclusive.

```
modify dfs,send dfskern,query,vnode

 VNODE Statistics
--------------------

 Max vnodes:          6144
 Vnodes in use:       16
 Root vnodes:         12
 Directory vnodes:    17
--------------------

IOEN00119I DFSKERN: SEND command - QUERY,VNODE completed successfully.
```

VOLREG
Queries the file system information in the volume registry. It includes the volume ID to file system name association and one the following status indicators:
- incorrect association
- exported
- not exported

RESET command

The **RESET** command clears the statistics monitored by the SMB file server. The **QUERY** command always displays the statistics since the server startup or since the last **RESET**. This allows an administrator to query performance for a given time period, such as during peak usage times.

The syntax for the server is as follows, where *report* is one of the following reports.

```
f dfs,send dfskern,reset,<report>
```

- ADAPTERS
- FILESETS
- LOCKING
- RFS
- SMBCOMM
- SMBINT
- SMBPRT
- SMBTKC
- STORAGE
- TKM
- VM

For example, the following command would clear the statistics kept for the STORAGE report.

```
f dfs,send dfskern,reset,storage
```

TRACE command

Restriction: You should only change tracing options under the advisement of your IBM service representative.An important aspect of SMB problem determination is its tracing capability. SMB has an internal (wrap around) trace table that is always tracing certain events. The initial size of the trace table is controlled by the **_IOE_TRACE_TABLE_SIZE** environment variable, but can be adjusted using the *tsize* subcommand on the **TRACE** command. For complete details, see "modify dfs processes" on page 93.

SMB hang detection

Restriction: Hang detection environment variables should only be changed under the advisement of your IBM service representative.

The SMB hang detector monitors the current location of the various tasks that are processing in SMB. At intervals, which are set through the **_IOE_HANG_DETECT_INTERVAL** environment variable, the hang detector scans the current user requests called into SMB. When the hang detector determines that a thread has remained in the same location for a predefined period of time, it flags the task as a potential hang and produces a dump to preserve problem data.

Occasionally, SMB hang detection might produce false hang dumps and an actual hang might go undetected. If **dfskern** issued a hang detection dump, or you suspect that **dfskern** is hung for some other reason (users are not getting a response), you can use the following procedure to determine if you have a hung thread.

1. In **dfskern**, issue the **f dfs,send dfskern,query,threads** command at 30 second intervals for two to three minutes and compare the output.
2. Compare tasks that remain in the same status (tcb,stack,routine,state,offset) across several different instances of **query,threads** output. If an entry is identical in each of **query,threads** output, there is presumably a hang.

   Figure 19 on page 140 shows some typical threads in wait for work states that are common to see persist across multiple instances of **query,threads** output:

```
f dfs,send dfskern,query,threads

            Currently Running Threads
            -------------------------
TCB       Stack     Size      Routine              State
--------  --------  --------  --------             --------
006D0370  1440FC20     77824  async_service_thread  WAITECB   off=F366CE
D0  rtn=async_service_thread
006BEE88  1459AC20      8192  keep_alive_thread     SLEEP     off=00E8
rtn=keep_alive_thread
006B6D90  0C4CEC20      8192  tpq_HelperThread      SLEEP     off=0E6E
rtn=tpq_HelperThread
006CEBF8  0CE2C210     16384  ioeglue_service_thre  WAITECB   off=041C
rtn=ioeglue_service_thread
006D3738  0CE6E5F0      8192  tkm_TokenGcThread     SLEEP     off=0C30
rtn=tkm_TokenGcThread
006CDE88  1363EC20      8192 ioe__modifyClient     RUNNING
006CD748  136CDC20      8192 ipc_listen_thread     RUNNING
006C6350  1451CC20      8192  zlsNSReqResp          SLEEP     off=044C
rtn=MsgQueue::receive(int*,int)
006B65E0  0C4DDC20      8192  tpq_DispatcherThread  SLEEP     off=108E
rtn=tpq_DispatcherThread
006D6588  0CE80208         0  main                  PAUSED    off=80001C
84  rtn=main
006CD9D8  136C1C20      8192  fileset_timeout_hand  SLEEP     off=012C
rtn=fileset_timeout_handler
006BE968  145A4C20     16384  hang_daemon           SLEEP     off=02DC
rtn=hang_daemon
006D6258  0CD9A690      8192 sleeper_thread        CONDWAIT
006CD528  13C19C20      8192  ScavengerThread       SLEEP     off=02DA
rtn=TSrvEvent::waitEvent(unsigned
006CDC68  136C6C20     16384  sync_daemon           SLEEP     off=01E2
rtn=sync_daemon
006BE138  1513DC20      8192  HostAnnounceThread    SLEEP     off=02DA
rtn=TSrvEvent::waitEvent(unsigned
```

Figure 19. Typical thread states

Figure 20 shows an entry for task 6856D8 that indicates a hang in **dfskern** when its status persists across multiple instances of **query,threads** output.

```
f dfs,send dfskern,query,threads

            Currently Running Threads
            -------------------------
TCB       Stack     Size      Routine              State
--------  --------  --------  --------             --------
006B56D8  1454EC20     77824  async_service_thread  SLEEP     off=2BF0
rtn=internal_item_obtain
```

Figure 20. Possible abnormal thread state

Conversely, if you reviewed the outputs and task 6856D8 changes status between outputs, you do not have a hang. For example, if the first **query,threads** command shows:

```
006B56D8  1454EC20     77824  async_service_thread  SLEEP     off=2BF0
rtn=internal_item_obtain
```

Then, 30 seconds later it shows the following output; this indicates the thread is progressing and not hung in `rtn=internal_item_obtain`.

```
006B56D8  1454EC20     77824  async_service_thread  WAITECB   off=WAITECB   off=F366CED0
rtn=async_service_thread
```

Data normalization

When analyzing performance information, you can normalize a performance metric with respect to the external request rate. For example, the SMBINT report records the number of SMBs received by the file server and the average SMB response time. Additionally, SMB provides many other fields that show count and average response times. When analyzing data, it is often easier to normalize a statistic per the average request response time.

For example, from the SMBINT report, the following was shown from a **QUERY** command:

```
Total SMB calls    5649396
Average DFS Response Time per SMB      2.158
```

Additionally, from the same QUERY command, the LOCK report showed the following:

```
Total waits for locks:                        1366750
Average lock wait time:          0.535 (msecs)
```

Therefore, to get average SMB lock wait time per SMB, you could perform the following calculation. In this example, SMB lock wait time is approximately 25% of the overall response time.

```
Avg. Lock Waits per SMB  =  Total waits for locks  =  0.242
                            ----------------------
                            Total SMB calls

Avg. DFS lock wait time per SMB  = Avg. Lock Waits per SMB X Average lock wait time
                                 = 0.522
```

SMB server tuning

The following sections discuss considerations for tuning the SMB server.

Workloads

File server performance is controlled by many factors. Some factors are external, such as the requests the clients present to the server. The client workload determines much of the file server performance characteristics. For example, are there many large file reads or writes? Do the clients request many metadata operations such as file renames or deletions. Other factors include the file read to write ratio and write intensity of the workload.

SMB workloads

SMB Windows clients do not cache file status or directory contents nearly as often as other clients do, therefore SMB workloads characteristically send much more file lookups and directory reads than other clients. While SMB workloads have much higher message rates per client user task or command, SMB caches data to turn around requests as fast as possible.

Tuning options

The following tuning options control the type of SMBs that flow from clients to servers. These are all environment variables that would be specified in the `dfskern` environment variable file. Refer to Appendix A, "Environment variables in SMB," on page 163 for possible values.

_IOE_SMB_RAW
: Controls if the client can use the raw mode SMBs that send data in large amounts to the server. Enabling raw mode improves large file read and write performance but the administrator can disable this if desired. Enabling raw mode allows the client or server to send or receive up to 64K of data in one packet. Additionally, the transmission is optimized. For some networks, disabling raw mode improves performance and, therefore, SMB allows the administrator to disable raw mode transmission.

_IOE_SMB_MAXXMT
: Controls how much data can be sent in one SMB. This variable controls how much data a client or the server sends in one SMB and determines the maximum size of single file read or write request. Any time a client desires to send more data in one SMB, it uses raw mode transmission for the read or write. A good test to determine the optimal settings for your network would be to make a simple test to copy large files to, or from, an idle server and track the response time. The test could be repeated with different options enabled at the server.

_IOE_SMB_OPLOCKS
: Controls if clients are allowed to request an opportunistic lock when a file is opened for read or write. Opportunistic locks allow clients to cache the file data in certain cases, cache byte range lock requests, and perform read-ahead and write-behind optimizations. It is recommended that you run with oplocks enabled, which is the default.

_IOE_SMB_PROTOCOL_LEVEL
: Determines the level of protocol negotiated with remote clients. It is the dialect that the client and server use to communicate with each other. The default is NTLMv2, which is the highest level of authentication protocol available.

Diagnostics

Figure 21 on page 143 shows sample output from the SMBINT report. Use this information to determine the amount and type of file related calls made to the SMB file server by clients. Headings that refer to HFS can refer to any z/OS UNIX physical file system, such as HFS, zFS, and others.

```
                            Average DFS Time (msecs)
SMB Call                       Count   Response Time       CPU Time
------------------------  ----------   -------------    ------------
smb_mkdir                         20         144.581          31.926
smb_rmdir                         20          96.246          19.172
smb_close                       1779           3.310           2.195
smb_unlink                       207         151.634          35.641
smb_mv                            15         133.500          14.208
smb_getattr                     1872          11.284           0.778
smb_setattr                      469           8.206           0.872
smb_write                       8644         343.186          12.129
smb_chkpath                       45           6.176           0.516
smb_readBraw                     154           6.149           4.993
smb_writeBraw                     21         274.119          63.503
smb_setattrE                     404          46.993           0.544
smb_getattrE                     445           0.556           0.445
smb_lockingX                      37           0.305           0.245
smb_openX                       2008          30.339           6.214
smb_readX                      20883           4.295           1.850
smb_trans2                      2219          10.424           0.636
T2: find first                   941          12.764           0.366
T2: find next                     20           0.173           0.151
T2: qfsinfo                      344           0.030           0.027
T2: qpathinfo                    540           5.778           0.183
smb_findclose                     20           3.793           3.670
smb_negprot                        0           0.000           0.000
smb_sesssetupX                     0           0.000           0.000

Total SMB calls      39262
Average DFS Response Time per SMB     82.411
Average DFS CPU Time per SMB           4.442

TCP/IP Session read count:       47751    Avg Time:         0.136 (msecs
TCP/IP Session write count:      39301    Avg Time:         0.252 (msecs

Average TCP/IP call time per SMB       0.418
```

Figure 21. Sample output from SMBINT report

This report shows the workload presented to the server from the SMB clients. It shows the type and performance of each SMB and an overall average. Additionally, the session read and write counts show performance of the TCP/IP calls made to read or write data on the client socket. This call time is included in the overall SMB response time but is not controllable by the SMB file server.

The administrator could use this information in the report to determine the effect of the setting of a tuning option or a hardware change on server SMB performance. Excessive TCP/IP send/receive times relative to the average SMB response time might indicate a network bottleneck. If the TCP/IP send plus receive time is greater than 50% of the overall SMB average response time, a network bottleneck is probably present.

Note: Only the SMB types that were received by the server since the last statistics reset are shown.

