z/OS

Distributed File Service zSeries File System Administration

SC24-5989-00

z/OS

Distributed File Service zSeries File System Administration

SC24-5989-00

Note!
Before using this information and the product it supports, read the information in “Appendix. Notices” on page 87.

First Edition (October 2001)

This edition, SC24-5989-00, applies to Version 1 Release 2 of z/OS Distributed File Service (program number 5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

International Business Machines Corporation
Information Development, Dept. C9MG
1701 North Street
Endicott, NY 13760-5553
United States of America

If you prefer to send comments electronically, use one of the following methods:
- IBMLink™ (United States customers only): GDLVME(PUBRCF)
- Internet e-mail: pubrcf@vnet.ibm.com
- World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/

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

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

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

Contents

Figures

About this book

The purpose of this book is to provide complete and detailed guidance and reference information. This information is used by system administrators that work with the zSeries File System (zFS) component of the IBM z/OS Distributed File Service product.

How this book is organized

This book is divided into parts, each part divided into chapters:

- "Part 1. zFS administration guide" on page 1 discusses guidance information for the zSeries File System (zFS).
- "Part 2. zFS administration reference" on page 35 discusses the zSeries File System (zFS) reference information which includes z/OS system commands, zFS commands, and zFS data sets.

Conventions used in this book

This book uses the following typographic conventions

Bold — **Bold** words or characters represent system elements that you must enter into the system literally, such as commands.

Italic — Italicized words or characters represent values for variables that you must supply.

`Example Font` — Examples and information displayed by the system are printed using an example font that is a `constant width typeface`.

[ ] — Optional items found in format and syntax descriptions are enclosed in brackets.

{ } — A list from which you choose an item found in format and syntax descriptions are enclosed by braces.

| — A vertical bar separates items in a list of choices.

< > — Angle brackets enclose the name of a key on a keyboard.

... — Horizontal ellipsis points indicated that you can repeat the preceding item one or more times.

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

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

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

This book used the following keying convention:

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

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

Where to find more information

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

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

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

Softcopy publications

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

Internet sources

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

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

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

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages. You can also use LookAt to look up explanations of system abends. The IBM LookAt development team is investigating other forms of reference information, such as commands.

Using LookAt to find information is faster than a conventional search because LookAt goes directly to the explanation.

LookAt can be accessed from the Internet or from a TSO command line.

You can use LookAt on the Internet at:

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

To use LookAt as a TSO command, LookAt must be installed on your host system. You can obtain the LookAt code for TSO from the LookAt Website by clicking on the **News and Help** link or from the *z/OS Collection*, SK3T-4269.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id* as in the following:

```
lookat iec192i
```

This results in direct access to the message explanation for message IEC192I.

To find a message from the LookAt Web site, simply enter the message ID and select the release you are working with.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS: MVS Routing and Descriptor Codes*, SA22-7624. For such messages, LookAt prompts you to choose which book to open.

Accessing licensed books on the web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link site at:

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

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link user ID, password, and z/OS licensed book key code. The z/OS order that you received provides a memo that includes your key code.

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

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

To register for access to the z/OS licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.
3. Select **Access Profile**.
4. Select **Request Access to Licensed books**.
5. Supply your key code where requested and select the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed.

After your request is processed you will receive an e-mail confirmation.

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

To access the licensed books:

1. Logon to Resource Link using your Resource Link user ID and password.
2. When prompted, enter the key code.
3. Select **Library**.
4. Select **zSeries**
5. Select **Software**.
6. Click on **z/OS**.
7. Access the licensed book by selecting the appropriate element.

Part 1. zFS administration guide

This part of the book discusses guidance information for the zSeries File System (zFS).

