

z/OS
3.2

Data Set File System Administration



Note

Before using this information and the product it supports, read the information in [“Notices” on page 123.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 2022, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	vii
Tables.....	ix
About this document.....	xi
How this document is organized.....	xi
Conventions used in this book.....	xi
z/OS information.....	xii
How to provide feedback to IBM.....	xiii
Summary of changes.....	xv
Summary of changes for z/OS 3.2.....	xv
Summary of changes for z/OS 3.1.....	xv
Part 1. DSFS administration guide.....	1
Chapter 1. Overview of z/OS Data Set File System.....	3
Components of the DSFS tree.....	5
File processing.....	8
Directories.....	8
Creating new data sets.....	9
Removing and renaming data sets.....	9
Holding dynamic allocations and enqueues on PDS and PDSE directories.....	9
Serialization and caching.....	9
Caching data set contents.....	10
Event notification facility (ENF).....	10
Security caching.....	10
UNIX security.....	11
z/OS UNIX attributes versus data set attributes	11
UNIX attribute mapping.....	11
DSFS background tasks.....	12
Tailoring high-level qualifiers.....	12
Program module support.....	13
Chapter 2. Installing and configuring DSFS.....	15
Steps for installing and configuring DSFS.....	15
Steps for creating a BPXPRMxx entry for DSFS.....	16
Steps for creating the utility file system	17
(Optional) Steps for creating or updating the DSFS configuration file.....	17
Specifying IDFPRMxx in the parmlib.....	17
Using an IDFZPRM DD file in the DSFS PROC.....	17
Defining the DSFS user ID.....	18
Creating the root directory.....	18
Coexistence and functionality APARs for DSFS.....	18
Chapter 3. Managing DSFS processes.....	21
Starting DSFS.....	21
Stopping DSFS.....	21

Determining DSFS status.....	22
Chapter 4. Managing the utility file system.....	23
Understanding the utility file system.....	23
Defining the utility file system.....	24
Allowing for dynamic growth of the utility file system.....	25
Allowing for encryption of the utility file system.....	26
Naming convention.....	27
Mounting the utility file system.....	27
The shared file system environment	27
Mounting and unmounting utility file systems.....	28
Changing the utility file system data set for a system.....	29
Encrypting the utility file system.....	29
Compressing transparent data.....	29
Monitoring space in the utility file system.....	29
Displaying usage information for the utility file system.....	30
Verifying and repairing the utility file system.....	31
Chapter 5. Creating new data sets with DSFS.....	33
Saving creation parameters for data sets.....	33
DSFS restrictions to BPXWDYN text strings in creation parameters.....	34
Displaying saved creation parameters.....	34
Chapter 6. Extended attributes.....	37
Chapter 7. Using the dsadm fileinfo command.....	41
Storing creation parameter information.....	42
Data set information.....	42
Timestamps.....	43
Chapter 8. Monitoring and tuning performance.....	45
Performance tuning.....	45
Monitoring DSFS performance.....	45
FILECACHE report.....	46
DSCACHE report.....	47
KNPFS report.....	48
Resetting performance statistics.....	48
Displaying memory usage.....	49
STORAGE report	49
Delays and hangs in DSFS.....	50
Steps for diagnosing and resolving a DSFS hang.....	51
Identifying storage shortages in DSFS.....	52
Disabling the utility file system.....	53
Chapter 9. JES spool data sets.....	55
Part 2. DSFS administration reference.....	59
Chapter 10. z/OS system commands.....	61
MODIFY DSFS PROCESS.....	62
MODIFY DSFS,FSINFO.....	63
MODIFY QUERY reports.....	65
SETOMVS RESET.....	81
Chapter 11. DSFS commands.....	83
MOUNT.....	83
The dsadm command suite.....	85

dsadm apropos	89
dsadm config	90
dsadm configquery.....	93
dsadm createparm	96
dsadm fileinfo.....	98
dsadm fsinfo.....	105
dsadm help.....	106
dsadm jobid.....	108
dsadm query.....	109
dsadm salvage	111
Chapter 12. The DSFS configuration file (IDFPRMxx or IDFFSPRM).....	113
Processing options for IDFFSPRM and IDFPRMxx.....	114
Appendix A. Accessibility.....	121
Notices.....	123
Terms and conditions for product documentation.....	124
IBM Online Privacy Statement.....	125
Policy for unsupported hardware.....	125
Minimum supported hardware.....	125
Trademarks.....	126
Index.....	127

Figures

1. DSFS directory structure..... 5

2. Defining a utility file system with secondary extents..... 25

3. Adding volumes to an existing utility file system..... 26

4. Creating a utility file system with a key label..... 27

5. Example of output from dsadm fsinfo..... 30

6. Example of dsadm query -storage.....49

7. Sample QUERY,THREADS output..... 52

Tables

1. Strings for system.jobstatus 38

2. Status codes for file system state..... 64

About this document

The purpose of this document is to provide complete and detailed guidance and reference information. This information is used by system administrators who work with z/OS Data Set File System (DSFS).

How this document is organized

- Part 1, “DSFS administration guide,” on page 1 provides guidance information for Data Set File System (DSFS).
- Part 2, “DSFS administration reference,” on page 59 provides reference information about system commands, DSFS commands, and the DSFS configuration options file.

Conventions used in this book

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 nonblank character on the line to be continued, and continue the command on the next line.

Note: When you enter a command from this document that uses the backslash character (\), make sure that 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 Information Roadmap*.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

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.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) (www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy).

Summary of changes for z/OS 3.2

The following content is new, changed, or no longer included in z/OS 3.2.

New

The following content is new.

September 2025 release

- None.

Changed

The following content is changed.

September 2025 release

- [“Coexistence and functionality APARs for DSFS” on page 18](#) replaces the original APAR information.
- [“Stopping DSFS” on page 21](#) contains a new reference.

Deleted

The following content is deleted.

September 2025 release

- None.

Summary of changes for z/OS 3.1

The following content is new, changed, or no longer included in z/OS 3.1.

New

The following content is new.

December 2023 refresh

- DSFS supports access to JES spool data sets in the primary JES subsystem. For more information, see Chapter 9, [“JES spool data sets,” on page 55](#) and [“dsadm jobid” on page 108](#). [“Stopping DSFS” on page 21](#) is also updated. (APAR OA65560, which applies to z/OS 3.1 and z/OS 2.5)

September 2023

- Support is added for extended attributes. For more information, see [Chapter 6, “Extended attributes,” on page 37](#).

Changed

The following content is changed.

September 2024 refresh

- Clarification is added to documentation about the DSFS QUERY reports. See [“MODIFY QUERY reports” on page 65](#).

May 2024 refresh

- The instructions are updated in [“Steps for creating a BPXPRMxx entry for DSFS” on page 16](#).

April 2024 refresh

- [“Starting DSFS” on page 21](#) is updated with additional information.

February 2024 refresh

- [“Steps for installing and configuring DSFS” on page 15](#) is updated.

December 2023 refresh

- A recommendation is added to [“Steps for creating a BPXPRMxx entry for DSFS” on page 16](#).

October 2023 refresh

- Minor updates are made.

September 2023

- Instructions for ServerPac installation are updated. For more information, see [“Steps for installing and configuring DSFS” on page 15](#).

Deleted

The following content is deleted.

September 2023 release

- None.

Part 1. DSFS administration guide

This part of the document discusses guidance information for Data Set File System (DSFS).

Chapter 1. Overview of z/OS Data Set File System

With z/OS Data Set File System (DSFS), z/OS UNIX applications can access data sets by presenting the data sets as a tree-structured file system that is mounted at mount point `/dsfs` in the z/OS UNIX file system tree. A *utility file system* is used by DSFS to contain POSIX information about the data sets accessed by applications to assist in the presentation of data sets as a tree in the z/OS UNIX file system space.

Colony address space

DSFS runs as a physical file system (PFS) in a z/OS UNIX colony address space. It cannot run inside the z/OS UNIX address space. For more information about colony address spaces, see [Running a physical file system in a colony address space](#) in *z/OS UNIX System Services Planning*.

Supported data sets

DSFS supports the following data sets:

- Fixed and variable-length record physical sequential data sets (PS)
- Partitioned data sets (PDS)
- Partitioned data set extended (PDSE)

DSFS does not support variable-spanned record data sets. It does not allow access to encrypted data sets unless the utility file system data set is also encrypted. DSFS supports only cataloged data sets. If the catalog contains more than one entry for a data set, DSFS will use the entry in the master catalog. It does not support aliases, VSAM data sets, or migrated data sets. Because those data sets are not shown in the directory tree, they will not be accessible to DSFS users.

Case sensitivity

DSFS is a case-insensitive physical file system. File names and directory names are returned in lowercase. It also converts any input name to lowercase and stores it as lowercase. When DSFS interacts with the z/OS system through DFSMS, it converts the names to uppercase.

When specifying names in the DSFS tree, use lowercase. If uppercase names are specified, unpredictable results can occur. For example, an **ls** command with a wildcard and uppercase characters will result in failed pattern matching.

Security

DSFS runs with the requesting application user credentials for all access to data sets. For those cases where DSFS might cache data to reduce access calls to data sets, DSFS will make specific SAF calls to verify that the user has the required authority for all accesses to cached data. DSFS does not use z/OS UNIX security protocols.

Utility file system

DSFS requires the administrator to define a linear data set by using the `DEFINE CLUSTER` command with the `ZFS` keyword to define a utility file system. This file system is used to store the directory tree that is used to represent the accessed data sets as a file system tree and will cache POSIX byte-stream representations of the data sets that are being read and written by applications. Each system in a shared file system will use its own utility file system for its processing. If the user defines a key label for the data set, then DSFS encrypts the data that it stores in the utility file system. DSFS also allows an option for transparent data compression of the files that it stores in its utility file system.

Parameters file (IDFFSPRM)

DSFS allows specification of a parameters file or allows for parmlib search to be used to specify parameters that control DSFS behavior. Many of these parameters can also be dynamically modified while DSFS is running.

The dsadm commands

DSFS provides commands that can query utility file system usage or statistics for a particular file or directory. Commands are also provided to enable the administrator to configure and monitor DSFS.

Components of the DSFS tree

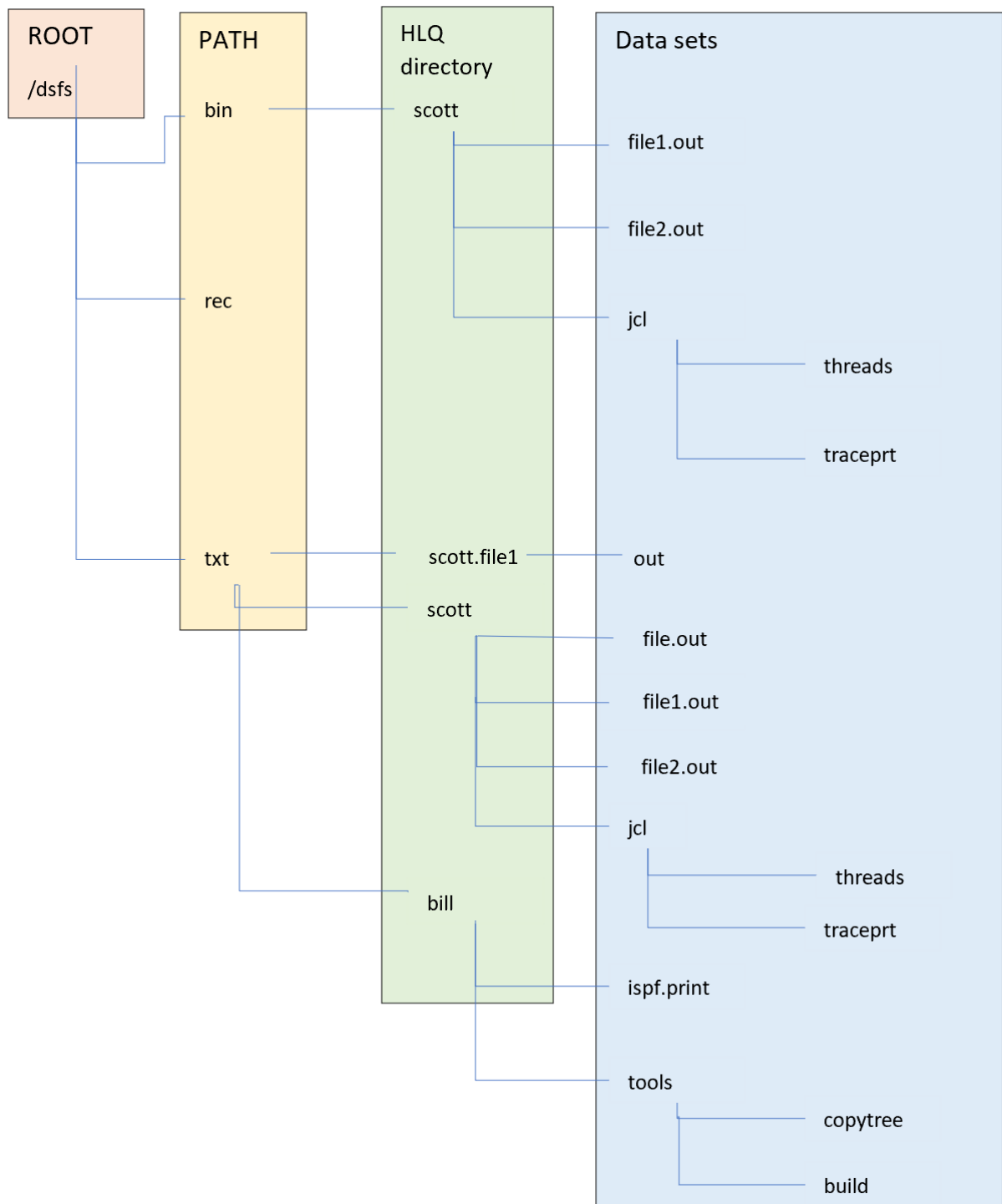


Figure 1. DSFS directory structure

The DSFS tree has four conceptual levels: root, path, HLQ directory, and data sets.

Root directory

Mount the utility file system at `/dsfs` in the z/OS UNIX file system tree, which results in the root of the DSFS tree at path `/dsfs`. Because this directory is read-only, users cannot store files or directories in it.

DSFS places the /txt, /bin, and /rec path directories in this directory at mount time if they are not already present.

All DSFS users have access to the root directory.

Path directory

The path directories determine the processing mode of data sets that are accessed through that path. These directories are automatically created by DSFS if they do not exist in the utility file system. Three processing modes (binary, record, and text) are available with DSFS. All users of DSFS are allowed access to a path directory.

Binary (/bin)

Data sets that are accessed by this path are treated by DSFS as binary data. When a data set is opened by DSFS on behalf of users who are accessing the data set through this path, DSFS reads the records from the data set. Next, it stores the records sequentially in a file in the utility file system as a POSIX byte stream. Updates made by applications to this file are stored in the utility file system file. When the application closes the file, DSFS writes the data to the data set, storing the bytes into sequential records in the data set.

- For variable-length records, the maximum record size is used for each record except for the last record, which can be partial.
- For fixed records, the last record is padded with binary zeros if the last record was short.

Record (/rec)

Data sets that are accessed by this path are treated by DSFS as z/OS record format files. When it accesses a data set, DSFS prepends a 4-byte header to each record it reads from the data set and stores the updated records into the POSIX file that is used to represent the data set.

- For fixed record data sets, each header must specify a length equal to the LRECL of the data set.
- For variable-length data sets, each header is used to determine the size of the record that is written to the data set.

Text (/txt)

Data sets that are accessed by this path are treated by DSFS as text files. Upon access to a data set, DSFS reads each record from the data set. If it is in fixed format, it strips the trailing blanks. For both fixed and variable record formats, it appends a newline character to the end of the record in the POSIX byte stream. Updates made by applications to this file are stored in the corresponding utility file system file. When applications close the file, DSFS scans the file's byte stream for newline characters, where each newline character delineates records to be written to the data set. The newline character is not written to the data set. For fixed-length record data sets, DSFS pads with blanks each record to ensure it matches the LRECL.

High-level qualifier directories

DSFS places a high-level qualifier directory name in a path directory if the user application changes the working directory or uses a path name that specifies the high-level qualifier directory name in that path. A HLQ directory name can consist of one or more qualifiers of a data set name. Once a HLQ directory has been created, it can only be deleted if the HLQ directory name is excluded by the HLQ list.

Restriction: New HLQ directories are created only when there is at least one existing PDS, PDSE, or sequential data set whose name begins with the HLQ directory name.

Tip: Users might want to use more than one qualifier to reduce the amount of catalog processing needed to populate or refresh the HLQ directories.

To control the data sets that DSFS can add to its tree, the administrator can create a list of excluded HLQ directory names by using both the HLQ_LIST parameter in the IDFFSPRM file and the **dsadm config** -hlq_list_add or -hlq_list_remove command. The user must have authority to the DFSMS catalog that contains the high-level qualifiers to access the HLQ directory. Upon first access, DSFS will populate the HLQ directory in its utility file system with names of supported data sets whose names begin with the

HLQ directory name and have at least one other qualifier. The supported data sets are fixed or variable record PS/PDS/PDSE.

Restriction: In order to add names of encrypted data sets that begin with the HLQ directory name, the DSFS utility file system must be encrypted.

The names that appear in the data set level of the DSFS directory tree do not have the HLQ directory name portion of the data set name. In [Figure 1 on page 5](#), physical sequential data set `scott.file1.out` is represented as a file called `file1.out` in the HLQ directory named `scott`. The sequential data set `scott.file1.out` could also be accessed as `scott.file1/out` if `scott.file1` were used as the HLQ directory name.

The same HLQ directory names can be accessed from multiple paths to allow different data sets of that HLQ directory name to be processed in different formats. Dynamic allocation (DYNALLOC) and ENQUEUE (ENQ) serialization is used to prevent the same data set from being processed by multiple paths at the same time. Similarly, this serialization is also used to prevent the same data set from being processed by multiple HLQ directories at the same time.

Creation parameters

With the **dsadm createparm** command, users can specify the data set attributes to use when PS or PDS/PDSE data sets are created. Creation parameters are assigned to an HLQ directory and apply to all data sets created with the same HLQ directory name.

These creation parameters are permanently stored in the utility file system and associated with the HLQ directory. They are permanently stored in the utility file system for the directory that represents the high-level qualifier directory name. The user ID of the issuer of the **dsadm createparm** command must match the HLQ in order to set or replace the creation parameters.

Data sets

The data sets belonging to an HLQ directory that is accessed by a DSFS user has a corresponding file (PS) or directory (PDS or PDSE) created inside the HLQ directory with the high-level qualifier directory name removed. Names appear in lowercase, but users can access the names by using uppercase, lowercase, or mixed-case because DSFS is case-insensitive.

Note: Using lowercase is preferable when accessing the names. If you use mixed case or uppercase, shell processing might not find the specified files or directories, especially when wildcards are used. For example, if you have a file that is named `my.file` in the POSIX shell, you will see the following lines:

```
# ls my*
my.file
# ls -l my*
-rwxrwxrwx 1 BPXROOT SYS1          1 Jul 23 00:00 my.file
# ls -l My*
ls: FSUM6785 File or directory "My*" is not found
# ls -l My.file
-rwxrwxrwx 1 BPXROOT SYS1          1 Jul 23 00:00 My.file
```

Any access to a data set is run under the requesting user's credentials to ensure that they have proper access. If access is made to a cached DSFS file system object (for example, a PS data set was read-in and stored in the corresponding file in the DSFS utility file system), then DSFS will make an explicit SAF call to ensure that the user has appropriate authority.

Users can create, rename, and remove data sets with the standard UNIX commands such as **rm** or **mv**.

Partitioned data sets (PDS and PDSE) are represented as directories in DSFS, with the members represented as files in DSFS. Users can read, write, rename, create, and remove members.

File processing

Memory cache

DSFS converts the contents of a physical sequential data set (or PDS/PDSE member) into its POSIX file equivalent based on the directory path that was used to access the data set. To avoid the overhead of transferring to and from the utility file system disks, DSFS attempts to cache the file in memory. It also avoids writing the contents to the utility file system. If the demand for the memory cache exceeds capacity, then DSFS writes out segments of files to the utility file system in a least recently used (LRU) fashion to meet demand. The administrator can use the `FILECACHE_SIZE` parameter to tailor the memory cache.

Retrieval

Upon first access to a data set, DSFS ensures that there is a file in the utility file system directory tree to represent it. It reads in the entire data set and converts it to a POSIX byte-stream file based on the path that it is accessed from.

Updates

When a sync operation is done or the last user closes the file after an application makes updates to the POSIX file that represents a data set, it is stored back to the data set by using the conversion method based on the path that was used to access the file. Both DFSMS and DSFS might issue messages if an error occurs when the file contents are stored.

SIGDSIOER signal at close

If errors are received during the storing of the file into the corresponding data set, DSFS raises a SIGDSIOER signal to the process. Raising the signal delivers an exception to the process, which often causes the process to be terminated. This action makes it clear that a problem occurred because many POSIX applications ignore the return code of the close function call.

Data format exceptions

When DSFS attempts to store the contents of a modified file in the utility file system back to the corresponding data set, it needs to scan the file contents for record boundaries. If a record is too large, then DSFS terminates the storing of the file, issues error messages that indicate the offset in the file with the problem and raises the SIGDSIOER signal. (For example, the record is larger than the LRECL of a fixed record data set or larger than the LRECL – 4 for a variable record data set.) Thus, not all POSIX access patterns can be handled by DSFS.

Restriction: DSFS cannot store sparse files and records that are too large back into the data set.

Application considerations for z/OS UNIX

For applications that scan files in the z/OS UNIX file system tree, ensure that they can tolerate the additional support of DSFS accessing data sets with z/OS UNIX files. For example, the Tivoli® Asset Management Inquisitor might scan the entire z/OS UNIX file system tree and experience failures when it accesses DSFS files.

Directories

Refreshing the HLQ directory

DSFS searches the catalogs for the directory names upon first access. Each name that represents a DSFS-supported data set and has at least one other qualifier is added to the HLQ directory in the utility file system. Because z/OS users and programs outside of DSFS might add, rename, or remove data sets,

an HLQ directory is refreshed when read (for example the **ls** command) on a periodic basis. The time period is 60 seconds by default and is controlled by the `DIRECTORY_REFRESH_TIMEOUT` parameter. Thus, you might see a delay before DSFS notices a change to an HLQ directory that was made outside of DSFS when directory contents are listed.

HLQ directory names search validation

Because z/OS users and programs outside of DSFS might add, rename, or remove data sets, DSFS searches the catalogs on path name search calls to DSFS even if the specified HLQ name is not listed in its cached POSIX directory. DSFS does the search to ensure that a DSFS user does not wait to access a new data set when it is created outside of DSFS. DSFS also performs validation to ensure that it is the same type of data set (file or directory) in case a data set name is deleted and re-created as a different type of data set. DSFS automatically corrects its directory contents as required when changes are made outside of DSFS.

Creating new data sets

DSFS uses the z/OS UNIX BPXWDYN function to create new data sets. For more information about BPXWDYN, see [BPXWDYN: A text interface to dynamic allocation and dynamic output in z/OS Using REXX and z/OS UNIX System Services](#).

DSFS provides the **dsadm createparm** command. With that command, users can store a string permanently with an HLQ directory in a path directory, which will define the DFSMS parameters to use for data set creation in BPXWDYN format. The string persists across a system restart or an unmount and subsequent mount of the utility file system. Because creation parms are specific to HLQ directories, different HLQ directories can have different creation parms. You can update creation parms by issuing a new **dsadm createparm** call.

DSFS allows a model file creation parameter string for physical sequential data sets and a model directory creation parameter string for PDS and PDSE data sets. DSFS also provides a **dsadm fileinfo** command, which shows the values that are stored with an HLQ directory.

Removing and renaming data sets

DSFS allows its users to remove or rename data sets following the same rules as DFSMS for renaming or removing data sets. DFSMS does not allow the removal or rename of a data set (or data set member) if it is being used by a program or user in the system. This restriction includes users who are accessing these data sets through DSFS. Thus, DSFS cannot remove or rename an open data set or an open utility file system file that represents that data set.

Rename in POSIX is provided by the **mv** command. DSFS only allows a singular file or directory to be renamed in the current directory, and the new name must not currently exist in the directory. Thus, the **mv** command is a simple rename command in DSFS, not the full POSIX version move or rename command.

Holding dynamic allocations and enqueues on PDS and PDSE directories

DSFS tends to hold the dynamic allocations and enqueues on PDS and PDSE directories, which ensure that they are not deleted or renamed while DSFS users are accessing them. Like HLQ directories, PDS and PDSE directories might be updated outside of DSFS. DSFS periodically refreshes its cached directory contents every 60 seconds. The frequency of the refresh is controlled by the IDFFSPRM parameter `DIRECTORY_REFRESH_TIMEOUT`.

Serialization and caching

DSFS uses dynamic allocation and enqueues to serialize access between DSFS and users who are accessing data sets outside of DSFS. It also uses dynamic allocation and enqueues to serialize access between DSFS and users within DSFS who are accessing data sets by using different paths or different HLQ directories. If DSFS has the proper dynamic allocation and enqueues on a data set, it caches its

contents in the utility file system. If it must release the dynamic allocation or enqueues, it will no longer cache the contents of a data set or a PDS/PDSE directory listing.

DSFS obtains the following dynamic allocation and enqueues based on the object that is being accessed:

- For physical sequential data sets (PS), DSFS allocates the data set OLD if it is being updated, removed, or renamed by DSFS users. If it is only being read by DSFS users, it allocates the data set SHR.
- For PDS and PDSE directories, DSFS takes the following actions:
 - Obtains a DYNALLOC SHR ENQ on the PDS/PDSE data set itself. If the data set is being removed or renamed, the DYNALLOC ENQ is upgraded to OLD.
 - Obtains a DYNALLOC SHR ENQ for any PDS/PDSE members that are being accessed.
 - Obtains an SPFEDIT ENQ in EXCL mode if the member is being updated, removed, or renamed. If the member is being read, the SPFEDIT ENQ is obtained in SHR mode.
- For PDS only, DSFS makes a RESERVE call on the DASD volume that contains the PDS if it must update the PDS in any fashion. If the call fails, then DSFS cannot update the PDS and will return an error message.

Extended TIOT processing (XTIOT)

DSFS uses extended TIOT (XTIOT) processing if it is enabled on the system. Otherwise, DSFS will not use it but this greatly limits the number of objects that can be accessed by z/OS UNIX users through DSFS. For this reason, extended TIOT processing is recommended. If extended TIOT is not used and the TIOT limit is reached on the system, then users cannot access data set based directories or files in the DSFS tree. For more information about enabling or disabling extended TIOT processing, see [DEVSUPxx \(device support options\)](#) in *z/OS MVS Initialization and Tuning Reference*.

Caching data set contents

To ensure that data sets remain cached after the DSFS user processes data sets, DSFS will hold dynamic allocations and enqueues on data sets in case the user processes them again. This action is typical in a POSIX file system and improves performance by reducing calls to DFSMS. To control the time that enqueues are held on PDS and PDSE data sets, the administrator can use the PDS_ENQ_DURATION parameter. The default is 60 seconds. For PDS and PDSE members and PS data sets, the administrator can use the PS_DYN_DURATION parameter, which also defaults to 60 seconds.

Event notification facility (ENF)

The event notification facility (ENF) allows an authorized program to listen for the occurrence of a specific system event. DSFS uses it to listen for contention on a data set that is accessed by a user outside of DSFS. As soon as the data set is no longer in use by any user of DSFS, it releases its dynamic allocations and enqueues on the data set.

Security caching

DSFS calls DFSMS with the requesting user's credentials to ensure that the user has the proper security when opening and processing data sets. DSFS also caches the directory and file contents, which can reduce calls to DFSMS. But if that cached data is accessed by a user, an access check is required. In this case, DSFS calls the SAF product to determine the access that the user is allowed on the object and cache that result. It keeps the result of this query for a default of 60 seconds (controllable by the SECURITY_TIMEOUT parameter). Any access by the user is checked against the cached result to avoid future SAF calls until the timeout is reached. At that point, another SAF call is made to determine the access that the user has to the data set. Thus, there may be a short delay before DSFS notices that the security to the data set was changed.

UNIX security

DSFS does not use z/OS UNIX security. It will provide values for UNIX permissions (777) and owner ID and group ID, but they exist only for compatibility with z/OS UNIX. They are not used by DSFS to make security decisions and have no effect on DSFS functions.

z/OS UNIX attributes versus data set attributes

A z/OS UNIX file system will typically handle attributes such as `atime`, `mtime`, `ctime`, `link count`, and `file mode` in a POSIX-compliant manner. User applications can access these attributes with the **dsadm** command or standard z/OS UNIX commands or programming interfaces. For the POSIX files and directories representing data sets, DSFS must provide the same capability by mapping applicable data set attributes and storing them in its utility file system. In addition, DSFS supports retrieval and update of ISPF statistics for PDS and PDSE members even though some statistics might not have corresponding z/OS UNIX object attributes.

DSFS updates the data set attributes with values from the UNIX attributes at store time.

For more information about attributes for the utility file system, see [“dsadm fileinfo” on page 98](#).

UNIX attribute mapping

A z/OS UNIX file system will typically handle attributes such as `atime`, `mtime`, `ctime`, `link count`, and `file mode` in a POSIX-compliant manner. User applications can use the **dsadm** command to access these attributes command. They can also use standard z/OS UNIX commands or programming interfaces. For the POSIX files and directories representing data sets, DSFS must provide the same capability by mapping applicable data set attributes and storing them in its utility file system. In addition, DSFS supports retrieval and update of ISPF statistics for PDS and PDSE members even though some statistics may not have corresponding z/OS UNIX object attributes.

DSFS updates the data set attributes with values from the UNIX attributes at store time.

atime, mtime, ctime

- For PDS and PDSE members, DSFS maps the `moddate` and `modtime` ISPF attributes to the UNIX `atime`, `mtime`, and `ctime` attributes.
- For sequential data sets and PDS/PDSE, DSFS maps the `atime`, `mtime`, and `ctime` attributes to the last **alt** date in the corresponding catalog entry for the data set if it is used. Otherwise, it uses the last reference date for those fields.

Mode

DSFS sets the UNIX object type to a file for data set members and for PS data sets, and to a directory for PDS/PDSE data sets.