Figure 22 on page 144 shows sample output from the SMBPRT report. The SMBPRT report shows the calls made to the Infoprint Server on behalf of print requests received by the SMB server, the average response time, and CPU time of each call. This information can be used to determine the performance of the Infoprint Server and the portion of the SMB response time represented by Infoprint Server calls.

```
                            Average Print Time (msecs)
Infoprint Server API          Count   Failed   Response Time       CPU Time
------------------------  ----------   ------   -------------    ------------
AbortPrintFile                    0        0           0.000           0.000
BeginEnumJobs                    30        0           1.395           0.496
BeginEnumPrinters                 0        0           0.000           0.000
CancelJob                         1        0          23.560           2.879
ClosePrintFile                    3        0          31.473           2.806
CreatePrintFile                   3        0          15.222           2.475
EndEnumJobs                      30        0           0.042           0.041
EndEnumPrinters                   0        0           0.000           0.000
EnumJobs                         60        0           0.574           0.314
EnumPrinters                      0        0           0.000           0.000
GetJobInfo                        8        0           1.422           0.798
GetPrinterInfo                  110        3           1.886           0.530
HoldJob                           0        0           0.000           0.000
InitAPI                           2        1         558.078          37.522
ReleaseJob                        0        0           0.000           0.000
SetTerminationHandler             1        0           0.016           0.016
TermAPI                           0        1           0.000           0.000
WritePrintFile                   55        0           0.271           0.253

Total Infoprint Server calls        303
Average Response Time per call      5.251
Average CPU Time per call      0.684
```

Figure 22. Sample output from SMBPRT report

The calls to `CreatePrintFile`, `ClosePrintFile`, and `WritePrintFile` occur when files are being printed by PC clients on printers managed by the Infoprint Server. The response time listed is the average time the print request actually spent in the Infoprint server, processing the request, or waiting for a request to complete.

Having a PC client printer queue window open results in a high volume of `GetPrinterInfo` and `xxxEnumJobs` requests to be made to the Infoprint Server. A high value for this number can represent the fact that these windows are left open for extended periods of time. Doing this can lead to unnecessary network traffic, and network contention.

Each of these remote printer queue requests uses a thread represented by the dfskern environment variable _IOE_SMB_MAIN_POOL to execute. The system administrator should use this information to ensure that enough threads are present to process these requests along with the file related requests from PC clients. If this number is too low, it can appear that the response time of the File portion of the SMB server is slow, but actually the file related requests are being queued instead of immediately handled. See the SMBINT (Figure 21 on page 143) and SMBCOMM (Figure 23 on page 145) reports for more information about how the SMB requests are actually being received and handled by the SMB server.

SMB service threads

The SMB server has two thread pools to service incoming requests. If all threads are busy then incoming requests are queued until a service thread becomes available. There is a primary pool that is used to service most SMB requests and a secondary pool to handle callbacks from clients and is used to ensure that client callbacks have available threads to process them.

Tuning options

The following environment variables control the size of the SMB file server main and secondary thread pools. The storage requirements of an SMB processing thread is shown in the SMBCOMM report.

_IOE_SMB_MAIN_POOL
A whole number that indicates the number of threads to assign to the SMB file server main thread pool; the default is 14. This is shown as pool number 0 in the SMBCOMM report.

_IOE_SMB_CALLBACK_POOL
A whole number that indicates the number of threads to assign to the SMB file server secondary thread pool; the default is 2. This is shown as pool number 1 in the SMBCOMM report.

The default may not be good for all your workloads. If you have a large number of active clients you want to increase the size of the main thread pool. The following section provides a diagnostic aid that can help an administrator determine the optimal thread pool size.

Diagnostics

Figure 23 shows sample output from the SMBCOMM report. This report indicates the incoming requests to the SMB file server and the amount of queuing that occurs.

```
             SMB Asynchronous I/O Statistics
             -------------------------------
Service Thread Stack Size=76K   Number of Pools=2
Pool:   0   Threads:    40     Pool:   1   Threads:     2

        SRB Accepts :         0         SRB Requests :     387008
         SRB Queued :         5
          Schedules :         0              Accepts :          0
              Reads :         0               Readvs :          0
              Recvs :    386875            Recvfroms :        136
             Writes :         0              Writevs :          0
              Sends :         0              Sendtos :          0
            Cancels :         0        Cancelsockets :          0
```

Figure 23. Sample output from SMBCOMM report

The number of threads in the pool and the size of each thread's stack is shown to allow for an approximate determination of the amount of storage required if a thread pool is adjusted in size. The `SRB Accepts` field indicates the number of new clients connecting since the statistics were reset (or since startup). The `SRB Requests` field is a raw count of client requests received by the server. The `SRB Queued` field indicates the number of requests that needed to be queued due to lack of service threads. The administrator should ensure that the percentage of requests queued is not more than 5% of the total requests since this represents increased response time to process SMB requests.

Token management

The following sections discuss token management.

Central SMB token manager

The SMB file server central token manager is a cache of tokens with a fixed maximum size. The administrator can tailor the maximum size. A large amount of file access activity increases the cache of tokens. When the central token manager runs out of tokens, it performs garbage collection to reclaim a large amount of tokens from clients. For more information, see **_IOE_TKM_MAX_TOKENS** in Table 9 on page 163. It is possible to ensure garbage collection does not happen at all.

Local user glue

Local user access to file systems exported by SMB also obtain tokens to ensure remote client caches remain correct. The Distributed File Service glue code caches tokens in the UNIX System Services kernel address space. For all local user access to file systems exported by SMB, the glue code gets control and ensures needed tokens are obtained before the request is processed by the file system. If the tokens are in the cache then no call to the central token manager is needed. Administrators can set the size of this cache. When a token is needed from the central token manager a task switch is required. There are a pool of threads in the SMB file server address space reserved to process local user token requests and the administrator can set the size of this thread pool.

SMB token cache

To ensure proper file sharing semantics and client cache consistency, SMB uses token cache to obtain tokens for requests from the central token manager. Windows (SMB) clients use a form of caching called opportunistic locking. These opportunistic lock requests are mapped to SMB token requests. The server caches SMB tokens from the central token manager to reduce path lengths. Opportunistic locks may require a callback to the client. The frequency of these callbacks and their average round-trip response time are shown in the SMBTKC report. The administrator can control the size of this cache by using tuning options. Although the SMB token cache can be set larger than the central token manager cache, the SMB token cache never uses more tokens than are available in the central token manager cache. The central token manager does not allow the cache maximum to be exceeded.

Tuning options

The central token manager cache maximum size can be set by using the following tuning options For defaults, see Table 9 on page 163.

_IOE_TKM_MAX_TOKENS
: Maximum number of tokens that the central token manager can hold. The size of each token (in bytes) is shown in the TKM report.

The following tuning option controls the number of threads available to process local PFS user token requests. This is important if PFS file systems are updated concurrently from remote SMB clients and local PFS users.

_IOE_TKMGLUE_SERVER_THREADS
: The number of threads available for processing requests from local PFS users to get tokens from the central token manager.

_IOE_TKCGLUE_CACHE_SIZE
: Maximum number of files for the DFS glue token cache. When more files are cached, the need to obtain tokens from the central token manager decreases. The glue provides this cache in the z/OS UNIX Systems Services kernel address space.

The following tuning option controls the size of the SMB token cache.

_IOE_SMB_TOKEN_FILE_MAX
: The SMB token cache allows the administrator to set the number of files that can have tokens cached. Although the maximum size is allowed to be exceeded (the central token manager really determines the maximum number of tokens and files with tokens), the SMB token cache tries to reuse its internal file structures before exceeding the maximum size. Setting the SMB token cache too small is permitted, though path lengths are larger because of the execution of the file reuse logic trying to keep the number

of files or tokens cached below the maximum. The storage for an internal file and token structure is shown in the SMBTKC report (Figure 25 on page 148).

Diagnostics

Figure 24 shows sample output from the TKM report. The TKM report shows the maximum number of tokens allowed in the cache (`Max Tokens`) which is the administrator setting. The `Allocated Tokens` field indicates how many tokens are allocated and `In Use` indicates how many are in use by clients at the given time. Similarly, the `Max Files`, `Allocated Files`, and `Files in Use` indicate the file maximum, the number currently allocated in memory, and the number in use to hold client tokens. The most important fields in this report are `GC Invocations` and `GC Failures`. These numbers should be as close to 0 as possible. (0 is optimal). Do not invoke garbage collection more than a few times per day or peak period.

```
                 Token Management Statistics
                 ---------------------------
Token Ceiling:       262144       Allocated Tokens:       7168
Max Tokens:           17408       In Use:                  111
Min Tokens:             128       Free Tokens:            7057
GC Invocations:           0       GC Failures:               0
Tokens Obtained:     220035       Tokens Returned:      208916
Tokens Revoked:        2237       Revokes Refused:          95
Max Files:            18496       Files Allocated:        5120
Min Files:             3082       Files Freed:            5107
Files In Use:          3095       Files Recycled:        57964
Max Filesets:         18496       Filesests in use:         43

Token struct size: 80   File struct Size: 120   Fileset struct size: 168

IOEN00119I DFSKERN: SEND command - QUERY,TKM completed successfully.
```

Figure 24. Sample output from TKM report

Figure 25 on page 148 shows sample output from the SMBTKC report, which shows a number of statistics related to SMB token cache performance. The important fields are the `Total Callbacks To Clients` and `Average Callback Response Time`. The counts should not be much more than a few percent of the total number of SMBs that are shown on the SMBINT report. The `Files` allocated and in-use count should be examined. If these values are greater than the maximum, then the SMB token cache is low on tokens. Server response times are increased slightly because of increased pathlength.

```
                SMB Token Cache Statistics
                --------------------------

            Allocated        In-use
            ----------      ----------
Hosts             3               3
Files           190              68      (Max: 4096, Struct size: 152)
Tokens          189              67      (Struct size: 72)
Open Holds       21               0      (Oplocked: 0)
Lock Holds       21               0
Data Holds       55               1

                 Count                               Count
               ----------                          ----------
Token Obtains:        239        Token Returns:           212
Fast Revokes:         217        Slow Revokes:              0
File Opens:          1779        Compatibility:             0
Oplocks:             1779        Exclusive Grants:          0
Batch Grants:           0        Pre-CIFS Grants:        1509
Level2 Grants:          0        Failed Opens:              0
File Locks:             0        Permitted Checks:      29702
Tkset Adds:         43277        Longterm Holds:           40

Total Callbacks to clients:        72      Response Waits:         72
Average Callback Response Time: 20.335 (msecs)
```

Figure 25. Sample output from SMBTKC report

Virtual memory file cache

The SMB file server has a cache that is used to contain file data to reduce disk I/O and perform read-ahead and write-behind of file data. This cache is hereafter referred to as the virtual memory (VM) cache. The administrator can tailor the maximum number of files to cache and how much storage is used to contain the cache. This disk cache is like most disk caches. It has least recently used (LRU) algorithms used to keep to most recently used file data in memory. This cache is used for all SMB server file systems: zFS, HFS, and RFS. The following sections are guidelines for tuning this cache for the different types of workloads and file systems.

SMB workloads

Windows (SMB) clients cache directory and file status information in some cases. It performs limited file data caching but the data is not cached. Therefore, the VM cache is important to these workloads. As with all data caching systems, the base measure of performance is the file read hit ratio. This is the percentage of times that a read request is satisfied from the cache rather than requiring a disk I/O. This ratio for SMB workloads should be at least 80-90%. The VM cache provides diagnostics and tuning options that allows the administrator to determine the performance of the cache and alter the cache performance. The PFS file systems also caches data in the address space, but the VM cache is used for the PFS file systems to reduce calls and hence reduce pathlength. A production SMB file server should be run with a larger VM cache for SMB workloads than the current 1 M default.

When running large workloads from many SMB remote clients, especially if accessing (create, write, delete) the same files, it might be beneficial to change defaults for a number of environment variables. You must ensure that you have sufficient real storage so that paging is not significantly increased. For example, in a test environment for 50 SMB clients running up to 100 active sessions, generating over 2 billion SMB calls in 24 hours, the following environment variable settings showed a significant improvement:

```
_IOE_SMB_MAIN_POOL=64 (This is related to the z/OS UNIX MAXTHREADS value.)
_IOE_TKM_MAX_TOKENS=256000
_IOE_VM_CACHE_SIZE=256M
_IOE_VM_MAX_FILES=32768
_IOE_VNODE_CACHE_SIZE=18432
```

Tuning options

The basic VM cache tuning options control how much storage is used for the cache and the maximum number of files allowed to be cached. The number of files can be set large because each in-memory structure that represents a file in the VM cache is not too large. The size of each file structure is shown in the VM report (Figure 26 on page 150). For the default values of these environment variables, see Appendix A, "Environment variables in SMB," on page 163.

_IOE_VM_MAX_FILES
Maximum number of files to cache.