Chapter 1. zSeries File System (zFS) overview

The z/OS Distributed File Service zSeries File System (zFS) is a z/OS UNIX file system that can be used in addition to the Hierarchical File System (HFS). zFS file systems contain files and directories that can be accessed with the z/OS hierarchical file system file Application Programming Interfaces (APIs). zFS file systems can be mounted into the z/OS UNIX hierarchy along with other local (or remote) file system types (for example, HFS, TFS, AUTOMNT, NFS, etc.).

zFS does not replace HFS, rather zFS is complimentary to HFS. HFS is still required for z/OS installation and the root file system must be HFS.

Features

zFS provides many features and benefits:

Performance zFS provides significant performance gains in many customer environments accessing files approaching 8K in size that are frequently accessed and updated. The access performance of smaller files is equivalent to HFS.

Restart zFS provides a reduced exposure to loss of updates. zFS writes data blocks asynchronously and does not wait for a sync interval. zFS is a logging file system. It logs metadata updates. If a system failure occurs, zFS replays the log when it comes back up to ensure that the file system is consistent.

Space Sharing
As an optional function, zFS allows the administrator to define multiple zFS file systems in a single data set. This allows space that becomes available from erasing files in one file system to be available to other file systems in the same data set.

Note: This function is currently only available in non-sysplex sharing environments.

Cloning As an optional function, zFS allows the administrator to make a read-only clone of a file system in the same data set. This clone file system can be made available to users to provide a read-only point-in-time copy of a file system. The clone operation happens relatively quickly and does not take up too much additional space because only the metadata[1] is copied.

Note: This function is currently only available in non-sysplex sharing environments.

Terminology

In order to discuss the details of zFS administration, a new concept and some new terminology is introduced. The new concept is multiple file systems in a single data set. This is in contrast to HFS which always has a single file system per data set.

The data set that contains zFS file systems is called a ***zFS aggregate***. A zFS aggregate can contain one[2] or more zFS file systems. A zFS aggregate is a Virtual Storage Access Method Linear Data Set (VSAM LDS). Once the zFS aggregate is defined and formatted, one or more zFS file systems can be created in the aggregate. A zFS aggregate that contains only a single zFS file system can be defined and is called a compatibility mode aggregate. Compatibility mode aggregates are more like HFS. It is recommended that as you begin to use zFS, you use compatibility mode aggregates. Later, if appropriate, you can use aggregates that contain multiple file systems. These aggregates are called multi-file system aggregates.

1. Metadata consists of things like owner, permissions and data block pointers.

2. Actually, a zFS aggregate can contain zero or more zFS file systems.

The term ***zFS file system*** refers to a hierarchical organization of files and directories that has a root directory and can be mounted into the z/OS UNIX hierarchy. zFS file systems reside on DASD. The term ***zFS Physical File System (PFS)*** refers to the code that runs in the zFS address space. The zFS PFS can handle many users accessing many zFS file systems at the same time.

Chapter 2. Post installation processing

zFS is part of the Distributed File Service base element of z/OS. Before using the zFS support, you must install the z/OS release, the Distributed File Service, and the other base elements of z/OS using the appropriate release documentation.

Note: If you are only using the zFS support of the Distributed File Service (and not the DCE DFS support nor the SMB server support of the Distributed File Service), DCE DFS and SMB do not need to be configured and DCE does not need to be configured. For more information on DCE DFS, refer to the *z/OS: Distributed File Service DFS Administration* book. For more information on SMB, refer to the *z/OS: Distributed File Service SMB Administration* book.