Create time

For new data set members, DSFS saves the date of the UNIX create time in the **cjdate** field of the ISPF statistics for the member. For PS/PDS/PDSE, it uses the **creation_date** from the catalog entry for the creation time.

Length and block size

These UNIX attributes are the space information for the object in the utility file system. They are the length of the data set object after it has been mapped to a UNIX object. For files, this means how large it is after the data was converted from the data set format to POSIX byte stream format, which is dependent on the path that is chosen to access the data set. The block size for DSFS is always 1 K and the number of blocks is always shown in 1 K units.

Link count

The link count for PS and PDS/PDSE members is always 1. For PDS/PDSE directories, it is always 2 and for HLQ directories it is equal to the number of subdirectories (PDS/PDSE data sets) that belong to the HLQ directory + 2.

There is no corresponding data set attribute for this z/OS UNIX attribute.

Data set attributes

Data sets have some attributes that do not have a similar POSIX attribute. DSFS supports ISPF statistics and ISPF extended statistics.

ISPF statistics

Following are some ISPF statistics and the actions that DSFS takes when PDS/PDSE members are updated:

Version/Modification

DSFS increments the modification number of the member each time it is stored by DSFS until the maximum value of 99 is reached. Members that are created by DSFS have an initial version of 1.

Initlines

This ISPF statistic is the initial number of lines of the member at creation time. DSFS sets this value for members that it creates. Otherwise, it does not change the value.

Curlines

The curlines statistic indicates the current number of lines. DSFS sets this value to the number of records that it writes during file store time. The *store time* occurs when all applications close the file that represents the data set.

Userid

DSFS stores the 8-byte alphanumeric user ID of the user who wrote to the data set.

ISPF extended statistics

DSFS also updates the extended ISPF statistics for a data set member if the ISPF_EXTENDED_STATISTICS parameter is enabled. In this case, it also updates the extended Initlines and Curlines statistics.

DSFS background tasks

DFSMS and many SAF calls require work to be performed by DSFS tasks. Direct DFSMS calls by z/OS UNIX users are not allowed. To facilitate this, DSFS runs much of the DFSMS calls and SAF calls on DSFS worker tasks.

- The *security pool* handles SAF calls and its size is controlled by the SECURITY_POOL_SIZE parameter.
- The *directory pool* handles catalog and PDS/PDSE calls. The size of the directory pool is controlled by the DIRECTORY_POOL_SIZE parameter.
- The *file IO* pool handles I/O to PS data sets and PDS/PDSE members and its size is controlled by the IO_POOL_SIZE parameter.

Tailoring high-level qualifiers

The administrator can use the HLQ_LIST and HLQ_MODE parameters to tailor the HLQ directories that can be accessed in its path directories, which can be **/txt**, **/bin**, or **/rec**.

- The HLQ_LIST lists the HLQ directory names to be excluded from DSFS user access.
- The HLQ_MODE parameter indicates that the list is a list of excluded HLQs.

For example:

```
HLQ_MODE=EXCLUDE
HLQ_LIST=JOE,MARY,TOM
HLQ_LIST=SCOTT.FILE1,SCOTT.FILE2
```

HLQ_LIST is an exclude list in which the HLQs JOE, MARY, TOM are not allowed in the DSFS tree. User access to those HLQs is prevented in DSFS no matter which path is chosen. Data sets with HLQ SCOTT can be accessed as long as their names do not begin with SCOTT.FILE1 or SCOTT.FILE2.

Program module support

DSFS supports executable PDS/PDSE modules but they must be defined with RECFM=U. Those members are called *program modules*. The data sets might also contain nonexecutable data members, which DSFS does not support. Program modules may also be referred to as *program objects* for PDSE data sets and *load modules* for PDS data sets.

- Program module data sets are only accessible from the /bin path of the DSFS file system tree.
- PDS data sets can simultaneously contain load modules and data members. DSFS only displays load modules. Other members are not displayed in the file system tree. If a PDS defined with RECFM=U does not contain any load modules, it will appear empty in the DSFS file system tree.
- PDSE data sets can contain either program objects or data members, but not both. If a PDSE data set is defined with RECFM=U, DSFS will display them only if they contain program objects. If a PDSE defined with RECFM=U contains nonexecutable data members, it is not displayed in the DSFS file system tree.
- PDSE program objects cannot be placed into PDS data sets if they incorporate program management features. When you attempt to execute this type of operation, you will receive the following error:

```
IEW2606S 4B39 MODULE INCORPORATES PROGRAM MANAGEMENT  
FEATURES AND CANNOT BE SAVED IN LOAD MODULE FORMAT
```

- Program modules are not compatible with encrypted data sets. An encrypted UTFS can still be used to access load modules in unencrypted data sets.

When you create data sets defined with RECFM=U to contain program modules, you need to consider what constitutes a valid data set that is compatible with program modules. For more information, see [Starting the binder in z/OS MVS Program Management: User's Guide and Reference](#).

Chapter 2. Installing and configuring DSFS

z/OS Data Set File System (DSFS) is a base element of z/OS. To use the DSFS support, you must configure the support on the system. Configuration includes the following administrative tasks:

- Setting up DSFS to run in a colony address space.
- Defining the DSFS physical file system to z/OS UNIX.
- Creating or updating the DSFS parameter data set IDFFSPRM.
- Defining the DSFS utility file system.
- Creating the mount point for DSFS.
- Adding a MOUNT statement in the BPXPRMxx member to mount the DSFS utility file system at IPL.

Steps for installing and configuring DSFS

Two methods of installing z/OS are provided with your z/OS license: z/OSMF portable software instance (ServerPac) and CBPDO. For each of these, *z/OS Planning for Installation* describes what IBM does for you, what you receive from IBM, and what actions you need to take.

In your z/OS order, DSFS uses the following target and distribution libraries. The target libraries are as follows:

Target	Member type
SIEALNKE	LMOD
SIOEEXEC	EXEC
SIOEMJPN	MSG
SIOEPROC	PROC

The UNIX System Services path is as follows:

SIOEHLMDL (/usr/lpp/dfs/global/bin/IBM)

The distribution libraries are as follows:

AIEALNKE
AIOEEXEC
AIOEMJPN
AIOEPROC
AIOESAMP

Steps to follow:

1. Prior to the initialization, ensure that the load library (hlq.SIEALNKE) is APF-authorized and in the link list.

The sample file examples are installed in hlq.SIOESAMP.

The sample PROC examples are installed in hlq.SIOEPROC.

2. Add these RACF commands. These commands define what are referred to as the DSFS user ID. For more information about specifying DSFS user IDs, see [“Defining the DSFS user ID” on page 18](#).

```
ADDGROUP DSFSGRP SUPGROUP(SYS1)
ADDUSER DSFS DFLTGRP(DSFSGRP) AUTHORITY(USE) UACC(NONE)
RDEFINE STARTED DSFS.** STDATA(USER(DSFS))
```

```
SETRPTS RACLIST(STARTED)
SETRPTS RACLIST(STARTED) REFRESH
```

-
3. Create a JCL PROC for the DSFS started task in SYS1.PROCLIB by copying the sample PROC.

When you are done, you have completed the installation.

Steps for creating a BPXPRMxx entry for DSFS

As part of installing and configuring DSFS, you need to create a BPXPRMxx entry for DSFS. This entry defines the DSFS physical file system to z/OS UNIX.

To create an entry in the BPXPRMxx parmlib member for DSFS, follow these steps:

1. Add a FILESYSTYPE statement to the BPXPRMxx parmlib member. For example:

```
FILESYSTYPE TYPE(DSFS) ENTRYPOINT(IDFFSCM) ASNAME(DSFS)
```

If there is no BPXPRMxx parmlib member that contains a FILESYSTYPE statement for DSFS, you might receive message BPXF218I.

To store data state allocation above the 16 MB line, specify SWA(ABOVE) in the BPXPRMxx parmlib member. For more information about the SWA parmlib statement, see [BPXPRMxx \(z/OS UNIX System Services parameters\)](#) in *z/OS MVS Initialization and Tuning Reference*.

-
2. Update the IEASYSxx parmlib member to contain the OMVS=(xx,yy) parameter for future IPLs.

-
3. **Recommended:** To eliminate the possibility of the JES spool filling up from output lines associated with DSFS, specify SUB=MSTR as in the following example:

```
FILESYSTYPE TYPE(DSFS) ENTRYPOINT(IDFFSCM) ASNAME(DSFS, 'SUB=MSTR')
```

With that specification, DSFS will not run under JES control or interfere with JES shutdown.

-
4. **Recommended:** At this point, you might not know whether you want to change some configuration options. In case you would like to change them later, you should create an "empty" configuration file now. Do not specify any options in the empty configuration file. It must contain only one line, a comment that is an asterisk (*) in column 1.

To create the configuration file, choose one of the following methods.

- **Use a DD card (IDFZPRM) to identify the data set.** The IDFZPRM DDNAME identifies the optional DSFS configuration file. For now, it is suggested that this DD refer to a PDS with a member called IDFFSPRM that has a single line that begins with an asterisk (*) in column 1. Subsequent modifications can be made to the IDFFSPRM member. For more information, see [“Using an IDFZPRM DD file in the DSFS PROC”](#) on page 17. You can modify the IDFZPRM member at any time.
- **Put the options in the IDFPRMxx parmlib member.** If you select this option, you cannot have an IDFZPRM DD statement in the DSFS PROC. In the BPXPRMxx parmlib member, specify the suffixes in the FILESYSTYPE statement of the IDFPRMxx members that contain the DSFS configuration options. The following example would be used if the configuration options were in members IDFPRM01, IDFPRM02, and IDFPRM03.

```
FILESYSTYPE TYPE(DSFS) ENTRYPOINT(IDFFSCM) ASNAME(DSFS, 'SUB=MSTR') PARM('PRM=(01,02,03)')
```

For more information about the different methods of specifying a DSFS configuration file, see [“\(Optional\) Steps for creating or updating the DSFS configuration file”](#) on page 17. For more information about the configuration options for the DSFS PFS, see [Chapter 12, “The DSFS configuration file \(IDFPRMxx or IDFFSPRM\),”](#) on page 113.

5. **Recommended:** To mount the DSFS utility file system at system IPL time, add a MOUNT statement. For example:

```
MOUNT FILESYSTEM('UTILITY.FS') TYPE(DSFS) MOUNTPPOINT('/dsfs') MODE(RDWR)
```

For information about creating a utility file system, see [“Steps for creating the utility file system” on page 17](#).

Steps for creating the utility file system

DSFS requires a utility file system, which is defined by the IDCAMS DEFINE CLUSTER command. This linear data set must be defined with the ZFS keyword and be mounted RDWR. If you want access to encrypted data sets, then define it with a key label where the DSFS user ID has access. For more information about DEFINE CLUSTER, see [DEFINE CLUSTER](#) in *z/OS DFSMS Access Method Services Commands*.

Restriction: DSFS utility file systems are linear data sets defined with the ZFS keyword, but once a utility file system is used by DSFS, it cannot be used by zFS. A file system that is mounted to zFS can never be used as a utility file system.

For more information about working with the utility file system, see [Chapter 4, “Managing the utility file system,” on page 23](#).

(Optional) Steps for creating or updating the DSFS configuration file

The DSFS configuration file (IDFPRMxx or IDFFSPRM) is optional. To specify the file, either use IDFPRMxx in the parmlib or use an IDFZPRM DD statement in the PROC that is used to start the address space where DSFS is running.

It is recommended that you create an empty configuration file in case one is needed in the future. Options are only required in the configuration file if you want to override the default DSFS options.

Important: : Do not specify any options in the empty configuration line. It must contain only one line. That line is a comment, which is an asterisk (*) in column 1.

Specifying IDFPRMxx in the parmlib

As the preferred alternative to the IDFZPRM DD statement, you can specify a configuration file as a true parmlib member.

- The member has the name IDFPRMxx, where xx is specified in the parmlib member list.
- Omit the IDFZPRM DD statement in the PROC that is used to start the address space in which DSFS will run.

For more information about specifying the IDFPRMxx member to be used when DSFS is initialized, see [Chapter 12, “The DSFS configuration file \(IDFPRMxx or IDFFSPRM\),” on page 113](#).

Using an IDFZPRM DD file in the DSFS PROC

If you use the IDFZPRM DD statement, the data set to which it points must have a record format of FB with a record length of 80. The block size can be any multiple of 80 that is appropriate for the device.

A sample IDFFSPRM is provided in hlq.SIOESAMP(IDFFSPRM). For a description of the IDFFSPRM options, see [“Processing options for IDFFSPRM and IDFPRMxx” on page 114](#). Update the IDFZPRM DD statement in the DSFS PROC to contain the name of the IDFFSPRM member, as shown in the following example:

```
IDFZPRM DD DSN=SYS4.PVT.PARMLIB(IDFFSPRM),DISP=SHR
```

If you are running a sysplex, you must have different DSFS configuration files for different systems. In this case, you should also specify a system qualifier in the data set name in the IDFZPRM DD, as shown in the following example:

```
IDFZPRM DD DSN=SYS4.&SYSNAME..PARMLIB(IDFFSPRM),DISP=SHR
```

For more information about the DSFS configuration options file, see [“Processing options for IDFFSPRM and IDFPRMxx” on page 114.](#)

Defining the DSFS user ID

You can specify DSFS as the user ID, or you can specify a user ID other than DSFS to run the DSFS started task if it is defined with the same RACF characteristics as in [“Steps for installing and configuring DSFS” on page 15.](#) For example, you could use the ZFS user ID since it would likely have the same RACF characteristics.

- The DSFS user ID must have at least ALTER authority to any VSAM linear data sets that are used as a utility file system.
- The DSFS user ID must also have at least READ access to any CSFKEYS profiles for any utility file systems with key labels.

If ICSF is configured with CHECKAUTH(YES), the DSFS user ID must also have at least READ access to the CSFKRR2 CSFSERV profile. For more information about the CSFKEYS and CSFSERV profiles and the encryption of data sets, see [Data set encryption in z/OS DFSMS Using Data Sets.](#)

As an alternative to permitting the DSFS user ID to all the necessary security profiles, you can assign the TRUSTED attribute to the DSFS started task.

Creating the root directory

The mount point for DSFS is /dsfs. If this directory does not exist in the root file system, then it must be created. Use a permission setting of 755.

Coexistence and functionality APARs for DSFS

z/OS 3.2

In z/OS 3.2, you do not have to apply any coexistence APARs after you complete the installation steps in Chapter 2, [“Installing and configuring DSFS,” on page 15.](#)

z/OS 3.1

You do not have to apply any coexistence APARs for z/OS 3.1.

Functionality APARs are available for z/OS 3.1.

OA65560

Enables access to JES spool data through DSFS. For more information, see Chapter 9, [“JES spool data sets,” on page 55](#) and [“dsadm jobid” on page 108.](#) (OA63902: NEW FUNCTION - Exploit new extended attribute support supplied by OA62733 (www.ibm.com/support/pages/apar/OA63902))

z/OS 2.5

You do not have to apply any coexistence APARs for z/OS 2.5.

Functionality APARs are available for z/OS 2.5.

OA63902

Extended attributes support. See Chapter 6, “Extended attributes,” on page 37. (OA63902: NEW FUNCTION - Exploit new extended attribute support supplied by OA62733 (www.ibm.com/support/pages/apar/OA63902))

OA65560

Enables access to JES spool data through DSFS. For more information, see Chapter 9, “JES spool data sets,” on page 55 and “dsadm jobid” on page 108. (OA63902: NEW FUNCTION - Exploit new extended attribute support supplied by OA62733 (www.ibm.com/support/pages/apar/OA63902))

OA63218

Multiple qualifiers can be represented as a single directory. For more information, see “High-level qualifier directories” on page 6. (OA63218: NEW FUNCTION - Allow multiple qualifiers to be represented as a single directory (www.ibm.com/support/pages/apar/OA63218))

OA64023

Support for the UTF8_NAME parameter in the IDFFSPRM configuration option. For a description of UTF8_NAME, see “Processing options for IDFFSPRM and IDFPRMxx” on page 114. (OA64023: Unable to use DSFS with system names that begin with a numeric character (www.ibm.com/support/pages/apar/OA64023))

Chapter 3. Managing DSFS processes

Managing DSFS processes includes starting and stopping DSFS, as well as determining DSFS status.

Starting DSFS

DSFS is started by z/OS UNIX, based on the FILESYSTYPE statement for DSFS in the BPXPRMxx parmlib member.

Requirement: DSFS must run in its own colony space. Before it can be started in its own colony address space, a DSFS PROC must be available.

DSFS can be started at IPL if the BPXPRMxx parmlib member is in the OMVS=(xx,yy) list in the IEASYSxx parmlib member. To start it later, use the SETOMVS RESET=(xx) operator command. To start DSFS and mount the utility file system, use the SET OMVS=(xx) operator command that has both the DSFS FILESYSTYPE statement and the MOUNT command for DSFS.

Stopping DSFS

DSFS stops automatically when you shut down z/OS UNIX. To shut down an LPAR or to reIPL an LPAR, use the MODIFY OMVS,SHUTDOWN operator command to shut down z/OS UNIX. This action synchronizes data to the utility file system and unmounts it from the local system. DSFS is still accessible on other systems where it is defined and running. For shutdown procedures that use F OMVS,SHUTDOWN, see *Planned shutdowns using F OMVS,SHUTDOWN in z/OS UNIX System Services Planning*. That section includes a link to the steps ([Steps for shutting down z/OS UNIX using F BPXOINIT,SHUTDOWN=...](#)). You can also stop DSFS with the MODIFY OMVS,STOPPFS=DSFS operator command.

Tip: If DSFS is being used to access JES spool data sets when running SUB=MSTR, you should shut down DSFS prior to shutting down or terminating JES. Otherwise, a hang can occur when DSFS is being shut down.

DSFS unmounts its utility file system during shutdown. Shutting down DSFS on one system in a shared file system environment does not affect DSFS on other systems. When DSFS is stopped, you receive the following message after replying Y to message BPXI078D:

```
nn BPXF032D FILESYSTYPE DSFS TERMINATED. REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO IGNORE.
```

When an LPAR is shut down without the orderly shutdown of DSFS, recovery actions are likely necessary to bring the DSFS utility file system back to a consistent state. In addition, some file activity might be lost. To restart DSFS, reply `r` to message `nn`. For example:

```
r 1,r
```

If you want DSFS to remain stopped, reply `i` to remove the prompt. In this case, DSFS can be redefined later with the SETOMVS RESET=(xx) operator command. If you plan to restart DSFS, reply `r` to the message.

If DSFS has an internal failure, it does not terminate unless the utility file system is damaged or the failure would affect multiple subcomponents of DSFS. If the error might result in damage to the utility file system, DSFS will disable access to the file system and then restart.

- In a shared file system environment, z/OS UNIX locally remounts the DSFS file system automatically.
- In a non-shared file system environment, the operator must manually remount DSFS.

If the error is severe and might affect multiple internal DSFS subcomponents, DSFS automatically stops and then requests that z/OS UNIX restart it.

- For a shared file system environment, z/OS UNIX automatically remounts the utility file system.

- For a non-shared environment, z/OS UNIX does not remount the utility file system. The administrator must manually remount it.

If a severe error occurs and DSFS encounters more errors during restart, it might stop with message BPXF032D. It is the same message that you receive when the MODIFY OMVS,STOPPFS=DSFS operator command is used.

Determining DSFS status

To determine whether DSFS is active, issue the D OMVS,PFS command. The column titled **ST** (for **Status**) contains an A if DSFS is active. It contains an S (Stopped) if it is not.

You can issue D OMVS,P to display the state of the PFS, including the start or exit timestamp. Message BPXO068I returns the PFS in one of the following possible states:

- A** Active; the timestamp is the start time of the PFS.
- I** Inactive. When the PFS is inactive with no timestamp, the PFS address space has not yet started. When the PFS is inactive with timestamp, the PFS has to stop at that time.
- S** Stopped. It is waiting for a reply of R to restart or I to terminate the PFS.
- U** Unavailable.

Chapter 4. Managing the utility file system

DSFS requires a *utility file system* to assist it with presentation of z/OS data sets as a file system tree. The utility file system stores UNIX files and directories in a tree that represents data sets and directories accessed by z/OS UNIX users.

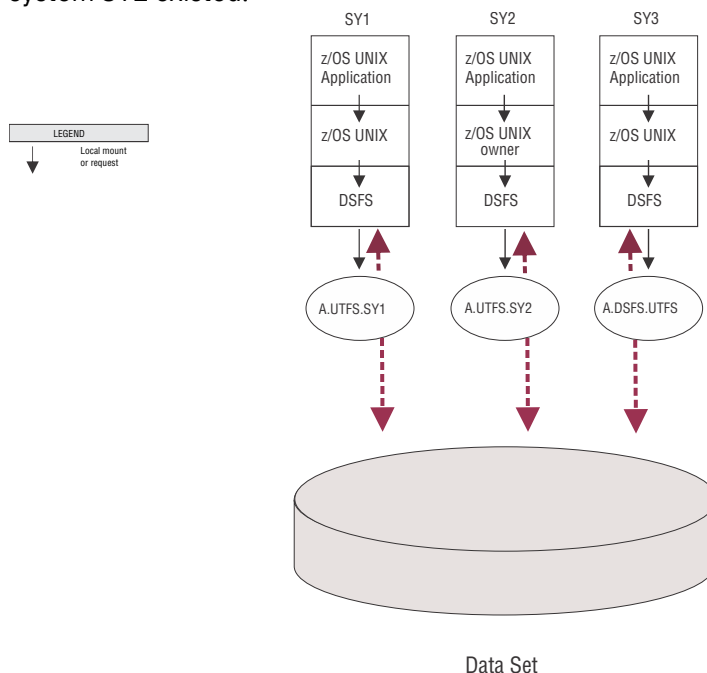
Important: Each active DSFS member in a sysplex must have its own utility file system.

To mount DSFS in the z/OS UNIX tree, mount the utility file system at mount point /dsfs. Even though each system in a shared file system environment will have its own utility file system, z/OS UNIX treats the file system mounted at /dsfs as a single file system.

Understanding the utility file system

The DSFS PFS does not use Sysplex Services to communicate with the DSFS PFS on another system. Therefore, the DSFS PFS is inherently sysplex unaware.

The following diagram depicts DSFS running on three systems in a sysplex, SY1, SY2, and SY3. Since the z/OS UNIX owner is SY2, the utility file system (UTFS) was mounted on SY2. z/OS UNIX sent mounts to SY1 and SY3. This configuration would be virtually the same in a single system environment where only system SY2 existed.



The diagram shows that application requests on each system are forwarded through z/OS UNIX to the local DSFS. There is no function shipping. To access z/OS data sets on a system, DSFS must be active on the system and a utility file system must be mounted. No function shipping will occur if the utility file system is not mounted on a system. In other words, if the utility file system is not mounted on a given system, DSFS will not be able to access any z/OS data sets on that system.

The diagram also depicts the naming conventions that are used for the utility file system. Each system running DSFS must have a VSAM linear data set defined for it to use as its utility file system. In the diagram, a utility file system with the name A.UTFS is mounted in the sysplex. Systems SY1 and SY2 are using the default naming convention. System SY3 is using the UTFS_NAME option in the IDFFSPRM configuration file. (For more information about defining these linear data sets and mounting them, see

“Naming convention” on page 27 and “Mounting and unmounting utility file systems” on page 28.) To mount the utility file system in the diagram, the following command would be used only on SY2:

```
MOUNT FI('A.UTFS') MOUNTPOINT('/dsfs') MODE(RDWR) TYPE(DSFS)
```

Because DSFS cannot access data sets without a utility file system, it should only be locally unmounted by shutting down DSFS (or OMVS) on a system. Do not unmount it with an unmount command. When DSFS (or OMVS) is restarted, the utility file system is remounted if z/OS UNIX delivers a mount to that system. Otherwise, another mount command will need to be issued to remount it.

To prevent an unintentional unmount of the utility file system, the mount command should not specify any AUTOMOVE options.

Defining the utility file system

The utility file system must be defined before it can be mounted. Use the IDCAMS program to define it. IDCAMS is the program name for access method services (AMS). For more information about IDCAMS, see [Using access method services in z/OS DFSMS Access Method Services Commands](#).

Restriction: DSFS utility file systems are linear data sets defined with the ZFS keyword, but once a utility file system is used by DSFS, it cannot be used by zFS. A file system that is mounted to zFS can never be used as a utility file system.

When you define the utility file system, consider the following factors:

Secondary space allocation

Provide the utility file system with a secondary extent allocation value to allow for dynamic growth of the data set if it becomes full.

Additional candidate volume. Allowing secondary extents on more than one candidate volume makes it more likely that the utility file system can be grown if the primary volume is full.

Key label

Any access to a z/OS data set that is encrypted requires the utility file system to be encrypted. The utility file system data set must be defined with a key label. It does not need to be the same key label as the data sets accessed through DSFS. If the data sets accessed through DSFS are not encrypted, then you do not need a key label on the utility file system data set.

Extended format. If a key label is used with the utility file system, then define the utility file system data set as *extended format*. Extended format records extra information in each control interval for improved integrity checking.

Extended addressability

If you want the utility file system to be larger than 4 GB, then use SMS DATACLASS for extended addressability.

Name

DSFS requires that each system have its own uniquely named VSAM LDS to use as a utility file system. By default, DSFS has a specific naming convention that requires the system name to be the last qualifier in the name of the data set. In a shared file system environment, the prior qualifiers must match the utility file system data set names that are being used by the other sysplex members. The default naming convention can be overridden by specifying the UTFS_NAME option in the IDFFSPRM file. The value of this option will be used as the full name of the VSAM LDS. For an example of utility file system data set names in a shared file system environment, see [“Naming convention” on page 27](#).

Additional requirements

- DSFS must also have access to the defined data set.
- The DSFS user ID must have at least ALTER authority for any VSAM linear data sets that is used as a utility file system.
- Utility file system data sets with key labels have additional requirements as described in [“Allowing for encryption of the utility file system” on page 26](#).

Allowing for dynamic growth of the utility file system

DSFS will attempt to dynamically grow a utility file system that is full. It extends and formats the utility file system in 25 MB increments as needed to satisfy demand for space on the utility file system.

- If the current extents are full, DSFS attempts to create a new extent (or extents) for the utility file system and format 25 MB of that extent or extents.
- If there is more space in the last extent to allow 25 MB to be formatted, then DSFS uses that space and avoids extending the data set. DSFS issues message IDFS00096I when attempting to grow the utility file system and message IDFS00095I if the file system is successfully grown. If the utility file system grow attempt failed, it issues message IDFS00094E.

To allow DSFS to extend the data set, a secondary space allocation is required, possibly with additional candidate volumes to allow extension if the primary DASD volume is full. An example of a definition that allows for dynamic growth is shown in the following figure:

```
//USERIDA JOB,'Multi-Volume',  
  
//    CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)  
  
//DEFINE EXEC PGM=IDCAMS  
  
//SYSPRINT DD  SYSOUT=H  
  
//SYSUDUMP DD  SYSOUT=H  
  
//AMSDUMP DD  SYSOUT=H  
  
//SYSIN DD  *  
  
        DEFINE CLUSTER (NAME(UTILITY.FS.SYSTEM1) -  
        VOLUMES(PRV000 PRV001 PRV002 PRV003 PRV004 -  
        PRV005 PRV005 PRV006 PRV007 PRV008 PRV009) -  
        DATACLASS(EXTATTR) -  
        ZFS CYL(2000 100))
```

Figure 2. Defining a utility file system with secondary extents

NAME

The VSAM cluster name to give to the utility file system. Guidelines to follow when naming the utility file system are described in [“Naming convention”](#) on page 27.

VOLUMES

The administrator specified multiple candidate volumes, which will allow for this utility file system to be grown much larger if needed.

DATACLASS

The administrator requested an SMS DATACLASS that allows for extended addressability, which enables the utility file system data set to be larger than 4 GB.

CYL (P, S)

The administrator provided a secondary extent value to allow dynamic growth of the data set.

ZFS

This keyword is required for utility file system definitions.

Adding volumes for a utility file system

You can add candidate volumes to an existing utility file system to allow the utility file system to be extended onto more volumes. Use the IDCAMS ALTER command with the ADDVOLUMES parameter to define the new volumes as in the following example.

```
//SUIMGVMAJOB (ACCTNO),'SYSPROG',CLASS=A,  
  
//    MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID  
  
//STEP01 EXEC PGM=IDCAMS  
  
//SYSPRINTDD SYSOUT=*  
  
//SYSIN DD *  
  
        ALTER UTILITY.FS.SYSTEM1.DATA -  
  
        ADDVOLUMES(* *)  
  
/*
```

Figure 3. Adding volumes to an existing utility file system

In this case, DFSMS is choosing the candidate volumes. If you want to specify the volumes, use their volume serial numbers in place of the asterisks. For more information about IDCAMS ALTER ADDVOLUMES, see [ALTER](#) in *z/OS DFSMS Access Method Services Commands*.

In DFSMS, if you use ALTER ADDVOLUMES to add volumes to a data set that was already opened and allocated, the data set must be closed, deallocated, reallocated, and reopened before VSAM can be extended onto the newly added candidate volume. This limitation means that the DSFS utility file system must be unmounted and mounted again. You can use the remount capability of z/OS UNIX to remount the utility file system. For more information, see [Remounting a mounted file system](#) in *z/OS UNIX System Services Planning*.

Allowing for encryption of the utility file system

Encrypting data in the utility file system is a requirement to allow DSFS to access any z/OS data set that is encrypted. If you plan on accessing encrypted z/OS data sets with DSFS, then you must assign a key label to the utility file system data set when you define it. The DSFS user ID must have a minimum of READ access to any CSFKEYS profiles for any utility file system with key labels. If ICSF is configured with CHECKAUTH(YES), the DSFS user ID must also have at least READ access to the CSFKRR2 CSFSERV profile. For more information about the CSFKEYS and CSFSERV profiles and the encryption of data sets, see [Data set encryption](#) in *z/OS DFSMS Using Data Sets*.

As an alternative to permitting the DSFS user ID to all the necessary security profiles, you can assign the TRUSTED attribute to the DSFS started task.

Assigning key labels to data sets

Before you assign a key label to a data set, these requirements must be met.