_IOE_VM_CACHE_SIZE
Maximum size in bytes of the VM cache. The user can prefix the size with K or M to indicate kilobytes or megabytes, respectively. The following example sets the cache size to 256 MB.

```
_IOE_VM_CACHE_SIZE=256M
```

Diagnostics

Figure 26 on page 150 shows sample output from the VM report. This report shows the performance of the virtual memory caching system since last reset or since server startup (whichever is most recent). Table 6 on page 150 describes the important fields in the report.

```
                   Virtual Memory Caching System Statistics
                   ----------------------------------------

External Requests:
------------------
Reads         37871     Fsyncs          251     Opens        119818
Writes        35362     Truncates     12723     Unmaps         3805
Asy Reads      2882     Getattrs     283760     Schedules      7610

File System Reads:
------------------
Reads Faulted      28075     (Fault Ratio 74.13%)
Writes Faulted         0     (Fault Ratio  0.00%)
Read Waits          1829     (Wait Ratio  4.83%)
Total Reads        30766

File System Writes:
-------------------
Scheduled Writes         35362     Sync  Waits         223
Error Writes                 0     Error Waits           0
Page Reclaim Writes          0     Reclaim Waits       168
Write Waits                  3     (Wait Ratio  0.01%)

File Management: (File struct size=128)
---------------------------------------
Total Files        4096     Free              3447     Pending             0
Unreferenced        649     Referenced           0     File Waits          0
Lookups     119818     Hits       80213 (Hit Ratio   66.95)
Misses       39605     Reuses         0 (Reuse Ratio  0.00)

Page Management (Segment Size = 64K) (Page Size = 4K)
-----------------------------------------------------
Total Pages        8192     Free                10     Pending             0
Unreferenced       8182     Referenced           0     Page Waits          0
Steal Invocations      15640     Steals From Self          0
Files Stolen From      40164     Steals From Others      40164
Waits for Reclaim         17
```

Figure 26. Sample output from VM report

Table 6. Sections of the VM report

Report section	Contents
External Requests	Number of requests presented to the VM cache system. The most significant fields are `Reads` and `Writes`, which represent read and write requests made to the cache.
Reads Faulted	Read miss ratio, which is the opposite of hit ratio. Obtaining the hit ratio, you subtract the miss ratio from 100%. In this case, the hit ratio would be 100-74 = 26%.
Reclaim Waits	Number of times a thread had to be suspended when reclaiming cache pages for an in-progress I/O (an I/O for file data written in write-behind mode). This number should be low relative to the number of external requests. It represents a thread waiting on disk I/O. A higher number means slower general response time
File Waits	Number of times a thread had to be suspended waiting to obtain an in-memory VM cache file structure. It means that all the VM cache files are in-use and represents an extreme shortage of VM cache file structures. The preferred value is 0.
Total Pages	Total number of pages allocated to the VM cache. A page is 4K in size. In this case, there are 8192 pages, therefore, 32M.

Table 6. Sections of the VM report (continued)

Report section	Contents
Page Waits	Number of times a thread had to wait for a free page in the cache to become available. It represents cases where there is an extreme shortage of pages in the cache. This number should be low, preferably 0, and should not be more than 1% of the total external requests.
Waits for Reclaim	The VM cache allows only one thread at a time to reclaim pages from the cache for use for a new file read or write request. If another thread requires a page and it finds another thread is already running reclaim processing, it waits. The occurrence of this depends on how many concurrent threads are running at a time and how often reclaim needs to run. This count should be low (not more than a few percent of the total number of external requests).

The administrator can use these indicators and set the cache size to obtain an optimal size of the cache for their environment. If `File Waits` is high, then the number of files should be increased by setting the _IOE_VM_MAX_FILES environment variable to a higher number. If the number of `Waits For Reclaim` or `Page Waits` is high, then the maximum storage should be increased by setting the _IOE_VM_CACHE_SIZE environment variable to a higher number. See Appendix A, "Environment variables in SMB," on page 163 for more information about these environment variables.

File system caches

Any z/OS UNIX System Services file system can share a cache that is used to store file status information, such as: owner ID, file change time, file permissions, directory contents, and directory name to vnode mappings. A vnode in SMB is much like a UNIX inode. It is the data structure used to represent a file system object, such as a directory or file. The RFS file system uses MVS access methods to manipulate data sets, while zFS is accessed through z/OS UNIX System Services. This cache hides zFS/RFS implementations that do not use the same interfaces as the rest of SMB. It also provides pathlength reduction because it reduces calls substantially to MVS access methods or zFS. The performance of this cache is essential for RFS performance.

Status

File status is cached in an extension to the vnode. By caching this status, calls are reduced to z/OS UNIX System Services or z/OS data set access methods. By caching vnode handles, the number of calls are reduced to z/OS UNIX System Services to obtain and return handles. The size of this cache is controlled by the administrator.

Permissions

The file system cache stores user permissions to file system objects. This is done by querying the permissions from SAF or z/OS UNIX System Services. By saving the permissions, calls to z/OS UNIX System Services or SAF are reduced.

Directory contents

The directory cache stores directory contents in SMB format. Use of this cache reduces lookup and directory read calls made to z/OS data set access methods or zFS.

Name lookup cache

The name lookup cache saves the directory entry name to associated vnode mappings; this reduces calls to obtain vnode handles.

Tuning options

An important tuning option for the shared SMB cache is the size of the directory cache. A larger cache allows directories to be cached and results in less calls to the associated physical file system routines. However, for large production systems, using a large vnode cache might be desirable.

_IOE_VNODE_CACHE_SIZE
: Number of vnodes provided in the cache; the default is 6144. More vnodes are provided, if needed, resulting in less calls to physical file system to read file status, file permissions, and obtaining vnode handles. The size of a vnode structure is shown in the ADAPTER report. Also see, VNODE report.

_IOE_DIRECTORY_CACHE_SIZE
: Number of 512 byte blocks that can be used to store directory contents and name to vnode mappings. The default is 2048 blocks, which is one megabyte of storage.

Diagnostics for file system cache

The following output is sample output from the ADAPTER report. Use the ADAPTER command to determine the performance of the shared PFS and RFS cache and the performance of the PFS. For an explanation of the report's contents, see Table 7 on page 154.

```
                         Adapter Caching Statistics
                          -------------------------
Vnode Cache Size = 6144 vnodes, structure size = 488 bytes
Vnode lookups = 1368121, hits = 1326732, ratio = 97%
Vnode invalidations = 41342
Directory Cache Size = 2048 (512 byte) blocks, Free = 729
Directory buffer refreshes = 0
Directory buffer reads = 5494594, hits = 5494594, ratio=100%
Get attributes calls = 2451296, hits = 2394971, ratio=98%
Name cache lookups = 5300598, hits = 5054509, avoided=246089, ratio=100%
Avoided set attributes calls = 1462
Access checks = 8145852, hits = 8063074, ratio=99%
Access cache invalidates = 82731

                            HFS Adapter Vnode Op Counts

Vnode Op                       Count   Vnode Op                       Count
------------------------  ----------   ------------------------  ----------
xoefs_hold                   1158433   xoefs_readdir                 143600
xoefs_rele                   7651388   xoefs_create                   41387
xoefs_inactive                     0   xoefs_remove                   41343
xoefs_getattr                2043702   xoefs_rename                    8684
xoefs_setattr                 316196   xoefs_mkdir                        0
xoefs_access                  519908   xoefs_rmdir                        0
xoefs_lookup                 5313507   xoefs_link                         0
xoefs_getvolume                    0   xoefs_symlink                      0
xoefs_getlength                    0   xoefs_readlink                     0
xoefs_afsfid                 8149223   xoefs_rdwr                         0
xoefs_fid                          0   xoefs_fsync                    14362
xoefs_vmread                  150510   xoefs_translate                    0
xoefs_vmwrite                 101034

Total HFS Vnode Ops   25653277                          Average HFS Time (msecs)
VFS Call                       Count   Response Time       CPU Time
------------------------  ----------   -------------    ------------
v_access                       82778           1.443           0.021
```

```
v_lookup                       0          0.000          0.000
v_get                          0          0.000          0.000
v_getattr                  56325         18.862          0.024
v_setattr                 356120          7.694          0.065
v_create                   41387         41.396          0.196
v_link                         0          0.000          0.000
v_mkdir                        0          0.000          0.000
v_rdwr                    265906         49.819          0.171
v_readdir                      0          0.000          0.000
v_readlink                     0          0.000          0.000
v_rel                      41342          3.549          0.041
v_remove                   41343         26.775          0.154
v_rename                    8684         28.960          0.349
v_rmdir                        0          0.000          0.000
v_rpn                          0          0.000          0.000
v_export                       0          0.000          0.000
v_symlink                      0          0.000          0.000
v_fstatfs                  10140          0.059          0.041

Total VFS calls                 904025
Average HFS Response Time per Call        22.553
Average HFS CPU Time per Call         0.101

                         RFS Adapter Vnode Op Counts

Vnode Op                       Count   Vnode Op                       Count
------------------------  ----------   ------------------------  ----------
xrda_hold                          0   xrda_readdir                      40
xrda_rele                          0   xrda_create                      219
xrda_inactive                      0   xrda_remove                      197
xrda_getattr                   17692   xrda_rename                       10
xrda_setattr                    1479   xrda_mkdir                        20
xrda_access                    21456   xrda_rmdir                        20
xrda_lookup                     7555   xrda_link                          0
xrda_getvolume                     0   xrda_symlink                       0
xrda_getlength                     0   xrda_readlink                      0
xrda_afsfid                    50687   xrda_rdwr                          0
xrda_fid                           0   xrda_fsync                         0
xrda_vmread                     1591   xrda_translate                 10059
xrda_vmwrite                    8468

Total RFS Vnode Ops     119493

                              RFS Adapter VFS Op Counts

VFS Op                         Count   VFS Op                         Count
------------------------  ----------   ------------------------  ----------
r_get                              0   r_readdir                         20
r_rel                            222   r_create                         219
r_getattr                         57   r_remove                         197
r_setattr                       1259   r_rename                          10
r_access                         478   r_mkdir                           20
r_lookup                           5   r_rmdir                           20
r_export                           0   r_link                             0
r_fstatfs                          2   r_symlink                          0
r_rpn                              0   r_readlink                         0
                                       r_rdwr                         10059

Total RFS Operations           12568
```

Table 7. Sections of the ADAPTER report

Report section	Contents
Adapter Caching Statistics	Details the performance of the caches. The hit ratios of the various caches should be approximately 80% or more. If the hit ratios are low, then the size of the vnode or directory caches could be increased. **Vnode cache size** Size of the vnode cache. It determines how many files or directories status and file permissions can be cached in the zFS/RFS cache. It also displays the size of an zFS/RFS vnode. **Vnode lookups** Number of searches for vnodes based on the inode (file number) and the hit ratio; the higher the hit ratio, the better. **Directory cache size** Size of the directory cache. **Directory buffer reads** Number of directory buffer read requests and the percentage of time the directory buffer was in storage. **Get attributes calls** Number of calls made to obtain file status information and the percentage of time valid attributes were found in the cache. **Name cache lookups** Number of calls made to find the vnode associated with a given file name. The percentage of times the vnode was found without a call to the HFS or RFS physical file system is also shown. **Access checks** Number of permission checks made and the percentage of times the permissions for a user were found cached.
HFS Adapter Vnode Op Counts	Number of each type of call made by the SMB protocol servers to the zFS adapter. Many of these calls do not result in a zFS call due to the presence of the shared cache.
Average zFS or HFS Time	Average response time and CPU time in milliseconds for each HFS or zFS physical file system call. The administrator determines the portion of the overall SMB response time that is represented by zFS or HFS calls, if desired. Changes to disk configurations would affect the performance of these response times because disk I/O wait time would be the predominant factor in these response times. Large HFS call times relative to the average SMB server response time could indicate DASD I/O bottlenecks.
RFS Adapter Vnode Op Counts	Number of calls made to the SMB protocol servers to the RFS adapter. Many of these calls do not result in an RFS file system call due to the presence of the shared cache.
RFS Adapter VFS Op Counts	Number of calls made to the SMB RFS physical file system. Response times are not given for these calls because SMB provides the support for RFS. The RFS report shows the z/OS access method response times.

HFS

In addition to the shared file system/RFS cache, most of the performance and tuning considerations are a function of the HFS file system. Use the appropriate HFS publications to tune the HFS file system.