To use the zFS support, you must configure the support on the system. Configuration includes the following administrative tasks:

- Define the zFS physical file system to z/OS UNIX
- Create or update the zFS parameter data set (IOEFSPRM)
- Define zFS aggregates and file systems
- Create mount points and mount zFS file systems
- Change owner/group and set permissions on file system root
- Add **/usr/sbin/mount** commands to **/etc/rc** for all added file systems so that they are mounted when the system is re-IPLed.

zFS installation and configuration steps

To install, configure, and access zFS, you must perform the following administrative steps:

1. Install and perform post-installation of the Distributed File Service by following the applicable instructions in the *z/OS: Program Directory*, GI10-0669 or the *ServerPac: Installing Your Order*.
   a. Ensure that the target and distribution libraries for the Distributed File Service are available.
   b. Run the prefix.SIOESAMP(IOEISMKD) job from UID 0 to create the symbolic links used by the Distributed File Service. This job reads the member prefix.SIOESAMP(IOEMKDIR) to delete and create the symbolic links.
   c. Ensure that the DDDEFS for the Distributed File Service are defined by running the prefix.SIOESAMP(IOEISDDD) job.
   d. Install the Load Library for the Distributed File Service. The Load Library (hlq.SIOELMOD) must be APF authorized and must be in link list.
   e. Install the samples (hlq.SIOEASAMP).
   f. Install the sample PROC for ZFS (hlq.SIOEPROC).
   g. Create a JCL PROC for the ZFS Started Task in SYS1.PROCLIB by copying the sample PROC from the previous step.

      The DDNAME IOEZPRM identifies the optional IOEFSPRM data set. Although this DD statement is optional, it is recommended that it be included to identify the parameter data set to be used for ZFS. For now, it is suggested that this DD refer to a PDS with a member called IOEFSPRM that has a single line that begins with an asterisk (*) in column 1. Subsequent modifications can be made to the IOEFSPRM member, refer to “IOEFSPRM” on page 80.
   h. Add the following RACF commands:

```
ADDGROUP DFSGRP SUPGROUP(SYS1) OMVS(GID(2))
ADDUSER DFS OMVS(HOME(/opt/dfslocal/home/dfscntl) UID(0)) DFLTGRP(DFSGRP) AUTHORITY(CREATE) UACC(NONE)
RDEFINE STARTED DFS.** STDATA(USER(DFS))
```

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

Note: The DFS user ID must have at least ALTER authority to all VSAM LDSes that contain zFS aggregates. A user ID other than DFS can be used to run the ZFS started task if it is defined with the same RACF characteristics as shown for the DFS user ID.

2. Create a BPXPRMxx entry for ZFS.

   Add the following FILESYSTYPE statement to your BPXPRMxx:

   ```
   FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS)
   ```

   You should update your IEASYSxx parmlib member to contain the OMVS=(xx,yy) parameter for future IPLs.

3. Run the **dfs_cpfiles** program.

   Running this program as described in the program directory is recommended even if you plan to only use the zFS support. The only zFS configuration file is the **IOEFSPRM** data set and it is not created by the **dfs_cpfiles** program. But, to complete the installation of the Distributed File Service, the **dfs_cpfiles** program should be run to create other files needed by SMB or DCE DFS support. This avoids problems if the other support (SMB or DCE DFS) supplied by the Distributed File Service is subsequently activated.

   To run the **dfs_cpfiles** program:

   - Logon as root (UID 0) on the local machine.
   - From the z/OS UNIX shell, enter **/usr/lpp/dfs/global/scripts/dfs_cpfiles**.

4. Create or update the zFS parameter data set (IOEFSPRM).

   The zFS parameter data set is optional. The IOEZPRM DD can be omitted from the ZFS PROC or the **IOEFSPRM** data set can exist, with no parameters contained in it. Parameters are only required if you want to override the defaults for the zFS parameters. As mentioned previously, it is recommended that you create an empty **IOEFSPRM** member in a PDS. The **IOEFSPRM** member should have a single line in it that is a comment (an asterisk(*) in column 1). Update the IOEZPRM DD statement in the ZFS PROC to contain the name of the IOEFSPRM member. For example:

   ```
   IOEZPRM   DD   DSN=SYS4.PVT.PARMLIB(IOEFSPRM),DISP=OLD
   ```

   If you are running a sysplex, you may want to have different IOEFSPRM data sets for different systems. Refer to “Chapter 5. Sysplex considerations” on page 13 for reasons why you may need to use different **IOEFSPRM** data sets. In this case, you may want to specify a system qualifier in the data set name in the IOEZPRM DD. For example:

   ```
   IOEZPRM   DD   DSN=SYS4.&SYSNAME..PARMLIB(IOEFSPRM),DISP=OLD
   ```

   The PDS (organization PO) should 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 **IOEFSPRM** is provided in hlq.SIOESAMP(IOEFSPRM). Refer to “Chapter 12. zFS data sets” on page 79 for a full description of the options that can be specified in **IOEFSPRM**.