1. Integrated Cryptographic Service Facility (ICSF) must be active.
2. The key label must exist in ICSF.

To create a VSAM linear data set with a key label, use the IDCAMS command DEFINE CLUSTER command with the ZFS and KEYLABEL keywords. The ZFS keyword is required. The specification of a key label can be replaced with the specification of a data class that has a key label. For more information about the DEFINE CLUSTER command, see [DEFINE CLUSTER](#) in *z/OS DFSMS Access Method Services Commands*.

The following example shows the creation of a DSFS utility file system with a key label.

```
//USERIDA JOB,'Multi-Volume',
//    CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=H
//SYSUDUMP DD  SYSOUT=H
//AMSDUMP DD   SYSOUT=H
//SYSIN  DD   *

        DEFINE CLUSTER (NAME(UTILITY.FS.SYSTEM1) -
        KEYLABEL(PROTKEY.AES.SECURE.KEY.32BYTE)) -
        DATACLASS(EXTATTR) -
        ZFS CYL(2000 100))
```

Figure 4. Creating a utility file system with a key label

Tip: Extended format VSAM data sets record the encryption status for each control interval in the data set, which provides improved integrity checking. If you are using a key label, define the utility file system data set with the extended format option.

Naming convention

DSFS requires a unique VSAM linear data set on each system to use as its utility file system. To facilitate a single /dsfs mount point in a shared file system environment, DSFS adheres to the following rules for deriving the names of these linear data sets.

- If the UTFS_NAME option in the IDFFSPRM (or IDFPRMxx) configuration file is specified, its value is the name of the VSAM LDS to be used on this system. If the configuration file is shared with multiple systems, it is recommended that this value contain a system variable that will make the name a unique name in the shared file system environment.
- If the UTFS_NAME option does not have a value, the default name that DSFS constructs will be the value of the file system parameter on the MOUNT command, a period, and the local system name as the last qualifier. In Figure 4 on page 27, the system name is SYSTEM1. Hence, the value that is specified as the file system parameter on the MOUNT command must be UTILITY.FS in order for the VSAM linear data set UTILITY.FS.SYSTEM1 to be used as the utility file system name on SYSTEM1.

Mounting the utility file system

Mounting the utility file system means providing a name to z/OS UNIX that collectively represents the utility file systems that are being used. By default, the name that is used on the MOUNT command should be the VSAM LDS name without its last qualifier, which is the system name. If needed, this default can be overridden by specifying the UTFS_NAME option in the IDFFSPRM file. The value that is specified is the name of the VSAM LDS to be used on the local system as the utility file system name.

The shared file system environment

Only one MOUNT command is allowed for DSFS in a shared file system environment. However, each member will use its own utility file system VSAM LDS. The name of this data set is either the value of

the UTFS_NAME configuration option or the combination of the FILESYSTEM parameter on the MOUNT command and the system name as the last qualifier. If a new member joins the sysplex, or if DSFS is stopped and restarted on a member, z/OS UNIX automatically provides a mount to DSFS for this system. The mount uses the same FILESYSTEM value as the original mount command.

DSFS uses a separate utility file system data set on each system. When DSFS receives a mount request from z/OS UNIX, it determines the name of the local utility file system data set. Then, it attaches and mounts that data set for its use. For example, in a shared file system environment with member systems SYSTEM1, SYSTEM2 and SYSTEM3, and no UTFS_NAME configuration options specified on any members, the administrator should define three linear data sets such as UTILITY.FS.SYSTEM1, UTILITY.FS.SYSTEM2, and UTILITY.FS.SYSTEM3. The single MOUNT command would specify the name UTILITY.FS for the FILESYSTEM parameter.

If only two utility file system data sets are provided, DSFS is up on all three systems but the DSFS file system tree is accessible only from two systems. Requests from the third system are not forwarded.

Important: Do not issue unmount commands for the utility file system. The unmount can cause the utility file system on other members of the sysplex to also be unmounted.

Example of a MOUNT command

Using the preceding examples, the following example shows a mount of DSFS with the name prefix that is used for the utility file system data set.

```
MOUNT FILESYS('UTILITY.FS') MOUNTPoint('/dsfs') TYPE(DSFS) MODE(RDWR)
```

FILESYS

If a system in the sysplex has no value for the UTFS_NAME configuration option, this must be the name of the utility file system VSAM data set with its system name qualifier (that is, the last qualifier) removed. Otherwise, FILESYS can be any string accepted by the MOUNT command.

MOUNTPoint

The mount point and must be the value /dsfs.

TYPE

The type of the file system. It must be DSFS.

MODE

Because DSFS must be mounted in read/write mode, MODE must have the value RDWR. The MODE parameter defaults to RDWR if it is omitted.

Ownership of the DSFS file system

DSFS has a single mount in the sysplex. Therefore, it has a z/OS UNIX owner, which is initially owned by the system that initiated the mount in the sysplex. If DSFS abnormally terminates or is shut down on a sysplex member, z/OS UNIX moves its ownership of the file system. It does not forward requests from applications that are running on the system where DSFS was terminated. This behavior is unique to DSFS. DSFS itself has no owner because it has a separate utility file system for each sysplex member.

Mounting and unmounting utility file systems

When a utility file system is mounted or remounted, the following actions occur:

1. If the utility file system is being used for the first time, it is automatically formatted.
2. If the utility file system was cleanly unmounted the last time it was used, it is used as is.
3. If a salvage operation has marked the utility file system damaged, it is reformatted.
4. If the utility file system was not cleanly unmounted the last time it was used, it is searched for creation parameters. A *creation parameter* is a string in BPXWDYN format that is used to create a new file (PS) or directory (PDS/PDSE). If creation parameters are not found, the utility file system is reformatted.

If creation parameters are found, DSFS attempts to recover the utility file system in order to preserve them. If the recovery attempt fails, the utility file system is reformatted.

Changing the utility file system data set for a system

Because there is only a single mount point for DSFS in a shared file system environment, an unmount will remove access to the DSFS file system tree from all members in the sysplex. To change the utility file system data set on a single member while maintaining access to the DSFS file system tree on all other members, you must terminate DSFS on the one member.

The steps to moving to a new utility file system data set involve the following actions.

1. Stopping DSFS on the system with the `F OMVS,STOPPFS=DSFS` command.
2. Deleting the original utility file system data set (or renaming it).
3. Defining a new utility file system data set with the new DFSMS options following the required naming conventions. The name should follow the guidelines in [“Naming convention” on page 27](#).
4. Starting DSFS on the system by responding to the restart prompt for z/OS UNIX message BPXF032D.
5. Ensure that the utility file system is remounted.

Encrypting the utility file system

DSFS does not allow access to a data set that is encrypted unless the utility file system itself was defined with a key label.

If the utility file system data set has a key label, DSFS automatically stores any POSIX byte-stream representation of files in the utility file system in encrypted format. To do so, it uses the encryption key that is identified by the key label that is assigned to the utility file system data set. If the utility file system data set has a key label, it is known as an *encrypted utility file system*. When DSFS stores the data back for a file to its corresponding data set, it decrypts the data first. If the target data set itself is encrypted, DFSMS then encrypts it automatically using the encryption key that is specified by the key label of the data set.

Compressing transparent data

DSFS uses z/OS Enterprise Data Compression services (zEDC), if available. If the administrator requests compression by using the `COMPRESS=ON` parameter in the IDFP RMxx configuration file when the utility file system is first mounted, then DSFS will enable compression for all files that are stored in the utility file system. If zEDC is not available, DSFS cannot compress file contents in the utility file system and the data is stored uncompressed. zEDC compression can reduce disk space usage by an average of 65%, which reduces utility file system space usage.

Data compression affects only file space inside the utility file system. The utility file system also contains other data such as directory contents and internal control structures, which are never compressed.

Note: The `COMPRESS` parameter takes effect on the first mount of the utility file system and is saved for all subsequent usage. Removing this parameter before future mounts does not change whether the utility file system will be compressed. To enable or disable compression in this case, a new utility file system needs to be defined and used in place of the original file system.

Monitoring space in the utility file system

You can request that DSFS warn the administrator when the utility file system is low on space. The warning is important if the utility file system does not have secondary extent capability, which allows for dynamic growth. Use the **FSFULL**(*thresh, increment*) parameter in the IDFP RMxx configuration file where:

Thresh

The threshold, expressed as a percentage of usage, where DSFS issues message IDFS00043E to the system log warning that the utility file system is low on space. Message IDFS00044I is issued if file system space usage falls under this percentage.

Increment

The increment to the threshold value where DSFS repeats message IDFS00043E to again warn the administrator that the file system is low on space.

The default is OFF.

Example

If FSFULL (90,2) is specified, then DSFS warns the administrator with message IDFS00043E when the utility file system has used 90% of its space. It reissues the warning message when space usage reaches 92%, and again if it reaches 94% utilization until the file system is full. If space usage falls under 90%, it issues message IDFS00044I to indicate that is now under the threshold.

Displaying usage information for the utility file system

The **dsadm fsinfo** command displays information about the utility file system data set. An example of the output is as follows:

File System Name:		UTILITY.FS.DCEIMGHQ	
System:	DCEIMGHQ	Devno:	56
Size:	1440000K	Free 8K Blocks:	178156
Free 1K Fragments:	7	Log File Size:	14400K
Bitmap Size:	208K	Anode Table Size:	16K
File System Objects:	24	Version:	1.5
Overflow Pages:	0	Overflow HighWater:	0
Space Monitoring:	90,5		
ENOSPC Errors:	0	Disk IO Errors:	0
Status:	NE,NC		
File System Creation Time: Jan 27 16:47:41 2021			
Mount Time:		Feb 16 19:00:36 2021	
Last Grow Time:		n/a	
Legend: NE=Not encrypted, NC=Not compressed			

Figure 5. Example of output from **dsadm fsinfo**

Some of the information in the display is intended for service personnel. The following fields are the most important ones for the DSFS administrator.

File System Name

The full name of the utility file system data set.

Size

The size of the utility file system data set in kilobytes.

Free 8K Blocks

The number of unused 8K blocks in the utility file system (DSFS uses 8K blocks as a unit of store inside the utility file system). The lower this value, the higher the usage of the utility file system. If the value is 0, the utility file system is full.

File System Objects

The number of objects in the utility file system and hence the number of objects in the utility file system tree.

Space Monitoring

The FSFULL value that was specified in the IDFPRMxx configuration file or with the **dsadm config** command.

ENOSPC/Disk IO Errors

An indicator of how often DSFS had to fail an application because the utility file system ran out of space, or the disk media encountered IO errors.

Status

Shows whether the utility file system is encrypted or compressed and will have additional indicators for other file system events.

One important status relative to DSFS performance is the L indicator. If the L is shown, it means DSFS is low enough on space in the utility file system that it is using altered algorithms, which make it less likely the file system runs out of space but increases response times.

For more information about the **dsadm fsinfo** command, see [“dsadm fsinfo” on page 105](#).

Verifying and repairing the utility file system

To verify that the contents of the utility file system are correct and not corrupted, use **dsadm salvage**, which is a z/OS UNIX privileged user command. In the rare case that an internal error is suspected, the file system administrator can use this command to check the correctness of the contents inside the utility file system data set. Messages are sent to the system log to show the progress of the operation.

If you use the **dsadm salvage** command, the following actions take place:

1. Application write access to DSFS is suspended while the salvage operation is running.
2. The salvage operation verifies the contents of the utility file system data set and issues progress and status messages to the system log to show the results of its analysis.
3. If the utility file system was verified to be correct, then application write activity can resume without failure if the utility file system is mounted elsewhere in the sysplex.
4. If the verification failed, then the following steps take place:
 - a. All application access is failed.
 - b. DSFS marks the utility file system damaged.
 - c. DSFS stops and requests z/OS UNIX to restart it.
 - d. In a shared file system environment z/OS UNIX automatically remounts the utility file system. Otherwise, the administrator has to manually remount it. This remount will cause the utility file system to be reformatted and made available for use by applications.

For more information about the **dsadm salvage** command, see [“dsadm salvage” on page 111](#).

Chapter 5. Creating new data sets with DSFS

DSFS users can create new files and directories under the HLQ directory or data set portions of the DSFS tree. A new file represents either a PS or PDS/PDSE member. A new directory represents a PDS/PDSE.

- If the parent directory in the DSFS tree represents a PDS or PDSE, DSFS calls DFSMS to create the new member. The name must be 8 characters or less and follow DFSMS naming rules.
- If the parent directory is an HLQ directory in the DSFS tree, DSFS calls BPXWDYN to create the new data set. The type of data set created, whether PS or PDS/PDSE, can be specified by using creation parameters.

Saving creation parameters for data sets

With the **dsadm createparm** command, users can save two creation parameter strings for each HLQ directory, one to be used with creation of PS data sets and one to be used with creation of PDS/PDSE data sets. A *creation parameter string* is a string of BPXWDYN parameters. Multiple data sets can be created under an HLQ directory. DSFS uses BPXWDYN to create the data set. For information about the syntax and the allowed parameters in the stdin, see [BPXWDYN: A text interface to dynamic allocation and dynamic output in z/OS Using REXX and z/OS UNIX System Services](#).

A creation parameter string can be saved for new files (PS) and another creation parameter string can be saved for new directories (PDS/PDSE). These parameters are saved permanently in the utility file system for the HLQ directory. You can change these parameters by reissuing the **dsadm createparm** command. After the command is reissued, data sets that use different creation parameters can be created. Each path can have its own creation parameters for the same HLQ directory because those are represented as different directories in the utility file system directory tree. When data sets are created that can be accessed through multiple HLQ directories, these names will appear in those HLQ directories when they are refreshed according to DIRECTORY_REFRESH_TIMEOUT.

When DSFS receives a **createparm** command, it first calls BPXWDYN to ensure that the creation parameter string is valid. It also performs additional validation to ensure that a user is not allowed to create data sets that DSFS does not support. If those validations pass, then DSFS will save the creation parameter string in the utility file system data set and associate it with the HLQ directory in its tree.

The DSFS user uses the **dsadm createparm** command to save creation parameters for an HLQ directory. The user ID that issued the **dsadm createparm** command must match the HLQ. An example of saving the creation parameters for new files and directories is shown as follows:

```
> dsadm createparm -path /dsfs/txt/user1 -psmodel  
"cyl dsorg(ps) lrecl(300) recfm(vb) space(20,10) blksize(0)"  
  
> dsadm createparm -path /dsfs/txt/user1 -pdsmodel  
"cyl dsorg(po) lrecl(180) recfm(fb) space(20,10) blksize(0) dsntype(library)"
```

In the example, the user has issued two **dsadm createparm** commands for the USER1 HLQ directory in the txt path, one to save file creation parameters and another to save directory creation parameters. These parameters are explained as follows:

psmodel

In the first command, file creation parameters indicate that when an application attempts to create a file for HLQ USER1, a variable blocked (VB) data set should be created with 20 cylinders primary extent and a secondary size of 10 cylinders. The logical record length is 300 bytes and the system is allowed to choose the optimal block size for the records.

pdsmodel

In the second command, the directory creation parameters indicate that when an application attempts to create a directory for the HLQ USER1, then a fixed length blocked (FB) PDSE (because the **dsntype** is library) with a logical record length of 180 bytes should be created. The primary extent

size is 20 cylinders, with a secondary size of 10 cylinders and the system is again allowed to choose the optimal block size for the records.

DSFS restrictions to BPXWDYN text strings in creation parameters

DSFS imposes further restrictions to what is allowed in a BPXWDYN text string for creation parameters.

dsorg

DSFS supports only PS, PDS, and PDSE data sets. The only allowed values for **dsorg** are PS (physical sequential) or PO (PDS or PDSE).

recfm

DSFS supports only fixed and variable record data sets and not variable spanned. It accepts only F, FB, FBA, FS, FBS, V, VB, VBA, and U for **recfm** values. FS and FBS are supported only by PS data sets. U is supported only by PDS or PDSE data sets.

dsntype

DSFS supports the following types:

pdsmodel

- PDS
- LIBRARY | LIBRARY, 1 | LIBRARY, 2

psmodel

- BASIC
- LARGE
- EXTREQ
- EXTPREF

blksize and lrecl

When FB or FBS is specified for **recfm**, then **blksize** must be evenly divisible by **lrecl**.

If users attempt to create a file (hence a PS data set) in an HLQ directory that does not have file creation parameters, they will receive a failure with an EINVAL return code. If they create a directory in an HLQ directory (hence a PDS or PDSE) that does not have directory creation parameters, they will receive a failure with an EINVAL return code. Additionally, the resulting data set name must conform to DFSMS naming rules. (The data set name is the HLQ directory name added as a prefix to the name of the file or directory that the user is creating.)



CAUTION: Be careful when you save the creation parameters because BPXWDYN allows the creation of data sets that might not be usable. In this case, DFSMS will not allow the data set to be opened with the QSAM access method, which is what DSFS uses for file IO.

Displaying saved creation parameters

The user can query the creation parameters for an HLQ directory with the **dsadm fileinfo** command. This command shows the creation parameters (if saved) along with additional information about the object in the DSFS tree.

dsadm fileinfo is a general user command. Any user with read access to the object that is being queried can issue it. If the user is querying an HLQ directory, then the user must have authority to list data sets for that HLQ directory name. Example output from the **dsadm fileinfo** command is shown as follows.

```
# dsadm fileinfo -path /dsfs/txt/suimgea
path: /dsfs/txt/suimgea
fid          6,1          anode          49228,1776
length       8192        format          BLOCKED
1K blocks    8           dir tree status    VALID
PDS model anode 32,54    PS model anode 31,30
object type  DIR         object linkcount 21
object genvalue 0x00000000 dir version    1
dir name count 34        data set name type HLQ DIR
recfm        na         lrecl         na
data mode     TEXT      data set status  UNCACHED
data set name SUIMGEA
ENQ held      NO
direct blocks 0x00003497 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
                0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
indirect blocks none
mtime         Mar  1 14:33:40 2021  atime         Mar  3 10:30:59 2021
ctime         Mar  1 14:33:40 2021  create date   Feb  3 2021
not encrypted not compressed
PDS model          new lrecl(80) dsorg(po) recfm(fb) space(10,10) dir(2)
PS model          lrecl(80) dsorg(ps) recfm(fb)
vnode,vntok   0x00000050,,0x440001E0      0x01D999B0,,0x00000000
opens         oi=0          rd=0          wr=0
file segments na          file unscheduled na
meta buffers  0           dirty meta buffers 1
```

This example shows the **fileinfo** command output for the path /dsfs/txt/suimgea. The HLQ directory name (which is also the HLQ) is named SUIMGEA. The user in this case must have set both a directory model creation parameter and a file model creation parameter for this HLQ directory as shown by the **PDS Model** and **PS Model** output fields, which are highlighted in blue and underlined.

Chapter 6. Extended attributes

DSFS has objects that represent data sets in a z/OS UNIX file system tree format. Some of the attributes of these objects can be accessed as extended attributes by using the z/OS UNIX **extattr** command or in a REXX program with the xattr syscall commands.

DSFS supports the following attributes:

system.dstype

Type of UNIX object. If it is a file, it can represent a sequential data set or member of a PDS or a sysout file. If a directory, it can represent a PDS, PDSE, a HLQ directory or a JES job.

Subcommands: LIST, GET

Possible values: PDS, LIBRARY, MEMBER, SYSOUT FILE, JOB DIR, SEQ, HLQ DIR

system.dsname

The data set name if the UNIX object represents a data set. If not, the name of the HLQ or job directory.

Subcommands: LIST, GET

Possible values: The z/OS data set name the object represents, or directory name.

system.lrecl

The LRECL of the data set.

Subcommands: LIST, GET

Possible values: The LRECL number.

system.recfm

The RECFM of the data set.

Subcommands: LIST, GET

Possible values: A **recfm** value that DSFS supports.

system.allocseq

String of attributes used to create new SEQ data sets.

Subcommands: LIST, GET, SET, REMOVE

Possible values: The string is treated the same as **dsadm createparm** attribute strings, and is described in [“Creating new data sets” on page 9](#).

system.alloclib

String of attributes used to create new PDS or PDSE data sets.

Subcommands: LIST, GET, SET, REMOVE

Possible values: The string is treated the same as **dsadm createparm** attribute strings, and is described in [“Creating new data sets” on page 9](#).

system.jobcompletion

Status of a JES job completion. A job only has this attribute once it has completed its execution phase.

Subcommands: LIST, GET

Possible values: One of the following strings:

Normal completion cc: xxxx

Non-zero completion code cc: xxxx

ABENDED cc: xxxx

Failed with EOM cc: xxxx

Converter error

System failed
JCL error
Canceled
Converter ABEND
Security error

where xxxx is the hexadecimal maximum job completion code. If xxxx is a condition code, it is 4 decimal digits. If it is a system abend, it is the letter S followed by 3 hexadecimal digits. If it is a user abend code, it is the letter U followed by 4 decimal digits.

system.jobstatus

Status of a JES job.

Subcommands: LIST, GET

Possible values: one of the following strings:

<i>Table 1. Strings for system.jobstatus</i>	
Strings	Strings
Executing	MDS system select
Awaiting execution	Awaiting resource allocation
Awaiting setup	Awaiting unavailable volumes
In setup	Awaiting volume mounts
Input processing	MDS system verify
Awaiting conversion	Error during MDS
Converting	Awaiting output
Spin processing	Awaiting MDS restart
Awaiting output	MDS complete
Awaiting purge	Awaiting output
Purging	Awaiting output writer
Receiving NJE SYSOUT	Awaiting reserved services
Awaiting NJE job transmission	Output service complete
Transmitting NJE job	Awaiting demand selection
No subchain exists	Ending function waiting
FSS processing	Ending function not complete
Awaiting batch postscan	Maximum index value
Awaiting demsel postscan	Unknown
Awaiting volume fetch	None

To add or replace the creation parameters string for HLQ directory suimgfq.pslarge.dataset:

```
extattr --set system.allocseq --value 'dsorg(ps) dsntype(large) recfm(fb) lrecl(80)
blksize(27920) cyl space(5,1)' /dsfs/txt/suimgfq.pslarge.dataset
```

To retrieve the data set type for printjb1.job00057:

```
extattr --get system.dstype /dsfs/sysout/suimgfq/printjb1.job00057
JOB DIR
```

```
extattr --get system.dstype /dsfs/txt/suimgfq/private.file  
SEQ
```

To list all attributes that are assigned to HLQ directory `suimgfq.qual2.psbasic.dataset`:

```
extattr --list /dsfs/txt/suimgfq.qual2.psbasic.dataset
```

To remove the `psmodel` creation parameter from HLQ directory `suimgfq.qual2.psbasic`:

```
extattr --remove system.allocseq /dsfs/txt/suimgfq.qual2.psbasic
```


Chapter 7. Using the dsadm fileinfo command

Use the **dsadm fileinfo** command to view information about the DSFS file system object, and if the object represents a data set, information about the data set that is relevant to DSFS. The command display contains information that is relevant both to the user and service personnel.

Length information

The fields in the following example are the fields from the **dsadm fileinfo** command output that are related to the size and attributes of the object as stored in the utility file system data set.

length	8192
1K blocks	8
not encrypted	not compressed

The length in the **fileinfo** output indicates the number of bytes the object occupies inside the utility file system data set. In DSFS, *directories* are files that contain names of objects inside the directory and pointers to their location inside the utility file system.

- For HLQ, path and PDSE directories, the more names inside the directory, the larger it will be.
- For PS files and PDS/PDSE members, the length reflects the size of the file after it was converted to POSIX byte-stream format based on the path that was chosen to access the object. For files that have not been read and converted to POSIX byte-stream format, the length will be 0.

1K blocks indicates how much space is taken inside the utility file system, including internal control structures. DSFS uses additional control structures for larger objects on disk to locate sections of that object on disk and is the full measure of how much space that object occupies. For files (PS and PDS/PDSE members), DSFS attempts to keep data in its file cache and to avoid writing it to the utility file system. For that reason, **1K blocks** might be much smaller than the length would indicate due to this performance feature. If the file is fully cached in memory, this value might even be 0.

DSFS also shows the encryption and compression status for objects in the utility file system. The utility file system must have a key label as a prerequisite for any object to be stored in encrypted format inside the utility file system. Similarly, for compression, the COMPRESS=ON parameter in the IDFPRMxx file must have been set before the utility file system data set is formatted during its first use by DSFS. Only files will show up as encrypted or compressed. DSFS does not encrypt or compress directories that are stored in its utility file system. If a file is compressed, the number of kilobytes of disk space that is saved is shown.

Restriction: The **fileinfo** output only shows the encryption status of the DSFS object in the utility file system. It does not represent the encryption status of the data set the file represents.

z/OS UNIX information

The following example shows the z/OS UNIX-related fields from the **fileinfo** command output that are related to the type of object in the utility file system.

object type	DIR	object linkcount	2
dir name count	27	data set name type	PDS

DSFS presents data sets in a file system tree that simulates z/OS UNIX protocols. It maps data sets to these attributes as follows:

object type

One of two values.

- DIR (root directory, path directory, HLQ directory, or PDS/PDSE directory).
- FILE (PS or PDS/PDSE member).

object linkcount

Although data sets do not have link counts, z/OS UNIX objects do. The utility file system keeps the link count of directories equal to the number of subdirectories, and for PS and PDS/PDSE members the link count is always 1. Every z/OS UNIX directory has special entries for . (dot) and .. (dot dot) entries. PDS/PDSE directories will have a link count of 2 because the dot and dot dot entries are inserted into the utility file system directory that represents the PDS/PDSE.

dir name count

The number of objects in the directory, including the special dot and dot dot entries for directories. For files, the number is the value *na*.

data set name type

The type of data set that this DSFS object represents (PDS, LIBRARY, SEQ, MEMBER, or HLQ DIR).

Storing creation parameter information

As described in Chapter 7, “Using the dsadm fileinfo command,” on page 41, the user can store creation parameter information for an HLQ directory in the utility file system. The utility file system provides the DFSMS parameters that are needed to create a new data set. The **fileinfo** fields in the following example shows information that is relevant to creation parameters.

PDS model anode	34,68	PS model anode	31,58
PDS model	cyl dsorg(po) lrecl(2000) recfm(vb) space(20,10) blksize(0) dir(100)		
PS model	cyl dsorg(ps) lrecl(300) recfm(vb) space(20,10) blksize(0)		

PDS model anode

This field contains a pair of numbers, *loc,length*, where *loc* is location information that is intended for service personnel, and *length* is the length of the associated creation parameter string.

PS model anode

This field has the same format as PDS model anode, except that it is for the file creation parameters.

PDS model

The creation parameter string for new PDS/PDSE data sets in BPXWDYN.

PS model

The creation parameter for new PS data sets in BPXWDYN.

Creation parameters are only relevant to HLQ directories. All these fields would be *na* for any other type of object in DSFS. Creation parameters are stored in internal objects in the utility file system and are reflected in the **fsinfo** command output in the file system objects field.

Data set information

Data set information is related to objects that are data sets, data set members or HLQ directories. These fields would all be either the value *na* or 0 for DSFS-specific objects such as its root directory or one of its path directories (/txt, /bin or /rec).

recfm	V	lrecl	300
data mode	TEXT	data set status	RETRIEVED
data set name	SUIMGHQ.PRIVATE.DSFS.TRACE(PROOF)		
ENQ held	SHR,SYS00011,SUIMGHQ, Feb 17 19:04:04 2021		

The possible values for these fields depend on the type of data sets as follows:

recfm

The record format for a data set. It is the value for a utility file system object that represents a data set. If the object does not represent a data set, then the value is na.

lrecl

The logical record length of the data set. It is the value na for an HLQ directory or a directory that does not represent a data set such as a path directory (/txt, /bin, /rec).

data mode

The processing mode, which is also the path that is chosen in the DSFS file system tree to access the data set. The value could be TEXT, BINARY, or RECORD, depending on the path chosen.

data set status

For PS and PDS/PDSE members, this field indicates whether DSFS has retrieved the data set contents and converted them to a POSIX byte stream format. If true, then it has the value RETRIEVED. If not, then it has the value UNCACHED. For HLQ or PDS/PDSE directories, it has the value RETRIEVED. For other DSFS file system objects, the value na is shown.

data set name

Name of the data set, including member name if this is a member. For HLQ directories, this is the HLQ directory name, which consists of one or more qualifiers.

ENQ held

Indicates whether a dynamic allocation or ENQ is held on the data set. It is either the value NO, which means no ENQ is held, or in the form *mode,ddname,userid,timestamp* where:

- *mode* is the enqueue mode: OLD for exclusive or SHR for shared access.
- *ddname* is the dynamic ddname of the allocation.
- *userid* is the 8-byte user ID of the user who accessed the file that caused the allocation to be obtained.
- *timestamp* is the time when the ENQ was obtained.

Timestamps

DSFS attempts to use ISPF statistic timestamps and DFSMS catalog entry timestamps as the z/OS UNIX equivalents of *atime*, *mtime*, and *ctime*.

mtime	Feb 22 19:01:05 2021	atime	Feb 22 19:19:00 2021
ctime	Feb 22 19:01:05 2021	create date	Apr 4 2017

DSFS updates its corresponding utility file system object timestamps according to POSIX rules. The *atime* is updated after each directory or file read. The *mtime* and *ctime* fields are updated for each write or setting of object attributes. DSFS sets these values when it retrieves a data set or caches the contents of a PDS/PDSE directory listing. Upon access to a data set, DSFS sets time stamps in its UNIX objects as follows:

atime

This option is set to the last reference date if the object is a PS, PDS, or PDSE if available. Otherwise, it is set to the last alt date from the catalog entry. For PDS/PDSE members, this is set to the *moddate* and *modtime* that are available in the ISPF statistics for the member. If ISPF statistics are not available, it is set to the time that the object is accessed by DSFS.

mtime or ctime

These options are set to the last `alt` date of the object if it's a PS, PDS or PDSE if it is set. If they are not set, then these times are set to the time the object is accessed by DSFS. For members, they are set to the `moddate` and `modtime` if ISPF statistics are available. Otherwise, they are set to the time that the object is accessed by DSFS.

create date

This option is set to the creation date from the catalog if it's a PS, PDS, or PDSE if available. If not available, this option is set to the oldest of `atime`, `mtime`, or `ctime`, which often is the time that the object is accessed by DSFS. For members, this option is set to the creation date if ISPF statistics are available. Otherwise, it is set to the `moddate` if ISPF statistics are available. If the `moddate` in ISPF statistics is not available, it is set to the date that the object is accessed by DSFS.