RFS

The RFS file system maps POSIX based file I/O and maps them to z/OS data set access method calls. It uses z/OS access methods to store file status information into z/OS catalogs and retrieves the information later. The important factors regarding RFS performance is the number of access method and z/OS dynamic allocation calls made and their corresponding response times. SMB attempts to perform I/O efficiently to and from the corresponding data sets. It tries to read/write a large number of blocks when performing I/O and attempts to overlap processing with I/O as much as possible. However, much of RFS is gated by the access method performance and disk I/O performance and is therefore the most important measure of performance. Additionally, because of the nature of z/OS data sets, certain file access patterns such as random file access and file update patterns of Windows clients, RFS may be forced to perform small I/Os to the data sets or may perform synchronous I/O.

Tuning options

The adapter cache tuning options and VM cache tuning options are important for RFS performance because the larger those caches are the smaller the number of access method calls RFS needs to make. The following RFS tuning options are important.

_IOE_RFS_WORKER_THREADS
: Sets the number of worker threads required to process RFS data set open/close requests. z/OS has a restriction where a data set must be closed by the same task that opened the data set. Because the SMB protocol exporters manage the threads to process incoming requests there is no control available to RFS to ensure a close request is processed by the same thread that opened the data set. SMB handles this restriction by making available a pool of threads used exclusively for data set open/close processing. The default size of this pool is one thread.

bufhigh
: Many RFS processing options are specified in an RFS attributes file stored in the local z/OS UNIX physical file system. An important option is the size of the cache used to store BSAM data set blocks. The VM cache caches the file contents in POSIX byte stream format. BSAM record data sets require additional control information imbedded in the disk blocks; therefore, I/O to the data sets must use disk blocks properly formatted for the data set. RFS provides a buffer cache used to hold these blocks in raw data set format. A sufficient size for it should be provided. This size should be set to the average number of bytes being read/written at a moment in time by the server. The default is 2M.

blksize
: Another RFS processing option is the *blksize* data set creation parameter. This can be specified on the user command line and/or the RFS attributes file. This option is used for creation of BSAM files. Records should either be large, or should be grouped into large block sizes for efficient I/O performance. z/OS allows applications to specify a block size of 0 which means z/OS will determine optimal block size, this is the recommended setting by SMB for BSAM data sets (sequential data sets and PDS or PDS/E members).

recordsize
The RFS processing option specified in the RFS attribute file. This sets the size of the records for VSAM data sets. SMB will read/write up to 64K of data at a time to a VSAM data set. However, I/O is more efficient with larger records.

space Another RFS processing option allows the administrator or user to specify the primary and secondary extent sizes of a data set. Data set creation and initial write takes longer if secondary extents are needed. It is best to use only a primary extent if possible, or at least make the secondary extents large enough so contents of the data set are not scattered.

Diagnostics for RFS

The following output is a sample output from the RFS report. The important fields in the RFS report are the access method counts, response times, and the open/close service threads statistics. The following output provides more information about the fields in this report.

```
                          FILBLK Management Statistics
                          ----------------------------
Active NAMEBLKs = 108
Active FILBLKs = 66
FILBLK inactivations = 0
FILBLK invalidations = 0

                             Logical I/O Statistics
                             ----------------------
Logical Cache High = 4194304 (4096K) (4M)
Logical Cache Used = 0 (0K) (0M)
Cache Window Per File = 16
Logical Read Requests = 7752
Read-for-size Requests = 0
Read-for-offset Requests = 5
Out-of-sequence Reads = 0
Logical Write Requests = 12404
Out-of-sequence Writes = 0
Writes With Holes = 0
Logical File Syncs = 48
Logical File Sync Closes = 19

      RFS Open/Close Thread Statistics
             --------------------------------
Number of threads: 5  (stack size = 16K)
Open requests:           809   Queued:           0  (0.0%)
Close requests:          797   Queued:          24  (3.0%)
AsyClose requests:        12   Queued:           1  (8.3%)

                             Physical I/O Statistics
                             -----------------------
Buffer Cache High = 2097152 (2048K) (2M)
Buffer Cache Used = 0 (0K) (0M)
Buffer Cache Steal Percent = 20%
Buffer Cache Trims = 249
Buffer Steals = 1108
Buffer Closes = 12

Dynamic Allocation Calls = 289
Dynamic Unallocation Calls = 267                            Call Counts By Access Method
Access                                GETs        PUTs
Method        OPENs     CLOSEs       READs      WRITEs       CHECKs       WAITs      *Total*
```

```
------- ---------- --------- ---------- ---------- ---------- ----------  ----------
BPAM            40        40          0         15         30         15 |       140
BSAM           263       263        509        704       1276       1213 |      4228
QSAM            20        20         20          0          0          0 |        60
VSAM           526       526        640       4062        903          0 |      6657
------- ---------- --------- ---------- ---------- ---------- ----------  ----------
*Total*        849       849       1169       4781       2209       1228 |     11085

                       Average Time Per Call By Access Method
Access                              GETs        PUTs
Method       OPENs     CLOSEs       READs      WRITEs      CHECKs       WAITs     *Avg*
------  ----------  ---------  ----------  ----------  ----------  ----------  ----------
BPAM         4.901      7.276       0.000       0.163       0.015       2.708 |     3.790
BSAM        10.202     23.315       0.200       0.186       0.120       4.688 |     3.521
QSAM        11.428      2.705       0.008       0.000       0.000       0.000 |     4.714
VSAM        44.727     42.300       1.894       0.891       0.694       0.000 |     7.696
------- ---------- ---------   ----------  ----------  ----------  ----------  ----------
*Avg*       31.371    112.676      1.124       0.785      0.353         4.664 |     8.630

                      Average CPU Time Per Call By Access Method
Access                              GETs        PUTs
Method       OPENs     CLOSEs       READs      WRITEs      CHECKs       WAITs       *Avg*
------  ---------- ---------  ----------  ----------  ----------  ---------    ----------
BPAM         1.785     1.621       0.000       0.100       0.015       0.031 |      0.991
BSAM         2.097     1.705       0.112       0.104       0.055       0.028 |      0.292
QSAM         2.488     1.206       0.008       0.000       0.000       0.000 |      1.234
VSAM        13.534     9.203       1.628       0.495       0.581       0.000 |      2.334
------- ---------- ---------  ----------  ----------  ----------  ----------   ----------
*Avg*        9.177    26.340      0.940       0.436       0.269       0.028         1.532
```

Table 8. Sections of the RFS report

Report section	Contents
RFS Open/Close Thread Statistics	Shows the `Number of threads` available to process open/close requests, the number of requests, and the number of requests that needed to be queued (because all of the service threads were busy). If a large portion of open/close requests need to be queued, then response times increase because of open/close queue wait time when reading/writing files or directories. The size of each thread's program stack is shown to allow an approximation of the required storage if the number of service threads is changed.
Physical I/O Statistics	The `Buffer Cache Steals` and the `Buffer Cache Trims` sections indicate cases where storage needed to be stolen from other files in the cache, allowing an I/O to be performed for a given file. A high steal rate relative to the total number of RFS `xrda_vmread` and `xrda_vmwrite` calls (from the ADAPTER report) might indicate that the RFS buffer cache could be too small and the **bufhigh** RFS tuning option might need adjustment.
Access Method Statistics	RFS performance is determined by the response time of the z/OS access methods. SMB tries to read/write to files in large (64K) amounts but certain client access patterns might prevent that. From this report, and the reports that detail external client requests, an administrator can determine the relative amount of access method calls per SMB and determine what portion of the response time the access methods use.

Locking and serialization

The following sections describe the information that the SMB server provides about internal lock performance.

I/O wait time affects lock wait time

In certain cases, the file systems require a lock to be held over an I/O operation. I/O waits are long relative to processor speed. The slower the disk response time, the longer the I/O waits. Therefore, disk I/O wait time affects not only those threads waiting on the I/O, but also other threads if the I/O waiter is holding a lock. Improving I/O response time often improves lock wait time and then overall file server performance.

File operation queuing

The SMB file server monitors certain events. One event is file operation queuing for SMB clients. The SMB protocol exporter queues file write operations and truncation operations on an internal file handle. This is done to ensure client requests are presented to the physical file system in order. This queuing does not introduce additional serialization because the physical file system locks the file in write mode when performing a write operation. However, file operation queue waits are part of the overall server response time. Hence, the locking and serialization diagnostics provide an indication of file operation queuing. This queue time is directly related to physical file system performance, which is related to disk response times and the efficiency of physical file system file write algorithms.

Diagnostics

Figure 27 on page 159 shows sample output from the LOCK report. The important fields from the LOCK report are the `Total waits for locks`, `Average lock wait time`, `Total monitored sleeps`, `Average monitored sleep time`. These fields indicate average amount of time a thread processing an SMB request must wait on a lock or for access to a shared resource.

In Figure 27 on page 159, the `SMB CS File Operation Queue Wait` is not too large; therefore, the disk performance and the file system performance were good. File operation queue waits occur more frequently and the average wait time is longer when a disk is overloaded and occurs infrequently when the disk is not a bottleneck.

```
                                     Locking Statistics

 Untimed sleeps:      16766   Timed Sleeps:       4769   Wakeups:     443450

Total waits for locks:                        8900506
Average lock wait time:          0.552 (msecs)

Total monitored sleeps:                          7048
Average monitored sleep time:         5.347 (msecs)

       Top 15 Most Highly Contended Locks
Thread Wait     Async Disp.     Pct.      Description
-----------     -----------     -----     -----------
    4977280               0     55.1%     LFS vnode main lock
    1959305               0     21.7%     LFS main transaction lock
    1272775               0     14.1%     LFS main equivalence class lock
     254672               0      2.8%     SMB time of day services lock
     154204           79264      2.6%     SMB AS queue lock
      81346               0      0.9%     LFS log map lock
      34953           33162      0.8%     SMB TKC global file LRU and free list lock
      42055               0      0.5%     LFS volume handle lock
      24416               0      0.3%     LFS buffer lock
      18548               0      0.2%     VM cache all file lock
          0           17195      0.2%     OSI Global queue of threads waiting for locks
      14504               0      0.2%     LFS allocation handle lock
      12283               0      0.1%     LFS vnode lock
      11319               0      0.1%     SMB CS handle table lock
       8014               0      0.1%     LFS file zero lock


                Top 10 Most Common Thread Sleeps
Thread Wait     Pct.      Description
-----------     -----     -----------
       7048    100.0%     SMB CS File Operation Queue Wait
          0      0.0%     VM Page Wait
          0      0.0%     VM File Wait
          0      0.0%     VM Page Reclaim Wait
          0      0.0%     SMB TKC Tkset Revoke Wait for Tkset Holds
          0      0.0%     SMB TKC Tkset Wait For Pending Revoke
          0      0.0%     SMB TKC Get Token Wait
          0      0.0%     SMB TKC Open-in-progress Wait
          0      0.0%     SMB TKC Callback Response Wait
          0      0.0%     SMB CS Oplock Break Read-Raw Wait
```

Figure 27. Sample output from LOCK report

Storage usage

The SMB file server and client provides a command that reports storage usage of the runtime heap; this is storage that SMB allocates directly from base z/OS storage subpools. Therefore, all SMB allocated storage is shown, with the exception of storage that is allocated for thread stacks. There is no way for SMB to track that information, but Language Environment runtime options can show you that information. For more information, see *z/OS Language Environment Programming Guide* .

Diagnostics

Figure 28 on page 160 shows sample output from the STORAGE report. The `OSI Storage Statistics` indicates current SMB program use of heap storage. The `MVS Obtained Storage` and `TCB Owned Storage` statistics sections indicate the number of bytes allocated directly from z/OS storage subpools.

```
                           OSI Storage Statistics
                           ----------------------

Current allocated storage size: 49515778 (48355K) (47M)
Number of storage allocations: 722977
Number of storage frees: 721684

                           MVS Obtained Storage Statistics
                           -------------------------------

Current allocated storage size above 16M line: 303387036 (296276K) (289M)
Number of storage allocations above 16M line: 0
Number of storage frees above 16M line: 0

Current allocated storage size below 16M line: 1092 (1K) (0M)
Number of storage allocations below 16M line: 0
Number of storage frees below 16M line: 0

                TCB Owned Storage
                -----------------
Lock storage allocations: 157
Lock storage allocated:   11304
Non-lock storage allocations: 804
Non-lock storage allocated:   2317976
```

Figure 28. Sample output from STORAGE report

Part 3. Appendixes

Appendix A. Environment variables in SMB

Table 9 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: All the examples that are shown in the table must be entered on one line, with no blanks, even though some of them appear on multiple lines.