Updates and references that are made by DSFS applications will update the `atime`, `mtime`, or `ctime` in the utility file system object. If the object was a file, then the associated data set times are updated at close time. For PS, PDS and PDSE, the DFSMS access methods will set the times automatically per their normal processing, though the times set for last alt date or last reference date might differ slightly from the DSFS times. DSFS will store the `mtime` for PDS/PDSE members (in `moddate` and `modtime`) when it stores the contents of its utility file system file into the PDS/PDSE member or when an application sets the `mtime` to a specific time.

Chapter 8. Monitoring and tuning performance

To help debug problems, DSFS provides several queries that allow for monitoring and tuning DSFS performance and functions. For information about monitoring and problem resolution specific to the utility file system, see [Chapter 4, “Managing the utility file system,”](#) on page 23.

The administrator can use display commands and tuning options to optimize for performance. Either MODIFY DSFS QUERY or the **dsadm query** command can be used to display DSFS information. The same information is displayed by both commands in most cases. The examples use the **dsadm query** command in most cases.

Several program options control DSFS caching and performance. To change an option dynamically, use the **dsadm config** command, which will change the option immediately. However, the change will not persist across restarts of the DSFS product. If the administrator has determined that altering a DSFS option provides value to their site, then the administrator should update the IDFFPRMxx parameter file by specifying a new value for that option.

Performance tuning

DSFS performance depends on many factors. DSFS provides performance information to help the administrator determine bottlenecks. The IDFFSPRM file contains many tuning options that can be adjusted. The output of the system MODIFY DSFS QUERY commands provide feedback about the operation of DSFS. This section describes those IDFFSPRM options and the operator commands that relate to performance.

DSFS performance can be optimized by tailoring the size of its caches to reduce I/O rates and pathlength. It is also important to monitor DASD performance to ensure that there are no volumes or channels that are pushed beyond their capacity.

Monitoring DSFS performance

You can monitor DSFS performance with the MODIFY command. The output from the MODIFY DSFS,QUERY command is written to the system log. The syntax of this command and an explanation of the *report* and their *option* values, if any, are shown as follows.

```
MODIFY DSFS,QUERY,<report>,<option>
```

ALL

Shows all of the reports. However, for the STOR report, the DETAILS option is OFF and the FILE report indicates only active file systems.

DATASET

Displays DSFS statistics about data sets.

DSCACHE

Displays thread pools for data set operations and security cache information.

FILECACHE

Provides performance information for the file cache including cache hit ratios, I/O rates, and storage usage.

IOBYDASD

Displays the I/O statistics by currently attached DASD volumes including the total number of waits for I/O and the average wait time per I/O.

KNPFS (or KN)

Provides counts of calls that are made to DSFS from z/OS UNIX and the average response time of each call. This information is the basic measure of DSFS performance.

LFS

Provides detailed file system statistics including the performance of the DSFS metadata cache, the vnode cache, and compression statistics.

LOCK

Provides a measure of lock contention and how often z/OS UNIX threads wait for certain events such as file cache reclaim.

STOR

Provides a detailed breakdown of DSFS allocated storage by component. By default, this report lists only storage usage by DSFS component. If you use the DETAILS option, you get more detailed information for each DSFS component.

FILECACHE report

DSFS provides access to data sets with a primary role of allowing applications to read and write these data sets in a UNIX file system environment. It provides a file cache where it attempts to cache its POSIX byte-stream conversions of data sets in memory to avoid paging to the utility file system. Therefore, the performance of the file cache is important. You can use the **dsadm query -filecache** command (or MODIFY DSFS,QUERY,FILECACHE command) to show the performance of the cache as shown in the following example.

```
# dsadm query -filecache
File Caching System Statistics
-----

External Requests:
-----
Reads      12824155    Fsyncs      0    Schedules      0
Writes     13504496    Setattrs    276    Unmaps         1475
Asy Reads   3939233    Getattrs    6452    Flushes        0

File System Reads:
-----
Reads Faulted      0    (Fault Ratio  0.000%)
Writes Faulted     90    (Fault Ratio  0.001%)
Read Waits         0    (Wait Ratio   0.000%)
Total Reads       90

File System Writes:
-----
Scheduled Writes      0    Sync Waits      0
Error Writes          0    Error Waits      0
Scheduled deletes      0
Page Reclaim Writes    0    Reclaim Waits    0
Write Waits           0    (Wait Ratio  0.000%)

Page Management (Segment Size = (64K) ) (Page Size = 8K)
-----
Total Pages      131072    Free      131072
Segments         3936
Steal Invocations  0    Waits for Reclaim  0

Number of cache spaces used:  16  Pages per cache space:  8192

Space Address      Total 8K Pages    Free Pages    Assigned Segments    Fix Type
-----
5029B00000         8192         8192         0    Not Fixed
502DC00000         8192         8192         0    Not Fixed
5031C00000         8192         8192         0    Not Fixed
5035D00000         8192         8192         0    Not Fixed
5039D00000         8192         8192         0    Not Fixed
503DD00000         8192         8192         0    Not Fixed
5041E00000         8192         8192         0    Not Fixed
5045E00000         8192         8192         0    Not Fixed
5049E00000         8192         8192         0    Not Fixed
504DF00000         8192         8192         0    Not Fixed
5051F00000         8192         8192         0    Not Fixed
5056000000         8192         8192         0    Not Fixed
505A000000         8192         8192         0    Not Fixed
505E000000         8192         8192         0    Not Fixed
5062100000         8192         8192         0    Not Fixed
```

5066100000	8192	8192	0	Not Fixed
------------	------	------	---	-----------

Many DSFS queries provide information that is useful for service personnel. They also contain information that is useful to both service personnel and the administrator. Following are the fields of interest to the administrator:

Total Pages

The total number of 8 K pages in the file cache. The default size of the file cache is MIN(10% of real memory, 1 GB) and in this example the default was taken.

Reads Faulted

The count of read requests to the cache where the data was not in the cache (paged out to the utility file system). In this example cache performance is good, no read faults. Read faults are workload-dependent. If the read faults are higher than 10% and system memory is available, the administrator can try increasing the FILECACHE_SIZE IDPRMxx setting or use **dsadm config -filecache_size** to immediately increase the size of the cache.

Page Reclaim Writes/Steal Invocations

These fields show the number of times a request to access data in the cache that is faulted and data had to be cast out of the cache to make room for the new requested data. Hence, the data was paged to the utility file system. These would increase along with the reads faulted value and show that demand exceeds capacity of the cache and paging to the utility file system is present.

DSCACHE report

Tasking

DSFS must dispatch user application requests to worker tasks in various pools because DFSMS must be called by DSFS tasks. If a worker task is not available, then the request is queued. The more this occurs the more the associated DSFS response time grows.

Security cache

DSFS also provides a security cache, which allows for caching of access that a user has to a data set. The more times a request is added to the queue, the slower the associated DSFS response time will be.

ENQ times

DSFS can only cache data set and directory contents if it holds the proper serialization (DYNALLOC and ENQs on the data set). The administrator can control how long an ENQ is held, which makes it more likely that DSFS has the data in its cache and will not need to convert data set contents to POSIX byte-streams (files). It also will not need to reread the PDS/PDSE directory or DFSMS catalog (directory).

DSFS provides the **dsadm query -dscache** (and the equivalent MODIFY DSFS,QUERY,DSCACHE command) to allow the administrator to measure DSFS worker pools, security cache and how frequently it has to call DFSMS to read data sets and catalogs. An example is as follows:

DSFS Data Set Caching and Thread Pool Statistics				
	Called/ Dispatched	Miss/ Queued	Miss/ QPCT	Resync/ Retrieve
IO Requests	959	0	0%	16
Dir Requests	67	0	0%	7
Sec Requests	2038	0	0%	na
Sec Cache Requests	980	22	2%	na

All the fields in this report are relevant to DSFS performance with a goal of minimizing the values in the third and fourth column of the report:

IO Requests

This field shows how often DSFS needed to dispatch work to its data set IO worker pool and how often it had to queue the request (which means longer response time). It also shows the Retrieve count (the number of times DSFS had to retrieve data set contents to convert to POSIX byte-stream format). If the queue count is excessive, the administrator can increase the `IO_POOL_SIZE` configuration option. If the retrieval count is high relative to the overall open count (from the DSFS KNPFS report), then increasing the `PS_DYN_DURATION` configuration option might help (workload dependent).

Dir Requests

This field shows how often DSFS needed to dispatch directory work to its directory IO worker pool and how often it had to queue the request. It also shows the catalog/PDS/PDSE directory retrieval count, which shows the number of times DSFS had to call DFSMS to read directory-related information. If the queue count is excessive, the administrator can increase the `DIRECTORY_POOL_SIZE` configuration option. If the retrieval count is high relative to the overall open/lookup counts (from the DSFS KNPFS report), then increasing the `DIRECTORY_REFRESH_TIMEOUT` and `PDS_ENQ_DURATION` configuration option might help, depending on the workload.

Sec Requests

This field shows how often DSFS needed to dispatch a security call to its security worker pool and how many times all workers were busy and DSFS needed to queue the request. If the queue count is high relative to the number of requests, then increasing the `SECURITY_POOL_SIZE` configuration option might help.

Sec Cache Rq

This field is the measure of how many times DSFS searched its security cache for access permissions a user has to a data set, and how often it failed to find the information (called a cache miss). The higher the number of misses to the overall requests the longer DSFS response time and a security cache miss results in a dispatched security request. The user can allow DSFS to cache security information for longer time resulting in less cache misses by increasing the `SECURITY_TIMEOUT` configuration parameter.

Note: Improving performance might entail altering more than one option and altering one option might move the performance issue to another subcomponent. An example might be an increase in the `PS_DYN_DURATION` option. By allowing DSFS to hold enqueues longer, it will result in more file data being cached, and it might, for example, be good to increase the size of the file cache, depending on file cache memory usage (steal invocations from the filecache report).

KNPFS report

The most fundamental measure of DSFS activity and its associated response times is shown by the **dsadm query -knpfs** report (and also the `MODIFY DSFS,QUERY,KNPFS` command). This report shows call rates to DSFS and their response times and total number of bytes moved between applications and DSFS. An example report is shown as follows:

The report shows the individual calls made by applications to DSFS, many of which might be familiar to the reader (such as `mkdir` or `rename`) but the fundamental measure of DSFS performance is the `*TOTALS*` line which shows the total calls made to DSFS and the average DSFS response time in milliseconds. Any tuning change made to DSFS could affect these numbers.

Resetting performance statistics

By default, all statistics in DSFS are maintained (counted) since DSFS is started. However, the administrator might need to measure performance during a specific time interval, such as peak usage. The measuring is done by resetting statistics to start counting at a specific time and then issuing the associated query at the end of the measurement interval.

In general, it's best to reset all statistics for all reports at the same time because you might want to compare information from different reports. You also want them to be measuring the same time interval. To reset all statistics, the easiest method is to use the `MODIFY DSFS,RESET,ALL` command to reset all monitoring statistics in DSFS. For example, if you want performance of DSFS between 1 PM and 3 PM, issue `MODIFY DSFS,RESET,ALL` at 1 PM and then issue `MODIFY DSFS,QUERY,ALL` at 3 PM.

Displaying memory usage

DSFS also provides commands that display its use of memory in its address space.

STORAGE report

DSFS provides the **dsadm query -storage** command (and the similar MODIFY DSFS,QUERY,STORAGE command) to show detailed usage of memory by DSFS. Much of this information is intended for service personnel but there are some key fields of interest to the administrator that show overall memory usage of DSFS above and below the bar. Following is a brief example of the output to show these key fields.

```
> dsadm query -storage
      DSFS Primary Address Space <2G Stge Usage
      -----

Total Storage Below 2G Bar Allocated:           186467328

IDFFSCM Heap Bytes Allocated:                   38395856
IDFFSCM Heap Pieces Allocated:                      2189
IDFFSCM Heap Allocation Requests                    2190
IDFFSCM Heap Free Requests                          1

IDFFSKN Heap Bytes Allocated:                   6395736
IDFFSKN Heap Pieces Allocated:                      53437
IDFFSKN Heap Allocation Requests                    80448
IDFFSKN Heap Free Requests                          27011
. . .
. . .

      DSFS Primary Address Space >2G Stge Usage
      -----

Total Storage Above 2G Bar Allocated:           2475765880

Total Bytes Allocated by IDFFSCM (Stack+Heap):   3145728
IDFFSCM Heap Bytes Allocated:                      4194304
IDFFSCM Heap Pieces Allocated:                      130
IDFFSCM Heap Allocation Requests                    130
IDFFSCM Heap Free Requests                          0

Total Bytes Allocated by IDFFSKN (Stack+Heap):   1887745024
Total Bytes Discarded (unbacked) by IDFFSKN:       1400832
IDFFSKN Heap Bytes Allocated:                      1780599051
IDFFSKN Heap Pieces Allocated:                      264593
IDFFSKN Heap Allocation Requests                    284050
IDFFSKN Heap Free Requests                          19457
```

Figure 6. Example of dsadm query -storage

The key fields that show DSFS memory usage above and below the bar are highlighted in blue and underlined. The fields are described as follows.

Total Storage Below 2G Bar Allocated

The total memory that is allocated in the DSFS address space below the 2 G bar. It includes DSFS-allocated memory and all other memory that is allocated by the operating system, z/OS UNIX, program modules. In other words, it includes everything.

IDFFSCM Heap Bytes Allocated

The amount of memory that the IDFFSCM DSFS module has allocated for data structures it uses.

IDFFSKN Heap Bytes Allocated

The amount of memory that the IDFFSKN DSFS module has allocated for data structures it uses.

Total Storage Above 2G Bar Allocated

The amount of memory that is allocated above the 2 G bar in the DSFS address space. It includes DSFS-allocated memory and all other memory that is allocated by the operating system, z/OS UNIX, program modules. In other words, it includes everything.

Total Bytes Allocated by IDFFSCM (Stack+Heap)

The amount of memory DSFS module IDFFSCM has allocated for data structures and task-related storage that it creates above the 2G bar.

Total Bytes Allocated by IDFFSKN (Stack+Heap)

The amount of memory DSFS module IDFFSKN has allocated for data structures and task-related storage that it created above the 2G bar.

Altering the size of thread pools and the size of the file cache can affect memory usage by DSFS, as can application demand. This report can be used to determine the effect on DSFS memory when a tuning option is changed, or the command could be repeatedly issued to determine how DSFS memory changes over time. Additionally, you can reset the statistics and then issue a **dsadm query -storage** command after a predetermined time interval and determine whether storage is still growing over the measured time period. In general, DSFS memory should grow to reach a stabilization point after it is started and application usage reaches its peak.

Delays and hangs in DSFS

The DSFS hang detector automatically monitors the current location of the various tasks processing in DSFS. At a set interval, the following actions take place:

1. The hang detector thread wakes up and scans the current user requests that were called into DSFS.
2. It processes this list of tasks and notes various pieces of information to determine the location of the task.
3. If the hang detector determines that a task has remained in the same location for a predefined period, it attempts to determine why the task is not making progress. This might cause DSFS messages or dumps. Certain DSFS messages might remain on the screen while the delay continues.
4. If the hang detector recognizes that this task has finally progressed, it removes the DSFS message from the console. Removal of the DSFS message means that the delay has cleared and was just a slowdown because of a stressful workload or another issue. In this case, you can discard any DSFS dumps that occur because of this delay. Several DSFS messages warn of potential problems in the DSFS address space that are connected with the delays.
5. If DSFS determines there is a true deadlock, DSFS initiates a dump on that system. This system that detected the deadlock stops and restarts DSFS to clear the deadlock.

IDFS00xxx messages are issued by the DSFS hang detector and generally remain on the console until the situation is resolved. Resolution occurs in the following situations:

1. The delayed task completes without any external correction. This situation is a slowdown and not a hang. Discard any DSFS system dumps.
2. The delayed task is canceled. In these cases, you should supply any system dump taken by DSFS to IBM service for diagnosis.

For delays, DSFS issues several messages to attempt to diagnose what might be involved in the delay. A delay might occur when:

- DSFS invokes another component such as allocation, open/close, or global resource serialization. In this case, DSFS issues message IDFS00159I or IDFS00164I to recommend that you use the other component's diagnosis material to determine the cause of the delay. DSFS does not produce a dump.
- There is heavy system activity with higher priority tasks that are delaying lower priority tasks or a delay in another system service that is not covered by message IDFS00159I. In this case, DSFS issues message IDFS00160I, but does not produce a dump.

Steps for diagnosing and resolving a DSFS hang

Perform these steps when a hang condition is suspected.

1. Continually monitor for the following messages:
 - IDFS00146I. DSFS has a potentially hanging thread that is caused by *UserList*, where *UserList* is a list of address space IDs and TCB addresses causing the hang.
 - IDFS00159I or IDFS00164I. The delay is outside of DSFS. DSFS called the identified system service and is waiting for a response. Investigate the identified system service. The problem is likely not with DSFS.
 - IDFS00160I. The delay is either in DSFS or in a system service that DSFS did not specifically identify in message IDFS00159I. DSFS cannot determine whether there is a hang, a slowdown, or some other system problem. To act, look for other symptoms. For example, if you see messages about components that are using a significant amount of auxiliary storage, resolve the auxiliary storage shortage. If the message persists, continue to the next step.

2. Enter MODIFY DSFS QUERY,THREADS to determine whether any DSFS threads are hanging and why. The type and amount of information that is displayed because of this command is for internal use and can vary between releases or service levels. For an example of the command output, see [Figure 7 on page 52](#).

3. Enter the DISPLAY A,DSFS command to determine the DSFS ASID.

4. Enter MODIFY DSFS QUERY,THREADS,OLDEST at one to two-minute intervals for six minutes.

5. Check the output for any user tasks (tasks that do not show the DSFS ASID) that are repeatedly in the same state during the time you requested MODIFY DSFS QUERY,THREADS,OLDEST. If there is a hang, the task that is hanging persists unchanged over the course of this time span. If the information is different each time, there is no hang.

6. Check whether any user tasks are hung, focusing on the tasks that are identified by message IDFS00146I or message IDFS00164I. User tasks do not have the same address space identifier (ASID) as the DSFS address space. One or more threads consistently at the same location might indicate a hang (for example, Recov, TCB, ASID Stack, Routine, State). The threads in the DSFS address space with the DSFS ASID (for example, `xcf_server`) are typically waiting for work. It is typical for the routine these threads are waiting in to have the same name as the entry routine, as shown in the following example. If successive iterations of the MODIFY DSFS QUERY,THREADS,OLDEST command show that the STK/Recov, TCB, ASID, Routine, and State for a thread are constant, it is probable that this thread is hung.

```

F DSFS,QUERY,THREADS
IDFS00139I Starting Query Command THREADS. 578
          DSFS and z/OS UNIX Tasks
-----
STK/Recov      TCB      ASID      Stack      Routine      State
-----
506A878000 005CFA60 0057 506A878748 DSFRDWR      RUNNING
50000DCF00
    since Mar 2 18:28:01 2021
    Operation counted for 0EVFS=7E78D360 VOLP=506A910000
    fs=DSFSAGGR.BIGZFS.UTILITY.FS.DCEIMGHQ

506A870000 005CFA60 0057 506A870748 DSFRDWR      RUNNING
50000DD7C0
    since Mar 2 18:28:01 2021

    7E2E7000 005C3658 004A 7E2E8000 comm_daemon  RUNNING
5000001A40
    since Mar 2 18:28:01 2021
    7BC3D000 005816D8 004A 7BC3E000 xcf_server   WAITECB
50000D3A40
    since Mar 1 22:10:53 2021 Current DSA: 7BC3E5F8
    wait code location offset=1404 rtn=xcf_server

```

Figure 7. Sample QUERY,THREADS output

7. IBM Support must have dumps of DSFS, OMVS, and the OMVS data spaces and possibly the user address space identified on any preceding IDFS00160 message for problem resolution. Obtain and save SYSLOG and dumps of DSFS, OMVS, and the OMVS data spaces, and the user ASID using `JOBNAME=(OMVS,DSFS,user_jobname),DSPNAME=('OMVS.*')` in your reply to the DUMP command. Following is an example of the DUMP command:

```

DUMP COMM=(dsfs hang)
R
x,JOBNAME=(OMVS,DSFS,user_jobname,SDATA=(RGN,LPA,SQA,LSQA,PSA,CSA,GRSQ,TRT,SUM),DSPNAME=('OMV
S'.*'),END

```

You must capture dumps for IBM Support before taking any recovery actions such as CANCEL or ABORT.

8. If you know which user task is hung (for example, returned in IDFS00146I or determined to be hung after review of the output from repeated MODIFY DSFS QUERY,THREADS,OLDEST commands), consider issuing the CANCEL command to clear that task from the system.
9. Finally, if the previous steps do not clear the hang, issue the MODIFY DSFS,ABORT command to initiate a DSFS termination and restart. DSFS will be started in a new address space. If you are running in a shared file system environment, then z/OS UNIX will mount the DSFS file system automatically. Otherwise, you will need to manually mount the DSFS file system to make it available to applications. For more information about remounting the file system, see [“Mounting the utility file system”](#) on page 27.

Identifying storage shortages in DSFS

When DSFS can no longer obtain sufficient storage to complete a request, it issues message IDFS00072A, possibly creates a dump, and restarts. If you see message IDFS00072A before DSFS initialization is complete (before message IDFS00039I), either increase the REGION size in the DSFS PROC or decrease FILECACHE_SIZE parameter in the IDFFSPRM configuration file.

In addition, the DSFS hang detector periodically checks a warning limit and a critical limit. When it reaches the warning limit, message IDFS00165I is displayed. That message remains on the console until the situation is resolved, or until the critical limit is reached. If the critical limit is reached, message

IDFS00166I is displayed and remains on the console until storage usage goes under the critical limit to the warning limit and message IDFS00165I is displayed again. See [“Displaying memory usage” on page 49](#) for more information about how to determine the amount of storage that is being used in the DSFS address space.

A DSFS storage shortage can be caused by many issues. If the size of the file cache is larger than the default size, try using the **dsadm config** command to reduce its size. If this action does not relieve the problem, collect a dump and restart DSFS with the MODIFY DSFS,ABORT command. Report the problem to IBM service.

Tips:

- Changing the size of the file cache can cause delays. Try to change the size during low activity periods.
- In general, if you see a return code 132 (ENOMEM), DSFS is short on storage. Take steps to reduce DSFS storage usage. When storage shortages become critical, you can also see return code 157 (EMVSERR).
- Started subtasks, such as the DSFS colony address space, fall under SUBSYS STC. These address spaces might be subject to IEFUSI limitations if IEFUSI exits are allowed for SUBSYS STC. IBM strongly recommends that you always set REGION=0M and MEMLIMIT=NOLIMIT for the DSFS colony address space.

Disabling the utility file system

If DSFS detects a problem with the utility file system, or a problem that could lead to corruption of the utility file system, DSFS disables it for access and application requests are failed. Message IDFS00131E is issued, as shown in the following example.

```
IDFS00131E Utility file system PLEX.JMS.AGGR001.DCEIMGHQ disabled
```

In addition, a dump will likely be generated. You can contact IBM service and provide the dump and any other information that is useful for diagnosing the problem (for example, what was running on the system when the problem occurred).

When the utility file system is disabled, applications cannot read from or write to it. DSFS on the affected system is not available for reading and writing until it is unmounted and mounted. DSFS performs the following steps if the utility file system is disabled:

1. It rejects all application access with failures . In most cases, it returns EMVSERR.
2. If the disablement occurred because of a DSFS software error, then it will terminate the address space and request that z/OS UNIX restart DSFS. It stops if there are problems terminating the address space. The operator will receive the restart prompt message BPXF032D.
3. After it has been restarted, DSFS will receive a mount for the utility file system if sysplex file sharing is enabled. Otherwise, the operator has to manually mount the utility file system or use automation to mount it.

To determine how mount processing will affect the utility file system, see [“Mounting and unmounting utility file systems” on page 28](#).

Chapter 9. JES spool data sets

DSFS supports access to JES spool data sets in the primary JES subsystem. Users or applications can programmatically examine their own job output, determine the status of jobs they own, purge individual sysout data sets from the JES spool, or purge jobs they own. Administrators can globally enable or disable JES spool support with the IDFFSPRM option `ENABLE_SYSOUT`. (See [“ENABLE_SYSOUT”](#) on page 115.)

Accessing JES spool data sets

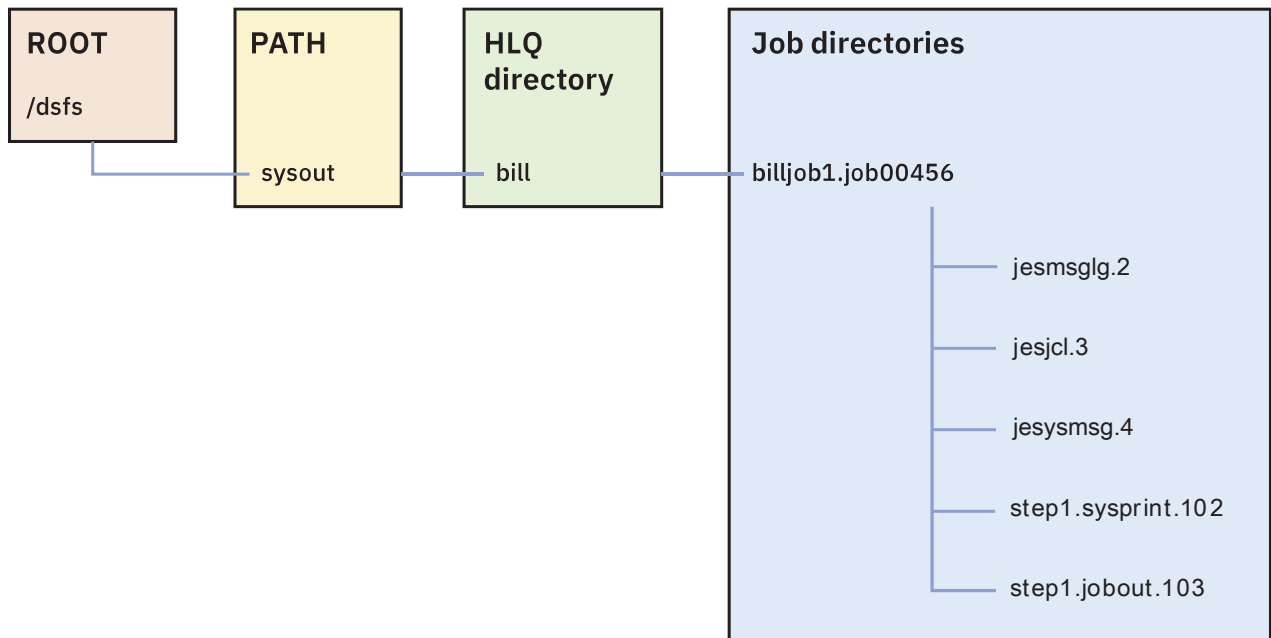
Access to JES spool data sets in the z/OS UNIX file system tree is done with the sysout path directory. DSFS automatically creates this directory, which is accessible by all users when `ENABLE_SYSOUT=ON` is specified.

1. DSFS places a high-level qualifier directory name in the sysout path directory if the user application changes the working directory or uses a path name that specifies the high-level qualifier directory name in that path. The high-level qualifier directory name is the user ID of the requesting user or application. Once the high-level qualifier directory is created, it cannot be removed. The user ID of the requesting user or application must match the name of a high-level qualifier directory name in order to be allowed access to it. The high-level qualifier directory names in the sysout path are not subject to the HLQ exclude list.
2. DSFS populates high-level directories with the jobs on the JES spool that are owned by the user ID and are not being purged. The name of these job directories takes the form `jobname.jesjobid`, where *jobname* is the job name that is specified on the JCL job statement and *jesjobid* is the job ID that is assigned to the job by JES. The contents of the high-level directory are refreshed each time the directory is accessed.
3. DSFS populates job directories with names that represent job output data sets. These names are also referred to as *sysout files*. Input and dummy data sets are ignored.
 - The naming convention for the sysout files that represent the system spool data sets is `ddname.dskey`, where the `ddname` and `dskey` values are both assigned by JES.
 - The naming convention for the regular sysout files is `stepname.ddname.dskey`, where *stepname* is the name from the JCL execution statement, `ddname` is the name from JCL data definition statements, and `dskey` is a data set key assigned to the JES spool data set by the primary JES.

By using JCL statements, JES can create duplicate spool data sets for use in JES output grouping. These duplicate spool data sets are considered by DSFS as one file and are represented by only one sysout file name. Job directory contents are refreshed each time the job directory is accessed. These sysout file names can be used with standard z/OS UNIX commands and tools to read the data in the spool data sets.

Restriction: DSFS does not support creation parameters on high-level qualifier directories in the sysout path directory.

The following graphic shows an example of the directory path and associated sysout files for a specified job.



1. User ID BILL submitted a job with the name BILLJOB1 on the JOB statement. The primary JES in this case is JES2.
2. JES2 assigned the job ID JOB00456 to the job.
3. JES2 created three system spool data sets for the job (JESMSG LG, JESJCL, and JESYSMSG) and assigned data set keys 2, 3, and 4 to them.

Usage notes for accessing JES spool data sets

1. DSFS treats sysout files as text files. When a sysout file is accessed, DSFS reads each record from the spool data set. If it is in fixed format, DSFS strips the trailing blanks. For both fixed and variable record formats, DSFS appends a newline character to the end of the record in the POSIX byte stream.
2. Updating sysout files is not supported. You cannot create, rename, or truncate sysout files. You also cannot modify the attributes of sysout files.
3. A sysout file can be removed with the z/OS UNIX **rm** command. When a sysout file that has duplicates is removed, DSFS attempts to remove all duplicate spool data sets as well. This operation can fail if the spool data set is involved in other JES processing. When the last sysout file of a job is removed, JES may queue the job for purge processing. If that happens, the job directory will be automatically removed in the next refresh of the high-level directory.
4. DSFS does not support modifying the attributes of job directories. DSFS also does not support renaming or creating job directories. DSFS supports the use of the z/OS UNIX **rmdir** command to remove job directories that have not yet been populated with sysout files. The job will be purged from the JES spool. Otherwise, the **rmdir** command should not be necessary to purge the job.
5. The only serialization mechanism is a dynamic allocation of the spool data set in SHR mode while the data is being retrieved from the spool data set and is placed in the file cache. This allocation is released when the read is completed. There is no IDFFSPRM option to control how long this dynamic allocation is held.
6. The values for **atime**, **mtime**, and **ctime** for job directories is initially set to the current time of day when the directory itself is created. The **atime** is updated each time the job directory is accessed. For sysout files, the **atime**, **mtime**, and **ctime** values are initially set to when the spool data set was created. The **atime** value is updated each time the sysout file is accessed.

Examples

The following example shows the SYSOUT-related fields for a **dsadm fileinfo** command of a SYSOUT high-level qualifier directory:

object type	DIR	object linkcount	141
dir name count	141	data set name type	HLQ DIR
recfm	na	lrecl	na
data mode	SYSOUT	data set status	RETRIEVED
data set name	SUIMGVY		

The following example shows the SYSOUT-related fields for a **dsadm fileinfo** command of a job directory:

object type	DIR	object linkcount	2
dir name count	5	data set name type	JOB DIR
recfm	na	lrecl	na
data mode	SYSOUT	data set status	RETRIEVED
data set name	SUIMGVY.SUIMGVY2.JOB01922		

The following example shows the SYSOUT-related fields for a **dsadm fileinfo** command of a sysout file:

object type	FILE	object linkcount	1
dir name count	n/a	data set name type	JOB DIR
recfm	FB	lrecl	80
data mode	SYSOUT	data set status	SYSOUT FILE
data set name	SUIMGVY.SUIMGVY2.JOB01922.D00000102.?		

The attribute fields have the following values:

object type

One of two values:

- DIR for HLQ or job directories.
- FILE for sysout files.

object linkcount

Although data sets do not have link counts, z/OS UNIX objects do. The utility file system keeps the link count of directories equal to the number of subdirectories. For SYSOUT files the link count is always 1. Every z/OS UNIX directory has special entries for . (dot) and . . (dot dot) entries. Job directories have a link count of 2 because the dot and dot dot entries are inserted into the utility file system directory that represents the job.

dir name count

The number of objects in the directory, including the special dot and dot dot entries for directories. For SYSOUT files, the number is the value na.

data set name type

The type of data set that this DSFS object represents (HLQ DIR, JOB DIR, or SYSOUT FILE).

recfm

The record format of the SYSOUT data set. For directories, the value is na.

lrecl

The record length of the SYSOUT data set. For directories, the value is na.

data mode

The processing mode, which is also the path that is chosen in the DSFS file system tree to access the data set. The value is SYSOUT for JES spool objects.

data set status

This field indicates whether DSFS has retrieved the data set contents and converted it to a POSIX byte stream format. If true, then it has the value RETRIEVED. If not, then it has the value UNCACHED. For HLQ and job directories, it has the value RETRIEVED.

data set name

For HLQ directories, the name of the HLQ directory itself (owning user ID).

For job directories, the HLQ directory name combined with the job name and JES job ID. For SYSOUT files, the name of the spool data set.

Part 2. DSFS administration reference

This part of the document discusses reference information for Data Set File System (DSFS).

Chapter 10. z/OS system commands

These system commands are available.

- MODIFY DSFS PROCESS queries internal counters and values. Use it to initiate or gather debugging information.
- MODIFY OMVS,STOPPFS=DSFS. Use it to stop DSFS.
- SETOMVS RESET starts the DSFS physical file system (PFS) if it has not been started at IPL, or if the PFS was stopped and the BPXF032D message was responded to with a reply of **i**.
- SET OMVS=xx where xx is the BPXPRMxx member that contains the FILESYSTYPE and/or MOUNT statement for DSFS.

Run these commands from the console or from System Display and Search Facility (SDSF).

MODIFY DSFS PROCESS

Purpose

Use the MODIFY DSFS PROCESS report to query internal DSFS counters and values and display them in the system log. You can also use it to initiate or gather debugging information. To use this command, the DSFS PFS must be running.

Format

Use any of the following formats for this command.

```
modify procname,query,{level|settings|threads[, {allwait|oldest}]} [{kn|filecache|lfs|lock|
storage|iobydasd|dataset|dscache|all}]
modify procname,reset
modify procname,abort
modify procname,dump
modify procname,fsinfo
```

Parameters

procname

The name of the DSFS PROC. The default *procname* is DSFS.

command

This parameter can have one of the following values:

abort

Causes DSFS to dump and restart.

dump

Causes the DSFS PFS to issue a dump that includes the z/OS UNIX address space and its associated data spaces.

fsinfo

Displays detailed information about the DSFS utility file system.

query

Displays DSFS counters or values.

level

Displays the DSFS level for the DSFS physical file system kernel.

settings

Displays the DSFS configuration settings, which are based on the IDFFSPRM file and defaults.

threads[, {allwait | oldest }]

Displays the threads that are monitored by the DSFS hang detector. To display all DSFS threads, use the MODIFY DSFS QUERY,THREADS,ALLWAIT command. The time of day values are shown in Greenwich mean time (GMT). To display the oldest thread of each system, use the MODIFY DSFS QUERY,THREADS,OLDEST command.

<report>

One of the following report options. These parameters all produce reports. For more information about these reports, see [“Monitoring DSFS performance” on page 45](#).

all

Displays all the DSFS counters.

dataset

Displays DSFS statistics about the data set calls that it makes.

iobydasd

Displays the DASD that the utility file system resides on and associated I/O rates.

kn

Displays the calls that were made to DSFS from z/OS UNIX.

lfs

Displays the file system statistics, including the performance of the DSFS metadata cache, the vnode cache, and the file compression statistics.

lock

Displays the lock contention values.

storage

Displays DSFS memory usage.

filecache

Displays the DSFS file cache statistics and storage usage.

dscache

Displays thread pool statistics for DSFS data set accesses and caching information.

reset

Resets all DSFS counters to zero.

No other options are allowed after **reset**.

Examples for MODIFY DSFS PROCESS

The following example queries all of the DSFS counters:

```
MODIFY DSFS,query,all
```

The following example displays information about the utility file system.

```
MODIFY DSFS,fsinfo
```

MODIFY DSFS,FSINFO

The FSINFO report shows usage of its utility file system. An example of this display is shown as follows.

```
IDFS00169I Starting FSINFO command.
IDFS00261I File System Status:
  File System Name:      DSFSAGGR.BIGDSFS.UTILITY.FS.DCEIMGHQ

  System:                DCEIMGHQ          Devno:                46
  Size:                  1440000K          Free 8K Blocks:      178142
  Free 1K Fragments:    7                  Log File Size:       14400K
  Bitmap Size:          208K              Anode Table Size:    56K
  File System Objects:  151                Version:              1
  Overflow Pages:        0                  Overflow HighWater:   0
  Space Monitoring:     90,5                Disk IO Errors:      0
  ENOSPC Errors:        0
  Status:                NE,NC

  File System Creation Time: Jan 27 21:47:41 2021
  Mount Time:            Mar  8 16:55:19 2021
  Last Grow Time:        n/a

Legend: NE=Not encrypted,NC=Not compressed
```

File System Name

The name of the utility file system used by DSFS on the system that ran the FSINFO command.

System

The system where the command was run.

Devno

The z/OS UNIX device number (*devno*) for the mount of the file system. z/OS UNIX assigns a unique number to each mounted file system.

Size

The size of the file system in kilobytes.

Free 8K Blocks

The number of whole blocks available in the file system. When this number is reduced to zero, the file system is considered out of space.

Free 1K Fragments

In some cases DSFS will store multiple object contents (maximum of eight) in one 8 K block in 1 K units called *fragments*. This is the number of free fragments available but note that most objects in DSFS require whole blocks. If the “Free 8 K Blocks” field is 0, the file system is considered out of space regardless of this value.

Log File Size

DSFS utility file systems are logging file systems and use a log file to record transactional updates to the file system, this is the size of that file in kilobytes.

Bitmap Size

DSFS uses a disk structure that is called a bitmap to track free space in the file system. This is the size of that object’s contents.

Anode Table Size

DSFS uses a structure that is called an *anode* to represent an object in the file system (a file or a directory or creation parameter contents). They are contained in a table and the size of this table is reported in kilobytes. The more objects that are added to the utility file system the greater the size of this table.

File System Objects

Total number of objects in the utility file system including files, directories, and creation parameters for HLQ directories.

Version

Version of the utility file system.

Overflow Pages/Overflow Highwater

DSFS uses extendible hash trees to represent directories. If the hashing of names creates too many hash keys with the same value, then DSFS will anchor overflow page linked lists to directory tree leaves which makes name lookup slower. Overflow pages should rarely be more than 0 and typically would only result due to extremely large directories. Overflow highwater is simply the high-water mark for the number of overflow pages that exist inside the utility file system.

Space Monitoring

Value that is used for the FSFULL configuration parameter. The value ‘na’ is shown if FSFULL reporting is not used.

ENOSPC Errors

Count of the instances when DSFS had to return ENOSPC error code to applications (file system out of space). This is an indicator that applications saw failures due to lack of space in the utility file system.

Disk IO Errors

Count of instances when DSFS had to return EIO error code to applications (disk I/O error). This is an indicator that applications saw failures due to I/O errors to its utility file system returned to DSFS from VSAM.

Status

A set of status codes that show the state of the file system.

Table 2. Status codes for file system state

Status code	File system state
CO	The utility file system is compressed.
DI	The utility file system is disabled for access. This error typically indicates that DSFS will restart itself to correct the error.
EN	The utility file system is encrypted, which means that file contents are stored in encrypted format by using encryption keys that are represented by the key label assigned to the utility file system.

Table 2. Status codes for file system state (continued)

Status code	File system state
GD	Dynamic grow is disabled. This value is set if the utility file system has had dynamic grow failure and will not attempt more dynamic grows for 3 minutes to avoid excessive grow attempts.
GF	The utility file system has failed dynamic grow attempts.
GR	The utility file system is being grown.
IE	The utility file system has encountered disk I/O errors.
L	The utility file system is low on space, which is defined as less than 1 MB of total space. In this state the utility file system will sync data more aggressively and reduced performance can be expected.
NC	The utility file system is not compressed.
NE	The utility file system is not encrypted.
OV	The utility file system has directories with overflow pages.
SE	The utility file system has run out of utility file system disk space and failed applications due to no space for the operation.
SL	The utility file system is being salvaged.

File System Creation Time

Time the utility file system was mounted for the first time (which formats the file system for use).

Mount time

The time the utility file system was mounted.

Last Grow Time

The time the utility file system was last grown. The value 'na' is used if the file system was never grown.

Legend

An explanation of the meanings of the status codes.

MODIFY QUERY reports

Use the MODIFY DSFS QUERY report to obtain various statistical information with a definition for each field. Output examples are shown for all the available modify queries with a definition of each field. DSFS will often postfix larger numbers that are too large to fit in their display field with one of the following values:

b

Multiply the number by 1,000,000,000.

G

Multiply the number by 1,073,741,824.

t

Multiply the number by 1000.

tr

Multiply the number by 1,000,000,000,000.

T

Multiply the number by 1,099,511,627,776.

m

Multiply the number by 1,000,000.

MODIFY DSFS QUERY,DATASET

K

Multiply the number by 1024.

M

Multiply the number by 1,048,576.

MODIFY DSFS QUERY,DATASET

The DATASET report shows the number of calls that DSFS makes to DYNALLOC and data set OPEN and CLOSE calls. Example output is shown as follows.

```
IDFS00139I Starting Query Command DATASET.
  Printing Dataset Allocation Stats
    Allocates                394205  (avg alloc 3.089 msecs)
    Allocates failed          0
    Allocates exceeding limit 0
    Unallocates              394205  (avg unalloc 1.779 msecs)
    Unallocates failed        0
    Current allocations        0
    Opens                     476119
    Open failures              0
    Closes                     476119
    Reserves                   3651
    Reserves failed            0
    Unreserves                 3651
    SPFEDIT enqs               58610
    SPFEDIT enqs failed        0
    Upgrade SPFEDIT enqs       140
    Upgrade SPFEDIT enqs failed 0
    SPFEDIT deqs               58610
    DESERV GET_ALL             26193
    DESERV GET_ALL failed      0
    DESERV GET                  146416
    DESERV GET failed          147258
    STOW REPLACES              0
    STOW REPLACE failures      0
    DESERV DELETES             0
    DESERV DELETE failures     0
    STOW DELETES               0
    STOW DELETE failures       0
    STOW CHANGES              0
    STOW CHANGE failures       0
```

```
IDFS00020I DSFS kernel: MODIFY command - QUERY,DATASET completed
successfully.
```

Allocates

The number of calls to the DYNALLOC service made by DSFS to allocate a data set, and the average time in milliseconds for each allocation.

Allocates failed

The number of times that DYNALLOC failed during an allocate call.

Allocates exceeding limit

The number of times an allocation was attempted that would have exceeded the DSFS limit for maximum concurrent allocations.

Unallocates

The number of calls to the DYNALLOC service made by DSFS to unallocate a data set, and the average time in milliseconds for each unallocation.

Current allocations

The number of data sets that are currently allocated by DSFS.

Unallocates failed

The number of times DYNALLOC failed during an unallocate call.

Opens

The number of calls to the OPEN macro made by DSFS for data set access.

Open failures

The number of calls to the OPEN macro that failed.

Closes

The number of calls to the data set CLOSE macro made by DSFS.

Reserves

The number of calls to the RESERVE service made by DSFS to reserve the volume that a PDS is on.

Reserves failed

The number of calls to the RESERVE service that failed.

Unreserves

The number of calls to the DEQ service made by DSFS to unreserve the volume that a PDS is on.

SPFEDIT enqs

The number of calls to the ENQ service made by DSFS to get the SPFEDIT ENQ for a PDS/PDSE member.

SPFEDIT enqs failed

The number of times that getting SPFEDIT ENQ failed.

Upgrade SPFEDIT enqs

The number of calls to upgrade an SPFEDIT ENQ from SHR to OLD for a PDS/PDSE member.

SPFEDIT enqs failed

The number of times that upgrading SPFEDIT ENQ from SHR to OLD failed.

SPFEDIT deqs

The number of calls to the DEQ service made by DSFS to remove the SPFEDIT ENQ held for a PDS/PDSE member.

DESERV GET_ALL

The number of calls to the DESERV GET_ALL service made by DSFS to retrieve all member names and directory entries from a PDSE/PDS.

DESERV GET_ALL failed

The number of calls to the DESERV GET_ALL service that failed.

DESERV GET

The number of calls to the DESERV GET service that was made by DSFS to retrieve directory information for a given PDS/PDSE member.

DESERV GET failed

The number of calls to the DESERV GET service that failed.

STOW REPLACES

The number of calls to the STOW service made by DSFS to store ISPF statistics for a PDS/PDSE member.

STOW REPLACE failures

The number of calls to the STOW service that failed.

DESERV DELETES

The number of calls to the DESERV DELETE service that was made by DSFS to delete PDSE members.

DESERV DELETE failures

The number of calls to the DESERV DELETE service that failed.

STOW DELETES

The number of calls to the STOW DELETE service made by DSFS to delete PDS members.

STOW DELETE failures

The number of calls to the STOW DELETE service that failed.

STOW CHANGES

The number of calls to the STOW CHANGE service that was made by DSFS to rename PDS or PDSE members.

STOW CHANGE failures

The number of calls to the STOW CHANGE service that failed.

MODIFY DSFS QUERY,DSCACHE

The DSCACHE report shows the activity to the various thread pools that DSFS uses to make calls to DFSMS and the SAF product along with security cache performance. You can use it to determine whether to change certain IDFPRMxx options to help improve performance. [“Performance tuning” on page 45](#) discusses in detail how to use the information in this report. An example of the output of this command is provided.

```
F DSFS,QUERY,DSCACHE
IDFS00139I Starting Query Command DSCACHE. 310
          DSFS Dataset Caching and Thread Pool
Statistics
```

	Called/ Dispatched	Miss/ Queued	Miss/ QPCT	Resync/ Retrieve
IO Requests	959	0	0%	16
Dir Requests	67	0	0%	7
Sec Requests	2038	0	0%	na
Sec Cache Requests	980	22	2%	na
Sysout IO Requests	80	0	0%	80
Sysout Dir Requests	926	0	0%	826

IO Requests

The number of I/O requests that were handled.

Request type	Description
Called/Dispatched	The total number of file I/O requests that needed to be dispatched onto a DSFS worker task.
Miss/Queued	The total number of file I/O requests that found that all worker tasks were busy and thus the request had to be placed in a queue.
Miss/QPCT	The percentage of file IO requests that had to be queued.
Resync/Retrieve	The number of file retrieval requests made that is one of the types of I/O requests to be dispatched. DSFS will retrieve a data set contents upon first access and convert it to POSIX byte-stream format.

Dir Requests

The number of directory-related requests.

Request type	Description
Called/Dispatched	The total number of directory-related requests (a directory being a PDS/PDSE or an HLQ directory) that had to be dispatched onto a DSFS worker task.
Miss/Queued	The total number of directory requests (such as reading a PDS directory or searching a catalog) that found that all worker tasks were busy and thus the request had to be placed in a queue.
Miss/QPCT	The percentage of directory requests that had to be placed in a queue.
Resync/Retrieve	The number of directory-resync requests that were made. A resync is necessary on first access to a PDS/PDSE or HLQ directory. It is necessary for an HLQ directory resync after a period that was determined by the DIRECTORY_REFRESH_TIMEOUT program option. You have to resync a PDS/PDSE directory if it was determined that programs outside of DSFS might have altered the PDS/PDSE directory contents.

Sec Requests

The number of calls to the DYNALLOC service made by DSFS to unallocate a data set.

Request type	Description
Called/Dispatched	The total number of security requests that required a SAF call, which must be done on a DSFS worker task.
Miss/Queued	The total number of SAF security requests that found that all worker tasks were busy and thus the request had to be placed in a queue.
Miss/QPCT	The percentage of security requests that were queued.
Resync/Retrieve	The value is always 'na' because it is not applicable to security requests.

Sec Cache Requests

The number of calls to the DSFS security cache to determine whether a user has access to a data set.

Request type	Description
Called/Dispatched	The total number of security calls made to the internal DSFS security cache.
Miss/Queued	The total number of times that a security cache entry was not found or was timed out for a particular user security cache query.
Miss/QPCT	The percentage of time that a security cache entry was not available to provide a security answer. In this case, a security request was generated and dispatched onto a DSFS worker task to obtain the security answer.
Resync/Retrieve	This field always has the value 'na' because it is not applicable to security cache requests.

MODIFY DSFS QUERY,FILECACHE

The FILECACHE report shows the performance of the file caching system in DSFS. DSFS converts data set contents to POSIX byte-streams and caches those contents in the file cache. Because the file cache is bounded in size, it might have to cast out data (paging) to the utility file system to make room for new data. This report is also described in [“Performance tuning” on page 45](#). An example of the command follows.

```

F DSFS,QUERY,FILECACHE
      File Caching System Statistics
-----

External Requests:
-----
Reads      12824155   Fsyncs      0   Schedules      0
Writes     13504496   Setattrs    276  Unmaps        1475
Asy Reads   3939233    Getattrs    6452  Flushes       0

File System Reads:
-----
Reads Faulted      0   (Fault Ratio  0.000%)
Writes Faulted     90   (Fault Ratio  0.001%)
Read Waits         0   (Wait Ratio   0.000%)
Total Reads       90

File System Writes:
-----
Scheduled Writes    0   Sync Waits      0
Error Writes        0   Error Waits     0
Scheduled deletes   0
Page Reclaim Writes 0   Reclaim Waits   0
Write Waits         0   (Wait Ratio    0.000%)

```

Page Management (Segment Size = (64K)) (Page Size = 8K)

```
-----
Total Pages          131072      Free          131072
Segments            3936
Steal Invocations      0      Waits for Reclaim      0
Number of cache spaces used:    16  Pages per space:    8192
```

Space Address	Total 8K Pages	Free Pages	Assigned Segments	Fix Type
5029B00000	8192	8192	0	Not Fixed
502DC00000	8192	8192	0	Not Fixed
5031C00000	8192	8192	0	Not Fixed
5035D00000	8192	8192	0	Not Fixed
5039D00000	8192	8192	0	Not Fixed
503DD00000	8192	8192	0	Not Fixed
5041E00000	8192	8192	0	Not Fixed
5045E00000	8192	8192	0	Not Fixed
5049E00000	8192	8192	0	Not Fixed
504DF00000	8192	8192	0	Not Fixed
5051F00000	8192	8192	0	Not Fixed
5056000000	8192	8192	0	Not Fixed
505A000000	8192	8192	0	Not Fixed
505E000000	8192	8192	0	Not Fixed
5062100000	8192	8192	0	Not Fixed
5066100000	8192	8192	0	Not Fixed

External Requests

Describes the requests that are made to the file cache to perform operations as requested by applications.

External Requests	Description
Reads, Writes	How often the cache was called to read or write files.
Asy Reads	How often file read-ahead is performed.
Setattr	The number of times a file had a set attributes call that affected its contents in the file cache (such as changing the size of the file).
Getattr	The number of queries of file length from the file cache. The utility file system contents for the file might not have recent changes that were not committed to disk from the file cache.
Schedules	The number of times a file's contents was requested to be written to the utility file system using asynchronous write-behind techniques.
Fsync	How often applications requested that a file's data be synced to the utility file system.
Unmaps	The count of times a file's contents was uncached in the file cache.

File System Reads

Shows how often the cache reads data from disk for a file. Cache misses and read I/Os degrade application response time and the goal is for these numbers to be as low as possible. Increasing the cache size is the typical method for making these numbers smaller.

File System Reads	Description
Reads Faulted	Count of read requests that needed to perform at least one I/O to read the requested portion of the file from disk.
Writes Faulted	Count of how often a write to a file needed to perform a read from disk. If a write only updates a portion of a page of a file on disk and that page is not in memory, then the page must be read in before the new data is written to the in-memory page. (The DSFS I/O driver can only perform I/O in whole pages.)

File System Reads	Description
Read Waits	How often a read had to wait for a pending I/O. For example, how often a read of a file found that the range of the file is pending read (probably because of asynchronous read ahead).
Total Reads	Total number of utility file system reads that were made for file requests.

File System Writes

Shows how often the cache wrote the data to disk. In general, try to minimize the Page Reclaim Writes and Reclaim Waits. If these occur often, relative to the external request rate (shown in the KNPFs report), then the cache might be too small.

File System Writes	Description
Scheduled Writes	Count of how often the cache wrote out dirty segments for a file.
Sync Waits	Count of how often a fsync request is needed to wait on pending I/O for dirty segments.
Error Writes and Error Waits	Count of the error handling paths and should almost always be 0 unless a disk hardware error occurs. If an unexpected error occurs for a file, all of its dirty segments are written and synced to the utility file system. (A utility file system that is running out of space is not an error condition that causes the cache to sync a file. The cache reserves storage for files as they are written, which ensures that no unexpected out of space conditions arise).
Scheduled Deletes	Count of times a pending I/O was canceled because a file was being deleted. In this case, the data is not appropriate to be on disk (because the file is 0 link count). Therefore, canceling the I/O is done to avoid an I/O wait. This is a performance optimization for file remove.
Page Reclaim Writes	Count of times that a segment had to be written to DASD to reclaim space in the cache.
Page Reclaim Waits	Count of times that the reclaim function waited on pending I/O to reclaim segment pages.
Write Waits	Count of times a write occurred to a page that was already pending I/O. In this case, the I/O must be waited upon before the page is updated with the new data.

Page Management

Shows how storage in the file cache is used. It is generally desirable to minimize the number of steal invocations (reclaims). To minimize the number of steal invocations, increase the size of the cache.

Page Management	Description
Total pages	The number of 8 K pages in the cache. That is, (filecache_size / 8K).
Free	The number of available 8 K pages in the cache.
Segments	The number of internal segment structures that represent 64 K sections of a file.
Steal Invocations	The number of times 8 K pages were reclaimed from the cache.
Waits for Reclaim	The number of times a task waited for space to be reclaimed from the cache.
Space Address	The address of a cache space used in the file cache. The file cache uses several cache space regions. Each one is managed separately and in-parallel for optimal performance.

Page Management	Description
Total 8K Pages	The number of pages in the associated cache space.
Free Pages	The number of unused pages in the associated cache space.
Assigned Segments	The number of 64 K file segments assigned to use pages from the cache space.
Fix Type	Indicates whether the file cache pages are fixed in real memory and the service used to fix those pages. The service is IARV64 (compression is not used for utility file system), ZEDC (compression is used for the utility file system), or Not Fixed, which means the file cache is not permanently fixed in memory. The FIXED option of the FILECACHE_SIZE IDFPRMxx parameter determines whether DSFS will attempt to fix the cache.

MODIFY DSFS QUERY,IOBYDASD

The IOBYDASD report shows the number of I/O requests made by DSFS to its utility file system and shows the I/O requests made to the individual DASD volumes that contain the utility file system. An example of this report is as follows.

```

F DSFS,QUERY,IOBYDASD
IDFS00139I Starting Query Command IOBYDASD. 320
          DSFS I/O by Currently Attached DASD/VOLs

DASD    PAV
VOLSER  IOs    Reads  bytes    Writes bytes    Waits    Average Wait
-----
ZFSD64   1     343  15348     626   2912     376     1.003
ZFSD65   1      25  10240       0     0       25     0.984

Total number of waits for I/O:         401
Average wait time per I/O:         1.001

```

DASD VOLSER

The DASD volumes that contain the utility file system.

PAV IOs

The maximum number of concurrent I/O requests to each volume.

Reads

The number of read I/O requests.

bytes

The number of bytes read or written in K units.

Writes

The number of write I/O requests.

Waits

The number of waits for I/O completion.

Average wait

The average wait time for I/O requests in milliseconds.

Total number of waits for I/O

Total of Waits column.

Average wait time per I/O

The average of the Average Wait times, in milliseconds.

MODIFY DSFS QUERY,KN

The KN report shows calls made to DSFS by applications and the time it takes DSFS to process those calls. This report is also described in [“Performance tuning” on page 45](#). An example of the command follows.

```
F DSFS,QUERY,KN
IDFS00139I Starting Query Command KN.
      DSFS Kernel PFS Calls
      -----
```

Operation	Count	Avg Time	Bytes
-----	-----	-----	-----
dsfs_opens	5651	3.384	
dsfs_closes	5651	16.028	
dsfs_reads	13175633	0.002	16T
dsfs_writes	13505094	0.006	16T
dsfs_ioctls	0	0.000	
dsfs_fileinfos	60	0.042	
dsfs_getattr	1630	0.003	
dsfs_setattr	6	0.854	
dsfs_accesses	1089	0.002	
dsfs_lookups	19982	0.742	
dsfs_creates	630	11.413	
dsfs_removes	630	9.188	
dsfs_links	0	0.000	
dsfs_renames	0	0.000	
dsfs_mkdirs	180	16.384	
dsfs_rmdirs	180	13.212	
dsfs_readdir	788	0.013	304.816K
dsfs_symlinks	0	0.000	
dsfs_readlinks	0	0.000	
dsfs_fsyncs	210	28.828	
dsfs_inactives	1013	0.003	
dsfs_truncs	60	4.748	
dsfs_recoveries	0	0.000	
dsfs_audits	0	0.000	
dsfs_pfscats	5353	0.012	
dsfs_statfss	1260	0.007	
dsfs_vgets	0	0.000	
dsfs_mounts	1	442.953	
dsfs_unmounts	0	0.000	
dsfs_vinacts	0	0.000	
dsfs_sync	0	0.000	
dsfs_creparms	390	0.123	
dsfs_extattr	0	0.000	
-----	-----	-----	
TOTALS	26725491	0.009	

Operation

The name of the DSFS entry point for the operation and is the list of services the DSFS PFS provides to z/OS UNIX and its applications.

Count

Number of calls for each operation.

Avg Time

Average response time in milliseconds.

Bytes

For operations that transfer data to and from application address spaces, the total number of bytes moved.

The ***TOTALS*** line shows the overall count of DSFS calls and the average DSFS response time.

MODIFY DSFS QUERY,LEVEL

Output examples are shown for the MODIFY DSFS QUERY,LEVEL report.

```
IDFS00017I DSFS kernel: z/OS    DSFS
Version 02.05.00 Service Level 000000 - HZFS450.
Created on Mon Mar 1 14:22:50 EST 2021.
IDFS00020I DSFS kernel: MODIFY command - QUERY,LEVEL completed
successfully.
```

Message IDFS00017I shows the name of the product subcomponent (DSFS Kernel) and the name of the product (z/OS DSFS) on line 1. On line 2, the OS version is shown, and the service level of the product is shown (000000 is the GA level of the product). HZFS450 is the FMID of the product. The final line shows the date, time, and time zone of the build of the product.

MODIFY DSFS QUERY,LFS

The LFS report shows various internal counters of events and resource usage as it interfaces with the utility file system. An example of the report is as follows.

```
F DSFS,QUERY,LFS
IDFS00139I Starting Query Command LFS. 335
                DSFS Vnode Cache Statistics

Vnodes      Requests      Hits      Ratio  Allocates      Deletes
-----
      1000        38642        37616   97.345%           0         813

DSFS Vnode structure size: 240 bytes
DSFS extended vnodes: 1000, extension size 1088 bytes (minimum)
Held DSFS vnodes:      1 (high      128)
Open DSFS vnodes:      0 (high      6)
Reusable:            999

Total osi_getvnode Calls:      1043 (high resp      0)
Avg. Call Time:                0.014 (msecs)
Total SAF Calls:                26805 (high resp      0)
Avg. Call Time:                0.043 (msecs)

                Metadata Caching Statistics

Buffers      (K bytes)  Requests      Hits      Ratio  Updates      PartialWrt
-----
      131072        1024      108795      104941   96.4%      13388          0

Compression calls:      6078  Avg. call time:      0.000
KB input                411216  KB output      0
Decompression calls:    5892  Avg. call time:      0.000
KB input                48864  KB output      0
```

This report has three sections that provide information on two internal caches used by the DSFS components that process the utility file system. The first report is the **DSFS Vnode Cache Statistics**, and a description of its fields is explained as follows.