Table 9. Environment variables in SMB

Name	Description
_EUV_AUTOLOG	This environment variable was formerly used to control whether DCE autologin processing is attempted. However, with the removal of DCE support in z/OS V1R13, this environment variable should always be set to NO for all SMB server processes. SMB users that use the **smbpw** command to store their SMB password should also specify NO in their z/OS UNIX environment. The valid value is: **NO** Do not enable single sign-on processing.
_EUV_SVC_MSG_LOGGING	This environment variable controls where SMB server messages are routed and displayed. **Default value** NO_LOGGING **Expected value** **NO_LOGGING** All messages are suppressed. It is not recommended to set (or default) to this value. **CONSOLE_LOGGING** This is the recommended setting. All messages are routed to the operator console. All messages are in English (the NLSPATH environment variable is not used). **STDIO_LOGGING** All messages are routed to the standard output file (the JES output). If it exists, NLSPATH is used for messages. **Example** `_EUV_SVC_MSG_LOGGING=CONSOLE_LOGGING` **Where used** All SMB server processes.

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 used
	All programs. If the default is not being used, this environment variable must be coded in the `envar` file for each process.

Table 9. Environment variables in SMB (continued)

Name	Description
LIBPATH	The path names used to find DLLs. **Default value** None **Expected value** Character string. **Example** `LIBPATH=/usr/lib:/usr/lpp/Printsrv/lib` **Where used** DFSKERN
NLSPATH	A POSIX environment variable that is used by SMB 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 used** All SMB server processes
TZ	Sets the time zone that is used by a process. The TZ environment variable is used for the process only and is not used to set the create or update time of the z/OS UNIX file. **Default value** localtime **Where 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** In this case, DFSKERN runs in its own address space `_IOE_DAEMONS_IN_AS=DFSKERN` **Where used** DFSCNTL
_IOE_DFS_MODIFY_PATH	The path that is 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 used** All programs. If the default is not being used, this environment variable must be coded in the `envar` file for each process.

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_DIRECTORY_CACHE_SIZE	The number of 512-byte blocks that are used to cache HFS directory entries. **Default value** 2048 **Expected value** The value that is specified must be a numeric value greater than or equal to 768. **Example** `_IOE_DIRECTORY_CACHE_SIZE=4096` **Where used** DFSKERN
_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 used** DFSKERN
_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 or equal to 480 (8 hours). **Example** `_IOE_EXPORT_TIMEOUT=OFF` **Where used** DFSKERN
_IOE_HANG_DETECT_DUMP_LIMIT	Specifies the maximum number of dumps for the hang detector to take. **Default value** 5 **Expected value** A number greater than or equal to five. **Example** `_IOE_HANG_DETECT_DUMP_LIMIT=5` **Where used** DFSKERN
_IOE_HANG_DETECT_INTERVAL	Specifies the interval, in seconds, for which hangs are checked. **Default value** 10 **Expected value** A number greater than or equal to zero. **Example** `_IOE_HANG_DETECT_INTERVAL=10` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_HANG_DETECT_OENODE_TIMEOUT	Specifies the period of time, in seconds, a thread must be hung before it is detected. When zero is specified, the hang detect function is off. **Default value** 0 **Expected value** A number greater than or equal to zero. **Example** `IOE_HANG_DETECT_OENODE_TIMEOUT=30` **Where used** DFSKERN
_IOE_HANG_DETECT_THREAD_TIMEOUT	Specifies the period of time, in seconds, a thread must wait before it is detected as a hung thread. When zero is specified, the hang detect function is off. **Default value** 0 **Expected value** A number greater than or equal to zero. **Example** `_IOE_HANG_DETECT_THREAD_TIMEOUT=50` **Where used** DFSKERN
_IOE_HFS_ATTRIBUTES_FILE	Specifies the path name of the `hfsattr` file that contains the definition of file name suffixes that controls if 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 used** DFSKERN This file contains AddType statements. 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.
_IOE_HFS_FILETAG	The valid values are as follows: **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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 that is 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 52for information about 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 (client write) or output from (client read) the SMB server is examined to determine if the data is text or not. When data is received by (client write) the SMB server, the first 255 bytes of data are compared against a table of valid text (ASCII) characters. If all 255 characters are valid characters, the file is marked for translation and all data is translated. Otherwise, the data is not translated. When data is output from (client read) the SMB server, the first 255 bytes of data are compared against a table of valid EBCDIC 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 106 (`auto` parameter) for a list of valid code points. **Example** `_IOE_HFS_TRANSLATION=ON` **Where 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 used** DFSKERN
_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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 used** DFSKERN
_IOE_PROTOCOL_RPC	An environment variable that controlled if the DCE DFS protocol is supported (using DCE RPC). As of z/OS Version 1 Release 13, with the removal of DCE support, this environment variable was disabled and is no longer used. **Default value** ON **Expected value** ON or OFF **Example** `_IOE_PROTOCOL_RPC=OFF` **Where used** DFSKERN
_IOE_PROTOCOL_SMB	An environment variable that controls if the SMB protocol is supported (using TCP/IP). **Default value** OFF **Expected value** ON or OFF **Example** `_IOE_PROTOCOL_SMB=ON` **Where used** DFSKERN
_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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	**Description**
_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 used** DFSKERN This environment variable can be overridden for specific file systems in the **devtab** file. See “devtab” on page 106.
_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 used** DFSKERN
_IOE_RFS_TRANSLATION	Specifies if 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 used** DFSKERN This environment variable can be overridden for specific file systems in the `devtab` file. Refer to “devtab” on page 106.

Table 9. Environment variables in SMB (continued)

Name	Description
_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 used** DFSKERN **Note:** This specification is related to the z/OS UNIX MAXTHREADS value. For more information, see step 5 in "Installing and configuring the SMB file and print server" on page 14.
_IOE_SERVER_NAME	The network name of the system. This environment variable should be used if there are several network connections on the z/OS system and the primary connection is not being used for the DFS system. **Default value** The name that is returned from the TCP/IP function, gethostname(). The name is the default name as determined from the TCP/IP TCPDATA file; it may not reflect the actual network name for the device being used for the DFS network. **Expected value** Character string containing the network name of the system without a domain. Because TCP/IP returns the name in uppercase letters, you should also enter this name in uppercase.) **Example** `_IOE_SERVER_NAME=RANDOM` **Where used** All programs that use the TCP/IP function, gethostname().
_IOE_SMB_ABS_SYMLINK	Specifies whether the SMB server can use absolute symbolic links. **Default value** OFF **Expected value** ON or OFF **Example** `_IOE_SMB_ABS_SYMLINK=OFF` **Where 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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_AUTH_SERVER	Specifies the IP address of the domain controller that is used to authenticate PC users. **Default value** None **Expected value** An IP address (*n.n.n.n*, where *n* is a number in the range of 0 - 255). **Example** `_IOE_SMB_AUTH_SERVER=9.200.150.99` **Where used** DFSKERN
_IOE_SMB_AUTH_SERVER_COMPUTER _NAME	Specifies the computer name of the Domain Controller that is used to authenticate PC users. This name is used to pass the NetBIOS computer name of the Domain Controller to the Domain Controller. This name is not used to find a Controller. _IOE_SMB_AUTH_SERVER and _IOESMB_BACKUP_AUTH_SERVER are used for that purpose. **Default value** None **Expected value** Computer name, 15 characters or less. **Example** `_IOE_SMB_AUTH_SERVER_COMPUTER_NAME=MYCOMPUTER` **Where used** DFSKERN
_IOE_SMB_BACKUP_AUTH_SERVER	Specifies the IP address of the domain controller that is 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 used** DFSKERN
_IOE_SMB_BACKUP_AUTH_SERVER _COMPUTER_NAME	Specifies the computer name of the backup domain controller that is used to authenticate PC users. This name is used to pass the NetBIOS computer name of the Domain Controller to the Domain Controller. This name is not used to find a Domain Controller. _IOE_SMB_AUTH_SERVER and _IOESMB_BACKUP_AUTH_SERVER are used for that purpose. **Default value** None **Expected value** Computer name, 15 characters or less **Example** `_IOE_SMB_BACKUP_AUTH_SERVER_COMPUTER_NAME= MYOTHERCOMPUTER` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_BLKSIZE	Specifies the block size value that is used by SMB clients. **Default value** 32 K **Expected value** 0 K, 8 K, 16 K, or 32 K (the 'K' is required). If you specify zero, the SMB server returns the block size that is assigned by the PFS. **Example** `_IOE_SMB_BLKSIZE=32K` **Where used** DFSKERN
_IOE_SMB_BROWSE_INTERVAL	Specifies the maximum browser announcement interval (in milliseconds). **Default value** 720000 (12 minutes) **Expected value** A number in the range of 600000 (10 minutes) - 720000 (12 minutes). **Example** `_IOE_SMB_BROWSE_INTERVAL=600000` **Where 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 used** DFSKERN **Note:** This specification is related to the z/OS UNIX MAXTHREADS value. For more information, see step 5 in "Installing and configuring the SMB file and print server" on page 14.

Table 9. 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. • NOTALLOWED specifies that passwords in the clear are not allowed. Authentication is attempted using encrypted passwords only. Because all supported clients include encrypted password support, the ALLOWED setting is not recommended. IBM suggests you use REQUIRED for clear passwords and NOTALLOWED for encrypted passwords. **Default value** REQUIRED **Expected value** REQUIRED, ALLOWED, or NOTALLOWED **Example** `_IOE_SMB_CLEAR_PW=NOTALLOWED` **Where used** DFSKERN If passthrough authentication is used, see “Using passthrough authentication” on page 61 for additional information.
_IOE_SMB_CONNECT_MSGS	This environment variable is for use in debugging or determining problems with connection issues. When in use, messages are issued for session connects, session configuration, and session cancels. **Default value** 0 (zero) **Expected value** 0 - No connection messages are issued. 1 - Only error connection messages are issued. 2 - All error and informational connection messages are issued. **Example** `_IOE_SMB_CONNECT_MSGS=1` **Where 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), the SMB computer name is used to identify this server to the WINS server. **Default value** The TCP/IP host name of this system. **Expected value** Character string, 15 characters or less. **Example** `_IOE_SMB_COMPUTER_NAME=OS390DATA1` **Where used** DFSKERN If you are using Windows clients, this environment variable must specify (or be defaulted to) the TCP/IP host name to allow Search Computername functionality to work.

Table 9. Environment variables in SMB (continued)

Name	Description
_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 used** DFSKERN
_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** In this example, there are embedded blanks. `_IOE_SMB_DESCRIPTION=z/OS File Server` **Where used** DFSKERN
_IOE_SMB_DIR_PERMS	The permissions for a directory that is created by the SMB server at the request of an SMB client. (This can also be specified on a shared directory basis. See "smbtab" on page 118.) **Default value** 777 **Expected value** The specified value 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 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** WORKGROUP **Expected value** Character string, 15 characters or less **Example** `_IOE_SMB_DOMAIN_NAME=OS/390DOMAIN1` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 118.) It also limits which write permission bits can be turned on by an SMB client. **Default value** 777 **Expected value** The specified value 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 used** DFSKERN 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.

_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`
	
	`_IOE_SMB_IDMAP=//'USERID.SMB.OPTIONS(SMBIDMAP)'`
	
	Where 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 in the range of 0 - 4294967295. **Example** `_IOE_SMB_IDLE_TIMEOUT=4000` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 used** DFSKERN This specification is related to the z/OS UNIX MAXTHREADS value. For more information, see step 5 in "Installing and configuring the SMB file and print server" on page 14.
_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 in the range of 1024 - 65535 **Example** `_IOE_SMB_MAXXMT=8192` **Where used** DFSKERN
_IOE_SMB_NT_SMBS	Specifies whether new SMBs in the NT protocol dialect are to be accepted from PC clients. There might 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 used** DFSKERN
_IOE_SMB_OCSF	Specifies whether OCSF is used for encrypted passwords. To use encryption hardware, OCSF (and, therefore, ON) is required. **Default value** ON **Expected value** ON or OFF **Example** `_IOE_SMB_OCSF=OFF` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_OPLOCK_TIMEOUT	Specifies the Opportunistic Lock Timeout period (in seconds). **Default value** 35 **Expected value** A number in the range of 1 - 4294967295 **Example** `_IOE_SMB_OPLOCK_TIMEOUT=60` **Where used** DFSKERN
_IOE_SMB_OPLOCKS	Specifies whether the server permits clients to use opportunistic locks on files and let some SMB clients cache data at the client. This can improve performance. **Default value** ON **Expected value** ON or OFF **Example** `_IOE_SMB_OPLOCKS=OFF` **Where 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 in the range of 0 - 255). **Example** `_IOE_SMB_PRIMARY_WINS=9.120.44.55` **Where 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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_SETATTR_OVERRIDE	Specifies whether the SMB server should appear to allow a user who is not the owner of a file to set the modtime of the file to an explicit value. z/OS UNIX does not allow a user who is not the owner of a file to explicitly set the modtime to a specific value. However, when any user updates a file, the modtime is updated implicitly to the current time. So normally, when a file is updated, there is no need to explicitly set the modtime to a specific time value. Some Windows applications attempt to set the modtime explicitly when a file is saved. When the user is not the owner of the file, this attempt will fail and this failure is reflected back to the PC user. As stated above, this attempt to update the modtime is unnecessary (because the modtime will be updated implicitly to the current time). When this environment variable is set to ON, the SMB server will ignore the failure (to set the modtime to an explicit value) and avoid the failure of saving the file. The modtime will be set to the current time. **Default value** OFF **Expected value** ON or OFF **Example** `_IOE_SMB_SETATTR_OVERRIDE=ON` **Where 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 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 in the range of 0 - 255). **Example** `_IOE_SMB_SECONDARY_WINS=9.120.66.77` **Where 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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_IOE_SMB_TRANSPORTS	Specifies whether SMB uses the TCP DIRECT mode (port 445) or NETBIOS mode (port 139) **Default value** NETBIOS **Expected value** NETBIOS, DIRECT, or BOTH **Example** `_IOE_SMB_TRANSPORTS=BOTH` **Where used** DFSKERN
_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). **Default value** OFF **Expected value** ON or OFF **Example** `_IOE_SMB_WINS_PROXY=ON` **Where 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 used** DFSKERN
_IOE_TKM_MAX_TOKENS	Specifies the maximum number of tokens that can be held in the SMB server memory. **Default value** 10240 **Expected value** A positive number. The minimum value is 10240. **Example** `_IOE_TKM_MAX_TOKENS=20480` **Where used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 specified value must be a numeric value greater than or equal to 5. **Example** `_IOE_TKMGLUE_SERVER_THREADS=8` **Where used** DFSKERN This specification is related to the z/OS UNIX MAXTHREADS value. For more information, see step 5 in "Installing and configuring the SMB file and print server" on page 14.
_IOE_TRACE_TABLE_SIZE	Specifies the size, in bytes, of the DFSKERN trace table. **Default value** 64 MB. Do not change this setting unless you are instructed to do so by IBM Service. **Expected value** A positive number 4 - 256 MB. The value can be a whole positive integer followed by a K (denoting kilobytes), or a whole positive integer followed by an 'M' (denoting megabytes). **Example** `_IOE_TRACE_TABLE_SIZE=64M` **Where used** DFSKERN
_IOE_USE_PTHREAD_SECURITY	When this environment variable is ON, pthread_security_np() is used to establish the user security environment. This results in security updates becoming active without restarting DFSKERN. **Default value** OFF **Expected value** ON or OFF **Example** `_IOE_USE_PTHREAD_SECURITY=ON` **Where used** DFSKERN
_IOE_VM_CACHE_SIZE	Specifies the size, in bytes, of the virtual memory used by DFSKERN for all file systems that are exported. The virtual memory is used for data in all file systems that are exported. **Default value** 10 M **Expected value** A positive number. The value can 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 used** DFSKERN

Table 9. Environment variables in SMB (continued)

Name	Description
_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 used** DFSKERN
_IOE_VNODE_CACHE_SIZE	The size of the HFS vnode cache, that is, the number of vnodes. **Default value** 6144 **Expected value** The specified value must be a numeric value greater than or equal to 6144. **Example** `_IOE_VNODE_CACHE_SIZE=6144` **Where used** DFSKERN
_IOE_WIRE_CODEPAGE	Specifies the code page to be used for text data on the wire for clients. **Default value** ISO8859-1 **Expected value** ISO8859-1 or *codepage* **Note:** The SMB server does not support translation of double-byte character set (DBCS) data. **Example** `_IOE_WIRE_CODEPAGE=ISO8859-1` **Where used** DFSKERN

Appendix B. Additional information about using the SMB server

Important Note: The information provided in this section is outside the scope of the z/OS Distributed File Service SMB server. However, it is included here because it might be helpful to the SMB administrator or the PC user.

Resolving authentication issues

If you are attempting to use unencrypted passwords (for example, you are using environment variable _IOE_SMB_CLEAR_PW=REQUIRED), the local security policy might need to be updated to permit unencrypted passwords. You might have to update the registry setting on your PC client.

If you are using the encrypted password support and your users entered their SMB password in RACF using the z/OS UNIX **smbpw** command, this registry change is not necessary. Users who make this change to the registry do not need to remove it to use encrypted 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.

Updating the registry for Windows clients

This section shows you how to update the registry for the supported Windows clients (see "Supported SMB clients" on page 3). The steps might vary depending on the Windows platform in use.

1. Click **Start**, then click **Control Panel**.
2. Double-click **Administrative Tools**.
3. Double-click **Local Security Policy**.
4. On the left pane, click the plus (+) to expand Local Policies (under Security Settings).

   On Windows 7 Professional/Enterprise/Ultimate clients, you would click an arrow, rather than a plus (+), to expand sections.
5. On the left pane, click **Security Options**.
6. Double-click Microsoft network client: Send unencrypted password to third-party SMB servers.
7. Click Enabled, and then click **OK**.

If the registry still does not permit unencrypted passwords, do the following tasks:

1. For Windows 7 and Vista Business/Enterprise: Type `Regedit` in the search box and the Registry editor will open in a new window.
2. On the left pane, click the plus (+) to expand HKEY_LOCAL_MACHINE
3. On the left pane, click the plus (+) to expand SYSTEM
4. On the left pane, click the plus (+) to expand CurrentControlSet
5. On the left pane, click the plus (+) to expand Services
6. On the left pane, click the plus (+) to expand Lanmanworkstation

7. On the left pane, click parameters
8. On the right pane, select EnablePlainTextPassword
9. Click Edit and select Modify
10. Update value data to hex "1"
11. Restart the client machine for the registry change to take effect

Checking whether the client can communicate

If your Windows client is not communicating with the SMB server, make sure that the SMB traffic can communicate over the following TCP/IP ports, depending on which method you are using:

137-139	NetBIOS
445	TCP/IP direct option

It is possible that the SMB traffic is being restricted from getting to the SMB server; for example, a firewall may be in place. Contact your network or system administrator to verify that this traffic is permitted.

Insufficient space errors

Before saving certain files, some client applications attempt to determine whether there is sufficient disk space available. An SMB request is sent to the server to query the space available in that file system. The request is against the file system at the share name; the request is not in the file system where the saved file will reside. As a result, you might receive a message similar to the following message, which indicates that insufficient space is available to save the file.

```
The disk you were saving to or the disk used for temporary files is full.
Free some space on this disk and try again, or save to a different disk.
```

When crossing through multiple file systems or mount points, the SMB query only looks for space available at the share name and not in the file system below that point. This behavior is consistent with SMB implementations on other supported platforms.

Use one of the following scenarios to resolve this issue:

1. Statically exported file systems (_IOE_DYNAMIC_EXPORT=OFF)

   Verify that sufficient space exists in the file system at the exported share name mount point. Increase the size of the file system defined by the entry in the **devtab** configuration file.
2. Dynamically exported file systems (_IOE_DYNAMIC_EXPORT=ON)

   Verify that sufficient space exists in the file system at the exported share name mount point that represents the parent file system to any dynamically mounted file systems. Dynamically mounted file systems under the parent file system have no bearing on the amount of space returned from the query.
3. Automounted file systems scenarios (_IOE_DYNAMIC_EXPORT=ON and automount)
   a. Accessing an SMB share using the smbtab &USERID keyword

      Verify sufficient space is available for the automounted file system reference. For example, refer to the sample configuration files in Figure 29 on page 185 on page 187. If the user smbuser performed a **net use** command to *myhomedir* and entered the **dir** command, the file system smbuser.auto.home is automounted and exported by the SMB server. Insure sufficient space in that file system exists when saving in this file system or in any file systems mounted below this point.

b. Accessing an SMB share at the automount file system

   z/OS UNIX System Services reserves this file system for automount processing. The response to a file system query at this point will return zero space available. In this scenario, add a static entry in your **devtab**, **dfstab**, and **smbtab** for that file system and access through that exported share name mount point to save the file.

   For example, refer to the sample configuration files in Figure 29. If the user smbuserperformed a **net use** command to *homedir*, and then proceeds to traverse into a directory called smbuser, that triggers the mount of the file system smbuser.home.auto. If the user attempts a save there, the query file response would return zero bytes available.

```
smbtab entry:
/dev/ufs10 myhomedir ufs "Each user's home directory" r/w 100 /&USERID
/dev/ufs10 homedir         ufs "Each user's home directory" r/w 100 /

dfstab entry:
/dev/ufs10 hfs10  ufs 0,,10

devtab entry:
"*AMD/home/automnt"  binary

u.map entry:
  name               *
 type              HFS
 filesystem     <uc_name>.AUTO.HOME
 mode             rdwr
 duration         nolimit
 delay              10

 auto.master entry:
 /home/automnt /etc/u.map
```

Figure 29. Sample configuration file

Editor or word processor changes the owner/permissions of the HFS file

When many editors and word processors save a file, they do not simply update the file. Rather, to avoid losing data if the system should fail during the save operation, they first create a file with a temporary name and save the data there; they then erase the original file and then rename the file with the temporary name to the original file name. Because a new file was created, it is owned by the user that created it. This user might be different from the owner of the original file. Also, the file permissions are changed to the default permissions, which might be different from the original permissions.

Editor or word processor cannot save a file to HFS

When many editors and word processors save a file, they do not simply update the file. Rather, 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 might 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. Microsoft WordPad correctly displays text files that have z/OS UNIX end of line characters. However, when the file is saved, the end of line characters are changed to Windows end of line characters. 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.

To determine which end of line characters are contained in a text file, you can use the following z/OS UNIX command to display the hexadecimal values of the characters in a file, where *textfile.txt* is the name of the text file you want to display.

```
od -cx textfile.txt
```

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. To do so, issue the following z/OS UNIX command, 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.

```
tr -d '\r' <textfile.txt >newfile.txt
```

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 shown in the preceding example 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 versus 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 a long 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, the following command sets the sync interval to 30 seconds for the HFS file system specified.

```
MOUNT FILESYSTEM('OMVS.ABC.DEF') MOUNTPOINT('/abc/def') TYPE(HFS) MODE(RDWR) PARM('SYNC(30)')
```

See the TSO/E MOUNT command in *z/OS UNIX System Services Command Reference* for information about the sync interval parameter.

Note: This does not apply to zFS file systems.

Writing of data appears successful but data is not there

If a write failure occurs due to the file system being full, the error will be returned on close. If the application / client ignores the error, the operation will appear to be successful. This can also occur if the end of line characters cannot be found in text data.

Opening Excel spreadsheets stored on z/OS

If a user (user A) opens a Microsoft Excel spreadsheet that is stored on z/OS, and then a second user (user B) opens the same spreadsheet, a popup window appears that gives user B three choices:

1. Open the file read-only
2. Open the file read-only and be notified when the file has been saved or closed
3. Cancel - do not open the file.