Vnodes

The number of vnodes in the cache. A vnode represents a file system object (file or directory, which for DSFS represents HLQs, data sets and data set members) and z/OS UNIX might point to these vnodes from its structures.

Requests

The number of requests to find a vnode in the cache for a file or directory.

Avg Time

Average response time in milliseconds.

Hits

The number of times the requested vnode was found in the cache. If the requested inode is not found, then a new one needs to be created. Creating a new vnode often requires using an existing vnode that represented another object, syncing any data for it to the utility file system, and then using it for the newly requested object.

Ratio

Ratio of hits to requests, expressed as a percentage, the higher the better for performance.

Allocates

The number of requests to allocate a new vnode for an object rather than reusing an existing vnode structure that represented another object.

Deletes

The number of times a request was made to delete a vnode from the cache.

Total osi_getvnode Calls

The number of times that DSFS paired one of its vnodes with the corresponding z/OS UNIX structure. Osi_getvnode is a z/OS UNIX function. The following fields are related to this field.

- **High resp.** Count of times that DSFS finds a high response time (greater than 1 second) for an osi_getvnode call.
- **Avg Call Time.** Average response time in milliseconds of osi_getvnode calls that DSFS makes.

Total SAF Calls

The number of times that DSFS makes a SAF call related to security checking of application access to data sets. These calls could be to set up or tear down a security context to represent a requesting user or make a security query of some sort. The following fields are related to this field.

- **High resp.** Count of times that DSFS finds a high response time (greater than 1 second) for a SAF call.
- **Avg Call Time .** Average response time in milliseconds of SAF calls that DSFS makes.

Additionally, the QUERY,LFS command shows the performance of its internal metadata cache that is used to cache directory contents and other utility file system disk structures. It caches entire blocks from the utility file system data set, where a block is 8 K in size. The description of these fields is as follows.

Buffers

The number of 8 K buffers in the cache.

K Bytes

The total number of bytes in the cache (number of buffers multiplied by 8 K).

Requests

The number of calls to the cache to search for a cached utility file system disk block.

Hits

The number of times a call to search for a disk block in the cache found that the block was cached.

Ratio

The percentage of times a call to search for a disk block in the cache found the block. The higher the percentage the better.

Updates

The number of times DSFS updated fields in a cached disk block in the metadata cache, which implies that disk block needs to be written to the utility file system disk.

Partial Wrt

The DSFS utility file system blocks data at 8 K. Thus, the utility file system is a large array of 8 K blocks. A VSAM linear data set tagged as ZFS type is blocked at 4K-control intervals. DSFS tracks a cached disk block and determines whether only the high half or low half of an 8 K block was updated. If so, DSFS can write only the high half or low half to disk to reduce the number of bytes transferred to disk.

Lastly, the QUERY,LFS command shows the compression statistics for files in the file cache and the utility file system.

Compression calls

The number of compression calls.

Decompression calls

The number of decompression calls.

Average call times

The average number of milliseconds per compression or decompression call.

KB input

The number of kilobytes sent to zEDC for compression or decompression calls.

KB output

The number of kilobytes returned from zEDC for compression or decompression calls.

MODIFY DSFS QUERY,LOCK

The LOCK report shows lock and resource contention in DSFS. Following is an example of the command.

```
F DSFS,QUERY,LOCK
IDFS00139I Starting Query Command LOCKING. 340
                Locking Statistics

Untimed sleeps:      1008 Timed Sleeps:      0 Wakeup:      1958

Total waits for locks:      1335428
Average lock wait time:      0.050 (msecs)

Total monitored sleeps:      18624
Average monitored sleep time:      7.025 (msecs)

Total starved waiters:      0
Total task priority boosts:      64

                Top 15 Most Highly Contended Locks
Thread Async  Spin
Wait  Disp.  Resol.  Pct.    Description
-----
1471591      0 440755    95.92%  Vnode-cache access lock
3503         0 76227     4.00%  File-cache segment slot lock
1189         0   14      0.06%  Vnode lock
38           0  225     0.01%  File cache main segment lock
7            0   58     0.00%  Anode bitmap allocation handle lock
34           0   2      0.00%  Service threads global block lock
0            0   3      0.00%  Metadata-cache buffer lock
0            0   2      0.00%  CDI0 bitmap lock
1            0   1      0.00%  Cache Services item refcount lock
0            0   1      0.00%  Size class partial lock
0            0   0      0.00%  Async IO device lock
0            0   0      0.00%  Async IO set free list lock
0            0   0      0.00%  Async IO event free list lock
0            0   0      0.00%  Async global device lock
0            0   0      0.00%  LVM global lock

Total lock contention of all kinds:      1993651

                Top 15 Most Common Thread Sleeps
Thread Wait  Pct.    Description
-----
17620      94.61%  Request completion on server thread
1002       5.38%  Task waiting for DS retrieval
1          0.01%  LOG wait for outstanding log page IO
1          0.01%  CTKC in-progress bitmap pool reservation
0          0.00%  XVOL file system unsure operation count
processing
0          0.00%  ADMIN DSFS dump is in progress
0          0.00%  XVOL file operation in progress
0          0.00%  XVOL file system command in progress
0          0.00%  XAGGR namespace altering command in progress
0          0.00%  XAGGR cache resize in progress
0          0.00%  CTKC user file pending IO wait
0          0.00%  Transaction allocation wait
0          0.00%  OSI cache item init wait
0          0.00%  OSI cache empty wait
0          0.00%  OSI cache item cleanup wait
```

This report is intended for service personnel. The fields are described as follows.

Untimed Sleeps

The number of times a task slept waiting for some event to occur.

Timed Sleeps

The number of times a task slept a fixed amount of time while waiting for an event to occur.

Wakeup

The number of times a task was woken up from its sleep.

Total wait for locks

The number of times a task had to wait for a lock that was in use in a noncompatible mode.

Average lock wait time

The average amount of time in milliseconds a task had to wait for a lock.

Total monitored sleeps

The number of times a task went to sleep where the length of time it slept was monitored for this report.

Average monitored sleep time

The average amount of time in milliseconds a task had to wait while sleeping.

Total starved waiters

Number of times it was detected that a task was being starved while access to a lock was being obtained. DSFS takes action to give these tasks priority.

Total task priority boosts

DSFS will boost the priority of user application tasks in its address space if it notices that they are not being dispatched. These tasks might be holding resources that are needed by other tasks.

Top 15 Most Highly Contended Locks

The most highly contended locks because contention on those negatively affect response time are listed. The columns in this report are as follows.

Thread Wait

The number of times a task was made to wait for a lock.

Async Disp

DSFS has an internal method to dispatch work to the lock instead of making a task wait for the lock. **Async Disp** is the count of how often the work was dispatched.

Spin Resol.

DSFS will attempt to make a task spin for a short time to get the lock. Spinning tasks is often more efficient than making the task wait and then waking it up.

Pct

The percentage of all lock waits that waiting on the described lock represents.

Description

Description of the lock that has the contention.

Top 15 Most Common Thread Sleeps

This report shows the number of times a task slept waiting for an event to occur. The columns in this report are described as follows:

Thread Wait

Number of times a task slept for the reason that is listed in the **Description** field.

Pct

The percentage of all sleeps that the sleep for the described reason represents.

Description

Description of the event that a task is waiting for.

MODIFY DSFS QUERY,SETTINGS

Output examples are shown for the MODIFY DSFS QUERY,SETTINGS report.

```

9I Starting Query Command SETTINGS. 499
      Program Options Used
Option      Value      Description
-----
COMPRESS    OFF      Initialize
utility file system for compression
DIRECTORY_POOL_SIZE  64      Number of tasks
to process directory requests
DIRECTORY_REFRESH_TI  60      Time in secs to
refresh directories
ENABLE_SYSOUT  ON      Allow access of
JES Spool data sets
FILECACHE_SIZE  10M     Size of cache
used to contain user file data
FSFULL      Indicates if
threshold monitoring will be performed when utility file system is low

```

MODIFY DSFS QUERY,STORAGE

on space		
IO_POOL_SIZE	128	Number of tasks
to do background file IO		
ISPF_EXTENDED_STATIS	OFF	Indicates if
storing num of lines (>65535) at file close time		
HLQ_MODE	EXCLUDE	Indicates if
listed HLQ directories should be excluded		
HLQ_LIST		List of HLQ
directory names excluded in path directory		
MSG_INPUT_DSN		Dataset name
where messages are located		
PDS_ENQ_DURATION	60	Time in seconds
to keep PDS ENQs		
PDS_INTERVAL	60	Number of
seconds between directory refresh		
PS_DYN_DURATION	60	Time in secs to
hold sequential data set DYNALLOC		
PS_INTERVAL	60	Time in secs to
unallocate sequential data set		
SECURITY_POOL_SIZE	32	Number of tasks
to create security caches		
SECURITY_TIMEOUT	60	Time in secs to
cache security info		
SYSOUT_DIRECTORY_P00	4	Number of tasks to
process sysout directory requests		
SYSOUT_IO_POOL_SIZE	3	Number of tasks to
do background file IO		
TRACE_TABLE_SIZE	1024M	Size of DSFS
kernel trace table		

The output has three columns, although the third column will wrap to the next line in the system log.

Option

The name of the corresponding kernel setting documented in [Chapter 12, “The DSFS configuration file \(IDFPRMxx or IDFFSPRM\)”](#), on page 113. Longer option names might be truncated due to the size of the display field.

Value

The current value used for this option. If the administrator lets the value default by not specifying it in the IDFPRMxx file, then this is the default value. However, this value will show any changes that were made dynamically by the administrator with the **dsadm config** command.

Description

Text description of the option.

MODIFY DSFS QUERY,STORAGE

The STORAGE report is generally used for problem diagnosis. Most of the fields are intended for service personnel, though the overall memory usage inside the DSFS address space might be useful for administrators. This report was also described in [“Performance tuning”](#) on page 45..

An example of the command is as follows.

```
F DSFS,QUERY,STORAGE
IDFS00139I Starting Query Command STORAGE. 184
          DSFS Primary Address Space <2G Stge Usage
          -----
Total Storage Below 2G Bar Available: 1936719872
Non-critical Storage Limit:          1915748352
USS/External Storage Access Limit:  1873805312
Total Storage Below 2G Bar Allocated: 60899328

IDFFSCM Heap Bytes Allocated:        18989800
IDFFSCM Heap Pieces Allocated:        1394
IDFFSCM Heap Allocation Requests:     1394
IDFFSCM Heap Free Requests:           0

IDFFSKN Heap Bytes Allocated:        4751750
IDFFSKN Heap Pieces Allocated:        52672
IDFFSKN Heap Allocation Requests:     52836
IDFFSKN Heap Free Requests:           164

          DSFS Primary Address Space >2G Stge Usage
          -----
```

```

Total Storage Above 2G Bar Available:      209715200K
Total Storage Above 2G Bar Allocated:      1288699904

Total Bytes Allocated by IDFFSCM (Stack+Heap): 1048576
IDFFSCM Heap Bytes Allocated:              1048576
IDFFSCM Heap Pieces Allocated:              1
IDFFSCM Heap Allocation Requests:           1
IDFFSCM Heap Free Requests:                 0

Total Bytes Allocated by IDFFSKN (Stack+Heap): 1272971264
Total Bytes Discarded (unbacked) by IDFFSKN: 0
IDFFSKN Heap Bytes Allocated:              1247628557
IDFFSKN Heap Pieces Allocated:              132949
IDFFSKN Heap Allocation Requests:           132964
IDFFSKN Heap Free Requests:                 15

```

Total storage below 2G bar available/Total storage above 2G bar available

Total virtual storage in the DSFS address space below or above the bar that is available for usage (such as caches, control blocks, and stacks).

Non-critical Storage Limit

The value that, when exceeded, will cause DSFS to issue the following message:

```
IDFS00166I DSFS is critically low on storage
```

USS/External Storage Access Limit

The value that, when exceeded, will cause DSFS to issue the following message:

```
IDFS00165I DSFS is low on storage
```

Total storage below 2G bar allocated/Total storage above 2G bar allocated

The current usage of virtual storage in the DSFS address space (requested by DSFS and other components that are running in the DSFS address space).

IDFFSCM Heap Bytes Allocated/IDFFSKN Heap Bytes Allocated

The current amount of storage that is allocated to the DSFS heaps.

IDFFSCM Heap Pieces Allocated/IDFFSKN Heap Pieces Allocated

The current number of storage pieces that are in the IDFFSCM and IDFFSKN heaps.

Total Bytes Allocated by IDFFSCM (Stack + Heap)/ Total Bytes Allocated by IDFFSKN (Stack + Heap)

The total bytes of storage that is allocated by the DSFS IDFFSCM and IDFFSKN components.

IDFFSCM Heap Allocation Requests/ IDFFSKN Heap Allocation Requests

Number of requests that DSFS made to obtain heap storage since the last DSFS storage statistics reset.

IDFFSCM Heap Free Requests/ IDFFSKN Heap Free Requests

Number of requests that DSFS made to free heap storage since the last DSFS storage statistics reset.

Total Bytes Discarded (unbacked) by IDFFSKN

Total number of bytes that IDFFSKN has discarded (made unbaked) from allocated storage.

MODIFY DSFS QUERY,THREADS

The THREADS report shows the state of all running tasks inside DSFS. It might be used by an administrator if a hang is suspected. Following is an example output.

```

F DSFS,QUERY,THREADS
IDFS00139I Starting Query Command THREADS. 578
          DSFS and z/OS UNIX Tasks
          -----
STK/Recov   TCB      ASID   Stack      Routine      State
-----
506A878000  005CFA60  0057   506A878748  DSFRDWR      RUNNING
50000DCF00
      since Mar 2 18:28:01 2021
      Operation counted for OEVS=7E78D360 VOLP=506A910000
      fs=DSFSAGGR.BIGZFS.UTILITY.FS.DCEIMGHQ

506A870000  005CFA60  0057   506A870748  DSFRDWR      RUNNING
50000DD7C0
      since Mar 2 18:28:01 2021

```

MODIFY DSFS QUERY,THREADS

```
7E2E7000 005C3658 004A 7E2E8000 comm_daemon RUNNING
5000001A40
    since Mar 2 18:28:01 2021
7BC3D000 005816D8 004A 7BC3E000 xcf_server WAITECB
50000D3A40
    since Mar 1 22:10:53 2021 Current DSA: 7BC3E5F8
    wait code location offset=1404 rtn=xcf_server
```

This command is provided mostly for service personnel, but the administrator can use several of the fields to determine whether a particular task is progressing. Here is an explanation of the fields of interest to the administrator:

STK/RECOV

The address of the data structures that the IDFFSCM (STK) and IDFFSKN (RECOV) use to represent a running task.

TCB/ASID

The structures that z/OS use to identify a running task (TCB) and its associated address space (ASID). If the ASID matches the DSFS ASID, then it is a DSFS task; else, it is a user or OMVS task who has called into DSFS.

Routine

The main routine of the task if it's a DSFS task. If it's a user task, then it's the main entry point that they called in DSFS and an indicator of the work that they requested. For example, DSFRDWR indicates a call to read or write a file.

State

The state of the task, either RUNNING or in some wait state.

Since <time>

The **since** field is important because it tells you how long that task has been in the given state. If the task is not a DSFS task, then a task that has been in the same state for a long time might be looping or hung. Thus, a hang detect or loop is analyzed by finding non-DSFS tasks that are in the same state for a long time and that state never changes.

The rest of the information in this query is intended for service personnel.

SETOMVS RESET

Purpose

Use SETOMVS RESET to start the DSFS physical file system (PFS) if it was not started at IPL. SETOMVS RESET can also be used to redefine it if it was terminated by replying **i** to the BPXF032D operator message after stopping the DSFS PFS.

Format

```
SETOMVS RESET=(xx)
```

Parameters

xx

The suffix of a BPXPRMxx member of PARMLIB that contains the FILESYSTYPE statement for the DSFS PFS.

Usage

The SETOMVS RESET command can be used to start the DSFS PFS.

Privilege required

This command is a z/OS system command.

Example

The following command starts the DSFS physical file system if the BPXPRMSS member of the PARMLIB contains the DSFS FILESYSTYPE statement:

```
SETOMVS RESET=(ss)
```

Related information

File: BPXPRMxx

The SETOMVS command also processes other statements. For more information, see [SETOMVS command](#) in *z/OS MVS System Commands*.

Chapter 11. DSFS commands

This section provides a description of DSFS commands. In the options section for each command, options are described in alphabetic order to make them easier to locate. This does not reflect the format of the command. The formats are presented the same as on your system. In addition to displaying z/OS UNIX reason codes, the z/OS UNIX shell command **bpixmttext** also displays the text and action of DSFS reason codes (EDxxnnnn) returned from the kernel. DSFS does not use the xx part of the reason code to display a module name. It always displays DSFS. If you only know the *nnnn* part of the DSFS reason code, you can use ED00nnnn as the reason code. The date and time that is returned with the DSFS reason code matches the date and time returned from the DSFS kernel (displayed with operator command MODIFY DSFS QUERY,LEVEL).

MOUNT

Purpose

MOUNT is a TSO/E command that mounts a file system into the z/OS UNIX hierarchy. This section documents only the MOUNT options that are unique to DSFS. MOUNT can also be invoked from the z/OS UNIX shell (/usr/sbin/mount). For more information about MOUNT, see [MOUNT - Logically mount a file system](#) in *z/OS UNIX System Services Command Reference*.

For DSFS, only one MOUNT is allowed per system in a nonshared file system environment. Only one MOUNT is allowed per sysplex in a shared file system environment, to the path: /dsfs. The FILESYSTEM keyword of the MOUNT command should be specified according to the guidelines in [“Mounting the utility file system”](#) on page 27.

First mount of a utility file system

The first time a utility file system is used by DSFS, it is formatted automatically by DSFS. DSFS provides a COMPRESS option in the IDFPRMxx configuration file that controls whether DSFS will enable the utility file system for compression. The COMPRESS option affects only this initial mount of the utility file system. It does not affect subsequent mounts of that same file system because it is already formatted.

Format

```
MOUNT TYPE(DSFS) MOUNTPoint('/dsfs') FILESYSTEM('utility_file_system_name') MODE(RDWR)
```

Options

TYPE

The type must be the string DSFS.

MOUNTPoint

The mount point must be the path '/dsfs'.

FILESYSTEM

For guidelines on specifying an appropriate value, see [“Mounting the utility file system”](#) on page 27.

MODE

The mode must be RDWR for DSFS or the MODE parameter that is omitted which defaults to RDWR.

Usage notes

1. The first time the utility file system is used, DSFS will format the file system for its use.

- If the data set has a key label, then DSFS will initialize the file system so that it encrypts any file contents that are stored in the utility file system. DSFS must have access to the key label.
 - If the COMPRESS=ON IDFPRMxx parameter option is specified and the file system has not been formatted yet (first mount) then the file system is initialized so that any file contents are stored in compressed format in the file system. COMPRESS only affects first-time mounts of the utility file system. DSFS also attempts to fix its file cache in memory by using zEDC services to allow for compression.
2. After the mount, DSFS will check the IDFPRMxx configuration file for the FSFULL parameter setting. If FSFULL is set, DSFS initializes the utility file system to report space usage based on the FSFULL setting.

Example 1

```
MOUNT TYPE(DSFS) MOUNTPPOINT('/dsfs') FILESYSTEM('MY.UTILITY.FS') MODE(RDWR)
```

On a system where the UTFS_NAME configuration option has no value, a VSAM linear data set with the ZFS keyword must have been defined with the name MY.UTILITY.FS.SYSNAME where SYSNAME is the name of the system where DSFS is started. If this is a shared file system sysplex implementation, there is only one singular mount. DSFS on each sysplex member will automatically append its local system name to the provided string in the FILESYSTEM mount parameter when it receives a mount from z/OS UNIX.

The name of the VSAM linear data set this system that is using can be seen with the **dsadm fsinfo** command, or in the Aggregate Name field displayed with the z/OS UNIX **df -v /dsfs** command.

```
Mounted on      Filesystem      Avail/Total    Files      Status
/dsfs           (MY.UTILITY.FS) 2256336/2264000 4294966679 Available
DSFS, Read/Write, Device:50, ACLS=N
File System Owner : SY1      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : MY.UTILITY.FS.SY1
```

Example 2

Alternatively, on a system where the UTFS_NAME configuration option has the value PLEX.UTILITY.FS1, a VSAM linear data set with the name PLEX.UTILITY.FS1 must have been defined with the ZFS keyword. This data set will be used as the utility file system. The name of the VSAM linear data set that this systems is using can be seen with **dsadm fsinfo** command, or in the Aggregate Name field displayed with the z/OS UNIX **df -v /dsfs** command.

```
Mounted on      Filesystem      Avail/Total    Files      Status
/dsfs           (MY.UTILITY.FS) 2256336/2264000 4294966679 Available
DSFS, Read/Write, Device:50, ACLS=N
File System Owner : SY1      Automove=Y      Client=N
Filetag : T=off  codeset=0
Aggregate Name : PLEX.UTILITY.FS1
```

Related information

Commands:

UNMOUNT

For more information about UNMOUNT, see [UNMOUNT - Remove a file system from the file hierarchy in z/OS UNIX System Services Command Reference](#).

Files:

IDFFSPRM

The dsadm command suite

Purpose

This section introduces the **dsadm** command suite. The **dsadm** command is run from the z/OS UNIX shell. You can also invoke it from TSO/E by using the program name IDFZADM or as a batch job by using PGM=IDFZADM. If PARM is coded in the JCL to pass options or arguments to IDFZADM and any of the options or arguments contain a slash (for example, fileinfo), you must specify a leading slash as the first character in the PARM string. For example,

```
PARM=(' /fileinfo -path /dsfs/txt/hlq')
```

Following is an example of invoking IDFZADM from a batch job.

```
//USERIDA JOB , 'DSADM FSINFO',
//          CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//DSFSF   EXEC   PGM=IDFZADM,REGION=0M,
// PARM=('fsinfo')
//SYSPRINT DD      SYSOUT=H
//STDOUT   DD      SYSOUT=H
//STDERR   DD      SYSOUT=H
//SYSUDUMP DD      SYSOUT=H
//CEEDUMP  DD      SYSOUT=H
//*
```

Command syntax

The **dsadm** commands have the same general structure:

```
command {-option1 argument...|-option2 {argument1|argument2}...}[-optional_information]
```

The following example illustrates the elements of a **dsadm** command:

```
dsadm salvage {-verifyonly | -cancel} [-trace <file_name>][-level][-help]
```

The following list summarizes the elements of the **dsadm** command:

Command

A command consists of the command suite (**dsadm** in the previous example) and the command name (**salvage**). The command suite and the command name must be separated by a space. The command suite specifies the group of related commands.

Options

Command options always appear in monospace type in the text, are always preceded by a - (dash), and are often followed by arguments. In the previous example, **-trace** is an option, with file name as its argument. An option and its arguments tell the program which entities to manipulate when running the command (for example, the name of a trace file). In general, the issuer should provide the options for a command in the order that is detailed in the format description. The { | } (braces separated by a vertical bar) indicate that the issuer must enter either one option or the other (**-verifyonly** or **-cancel** in the previous example). Command options are described in alphabetic order to make them easier to locate; this does not reflect the format of the command. The formats are presented the same as on your system.

Arguments

Arguments for options are highlighted in the text. The { | } indicate that the issuer must enter either one argument or the other (**-verifyonly** or **-cancel** in the preceding example). The ... (ellipsis) indicates that the issuer can enter multiple arguments.

Options

Some commands have optional, as well as required, options, and arguments. Optional information is enclosed in [] (brackets). All options except **-verifyonly** or **-cancel** in the previous example are optional.

Options

The following options are used with many **dsadm** commands. They are also listed with the commands that use them.

-help

Prints the online help for this command. All other valid options that are specified with this option are ignored. For more information about receiving help, see [“Receiving help” on page 87](#).

-trace file_name

Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.

-level

Displays the service level of the **dsadm** program. All other valid options that are specified with this option are ignored.

When an option is specified multiple times on one command, the first one is accepted and the subsequent ones are ignored. This can cause a subsequent argument to be interpreted as an option and be diagnosed as unrecognized.

Usage notes

1. Most **dsadm** commands are administrative-level commands that are used by system administrators to manage DSFS. You can issue commands from OMVS, TSO/E, or as a batch job. Use the IDZADM format for TSO/E and batch.
2. For a batch job, the **dsadm** options are specified in the EXEC PARM as a single subparameter (a single character string enclosed in apostrophes with no commas separating the options). You cannot put the ending apostrophe in column 72. If it needs to go to the next line, use a continuation character in column 72 (continuing in column 16 with the ending apostrophe on the second line). Remember that a JCL EXEC PARM is limited to 100 characters. For more information about EXEC PARM, see [PARM parameter in z/OS MVS JCL Reference](#).
3. **dsadm** commands are serialized with each other. When a **dsadm** command is in progress, a subsequent **dsadm** command is delayed until the active **dsadm** command completes. This also does not include long-running **dsadm** commands such as **dsadm salvage**. **dsadm** commands do not delay normal file system activity (except when the **dsadm** command requires it, such as **dsadm salvage**).
4. When supplying an argument to a **dsadm** command, the option (for example -path for the **dsadm createparm** command) that is associated with the argument can be omitted if:
 - All arguments that are supplied with the command are entered in the order in which they appear in the command's syntax. (The syntax for each command is provided.)
 - 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.
 - The first option cannot be followed by an additional option before the vertical bar.

In the case where two options are presented in

```
{ | }
```

(braces separated by a vertical bar), 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.

An exception is **dsadm config**; see usage note 5.

5. The **dsadm config** command is an exception to usage note 4 in that it requires at most one option to be set and the name of the option cannot be omitted. (See [“dsadm config” on page 90](#).)

If it must be specified, an option can be abbreviated to the shortest possible form that distinguishes it from other options of the command. For example, the `-path` option found in some **dsadm** commands can typically be omitted or abbreviated to be simply `-p`.

It is also valid to abbreviate a command name to the shortest form that still distinguishes it from the other command names in the suite. For example, it is acceptable to shorten the **dsadm salvage** command to **dsadm** because no other command names in the **dsadm** command suite begin with the letter `s`. However, there are two **dsadm** commands that begin with `f`: **dsadm fileinfo** and **dsadm fsinfo**. To remain unambiguous, they can be abbreviated to **dsadm fi** and **dsadm fs**.

The following examples illustrate three acceptable ways to enter the same **dsadm salvage** command:

- Complete command:

```
dsadm salvage -verifyonly
```

- Abbreviated options:

```
dsadm s -v
```

- Omitted argument:

```
dsadm salvage
```

6. The ability to abbreviate or omit options is intended for interactive use. If you embed commands in a shell script, do not omit options or abbreviate them. If an option is added to a command in the future, it might increase the minimum unique abbreviation that is required for an existing option or change the order of options.

Receiving help

There are several different ways to receive help for **dsadm** commands. The following examples summarize the syntax for the different help options available:

dsadm help

Displays a list of commands in a command suite.

dsadm help -topic *command*

Displays the syntax for one or more commands.

dsadm apropos -topic *string*

Displays a short description of any commands that match the specified *string*.

When the **dsadm** command displays help text or a syntax error message, it will show the name of the command as **IDFZADM**, instead of **dsadm**. This occurs because the **dsadm** command is not a binary module in the z/OS UNIX file system. Rather, it is a shell script that invokes **IDFZADM**. **IDFZADM** is an entry that has the sticky bit on in the permissions. The sticky bit means that the **IDFZADM** module is found and executed from the user's STEPLIB, link pack area, or link list concatenation. (**IDFZADM** is typically located in **SYS1.SIEALNKE**.) However, you cannot run **IDFZADM** from the shell because **IDFZADM** is not normally in your **PATH**.

Privilege required

dsadm commands that query information (for example, **fileinfo**, **fsinfo**) usually do not require the issuer to have any special authority. **dsadm** commands that modify (for example, **salvage**) usually require the issuer to have one of the following authorizations:

- UID of 0. If you are permitted READ to the BPX.SUPERUSER resource in the RACF® FACILITY class, you can become a UID of 0 by issuing the **su** command.
- READ authority to the SUPERUSER.FILESYS.PFCTL resource in the z/OS UNIXPRIV class.

Related information

Commands:

dsadm apropos
dsadm config
dsadm configquery
dsadm createparm
dsadm fileinfo
dsadm fsinfo
dsadm help
dsadm level
dsadm query
dsadm salvage

File:

IDFFSPRM

dsadm apropos

Purpose

dsadm apropos shows each help entry that contains a specified string.

Format

```
dsadm apropos -topic string [-level] [-help] [-trace file_name]
```

Options

-help

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

-level

Prints the level of the **dsadm** command. This is useful when you are diagnosing a problem. Except for -help, all other valid options that are specified with -level are ignored.

-topic

Specifies the keyword string to search. If it is more than a single word, surround it with quotation marks (" ") or another delimiter. Type all strings for **dsadm** commands in all lowercase letters.

-trace *file_name*

Specifies the name of the file that will have the trace records written into it. The trace file is a z/OS UNIX file.

Usage notes

The **dsadm apropos** command displays the first line of the online help entry for any **dsadm** command containing the string specified by -topic in its name or short description. To display the syntax for a command, use the **dsadm help** command.

Privilege required

The issuer does not need special authorization.

Results

The first line of an online help entry for a command lists the command and briefly describes its function. This command displays the first line for any **dsadm** command where the string specified by -topic is part of the command name or first line.

Examples

The following command lists all **dsadm** commands that have the word *file* in their names or short descriptions:

```
dsadm apropos file
fileinfo: Obtain information on a file or directory
fsinfo: Obtain utility file system information
salvage: salvage utility file system
```

Related information

Commands:

dsadm help

dsadm config

Purpose

dsadm config changes the value of the DSFS configuration file (IDFFSPRM) options in memory. For a complete list of IDFFSPRM options, see [Chapter 12, “The DSFS configuration file \(IDFPRMxx or IDFFSPRM\),” on page 113.](#)

Format

```
dsadm config [{-directory_pool_size <number> |
-dictionary_refresh_timeout <number> |
-filecache_size <cache_size[,fixed|edcfixed]> |
-fsfull <(threshold,increment) | (off)> |
-hlq_list_add <hlq1,hlq2,...> | -hlq_list_remove <hlq1,hlq2,...> |
-io_pool_size <number> | -ispf_extended_statistics <on | off> |
-pds_end_duration <number> | -pds_interval <number> |
-ps_dyn_duration <number> | -ps_interval <number> |
-security_pool_size <number> | -security_timeout <number> | }]
[-trace <file_name>] [-level] [-help]
```

Options

-directory_pool_size <number>

Specifies the number of processing threads used to handle directory requests that require access to DFSMS data sets or catalogs.

-dictionary_refresh_timeout <number>

Specifies the length of time in seconds between successive catalog searches to verify that the DSFS cached directory contents match those of the catalog for HLQ directories. It is also the length of time in seconds between successive PDSE directory searches to verify that the DSFS cached directory contents match those of the PDSE data set.

-filecache_size <cache_size[,fixed|edcfixed]>

Specifies the size, in bytes, of the cache that is used to contain file data. The **fixed** and **edcfixed** options can fix the file cache in real memory:

- The **fixed** option avoids page fix and page unfix for disk I/Os that do not use compression.
- The **edcfixed** option avoids page fix and page unfix for disk I/Os that use compression. It also avoids data movement for compression I/Os. DSFS will automatically fix the cache in **edcfixed** format if the utility file system is enabled for compression.

-fsfull <(threshold,increment) | off>

Specifies the threshold and increment when sending file system utilization messages to the operator.

-hlq_list_add [<hlq1,hlq2,...>]

Allows the administrator to add high-level qualifiers that are to be excluded. The **-hlq_list_add** and **-hlq_list_remove** options are mutually exclusive. They cannot be specified at the same time with the same **dsadm config** command.

- The high-level qualifiers that are listed must be separated by commas with no intervening spaces.
- The specified high-level qualifiers can consist of one or more qualifiers.
- After a high-level qualifier is added to the HLQ list, the associated HLQ directories are removed from the UNIX tree after all users finish accessing objects in the HLQ directory subtree.

-hlq_list_remove <hlq1,hlq2,...>

Allows the administrator to remove high-level qualifiers from the list. The **-hlq_list_add** and **-hlq_list_remove** options are mutually exclusive. They cannot be specified at the same time with the same **dsadm config** command.

- The high-level qualifiers that are listed must be separated by commas with no intervening spaces.

- The specified high-level qualifiers can consist of one or more qualifiers.
- io_pool_size <number>**
Specifies the number of DSFS worker threads that are used to process data set IO requests using the QSAM access methods.
- ispf_extended_statistics <ON | OFF>**
Specifies whether DSFS should retrieve and store ISPF extended statistics when processing PDS and PDSE members.
- pds_enq_duration <number>**
Specifies the number of seconds that DSFS will hold enqueues on PDS/PDSE data sets, which allows DSFS to cache their directory contents for that period of time and reduce the number of calls to read the PDS/PDSE data set directory.
- pds_interval <number>**
Specifies the number of seconds between the time a DSFS task will look at cached PDS and PDSE directories to determine whether they have exceeded their enqueue duration. If so, then it releases the enqueue.
- ps_dyn_duration <number>**
Specifies the number of seconds that DSFS will hold a dynamic allocation on PDS/PDSE members and PS data sets. Holding the dynamic allocation allows DSFS to cache their contents for that period of time and reduces the number of calls to read the data set.
- ps_interval <number>**
Specifies the number of seconds between the time a DSFS task will look at cached PDS/PDSE members and PS data sets to determine whether they have exceeded their dynamic allocation duration. If so, then it releases the allocation, which will cause DSFS to uncache the file's contents.
- security_pool_size <number>**
Specifies the number of DSFS worker threads that are used to process SAF security calls.
- security_timeout <number>**
Specifies the number of seconds that DSFS will use its cached security access query for a user's access to a data set. If the number of seconds since the last query exceeds this value, then DSFS will make another query to the SAF product to determine the user's access.
- trace <file_name>**
Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.
- level**
Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem. Except for **-help**, all other valid options that are specified with **-level** are ignored.
- help**
Prints the online help for this command. All other valid options that are specified with this option are ignored.

Usage notes

1. The **dsadm config** command changes the configuration options (in memory) that were specified in the IDFFSPRM file (or defaulted). The IDFFSPRM file is not changed. If you want the configuration specification to be permanent, you must modify the IDFFSPRM file because DSFS reads the IDFFSPRM file to determine the configuration values when DSFS is started. The values that can be specified for each option are the same as the values that can be specified for that option in the IDFFSPRM file.
2. The **dsadm config** command does not accept positional parameters.
3. The values for **-directory_pool_size**, **-security_pool_size**, and **-io_pool_size** are only allowed to be increased. They can only be decreased by updating the IDFFSPRM file, stopping DSFS and restarting it.

Privilege required

The issuer must be logged in as a root user (UID=0) or have READ authority to the SUPERUSER.FILESYS.PFSCCTL resource in the z/OS UNIXPRIV class.

Examples

The following example changes the size of the file cache:

```
dsadm config -filecache_size 64M
Successfully set -filecache_size to 64M
```

Related information

Commands:

dsadm configquery

Files:

IDFFSPRM

dsadm configquery

Purpose

dsadm configquery queries the current value of DSFS configuration options.

Format

```
dsadm configquery [-all] [-compress][-directory_pool_size]  
[-directory_refresh_timeout] [-filecache_size] [-fsfull]  
[-hlq_mode] [-hlq_list] [-io_pool_size]  
[-ispf_extended_statistics] [-msg_input_dsn] [-pds_enq_duration]  
[-pds_interval] [-ps_dyn_duration] [-ps_interval]  
[-security_pool_size] [-security_timeout][-trace_table_size]  
  
[-trace <file_name>][-utfs_name][-level] [-help]
```

Options

-all

Displays the full set of configuration options.

-compress

Displays whether a newly defined utility file system will be formatted to use compression upon its first mount.

-directory_pool_size

The number of processing threads used to handle directory requests that require access to DFSMS data sets or catalogs.

-directory_refresh_timeout

The length of time in seconds between successive catalog searches to verify that the DSFS cached directory contents match those of the catalog for HLQ directories. It is also the length of time in seconds between successive PDSE directory searches to verify that the DSFS cached directory contents match those of the PDSE data set.

-filecache_size

Displays the size, in bytes, of the cache that is used to contain file data.

-fsfull

Displays the threshold and increment for reporting file system full error messages to the operator.

-hlq_mode

Displays the HLQ mode that is used by DSFS.

EXCLUDE

The HLQ_LIST will list HLQ directory names that are not allowed in the DSFS tree. Any HLQ directory that is not in the HLQ_LIST is allowed in the DSFS tree. EXCLUDE is the default if HLQ_MODE is not provided in the IDFFSPRM file.

-hlq_list

Displays the list of HLQ directory names that are not allowed access by DSFS users. The default is an empty list.

io_pool_size

Displays the number of DSFS worker threads that are used to process data set I/O requests using the QSAM access methods.

-ispf_extended_statistics

Displays whether DSFS should retrieve and store ISPF extended statistics when processing PDS and PDSE members.

-msg_input_dsn

Displays the name of the data set that contains converted DSFS messages.

-pds_enq_duration

Specifies the number of seconds that DSFS will hold enqueues on PDS/PDSE data sets, which allows DSFS to cache their directory contents for that period of time and reduce the number of calls to read the PDS/PDSE data set directory.

-pds_interval

Displays the number of seconds between the time a DSFS task will look at cached PDS/PDSE directories to determine if they have exceeded their enqueue duration. If so, then it will release the enqueue.

-ps_dyn_duration

Displays the number of seconds that DSFS will hold a dynamic allocation on PS data sets and PDS/PSDE members, which allows DSFS to cache their contents for that period of time and reduce the number of calls to read the data set.

-ps_interval

Displays the number of seconds between the time a DSFS task will look at cached PS data sets and PDS/PSDE members to determine whether they have exceeded their dynamic allocation duration. If so, then it will release the allocation, which will cause DSFS to uncache the file's contents.

-security_pool_size

Displays the number of DSFS worker threads that are used to process SAF security calls.

-security_timeout

Displays the number of seconds that DSFS will use its cached security access query for a user's access to a data set. If the number of seconds since the last query exceeds this value, then DSFS will make another query to the SAF product to determine the user's access.

-trace_table_size

Displays the size, in bytes, of the internal trace table.

-trace_file_name

Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.

-utfs_name

The value that is specified in the configuration file for the utility file system name.

-level

Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem. Except for **-help**, all other valid options that are specified with **-level** are ignored.

-help

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

Usage notes

1. The **dsadm configquery** command displays the current value of DSFS configuration options. The value is retrieved from DSFS address space memory rather than from the IDFFSPRM file.

Privilege required

The issuer does not need special authorization.

Examples

1. The following command displays the current value of the file cache:

```
dsadm configquery -filecache_size
The value for -filecache_size is 64M.
```

2. The following command displays the current value of the io pool.

```
dsadm configquery -io_pool_size
The value for -io_pool_size is 128.
```

Related information

Commands:

dsadm config

Files:

IDFFSPRM

dsadm createparm

Purpose

Use the **dsadm createparm** command to save default creation parameters in an HLQ directory. *Creation parameters* are a string of options in BPXWDYN format that detail how a new data set is to be created; for example, `recfm` and `lrecl`).

Restriction: Creation parameters are only meaningful for an HLQ directory. An attempt to save creation parameters for any other type of object in the DSFS file system tree is not allowed.

Two types of creation parameters are allowed.

pdsmodel

This option allows the user to save default creation parameters for subdirectories for an HLQ directory. Subdirectories for an HLQ directory are PDS and PDSE data sets. The user can control whether a PDS or a PDSE data set is created with the `dsntype(library)` creation parameter option.

psmodel

This option allows the user to save default creation parameters for subfiles for an HLQ directory. Subfiles for an HLQ directory are PS data sets. The user can control the options that a new PS data set is created with.

For information about the syntax and options for the creation parameter strings, see [BPXWDYN: A text interface to dynamic allocation and dynamic output in z/OS Using REXX and z/OS UNIX System Services](#).

Format

```
dsadm createparm -path <path_name> [{-pdsmodel <sub-directory parameters> |
-psmodel <sub-directory parameters>}] [-trace <file_name>] [-level] [-help]
```

Options

-path *path_name*

The z/OS UNIX path to the HLQ directory in the utility file system tree that will store the creation parameters. The path follows typical POSIX rules in that it can be fully qualified from the file system tree root or local based on the current working directory.

-pdsmodel *sub-directory parameters*

A string enclosed in single or double quotation marks that list the default creation parameters for new subdirectories. The syntax of the options that are enclosed in quotation marks must meet the BPXWDYN syntax format. It must also be a data set type that DSFS supports. The supported types are PDS, PDSE data sets that have a record format of F, FB, FBA, V, VB, VBA, or U.

-psmodel *sub-file parameters*

A string enclosed in single or double quotation marks that list the default creation parameters for new files. The syntax of the options that are enclosed in quotation marks must meet the BPXWDYN syntax format. It must also be a data set type that DSFS supports. The supported types are PS data sets that have a record format of F, FB, FBA, FS, FBS, V, VB, or VBA.

-trace *<file_name>*

Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.

-level

Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem. Except for `-help`, all other valid options that are specified with `-level` are ignored.

-help

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

Usage notes

1. DSFS checks the syntax of the creation string before it attempts to save the string. Additionally, DSFS ensures it represents a data set that DSFS supports.
 - RECFM must have the value F, FB, FBA, FS, FBS, V, VB, VBA, or U and must exist in the string.
 - DSORG must have the value PS or PO and must exist in the string.
2. Although DSFS checks the syntax of the string before storing it, it is still possible for the user to create a data set string that might be created by DFSMS but are not usable by the QSAM access method.
 For example, a data set that has a RECFM(FB), LRECL and BLKSIZE that might specify a record length that is not evenly divisible by the block size. DFSMS will allow creation of the data set but the data set cannot be processed by DSFS. Thus, users must be careful when they specify the data set attributes.
3. DSFS will store at most one string in the utility file system for a psmode1 and at most one string for a pdsmodel creation string per HLQ directory. If a user performs a **dsadm createparm** for a psmode1/pdsmodel string and a psmode1 or pdsmodel creation string exists, it is replaced by the new string.
4. The **dsadm fileinfo** command can be used to display the current creation strings for an HLQ directory.

Privilege required

The user ID of the issuer of the command must match the HLQ (just the first qualifier) in order to be allowed to save creation parameters for the HLQ directory.

Examples

1. The following commands save the creation parameters that will by default create a VB LRECL(133) PDSE anytime a user attempts an **mkdir** operation on the associated HLQ directory. The HLQ is SUIMGHQ.

```
cd /dsfs/txt/suimghq
dsadm createparm -path . -pdsmodel "dsntype(library) dsorg(po) lrecl(133) recfm(vb)
blksize(0)"
```

2. The following commands save the creation parameters that will by default create a FB LRECL(80) PS data set if a user attempts to create a file on the associated HLQ directory. The HLQ is SUIMGHQ.

```
cd /dsfs/txt/suimghq
dsadm createparm -path . -psmodel "dsorg(ps) lrecl(80) recfm(fb) blksize(0)"
```

Related information

Commands:

dsadm fileinfo

dsadm fileinfo

Purpose

dsadm fileinfo displays information about a particular file or directory in DSFS. Because DSFS files and directories often represent data sets, it will also include some information about the data set that is represented by the file or directory. However, only information relevant to DSFS processing is shown. Data set attributes not related to DSFS processing are not shown and are available through other facilities in z/OS.

Some output fields for **fileinfo** are intended for service personnel.

Format

```
dsadm fileinfo -path <path_name>[-trace <file_name>][-level][-help]
```

Options

-path path_name

The z/OS UNIX path to the object in the utility file system tree that is being queried by the command. The path follows normal POSIX rules in that it can be fully qualified from the file system tree root or local based on the current working directory.

-trace file_name

Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.

-level

Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem. Except for **-help**, all other valid options that are specified with **-level** are ignored.

-help

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

Output

A *path directory* (/txt, /bin, /rec) is a DSFS-specific directory that is not backed by a data set that provides the indication of how file contents are to be processed by DSFS. The following examples show the output of this command for a path directory.

```
path: /dsfs/txt/.
  fid                2,1                anode                128436,768
  length             8192                format              BLOCKED
  1K blocks          8                   dir tree status    VALID
  PDS model anode    na                   PS model anode      na
  object type        DIR                  object linkcount     9
  object genvalue     0x00000000          dir version          1
  dir name count      9                   data set name type   na
  recfm              na                   lrecl                na
  data mode           TEXT                 data set status      na
  data set name       na
  ENQ held            NO
  direct blocks       0x00018170          0xFFFFFFFF          0xFFFFFFFF          0xFFFFFFFF
                        0xFFFFFFFF          0xFFFFFFFF          0xFFFFFFFF          0xFFFFFFFF
  indirect blocks     0xFFFFFFFF          0xFFFFFFFF          0xFFFFFFFF          0xFFFFFFFF
  mtime              Aug  4 12:01:54 2021  atime                Sep 28 17:51:29 2021
  ctime              Aug  4 12:01:54 2021  create date          May 10 2021
  not encrypted
  PDS model           na
  PS model            na
  vnode,vntok         0x00000050,,0x452000F0  0x01EF18E0,,0x00000000
  opens               oi=0                  rd=0                  wr=0
  file segments       na                    file unscheduled     na
  meta buffers        0                     dirty meta buffers    0
```

path

The path of the object that was queried.

fid

The z/OS UNIX file identifier. z/OS UNIX represents every object by a pair of numbers and these numbers are shown on application interfaces such as a stat call. The form is INODE,UNIQUE where:

INODE

The inode number of the object, which is an indicator of its location in the anode table in the utility file system data set.

UNIQUE

The uniquifier that represents reuse of the inode number as the number is reused when the original object represented by the inode number was deleted.

anode

The location of the anode on disk with the format BLOCK,OFFSET where:

BLOCK

The block number of the block that contains the anode. A file system is an array of blocks that are indexed by block number. The size of each block is 8 K.

OFFSET

Offset into the block where the anode is located.

length

The length of the utility file system object in bytes. DSFS tries to avoid placing file contents in the utility file system. Instead, it prefers to keep the contents cached until they can be written to the data set instead. The length depends on the type of object.

Directory

Directories are stored in the utility file system, so this length represents the size of the object in all cases, no matter what type of directory it is (HLQ, path, PDS, PDSE).

Files

Files show length only if the object was opened for processing by applications.

- A file that is not open and was not recently accessed by an application shows a length of zero regardless of the length of the actual data set the file represents.
- A file that is open or recently accessed by applications has a length that shows the length of the data set represented as a POSIX file.

format

The storage method that is used to store the object contents in the utility file system. It can have one of the following values:

INLINE

Object is small and stored inside the anode in the utility file system.

BLOCKED

Object is blocked and stored as an array of blocks in the utility file system. DSFS uses trees to locate the blocks on disk, which takes a modest amount of extra disk space. Empty objects are represented by the BLOCKED indication.

1K Blocks

The total amount of space that the object occupies on disk. It includes any internal trees that are used to locate blocks of the object. The length field might not be an indicator of the space that is used on disk. Additionally, file contents are only stored in the utility file system if the file cache does not have the required space to contain the file. If the file cache cannot contain the whole file, then it would write the least recently used regions to the utility file system.

dir tree status

This field is valid only for directories. It is one of the following:

VALID

Directory tree has no errors on disk, or object is a file.

BROKEN

The directory and its tree have a corruption, directories in this state cannot be written to, only read from.

PDS model anode

This field is valid only for HLQ directories. If the object that is being queried is not an HLQ directory, it shows the value 'na'. If the object is an HLQ directory and has saved PDS creation parameters, then this field shows a pair of numbers of the format ANODE, LENGTH where:

ANODE

Slot in the anode table that contains the anode for the PDS model object. Creation parameters are stored as special objects in the file system.

LENGTH

Length of the creation parameter string.

PS model anode

This field is valid only for HLQ directories. If the object being queried is not an HLQ directory, it shows the value 'na'. If the object is an HLQ directory and it has saved PS creation parameters, then this field will show a pair of numbers of the format ANODE, LENGTH where:

ANODE

Slot in the anode table that contains the anode for the PS model object. Creation parameters are stored as special objects in the file system.

LENGTH

Length of the creation parameter string.

object type

Type of object, either DIR or FILE.

DIR

Object is a directory.

FILE

Object is a file.

object linkcount

The link count of the object. For directories this is an indicator of the number of subdirectories. All POSIX directories have special entries '.' (dot) which represents the current directory and '..' (dot dot) which represents its parent directory and because of this an empty directory would have a link count of 2. Thus, all directories would have a minimum link count of 2. Files will always have a link count of one.

object genvalue

The **genvalue** is a 4-byte field that is made available to z/OS UNIX so they can use to store information in relative to the object. The contents of this field have no meaning to DSFS.

dir version

Software version of the directory, should have the value 1 for directories and 'na' for files.

dir name count

The number of names inside the directory. The names include the dot and dot dot entry. Files will display 'na' for this field.

data set name type

The type of data set (HLQ DIR, PDS, LIBRARY, SEQ, or MEMBER) that this DSFS object represents.

recfm

For utility file system objects that represent data sets, **recfm** is the record format of the associated data set. For objects that do not represent a data set the value 'na' is displayed.

lrecl

For utility file system objects that represent data sets, **lrecl** is the record length of the associated data set. For objects that do not represent a data set the value 'na' is displayed.

data mode

The type of conversion processing that will be performed on the object (or objects in its subtree if the object is a directory). The data mode is one of the following values:

TEXT

Object is processed as text data according to the DSFS conversion rules.

BINARY

The object is processed as binary data according to the DSFS conversion rules.

RECORD

The object is processed as record data according to the DSFS conversion rules.

data set status

An indicator if the data set is cached in DSFS with one of the following values:

DIRTY

The file was updated and needs to be written back to the data set when the file is closed.

RETRIEVED

For files, this value indicates that the associated data set contents were retrieved and converted to POSIX byte-stream format. Directories will always show the value RETRIEVED because DSFS must store the names in the directory in the utility file system.

RETRIEVING

The file is being read into the file cache. The DSFS task control block (TCB) of the task reading the file is displayed, along with the time that the task started reading it.

UNCACHED

The data set is not cached in DSFS, which means that DSFS does not have the data set opened or its contents converted and stored in the utility file system.

data set name

Name of the data set the object represents if the DSFS object represents a data set. If the object represents an HLQ directory, the qualifiers in the HLQ directory name are shown. Otherwise, the value 'na' will be shown.

ENQ held

An indicator if DSFS has the associated data set (shown in the **data set name** field) dynamically allocated or enqueued. If DSFS does not have the data set allocated or enqueues, the value NO is shown. If the data set is allocated, then the field has the format MODE,DDNAME,USER,TIME where:

MODE

Indicates whether the data set is allocated SHR or OLD.

DDNAME

The dynamic DDNAME of the allocation.

USER

The requesting user that caused DSFS to allocate the data set.

TIME

The time of day that the dynamic allocation was obtained.

direct blocks

This field lists the locations inside the utility file system of the first eight blocks of the object if the object is stored in blocked format. The value 0xFFFFFFFF is shown if there is no corresponding direct block. This field is intended for IBM service.

indirect blocks

This field lists the locations inside the utility file system of the anchors to the trees that locate additional blocks of the object inside the utility file system. The value 0xFFFFFFFF is shown if there is no corresponding tree. This field is intended for IBM service.

mtime

The POSIX modification time of the object. The time is the modification time that is stored in the utility file system object.

atime

The POSIX access time of the object that is stored in the utility file system object.

ctime

The POSIX metadata change time of the object stored in the utility file system object.

create date

The POSIX create date of the object, which is the date that the object was created. For PDS and PDSE members, it is the create date that was stored in the ISPF statistics. For PS/PDS/PDSE data sets, it is the create date as stored in the catalog. If the create date is not available, it is the date that the object was first accessed in DSFS.

encrypt/compress status

The encryption and compression status of the object that is stored inside the utility file system.

- For directories, not encrypted not compressed is shown as those objects are never encrypted or compressed.
- For files that are not encrypted and not compressed, then not encrypted not compressed is shown. If the utility file system is encrypted, then the value encrypted is shown as the encryption indicator. If the file is compressed, then it shows the value compressed xxK saved, which indicates how much disk space was saved due to compression.

PDS model

If the object is an HLQ directory, then a saved PDS model creation string is shown; else, the value 'na' is shown.

PS model

If the object is an HLQ directory, then a saved PS model creation string is shown; else the value 'na' is shown.

vnode/vntok

This field is the address of the internal data structures that represent the object in DSFS and z/OS UNIX.

opens

This field is the count of opens to the object.

oi

Count of internal opens to the file or directory.

rd

Count of read opens to the file or directory.

wr

Count of write opens to the file or directory.

file segments

The number of 64 K segments of the file cached in the file cache. It has the value 'na' for directories.

file unscheduled

Indicates the number of unscheduled (dirty) pages in the file cache for the file.

meta buffers

The number of blocks in the metadata cache for this object.

dirty meta buffers

The number of blocks in the metadata cache that are dirty for this object.

Additional examples: HLQ directories

```

path: /dsfs/txt/suimghq/.
fid          18,195          anode          137033,4800
length       8192           format          BLOCKED
1K blocks    8              dir tree status  VALID
PDS model anode 37,59       PS model anode 0,0
object type   DIR           object linkcount 11
object genvalue 0x00000000  dir version     1
dir name count 21          data set name type HLQ DIR
recfm        na            lrecl          na
data mode     TEXT         data set status  UNCACHED
data set name SUIMGHQ
ENQ held      NO
direct blocks 0x0000A7EF    0xFFFFFFFF    0xFFFFFFFF    0xFFFFFFFF
               0xFFFFFFFF    0xFFFFFFFF    0xFFFFFFFF
indirect blocks 0xFFFFFFFF  0xFFFFFFFF    0xFFFFFFFF
mtime         Feb 22 19:00:00 2021  atime         Mar 17 12:48:55 2021
ctime         Feb 22 19:00:00 2021  create time   Feb 22 19:00:00 2021
not encrypted not compressed
PDS model     dsntype(library) dsorg(po) lrecl(133) recfm(vb) blksize(0)
PS model      na
vnode,vntok   0x000000050,,0x289001E0    0x01AC85D0,,0x00000000
opens         oi=0          rd=0          wr=0
file segments na           file unscheduled na
meta buffers  0            dirty meta buffers 1

```

The fields of particular interest to HLQ directories are:

PDS model anode/PS model anode

Contains an indicator if a creation parameter exists and its length.

PDS model/PS model

Contains the BPXWDYN format-compatible string for new data set creation parameters.

data set name

Shows the high-level qualifiers in the HLQ directory name. In this case, it is just the HLQ itself.

data mode

Shows which DSFS path the HLQ directory is contained in.

Additional examples: PDS member

```

path: /dsfs/txt/suimghq/private.dumps.jcl/print
fid          54,195          anode          154080,5808
length       891           format          BLOCKED
1K blocks    8              dir tree status  VALID
PDS model anode na         PS model anode  na
object type   FILE          object linkcount 1
object genvalue 0           dir version     na
dir name count na          data set name type MEMBER
recfm        FB            lrecl          80
data mode     TEXT         data set status  RETRIEVED

```

```

data set name      SUIMGHQ.PRIVATE.DUMPS.JCL(PRINT)
ENQ held          SHR,SYS00041,SUIMGHQ,Mar 17 17:53:44 2021
direct blocks      0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
                  0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
indirect blocks    0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
mtime             Nov 15 14:32:13 2017      atime      Mar 17 17:53:44 2021
ctime             Nov 15 14:32:13 2017      create time  Nov 15 14:32:13 2017
not encrypted      not compressed
PDS model          na
PS model           na
vnode,vntok        0x00000050,,0x28902DF0      0x01619D00,,0x00000000
opens              oi=0      rd=0      wr=0
file segments      1          file unscheduled 1
meta buffers       0          dirty meta buffers 0

```

The fields of interest to a PDS member (and a PS data set and PDS/PDSE would be similar):

length

The length of the utility file system file that represents the data set contents that are converted to POSIX byte stream format.

recfm

Shows the record format. In this example, the record format is FB.

lrecl

Shows the record length. In this example, the record length is 80 bytes.

data mode

Shows the processing mode. It also indicates the path and TEXT in this example.

data set status

Shows the data set status. In this case, data set is RETRIEVED and cached by DSFS.

data set name

The full data set name is shown.

ENQ held

Displays if an allocation is made to the object, its DDNAME, when it was obtained and the user ID of the user whose access caused DSFS to retrieve the data set.

not encrypted/not compressed

For files, this field indicates whether the file contents are encrypted or compressed. In this example, neither is true. Directories are never stored encrypted or compressed.

Privilege required

If this command is run against the root, a path directory, or an HLQ directory, then no special authority is required. If the command is issued against a data set, then the issuer must have READ authority to the data set.

Related information

Commands:

dsadm createparm

dsadm fsinfo

Purpose

dsadm fsinfo displays information about the utility file system data set.

Format

```
dsadm fsinfo [-trace <file_name>] [-level] [-help]
```

Options

-trace file_name

Specifies the name of the file that will have the trace records written into it. The trace file must be a z/OS UNIX file.

-level

Prints the level of the **dsadm** command. This information is useful when you are diagnosing a problem. Except for -help, all other valid options that are specified with -level are ignored.

-help

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

Privilege required

The **dsadm fsinfo** command requires no special authorization.

Examples

The following is an example output from the command.

```
# dsadm fsinfo
File System Name:      ZFSAGGR.BIGDSFS.UTILITY.FS.DCEIMGHQ

System:                DCEIMGHQ          Devno:                75
Size:                  1440000K          Free 8K Blocks:      178139
Free 1K Fragments:    7                  Log File Size:       14400K
Bitmap Size:          208K               Anode Table Size:    64K
File System Objects:  192                Version:              1
Overflow Pages:       0                  Overflow HighWater:   0
Space Monitoring:     0,0
ENOSPC Errors:        0                  Disk IO Errors:      0
Status:               NE,NC

File System Creation Time: Jan 27 16:47:41 2021
Mount Time:             Mar 16 20:30:14 2021

Last Grow Time:        n/a
```

The meanings of the output fields are described in [“MODIFY DSFS,FSINFO” on page 63](#).

Related information

Commands:

MOUNT

Files:

IDFFSPRM

dsadm help

Purpose

dsadm help shows syntax of specified **dsadm** commands or lists functional descriptions of all **dsadm** commands.

Format

```
dsadm help [-topic <command ...>][-trace <file_name>][-level][-help]
```

Options

-help

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

-level

Prints the level of the **dsadm** command. This is useful when you are diagnosing a problem. Except for **-help**, all other valid options that are specified with **-level** are ignored.

-topic *command*

Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **fsinfo**, not **dsadm fsinfo**). Multiple topic strings can be specified. If this option is omitted, the output provides a short description of all **dsadm** commands.

-trace *file_name*

Specifies the name of the file that will have the trace records written into it. The trace file must be a z/OS UNIX file.

Usage notes

1. The **dsadm help** command displays the first line (name and short description) of the online help entry for every **dsadm** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.
2. The online help entry for each **dsadm** command consists of the following two lines:
 - The first line names the command and briefly describes its function.
 - The second line, which begins with Usage:, lists the command options in the prescribed order. Use the **dsadm apropos** command to show each help entry containing a specified string.

Privilege required

The issuer does not need special authorization.

Examples

The following command displays the online help entry for the **dsadm fsinfo** command and the **dsadm fileinfo** command:

```
dsadm help -topic fsinfo fileinfo
dsadm fsinfo: Obtain utility file system information
IDFS00090I Usage: dsadm fsinfo [-trace <file_name>] [-level] [-help]
dsadm fileinfo: Obtain information on a file or directory
IDFS00090I Usage: dsadm fileinfo -path <path name> [-trace <file_name>] [-level][-help]
```

Related information

Commands:

dsadm apropos

dsadm jobid

Purpose

If DSFS is not running under the control of JES (SUB=MSTR is being used), DSFS needs to have a JES job ID to use the subsystem interface (SSI). DSFS usually handles this job ID automatically but in certain circumstances, an administrator might need to assist with this.

Format

```
dsadm jobid {-refresh | -return } [-level][-trace][-help]
```

Options

-refresh

DSFS returns the current job ID if it has one and then obtains a new one.