In the popup window, the user that is indicated as having the file locked for editing might be inaccurate. If the current user that has the file open for write is using the z/OS SMB server and has not saved any updates, the editing user displayed is the last user that updated the file (possibly a while ago). See also "Editor or word processor changes the owner/permissions of the HFS file" on page 185. Excel apparently causes the owner to change when the spreadsheet is saved. A user (who is mapped to a different z/OS user) might be denied access to the file. The error message displayed did not indicate an authorization problem but that the file was read-only or encrypted.

Inconsistent file attributes on copy

When the file copy operation is initiated, the file attributes are queried. In this scenario, the file has r_x authority. After the new file is created, an SMB can be issued to reset the write file attribute. If the file is later edited by the user, attempts to save the file might result in messages about the file being denied access. This behavior might be confusing because the user is the owner and, therefore, should have write access to the file.

The application performing the copy can determine how the file write permission is to be handled and can issue a set file attribute SMB call to reset the write permission file attribute.

For example, the DOS XCOPY command by default resets the read-only file attributes. If the `/k` option is specified, the file attributes are preserved.

Keeping medium weight threads during pthread_exit processing

APAR OA39909 describes changes by z/OS UNIX that enable applications to specify how medium weight threads handle pthread_exit processing. APAR OA45175 describes the SMB updates that were made as a result of the change in threading behavior. Because of the changes, SMB users must now specify that medium weight threads are to be kept by using the _BPXK_UNUSEDTASKS=KEEP environment variable.

Perform the following steps to set up this option in the environment file:

1. Locate the environment variable file:
   ```
   /opt/dfslocal/home/dfscntl/envar
   ```
2. Set it as follows:
   ```
   _BPXK_UNUSEDTASKS=KEEP
   ```
3. Recycle the DFS/SMB server processes (DFSCNTL/DFSKERN) to pick up the _BPXK_UNUSEDTASKS environment variable. Issue these commands:
   ```
   C DFS
   S DFS
   ```
   If you try to STOP DFS (P DFS) instead of canceling it, you will see:
   ```
   IOEP01104I DFS kernel has received STOP command.
   IOEP01130A DFS kernel cannot end process 50331677.
   EDC5143I No such process
   ```
   If you try to RESTART DFSKERN without first canceling DFS(DFSCNTL), you will see:
   ```
   IOEP01119A Daemon DFSKERN cannot be started because it has status of READY
   ```

Appendix C. Customizable files

Table 10 summarizes the Distributed File Service example files that are supplied by IBM and where they are placed; you can customize these files to meet your local SMB support requirements. Files that apply to support that is not being used are created, but they 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, which is run as part of the Distributed File Service post installation processing. For more information, see Chapter 3, "SMB post installation processing," on page 13.

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 also 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.

Table 10. DFS customizable files

IBM-Supplied file	Customizable file
`/opt/dfs/global/examples/daemonct.envar`	`/opt/dfslocal/home/daemonct/envar`
`/opt/dfs/global/examples/devtab`	`/opt/dfslocal/var/dfs/devtab`
`/opt/dfs/global/examples/dfs.ioepdcf`	`/opt/dfslocal/etc/ioepdcf`
`/opt/dfs/global/examples/dfscntl.envar`	`/opt/dfslocal/home/dfscntl/envar`
`/opt/dfs/global/examples/dfsexport.envar`	`/opt/dfslocal/home/dfsexport/envar`
`/opt/dfs/global/examples/dfskern.envar`	`/opt/dfslocal/home/dfskern/envar`
`/opt/dfs/global/examples/dfstab`	`/opt/dfslocal/var/dfs/dfstab`
`/opt/dfs/global/examples/cmattr`	`/opt/dfslocal/var/dfs/hfsattr` Defined by envar **_IOE_HFS_ATTRIBUTES_FILE**
`/opt/dfs/global/examples/rfstab`	`/opt/dfslocal/var/dfs/rfstab`
`/opt/dfsglobal/examples/idmap`	`/opt/dfslocal/home/dfskern/smbidmap` Defined by envar **_IOE_SMB_IDMAP**
`/opt/dfs/global/examples/smbtab`	`/opt/dfslocal/var/dfs/smbtab`

Files and directories in /opt/dfsglobal

Table 11 lists the directories and files in the **/opt/dfsglobal** directory. Note that these files or directories are created during the installation process.

Table 11. Files and directories in /opt/dfsglobal

File or directory	Description
`/opt/dfsglobal/bin`	Contains commands: dfsshare, dfssyntax, and smbpw
`/opt/dfsglobal/examples`	Contains example files for devtab, dfstab, ioepdcf, and rfstab; it also contains the example envar files for the DFS server processes. In addition, the directory contains the following program files: **asc2eb.cmd** Converts ASCII characters to EBCDIC characters **eb2asc.cmd** Converts EBCDIC characters to ASCII characters
`/opt/dfsglobal/lib/nls/msg/En_US.IBM-1047`	Contains DFS message catalogs.
`/opt/dfsglobal/src/COPYRIGHT.DFS`	Contains the Distributed File Service copyright information.
`/opt/dfsglobal/script/dfs_cpfiles`	Contains the z/OS shell script for the z/OS Distributed File Service default configuration files creation program.
Note: The symbolic link **/opt/dfsglobal** refers to the directory **/usr/lpp/dfs/global**.	

Symbolic links to /etc/dfs created during installation

Table 12 lists the frequently-used symbolic links that are created during installation.

Table 12. Symbolic links created to /etc/dfs during installation

Symbolic link	Linked file
`/bin/dfsexport`	`../usr/lpp/dfs/global/bin/dfsexport`
`/bin/dfsshare`	`../usr/lpp/dfs/global/bin/dfsshare`
`/bin/dfssyntax`	`../usr/lpp/dfs/global/bin/dfssyntax`
`/etc/ioepdcf`	`../etc/dfs/etc/ioepdcf`
`/opt/dfslocal`	`../etc/dfs`
`/opt/dfsglobal`	`../usr/lpp/dfs/global`

Directories in /opt/dfslocal

Table 13 lists the directories in `/opt/dfslocal`. All of these directories are created by the installation process.

Table 13. Directories in /opt/dfslocal

Directory name	Description
`/opt/dfslocal/etc`	Contains the daemon configuration (**ioepdcf**) file.
`/opt/dfslocal/bin`	Contains binary files
`/opt/dfslocal/home`	Contains the home directories of the Distributed File Service daemons.
`/opt/dfslocal/home/daemonct`	Home directory of the daemonct process.
`opt/dfslocal/home/dfscm`	Home directory of the dfscm process

Table 13. Directories in /opt/dfslocal (continued)

Directory name	Description
/opt/dfslocal/home/dfscntl	Home directory of the dfscntl process.
/opt/dfslocal/home/dfsexport	Home directory of the dfsxport process.
/opt/dfslocal/home/dfskern	Home directory of the dfskern process.
/opt/dfslocal/lib/nls/msg	Contains the DFS message catalogs.
/opt/dfslocal/syscall	Used to contain the socket for AFS_SYSCALL.
/opt/dfslocal/var/dfs	Contains administration and configuration files.
/opt/dfslocal/var/dfs/adm	Contains log files for the DFS server processes.

Note: The symbolic link `/opt/dfslocal` refers to the directory `/etc/dfs`.

Customizable files in opt/dfslocal

Table 14 identifies the files in the `/opt/dfslocal` subdirectories that can be customized. Many of these files are copied from the IBM-supplied files in the `/opt/dfsglobal/examples` directory to the applicable operational directory by the `/opt/dfsglobal/scripts/dfs_cpfiles` program. These files can be updated by an administrator using an editor such as OEDIT to specify environment variable values and other configuration information. These files should only be updated using the commands.

Table 14. Customizable files in opt/dfslocal

File	Created by	Customizable
/opt/dfslocal/etc/ioepdcf	dfs_cpfiles Program	OEDIT
/opt/dfslocal/home/daemonct/envar	dfs_cpfiles Program	OEDIT
/opt/dfslocal/home/dfscntl/envar	dfs_cpfiles Program	OEDIT
/opt/dfslocal/home/dfsexport/envar	dfs_cpfiles Program	OEDIT
/opt/dfslocal/home/dfskern/envar	dfs_cpfiles Program	OEDIT
/opt/dfslocal/home/dfskern/smbidmap	dfs_cpfiles Program	OEDIT
/opt/dfslocal/syscall/rendezvous	Server processing	No
/opt/dfslocal/var/dfs/devtab	dfs_cpfiles Program	OEDIT
/opt/dfslocal/var/dfs/dfstab	dfs_cpfiles Program	OEDIT
/opt/dfslocal/var/dfs/hfstab	dfs_cpfiles Program	OEDIT
opt/dfslocal/var/dfs/hfsattr	dfs_cpfiles Program	OEDIT
/opt/dfslocal/var/dfs/rfstab	dfs_cpfiles Program	OEDIT
/opt/dfslocal/var/dfs/smbtab	dfs_cpfiles Program	OEDIT

Note: The symbolic link **/opt/dfslocal** refers to the directory **/etc/dfs**.

Appendix D. 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) and multi-volume data sets (except for striped) 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 that is a hierarchy of directories that can contain other directories or files. Record data, however, uses a non-hierarchical 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.

Because 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

The example in this section refers to the data sets shown in Figure 30.

```
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
```

Figure 30. Example of data sets to map to an RFS file system

The following sequence shows 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 the device parameter (in this example, it would be `/dev/ufs3`). 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
   ```

   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 the device name parameter (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 /
   ```

   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.

   ```
   C:\>net use r: \\computer1.abc.com\mvsusera password
   ```

   Figure 31 represents a view of the data set hierarchy.

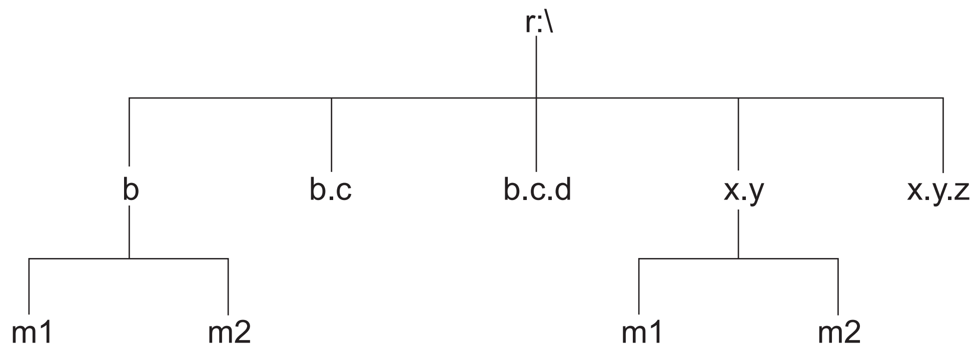


Figure 31. View of data set hierarchy

You can 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. That is, the data set remains allocated the entire time that the data set is exported. 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 a 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.

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. PC clients or record applications can write the data, but not both. In some cases, they can both read the data.

Note that 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 permit a batch job that needs access to the data set to continue. For example, the z/OS UNIX command **dfsexport /dev/ufs4 -detach** would unexport the data set (assuming that the **devtab** entry for the data set begins with **/dev/ufs4**).

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. See *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information about 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.)

Releasing data sets

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, which 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.

To make a data set available immediately, 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. Figure 32 on page 197 shows a simple batch job.

```
//JOBNAME JOB
//STEP1   EXEC   PGM=IEFBR14
//DD1     DD     DSN=USERA.PCDSNS.B.C,DISP=OLD
```

Figure 32. Sample batch job to make a data set available

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). See *z/OS JES3 Initialization and Tuning Reference* for information about 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. As the following example shows, if the release was successful, the message indicates that the system attempted to access the file but the system could not do so because the object already exists.

```
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, the SMB server sends back EACCES (the user does not have the required permission) to the PC client. For a Linux Samba client, the **mkdir release** command can also be used. For example:

```
linux001:/mntrfs # mkdir "b.c,release"
mkdir: cannot create directory `b.c,release': File exists
```

Forcing a data set to be freed by SMB

An operator force command (**modify dfs**) is provided to force the SMB server to free up a data set if an important batch job has been waiting too long. In the following example, *data-set-name* is the name of the data set that the SMB server should make available.