-return

DSFS returns the current job ID.

-level

Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem.

Except for `-help`, all other valid options that are specified with `-level` are ignored.

-trace <file_name>

Specifies the name of the file that will have the trace records written into it. The trace file should be a z/OS UNIX file.

-help

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

Privilege required

The issuer must be logged in as a root user (UID=0) or have READ authority to the SUPERUSER.FILESYS.PFSCTL resource in the z/OS UNIXPRIV class.

dsadm query

Purpose

dsadm query displays internal DSFS statistics (counters and timers) that are maintained in the DSFS Physical File System (PFS).

Format

```
dsadm query [-locking] [-reset] [-storage] [-filecache] [-iobydasd] [-knpfs]
[-metacache] [-vnodecache] [-compression] [-dscache] [-trace <file_name>]
[-level] [-help]
```

Options

-compression

Displays the compression statistics. For more information, see [“MODIFY DSFS QUERY,LFS” on page 74.](#)

-dscache

Displays data set operation thread pools and security cache information. For more information, see [“MODIFY DSFS QUERY,DSCACHE” on page 68.](#)

-filecache

Displays the file cache report. For more information about this report, see [“MODIFY DSFS QUERY,FILECACHE” on page 69.](#)

-help

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

-iobydasd

Displays the I/O count by direct access storage device (DASD) report. For more information about this report, see [“MODIFY DSFS QUERY,IOBYDASD” on page 72.](#)

-knpfs

Displays the kernel counters report. For more information about this report, see [“MODIFY DSFS QUERY,KN” on page 73.](#)

-level

Prints the level of the **dsadm** command. This is useful when you are diagnosing a problem. Except for -help, all other valid options that are specified with -level are ignored.

-locking

Displays the locking statistics report. For more information about this report, see [“MODIFY DSFS QUERY,LOCK” on page 76.](#)

-metacache

Displays the metadata cache counters report. For more information about this report, see [“MODIFY DSFS QUERY,LFS” on page 74.](#)

-reset

Resets all the DSFS monitoring statistics.

-storage

Displays the storage report. For more information about this report, see [“MODIFY DSFS QUERY,STORAGE” on page 78.](#)

-trace file_name

Specifies the name of the file that will have the trace records written into it. The trace file must be a z/OS UNIX file.

-vnodecache

Displays the vnode cache counters report. For more information about this report, see [“MODIFY DSFS QUERY,LFS” on page 74](#).

Usage notes

- 1. Use the **dsadm query** command to display performance statistics that are maintained by the DSFS Physical File System.

Privilege required

The issuer does not need special authorization to query the statistics. To reset statistics, the issuer must be logged in as a root user (UID=0) or have READ authority to the SUPERUSER.FILESYS.PFCTL resource in the z/OS UNIXPRIV class.

Examples

The following is an example of the **dsadm query -dscache** report.

dsadm query -dscache				
DSFS Data Set Caching and Thread Pool Statistics				
	Called/ Dispatched	Miss/ Queued	Miss/ QPCT	Resync/ Retrieve
I/O Requests	99	0	0%	4
Dir Requests	79	0	0%	12
Sec Requests	0	0	0%	na
Sec Cache Rq	65	12	18%	na

Related information

Commands:

MODIFY DSFS QUERY

dsadm salvage

Purpose

dsadm salvage verifies the correctness of the DSFS utility file system and takes corrective action if required.

Format

```
dsadm salvage [{-verifyonly | -cancel}][-trace <file_name>][-level][-help]
```

Options

-verifyonly

Initiates a salvage operation that will verify the contents of the file system. If the file system is found to be corrupted, then DSFS will restart. The next time the DSFS utility file system is mounted, which could be automatically in a shared file system environment as a result of the restart, DSFS will reformat it to make it usable again.

-cancel

Cancels a salvage verification operation that is in progress.

-trace file_name

Specifies the name of the file that will have the trace records written into it. The trace file must be a z/OS UNIX file.

-level

Prints the level of the **dsadm** command. This option is useful when you are diagnosing a problem. Except for **-help**, all other valid options that are specified with **-level** are ignored.

-help

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

Usage notes

1. The **dsadm salvage -verifyonly** command can take a long time for very large utility file systems. The length of time is determined primarily by the number of objects in the file system.
2. While the **dsadm salvage** verification is running, all operations that cause a write to the file system are suspended (both directory and file updates). Read requests can progress while the operation is in-progress. Lookup operations will fail.
3. If the salvage operation detects a corruption in the file system, all accesses are disabled and all requests are failed. DSFS will restart. The next time the utility file system is mounted, it is reformatted to allow it to be safely used again for DSFS.
4. Because the salvage operation is long-running, the operation can be canceled by the **dsadm salvage -cancel** command. It suspends the operation and allows user activity to resume.

Privilege required

The issuer must be logged in as a root user (UID=0) or have READ authority to the SUPERUSER.FILESYS.PFSCCTL resource in the z/OS UNIXPRIV class.

Example

```
dsadm salvage -verifyonly
IDFS00183I Utility file system was successfully verified.
```

Related information

Commands:

dsadm fsinfo
MOUNT

Chapter 12. The DSFS configuration file (IDFPRMxx or IDFFSPRM)

The IDFFSPRM file lists the configuration options for the DSFS PFS. Because this file does not contain any mandatory information in this file, it is not required. The options all have defaults. However, if you need to specify any options (for tuning purposes, for example), you must have an IDFFSPRM file.

DSFS allows for more than one method to specify the location of the IDFFSPRM configuration file. It uses the following criteria to determine which method to use.

- If an IDZPRM DD statement exists in the JCL, the data set that it defines will become the configuration file for the local system.
- If there is no IDZPRM DD statement, the IDFPRMxx parmlib members that are specified in the PARM string of the DSFS FILESYSTYPE statement are used.
- If there is no PARM string on the DSFS FILESYSTYPE statement, parmlib member IDFPRM00 is used.
- If there is no IDFPRM00 parmlib member, no DSFS configuration data is used.

The location of the IDFFSPRM file can be specified by the IDZPRM DD statement in the DSFS PROC. However, the preferred method for specifying the DSFS configuration file is to use the IDFPRMxx parmlib member as described in [“Processing options for IDFFSPRM and IDFPRMxx”](#) on page 114. If you still want to use a single IDFFSPRM file, specify the IDZPRM DD statement in your JCL. The IDFFSPRM file is typically a PDS member, so the IDZPRM DD statement might look like the following example:

```
//IDZPRM DD DSN=SYS4.PVT.PARMLIB(IDFFSPRM),DISP=SHR
```

If you need to have separate IDFFSPRM files and you want to share the DSFS PROC in a sysplex, you can use a system variable in the DSFS PROC so that it points to different IDFFSPRM files. The IDZPRM DD might look like the following:

```
//IDZPRM DD DSN=SYS4.PVT.&SYSNAME..PARMLIB(IDFFSPRM),DISP=SHR
```

Your IDFFSPRM file might reside in SYS4.PVT.SY1.PARMLIB(IDFFSPRM) on system SY1; in SYS4.PVT.SY2.PARMLIB(IDFFSPRM) on system SY2; and others.

If you want to share a single IDFFSPRM file, you can use system symbols in data set names in the IDFFSPRM file. For example, `msg_input_dsn=USERA.&SYSNAME..DSFS.MSGIN` results in `USERA.SY1.DSFS.MSGIN` on system SY1. Each system has a single (possibly shared) IDFFSPRM file.

Any line beginning with `#` or `*` is considered a comment. The text in the IDFFSPRM file is not case-sensitive. Any option or value can be uppercase or lowercase. Blank lines are allowed. Do not have any sequence numbers in the IDFFSPRM file. If you specify an invalid text value, the default value is assigned. If you specify an invalid numeric value, and it is smaller than the minimum allowed value, the minimum value is assigned. If you specify an invalid numeric value, and it is larger than the maximum allowed value, the maximum value is assigned.

The preferred alternative to an IDZPRM DDNAME is specifying the IDFFSPRM file as a parmlib member. In this case, the member has the name IDFPRMxx, where xx is specified in the parmlib member list.

When the IDFFSPRM is specified in a DD statement, there can be only one IDFFSPRM file for each member of a sysplex. With PARMLIB, DSFS configuration options can be specified in a list of configuration parmlib files. This allows an installation to specify configuration options that are common among all members of the sysplex (for example, `io_pool_size`) in a shared IDFPRMxx member and configuration options that are system-specific in a separate, system-specific IDFPRMxx member. If a configuration option is specified more than once, the first one found is taken.

The IDFPRMxx files are contained in the logical parmlib concatenation. The logical parmlib concatenation is a set of up to ten partitioned data sets defined by parmlib statements in the LOADxx member of

either SYSn . IPLPARM or SYS1 . PARMLIB. The logical parmlib concatenation contains DSFS IDFPRMyy members that contain DSFS configuration statements. Columns 72-80 are ignored in the IDFPRMyy member. The yy values are specified in the PARM option of the FILESYSTYPE statement for the DSFS PFS in the BPXPRMxx parmlib member. The only valid value that can be specified on the PARM option for the DSFS PFS is the parmlib search parameter **PRM=**. The PARM string is case-sensitive. As the following example shows, you must enter the string in uppercase.

```
FILESYSTYPE TYPE(DSFS) ENTRYPPOINT(IDFFSCM)
ASNAME(DSFS, 'SUB=MSTR')
PARM('PRM=(01,02,03)')
```

Up to 32 member suffixes can be specified. You can also use any system symbol that resolves to two characters.

```
FILESYSTYPE TYPE(DSFS) ENTRYPPOINT(IDFFSCM)
ASNAME(DSFS, 'SUB=MSTR')
PARM('PRM=(01,&SYSCONE.)')
```

If &SYSCONE . =AB, parmlib member IDFPRMAB is searched after parmlib member IDFPRM01. IDFPRM01 can contain common configuration options and IDFPRMAB can contain configuration options that are specific to system AB. If a parmlib member is not found, the search for the configuration option will continue with the next parmlib member. To specify 32 members, type the member suffixes up to column 71. Then, continue them in column 1 on the next line, as shown in the following example:

```
FILESYSTYPE TYPE(DSFS) ENTRYPPOINT(IDFFSCM) ASNAME(DSFS, 'SUB=MSTR')
PARM('PRM=(00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,
15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31)')
^ | col 1
```

If no PRM suffix list is specified (and no IDFZPRM DD is specified in their respective JCL), then parmlib member IDFPRM00 is read. Parmlib support is only used when no IDFZPRM DD is present in the JCL.

Processing options for IDFFSPRM and IDFPRMxx

Descriptions of the valid configuration options (also referred to as *parameters*) and their allowed values follow. If an IDFFSPRM file is not found, the default value for each configuration option is used.

COMPRESS

Indicates whether DSFS should format a utility file system to use zEDC compression to compress file contents it stores in the utility file system. The expected value is ON or OFF. For example,

```
COMPRESS=ON
```

COMPRESS has meaning only for the first mount of a newly defined utility file system. The first mount of a new utility file system will format it for use and will query the COMPRESS option to determine whether the contents of files should be compressed in the utility file system. Any future mount of the utility file system continues to use the compression value that was set at the first mount time. Any change to the COMPRESS option in the IDFFSPRM file does not affect utility file systems that were already mounted. If the utility file system needs to be reformatted as part of error recovery, the current setting of the COMPRESS option is used.

The default value is OFF.

Restriction: COMPRESS can be set only from IDFPRMxx. That is, **dsadm config** -compress is not allowed.

DIRECTORY_POOL_SIZE

Indicates the size of the DSFS worker task pool that is used to handle directory-related requests that require a call to DFSMS. Examples of these requests might be the need to scan a PDS/PDSE directory or read DFSMS catalogs. The expected value is a number in the range 1-100. For example,

```
DIRECTORY_POOL_SIZE=55
```

The default value is 64.

DIRECTORY_REFRESH_TIMEOUT

Determines how many seconds DSFS will use its cached version of an HLQ directory or PDS/PDSE directory in its utility file system before it rereads the directory data from the associated catalog or PDS/PDSE that is represented by the utility file system directory. A larger value results in fewer calls to DFSMS. However, a larger value also delays the time that it takes for DSFS to notice that a change was made outside of DSFS to a PDS/PDSE data set or to a HLQ directory in the catalog. The expected value is a number in the range 30-600. For example,

```
DIRECTORY_REFRESH_TIMEOUT=89
```

The default value is 60.

ENABLE_SYSOUT

Specifies whether DSFS will allow users to access JES spool data sets. The expected value is either the value OFF or ON. For example,

```
ENABLE_SYSOUT=OFF
```

The default value is ON.

FILECACHE_SIZE

Specifies the size in bytes of the cache that is used to contain file data. You can also specify a fixed option, which indicates that the pages are permanently fixed for performance. The `fixed` and `edcfixed` options can fix the file cache in real memory.

- The `fixed` option avoids page fix and page unfix for disk I/Os that do not use compression.
- The `edcfixed` option avoids page fix and page unfix for disk I/Os that use compression. It also avoids data movement for compression I/Os. If the `edcfixed` option is used, DSFS will wait during the initialization process for zEDC to be available. While it is waiting, DSFS will display message IDFS00306I. When zEDC is ready, DSFS will continue the initialization process.

The expected value is a number in the range 10 MB - 65536 MB (64 G) if the `edcfixed` option is not used. If the `edcfixed` option is used, the file cache size should be in the range 10 MB - 14336 MB (14 G) due to zEDC compression limitations. K or M can be appended to the value to mean kilobytes or megabytes. For example,

```
FILECACHE_SIZE=64M, FIXED
```

For the default value, DSFS calculates 10% of real storage the system has available during DSFS initialization. If this amount is less than 256 M, then the default is 256 M. If this amount is between 256 M and 2048 M, then the default is 10% of real storage. If the amount is greater than 2048 M, then the default is 2048 M.

FSFULL

Specifies the threshold and increment when sending file system utilization messages to the operator. The expected value is two numbers that are separated by a comma in the range 1-99 within parentheses. For example,

```
FSFULL=(90,5)
```

The default value is OFF.

IO_POOL_SIZE

Specifies the number of DSFS worker tasks that are used to read and write PS data sets and PDS/PDSE members. The expected value is a number between 1-1000. For example,

```
IO_POOL_SIZE=306
```

The default value is 128.

ISPF_EXTENDED_STATISTICS

Specifies whether DSFS will use and update the ISPF extended statistics when processing PDS/PDSE members. The expected value is either the value OFF or ON. For example,

```
ISPF_EXTENDED_STATISTICS=ON
```

The default value is OFF.

HLQ_MODE

Specifies the HLQ mode that DSFS is to run in. The options are as follows:

EXCLUDE

The HLQ directory names listed in the HLQ_LIST IDFFSPRM option are the data set name qualifier prefixes that are to be excluded from the DSFS tree. All others are allowed.

The default is EXCLUDE.

If HLQ_MODE is specified with an option that is not allowed, DSFS will terminate.

HLQ_LIST

Specifies the high-level qualifier directory names that are not allowed to be accessed by DSFS users. High-level qualifier directory names can consist of one or more data set name qualifiers, separated by periods. These names will also be used to exclude access to data set names that begin with these qualifiers. The expected value is a list of single or multiple qualifier names separated by commas. No blanks are allowed in the list, and the names must not be larger than 42 characters. DSFS allows multiple HLQ_LIST entries in the DSFS parameters file and will concatenate the lists automatically. For example,

```
HLQ_MODE=EXCLUDE
HLQ_LIST=CATHY,CHARLIE
HLQ_LIST=STEVE,MARY,VIVIAN
HLQ_LIST=SUIMGEA.PRIVATE,SUIMGHQ.PRIVATE
```

In the example, the listed HLQs in both lines are not allowed in DSFS: STEVE, MARY, VIVIAN, CATHY, CHARLIE. Data set names that begin with these HLQs cannot be accessed by DSFS users. Data sets of HLQs SUIMGEA and SUIMGHQ can be accessed as long as their names do not begin with SUIMGEA.PRIVATE or SUIMGHQ.PRIVATE.

The default value is no list. Since the default HLQ_MODE is EXCLUDE, this means that no HLQs are disallowed by DSFS if these options are defaulted.

MSG_INPUT_DSN

Specifies the name of a data set that contains converted DSFS messages. Specify this option when you use messages that are in languages other than English. Do not specify this option for English messages. It is read when DSFS or the batch job is started or restarted. Currently, Japanese messages are supported. The expected value is the name of the data set that contains converted DSFS messages. For example,

```
MSG_INPUT_DSN=USERA.SIOEMJPN
```

There is no default value.

PDS_ENQ_DURATION

Specifies the length of time in seconds for how long DSFS will keep its enqueues held on PDS/PDSE data sets to allow it to cache directory contents. Specifying a larger value will result in fewer reads of the PDS/PDSE directory but also increase the likelihood that an ISPF user might receive an in-use data set error. A smaller value will result in fewer PDS/PDSE directory reads. It will also result in a smaller likelihood of an ISPF user being prevented from access to an in-use PDS/PDSE data set accessed by DSFS.

The expected value is a number between 1-3600. For example,

```
PDS_ENQ_DURATION=2021
```

The default value is 60.

PDS_INTERVAL

Specifies the length in time in seconds where a DSFS worker task will periodically wake up to determine whether any PDS/PDSE data sets have exceeded their enqueue time as determined by PDS_ENQ_DURATION. Any PDS or PDSE data set that is no longer being accessed by z/OS UNIX and has been enqueued longer than PDS_ENQ_DURATION is dequeued by this worker task.

The expected value is a number in the range 1 - 3600. For example,

```
PDS_INTERVAL=2032
```

The default value is 60.

PS_DYN_DURATION

Specifies the length of time in seconds that DSFS will keep its dynamic allocations held on PDS/PDSE members and PS data sets so that file contents can be cached. If you specify a larger value, fewer reads of the PDS/PDSE members and PS data sets upon subsequent application access will occur. However, ISPF users might be more likely to receive an in-use data set error. A smaller value will result in fewer data set reads and in a smaller likelihood of an ISPF user being prevented from access to an in-use PDS/PDSE member or PS data set that is being accessed by DSFS.

The expected value is a number in the range 1 - 3600. For example,

```
PDS_ENQ_DURATION=2001
```

The default value is 60.

PS_INTERVAL

Specifies the length in time in seconds where a DSFS worker task will periodically wake up to determine whether any nonopen utility file system files whose associated PDS and PDSE members and PS data

sets have exceeded their enqueue time as determined by PS_DYN_DURATION. Any nonopen PDS/PDSE members and PS data sets that were allocated longer than PS_DYN_DURATION are deallocated by this worker task.

The expected value is a number in the range 1 - 3600. For example,

```
PS_INTERVAL=162
```

The default value is 60.

SECURITY_POOL_SIZE

Specifies the number of DSFS worker tasks that are available to process security calls to System Authorization Facility (SAF). DSFS will dispatch security calls to worker task to create user security contexts and check access to data sets. The expected value is a number in the range 1 - 100. For example,

```
SECURITY_POOL_SIZE=52
```

The default value is 32.

SECURITY_TIMEOUT

Specifies the number of seconds that DSFS will cache the results of a user access query. DSFS remembers whether a user has a particular access level to a data set for security_timeout seconds before it considers the cached information to be stale. If a security call is required and the cached data has timed out, DSFS calls the SAF product to query the user's access to the data set.

Using a larger value will result in fewer SAF calls, but it will also increase the elapsed time before DSFS notices a change in a user's access to a data set.

The expected value is a number in the range 1 - 3600. For example,

```
SECURITY_TIMEOUT=325
```

The default value is 60.

TRACE_TABLE_SIZE

Specifies the size, in bytes, of the internal trace table. This is the size of the wrap-around trace table in the DSFS address space that is used for internal tracing that is always on.

The expected value is a number in the range 1 M - 65535 M. For example,

```
TRACE_TABLE_SIZE=65M
```

The default is 64 M when no value is specified or when a nonnumeric numeric value is specified. If a numeric value is specified and is less than 1 M, then the default is 1 M. If a value is specified and is greater than 65535 M, then the default is 65535 M.

UTFS_NAME

Specifies the name of the VSAM linear data set to be used as the utility file system on the local system. Specify this option if the name of the local system is not a valid qualifier in a data set name. In a shared file system environment when the configuration data set is to be shared by multiple systems, a system variable should be used in the VSAM linear data set name specified in order to have a unique name for the utility file system on each system.

For example, where &ASYSVAR is a system variable name that contains unique characters that are also valid in data set names on each system:

```
UTFS_NAME=UTILITY.FS.&ASYSVAR.
```

The default value is determined when the system processes the local mount for the utility file system. The name used is the value of the FILESYSTEM parameter combined with the local system name, as defined in [“Naming convention” on page 27](#).

Appendix A. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation
Site Counsel
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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains 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.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, 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 \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special Characters

/bin directory [6](#)
/rec directory [6](#)
/txt directory [6](#)
\ (backslash) [xi](#)
(pound sign) [xi](#)

A

accessibility
 contact IBM [121](#)
active file system [22](#)
APARs
 coexistence [18](#)
 functionality [18](#)
assistive technologies [121](#)
attributes, mapping [11](#)

B

background tasks [12](#)
BPXPRMxx parmlib member
 creating entry for DSFS [16](#)
BPXWDYN
 creating new data sets [9](#)
 restrictions to text strings [34](#)

C

cache size
 identifying storage shortage [52](#)
caching
 data set contents [10](#)
 data sets [9](#)
 security [10](#)
case sensitivity [3](#)
colony address space [3](#)
command suite, dsadm [85](#)
commands
 dsadm apropos [89](#)
 dsadm config [90](#)
 dsadm configquery [93](#)
 dsadm createparm [96](#)
 dsadm fileinfo [98](#)
 dsadm fsinfo [30](#), [105](#)
 dsadm help [106](#)
 dsadm jobid [108](#)
 dsadm query [109](#)
 dsadm salvage [111](#)
 MODIFY DSFS,FSINFO [63](#)
 MOUNT [83](#)
 SETOMVS RESET [81](#)
 z/OS system [61](#)
COMPRESS [114](#)
configuring DSFS [15](#)

contact
 z/OS [121](#)
creating
 BPXPRMxx entries [16](#)
 data sets [9](#), [33](#)
 DSFS configuration file (IDFPRMxx,IDFFSPRM) [17](#)
creation parameter
 displaying saved parameters [34](#)
 dsadm createparm [96](#)
 explanation [7](#)
 saving [33](#)
 storing information in HLQ directory [42](#)
 utility file system [28](#)

D

data format exceptions [8](#)
data set attributes [12](#)
data set information [42](#)
data sets
 caching [9](#), [10](#)
 creating [9](#)
 explanation [7](#)
 removing [9](#)
 renaming [9](#)
 supported [3](#)
debugging
 storage shortage [52](#)
delays
 explanation of [50](#)
DFZPRM DD file [17](#)
directories
 PDS and PDSE
 dynamic allocations and enqueues [9](#)
directory pool [12](#)
DIRECTORY_POOL_SIZE [114](#)
DIRECTORY_REFRESH_TIMEOUT [115](#)
dsadm apropos command [89](#)
dsadm command suite [85](#)
dsadm commands
 overview [4](#)
dsadm config command [90](#)
dsadm configquery command [93](#)
dsadm createparm command [9](#), [96](#)
dsadm fileinfo command
 description [98](#)
 using [41](#)
dsadm fsinfo command [30](#), [105](#)
dsadm help command [106](#)
dsadm jobid command [108](#)
dsadm query command [109](#)
dsadm salvage command [111](#)
DSCACHE report [47](#)
DSFS (z/OS Data Set File System)
 command options, explanation of [83](#)
 configuring [15](#)
 data sets, creating [33](#)

DSFS (z/OS Data Set File System) *(continued)*

- determining status [22](#)
- installing [15](#)
- overview [3](#)
- reason codes [83](#)
- security [3](#)
- starting [21](#)
- stopping [21](#)
- tree structure [5](#)

DSFS user ID
defining [18](#)

E

ENABLE_SYSOUT [115](#)
ENF (event notification facility) [10](#)
event notification facility (ENF) [10](#)
extended attributes [37](#)
extended TIOT processing (XTIOT) [10](#)

F

file IO pool [12](#)
file processing

- data exceptions [8](#)
- memory cache [8](#)
- overview [8](#)
- retrieving data [8](#)
- SIGDSIOER signal at close [8](#)

file system

- active [22](#)
- status [22](#)

FILECACHE report [46](#)
FILECACHE_SIZE [115](#)
files

- IDFFSPRM [113](#)
- IDFPRMxx [113](#)

FSFULL [116](#)

H

hangs

- diagnosing and resolving [51](#)
- explanation of [50](#)

high-level qualifier (HLQ) directory

- explanation [6](#)
- refreshing [8](#)
- tailoring [12](#)
- validating searches [9](#)

HLQ_LIST [116](#)
HLQ_MODE [116](#)

I

IDFFSPRM configuration file

- creating or updating [17](#)
- processing options [114](#)
- specifying in parmlib [17](#)

IDFPRMxx configuration file

- creating or updating [17](#)
- processing options [114](#)

IDFZADM module [87](#)
installing DSFS [15](#)

IO_POOL_SIZE [116](#)
ISPF extended statistics [12](#)
ISPF statistics [12](#)
ISPF_EXTENDED_STATISTICS [116](#)

J

JES spool data sets
accessing [55](#)

K

keyboard

- navigation [121](#)
- PF keys [121](#)
- shortcut keys [121](#)

KNPFS report [48](#)

L

long-running operations

- dsadm salvage [111](#)
- progress indicators [105](#)

M

managing processes [21](#)
mapping attributes [11](#)
memory cache [8](#)
memory usage

- displaying [49](#)

MODIFY DSFS PROCESS report [62](#)
MODIFY DSFS QUERY,DATASET report [66](#)
MODIFY DSFS QUERY,DSCACHE report [68](#)
MODIFY DSFS QUERY,FILECACHE report [69](#)
MODIFY DSFS QUERY,IOBYDASD report [72](#)
MODIFY DSFS QUERY,KN report [73](#)
MODIFY DSFS QUERY,LEVEL report [73](#)
MODIFY DSFS QUERY,LFS report [74](#)
MODIFY DSFS QUERY,LOCK report [76](#)
MODIFY DSFS QUERY,SETTINGS command [77](#)
MODIFY DSFS QUERY,STORAGE report [78](#)
MODIFY DSFS QUERY,THREADS report [79](#)
MODIFY DSFS,FSINFO command [63](#)
MODIFY QUERY reports [65](#)
MOUNT command [83](#)
MSG_INPUT_DSN [117](#)

N

navigation
keyboard [121](#)

P

partitioned data set [7](#)
path directory

- binary (/bin) [6](#)
- explanation [6](#)
- record (/rec) [6](#)
- text (/txt) [6](#)

PDS directories

- PDS directories (*continued*)
 - holding dynamic allocations and enqueues [9](#)
- PDS_ENQ_DURATION [10](#), [117](#)
- PDS_INTERVAL [117](#)
- PDSE directories
 - holding dynamic allocations and enqueues [9](#)
- performance considerations
 - monitoring [45](#)
 - monitoring and tuning [45](#)
 - tuning [45](#)
- performance statistics
 - resetting [48](#)
- PFS (physical file system)
 - state [22](#)
- processing files [8](#), [11](#)
- program modules [13](#)
- PS data set [7](#)
- PS_DYN_DURATION [10](#), [117](#)
- PS_INTERVAL [117](#), [118](#)
- PSDE data set [7](#)

R

- report
 - MODIFY DSFS PROCESS [62](#)
 - MODIFY DSFS QUERY,DATASET [66](#)
 - MODIFY DSFS QUERY,DSCACHE [68](#)
 - MODIFY DSFS QUERY,FILECACHE [69](#)
 - MODIFY DSFS QUERY,IOBYDASD [72](#)
 - MODIFY DSFS QUERY,KN [73](#)
 - MODIFY DSFS QUERY,LEVEL [73](#)
 - MODIFY DSFS QUERY,LFS [74](#)
 - MODIFY DSFS QUERY,LOCK [76](#)
 - MODIFY DSFS QUERY,SETTINGS [77](#)
 - MODIFY DSFS QUERY,STORAGE [78](#)
 - MODIFY DSFS QUERY,THREADS [79](#)
 - MODIFY QUERY [65](#)
- root directory
 - explanation [6](#)
- root file system
 - creating [18](#)

S

- security
 - DSDF [3](#)
 - UNIX [11](#)
- security caching [10](#)
- security pool [12](#)
- SECURITY_TIMEOUT [118](#)
- serialization [9](#)
- SETOMVS RESET command [81](#)
- shortcut keys [121](#)
- SIGDSIOER signal at close [8](#)
- storage
 - identifying shortage [52](#)
- STORAGE report [49](#)
- summary of changes [xv](#)
- system commands
 - MODIFY DSFS,FSINFO [63](#)
 - SETOMVS RESET [81](#)

T

- timestamps [43](#)
- TRACE_TABLE_SIZE [118](#)
- tuning options, changing [45](#)
- typographic conventions [xi](#)

U

- UNIX attribute mapping [11](#)
- UNIX security [11](#)
- updating the DSFS configuration file (IDFPRMxx,IDFFSPRM) [17](#)
- user interface
 - ISPF [121](#)
 - TSO/E [121](#)
- UTFS_NAME [118](#)
- utility file system
 - adding volumes [26](#)
 - allowing dynamic growth of [25](#)
 - allowing for encryption [26](#)
 - changing the data set for system in a shared file system [29](#)
 - compressing transparent data with zEDC [29](#)
 - creating [17](#)
 - creation parameters [28](#)
 - defining [24](#)
 - disabling [53](#)
 - displaying usage information [30](#)
 - encrypting [29](#)
 - managing [23](#)
 - monitoring space [29](#)
 - mount processing [28](#)
 - mounting [27](#)
 - naming convention [27](#)
 - overview [3](#)
 - ownership of the DSFS file system [28](#)
 - repairing [31](#)
 - shared file system considerations [27](#)
 - understanding [23](#)
 - verifying [31](#)

W

- worker tasks [12](#)

X

- XTIOT (extended TIOT processing) [10](#)

Z

- z/OS
 - system commands [61](#)
- z/OS Enterprise Data Compression services (zEDC) [29](#)



Product Number: 5655-ZOS

GI13-5603-70