```
modify dfs,send dfskern,release,data-set-name
```

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).

Because 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.

This only affects commands that list file names and attributes. If a PC client creates an RFS file, that file can be directly referenced immediately after creation. Data sets or files created by a TSO user, a batch job or any z/OS application are not visible or immediately accessible to the PC client, until the deallocation time period (**_IOE_RFS_ALLOC_TIMEOUT**) and the refresh time interval (**_IOE_RFS_STATUS_REFRESH_TIME**) expire. In addition, if the data set or file remains opened by the z/OS user, batch job, or application after it is created, it is not visible or accessible to the PC client, until it is closed and both the deallocation time period and the refresh time interval expire. For details about the **_IOE_RFS_ALLOC_TIMEOUT** and **_IOE_RFS_STATUS_REFRESH_TIME** environment variables, see Appendix A, "Environment variables in SMB," on page 163.

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.

Determining file size 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 208 for more information of file size determination and to “Handling of the time stamps” on page 210 for more information about 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 193, 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 *z/OS DFSMS Using Data Sets* for information about 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 uppercase characters. This means that file names would be displayed to PC client users in uppercase and would need to be entered in uppercase. There is, however, a processing attribute in the attributes file (refer to “Attributes file (rfstab)” on page 99 called **maplower**. **maplower** is the default. When this attribute is specified or defaulted in the attributes file, users can enter lowercase letters for file names and they are mapped to uppercase by the SMB server. Also, when the (uppercase) file names are displayed to the user, they are mapped to lowercase by the SMB server.

You are cautioned, however, that when the **maplower** processing attribute is in effect, you should only use lowercase 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 have file names that 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.

Using the fastfilesize attribute

Read-for-size operations can be resource-intensive and can require long time cycles because the entire file must be read to obtain the exact file sizes. Consequently, the `fastfilesize` attribute for PDS, PDSE, DA, or non-system-managed data set might be used to improve performance. This attribute avoids read-for-size file operations by estimating the file size without opening and reading the entire file.

The method of obtaining directory and file information depends on what transactions are sent to the server by the SMB client or service (for example Windows File Explorer). If the `fastfilesize` attribute is defined, the file sizes might be displayed with a file size of zero. The client might not have requested the actual file size or the server might not have calculated an estimated file size. In addition, the size might not be current because the purpose of the `fastfilesize` attribute is to avoid a read-for-size file operation.

Due to this behavior, do not use the `fastfilesize` attribute. Instead, use the `nofastfilesize` attribute.

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 99
- Default installation data set creation attributes, specified by the system administrator in the attributes file (refer to "Attributes file (rfstab)" on page 99)
- DFSMS data class attributes
- Data set creation attributes specified for the fileset in the **devtab** file (refer to "devtab" on page 106)
- Data set creation attributes specified at file creation, for example, in the **mkdir**, **notepad** (edit), or **copy** commands (highest priority).

The following 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:

1. Keyed access to files is not supported by PC clients.
2. You can only share a PDSE between systems in the same parallel sysplex; z/OS does not support sharing PDSE between sysplexes.

Naming z/OS files

When naming conventional z/OS files, you must follow the file naming conventions, as described in *z/OS DFSMS Using Data Sets*.

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 Figure 32 on page 197, it is `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.txt,dsorg(ps)"
```

When you save the file using **`notepad`**, you have just created a new PS file named `SMITH.NEW.TXT`.

You must specify the .txt suffix on the file name. If you do not, **`notepad`** will automatically append it and will try to create `"new,dsorg(ps).txt"`, which will not be successful. You must also type a carriage return after the last line (or the only line) of the file. Notepad does not do this automatically.

An example for Linux Samba follows:

```
linux001:/mntrfs # vi "new.txt,dsorg(ps)"
```

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.

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. Table 15 shows examples of the how to do so correctly.

Table 15. Examples of commands to copy a file to an RFS file system

Form	Command example
Correct	`C:\>copy file1 "r:\newfile1,space(2,5),cyls"`
Incorrect	`C:\>copy file1 "r:\,space(2,5),cyls"`
Correct	`linux001:/mntrfs # cp file1 "newfile1,space(2,5),cyls"`

Incorrect	`linux001:/mntrfs # cp file1 ",space(2,5),cyls"`
	 `` cp: cannot create regular file `,space(2,5),cyls': Invalid argument ``

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.txt,dsorg(da),blksize(80),lrecl(80),recfm(f)"
```

You have just created a new DA file named `SMITH.NEW.TXT`

You must specify the `.txt` suffix on the file name. If you do not, **notepad** will automatically append it and will try to create `"new,dsorg(da),blksize(80),lrecl(80),recfm(f).txt"`, which will not be successful. You must also type a carriage return after the last line (or the only line) of the file. Notepad does not do this automatically.

An example of Linux Samba follows:

```
linux001:/mntrfs # vi "new.txt,dsorg(da),blksize(80),lrecl(80),recfm(f)"
```

When reading or changing data in a Direct Access 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 or defaulted to 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. 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.

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. For general information about PDSs and PDSEs, refer to *z/OS DFSMS Using Data Sets*.

Note:

1. You cannot create new directories within a PDS or PDSE, due to the nature of these data structures.
2. You can only share a PDSE between systems in the same parallel sysplex; z/OS does not support sharing PDSE between sysplexes.

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)"
   ```

   You can omit specifying the **dsntype(pds)** attribute, if **pds** has been specified for the **dsntype** attribute in the attributes file listed in Table 3 on page 102.

   Examples for Linux Samba follow:

   ```
   linux001:/mntrfs # mkdir "datalib,dsntype(library)"

   linux001:/mntrfs # mkdir "datalib,dsntype(pds),dir(20)"
   ```

2. You can use the **copy** command to create a PDS or PDSE member named `smith.datalib(member1)`:

   ```
   C:\>copy c:\otherfile.txt r:\datalib\member1
   ```

   An example for Linux Samba follows:

   ```
   linux001:/mntrfs # cp otherfile.txt datalib/member1
   ```

You have now created a PDS or PDSE member. You can use the **type** command (the **cat** command for Linux) to view the contents of your PDS or PDSE member.

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. The example shown in Figure 33 on page 204 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
```

Figure 33. Example commands to remove a PDS or PDSE

Figure 34 shows an example of Linux Samba commands that produce the same results as the commands shown in Figure 33.

```
linux001:/mntrfs # ls datalib
.  ..  butcvsmp  copyunld  su2aloc  test  testtext  testtxt3  textaloc
linux001:/mntrfs # rm datalib/*
linux001:/mntrfs # rmdir datalib
linux001:/mntrfs # ls datalib
/bin/ls: datalib: No such file or directory
```

Figure 34. Example of Linux commands to remove a PDS or PDSE

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 the command sequences shown in Figure 35; these two approaches are equivalent.

```
C:\>type r:\source\bigblue

or

C:\>cd r:\source
C:\>type r:bigblue
```

Figure 35. Example commands to access PDS or PDSE members

Figure 36 on page 205 shows two examples of similar commands for Linux Samba follow.

```
linux001:/mntrfs # cat source/bigblue

or

linux001:/mntrfs # cd source
linux001:/mntrfs/source # cat bigblue
```

Figure 36. Example of Linux commands to access PDS or PDSE members

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, such as the AIX and z/OS UNIX **vi** editor, follow the preceding steps. Other editors, such as the Notepad editor, do not follow those steps. In the latter case, the user must save the updated version into a new file. You cannot save the updated version into the PDS or PDSE, because Notepad will attempt to append .txt to the member name. You must save it somewhere else (for example, `c:\saved_updates.txt`) and then copy it into a new member in the PDS. For example:

```
C:\>copy c:\saved_updates.txt r:\datalib\member2
```

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; however, keyed access and relative-number access to the files are not supported.
- Key-sequenced (KSDS)
- Entry-sequenced (ESDS)
- Relative record (RRDS).

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 about VSAM files, refer to *z/OS DFSMS Using Data Sets*.

For example, in Figure 37, the attributes indicate the following information:
- 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.

```
notepad "r:\ksds.new2.txt,spanned,dsorg(indexed),keys(8,0),recordsize(1k,4k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 37. Example command to create a VSAM file

Similarly, Figure 38 on page 207 shows an example of a Linux Samba command.

```
linux001:/mntrfs # vi "ksds.new2.txt,spanned,dsorg(indexed),keys(8,0),recordsize(1k,4k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 38. Example Linux command to create a VSAM file

Figure 39 shows an example command that creates a VSAM ESDS file that has the following attributes:
- 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.

```
notepad "r:\esds.new3.txt,spanned,dsorg(nonindexed),recordsize(1k,4k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 39. Example command to create a VSAM ESDS file

Figure 40 shows a similar command example using Linux Samba.

```
linux001:/mntrfs # vi "esds.new3.txt,spanned,dsorg(nonindexed),recordsize(1k,4k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 40. Example Linux command to create a VSAM ESDS file

In Figure 41, the attributes specified on the command indicate the following information:
- 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.

```
notepad "r:\rrds.new4.txt,nonspanned,dsorg(numbered),recordsize(1k,1k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 41. Example command to create a VSAM ESDS file (non-spanned records)

Figure 42 on page 208 shows an example of a Linux Samba command.

```
linux001:/mntrfs # vi "rrds.new4.txt,nonspanned,dsorg(numbered),recordsize(1k,1k),
space(50,10),shareoptions(1,3),vol(d80cat)"
```

Figure 42. Example of a Linux command to create a VSAM ESDS file (non-spanned records)

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)"
```

An example of Linux Samba follows; this command results in a file created with a fixed-blocked format.

```
linux001:/mntrfs # vi "file,recfm(vb),recfm(fb)"
```

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 about striped files, refer to *z/OS DFSMS Using Data Sets*.

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 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 data sets

A data set that is migrated cannot be accessed. It must be recalled before it can be accessed. Whether a data set is recalled or not is based on the **retrieve** / **noretrieve** specification in the attributes file (rfstab). When a migrated data set is referenced, the request fails and the data set is recalled if retrieve (no wait) is specified in **rfstab**. Later, when the data set has been recalled, the data set can be accessed. If noretrieve is specified in **rfstab**, the request fails and the data set is not recalled.

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 Last time the file was accessed (read).

mtime Last time the file was modified (write).

ctime 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

Figure 43 shows how the SMB server generates the *atime* and *mtime* values for non-system managed PS and DA data sets from the dates.

```
atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

Figure 43. How time stamps for non-system managed PS and DA data sets are generated

The *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. Figure 44 shows how the time stamps for PDS members are generated

```
atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

Figure 44. How time stamps for PDS members are generated

ISPF is a z/OS base element that maintains some additional statistics for each member. They include the creation date and the last modification date and time. Figure 45 shows how the server generates the time stamps and initializes the z/OS UNIX times, if the ISPF time stamps are present for a PDS member.

```
atime = mtime = modification_date + modification_time
ctime = ISPF_creation_date + time_increment
```

Figure 45. How time stamps for PDS members are generated with ISPF times stamps

The *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 E. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the `FILE` keyword. In the example, if you include the `FILE` keyword, but do not specify an option, the default option `KEEP` is applied. A default option also applies to the next higher dotted decimal number. In this example, if the `FILE` keyword is omitted, the default `FILE(KEEP)` is used. However, if you hear the lines `2? FILE`, `2.1`, `2.1.1! (KEEP)`, and `2.1.1 (DELETE)`, the default option `KEEP` applies only to the next higher dotted decimal number, `2.1` (which does not have an associated keyword), and does not apply to `2? FILE`. Nothing is used if the keyword `FILE` is omitted.

`* indicates an optional syntax element that is repeatable`
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the `*` symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data area`, you know that you can include one data area, more than one data area, or no data area. If you hear the lines `3* , 3 HOST`, `3 STATE`, you know that you can include `HOST`, `STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loopback line in a railroad syntax diagram.

`+ indicates a syntax element that must be included`
The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the `+` symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST, and 2 STATE`, you know that you must include `HOST, STATE`, or both. Similar to the `*` symbol, the `+` symbol can repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

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:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

Index

Product Number: 5650-ZOS

Printed in USA

SC23-6886-01