5. Create a zFS (compatibility mode) file system.

   A zFS file system resides in a zFS aggregate. A zFS aggregate is a VSAM Linear Data Set. Refer to “Chapter 4. Creating and managing zFS file systems using compatibility mode aggregates” on page 11 for details on creating zFS file systems.

6. Create a directory and mount the zFS file system on it.

   A directory can be created with the OMVS **mkdir** command. (You can also use an existing directory.) The TSO/E **MOUNT** command or the OMVS **/usr/sbin/mount** command can be used to mount the zFS file system on the directory. Refer to “Chapter 4. Creating and managing zFS file systems using compatibility mode aggregates” on page 11 for details on mounting zFS file systems.

Note: Steps 5 on page 6 and 6 on page 6 can be repeated as many times as necessary for each permanently mounted zFS file system. Only step 5 on page 6 is needed for zFS automounted file systems (assuming that the automount file system has been set up.)

7. Add entries so that the zFS file systems are mounted automatically on the next IPL.

   MOUNT commands for zFS file systems cannot be placed in BPXPRMxx members. This is a restriction of zFS. It is recommended that you place **/usr/sbin/mount** commands in **/etc/rc** to cause zFS file systems to be mounted during IPL. For example:

   ```
   /usr/sbin/mount -f OMVS.PRV.COMPAT.AGGR001 -t ZFS /etc/mountpt
   ```

Chapter 3. Managing zFS processes

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

zFS runs as a z/OS UNIX colony address space. There must be an entry in a BPXPRMxx parmlib member for ZFS and the ZFS PROC must be available. zFS is started by z/OS UNIX based on the FILESYSTYPE statement for ZFS in the BPXPRMxx parmlib member.

zFS can be started at IPL if the BPXPRMxx parmlib member is in the IEASYSxx parmlib member's OMVS=(xx,yy) list. It can also be started later by using the **setomvs reset=(xx)** operator command.

zFS can be stopped using the **p zfs** operator command. When zFS is stopped, you receive the following message:

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

To restart zFS, reply **r** to message nn. (For example, **r 1,r**). If you want zFS to remain stopped, you can reply **i** and this removes the prompt. In this case, zFS may be restarted at a later time using the **setomvs reset=(xx)** operator command.

Note: Stopping zFS may have shared HFS (sysplex) implications. Refer to "Chapter 5. Sysplex considerations" on page 13 for information on shared HFS and zFS.

Chapter 4. Creating and managing zFS file systems using compatibility mode aggregates

This chapter discusses creating compatibility mode aggregates and file systems. Refer to “Chapter 8. Multi-file system aggregates” on page 23 for information on multi-file system aggregates.

Creating a compatibility mode aggregate

A zFS file system is created in a zFS aggregate (which is a VSAM Linear Data Set). When using compatibility mode aggregates, the aggregate and the file system are created at the same time. For simplicity, we refer to a file system in a compatibility mode aggregate as a compatibility mode file system. A compatibility mode file system is created using the **IOEAGFMT** utility. This is a two step process:

1. Create a VSAM Linear Data Set using **IDCAMS**.
2. Format the VSAM LDS as a compatibility mode aggregate and create a file system in the aggregate using **IOEAGFMT**.

The VSAM LDS, the aggregate, and the file system all have the same name and that name is equal to the VSAM LDS cluster name. The zFS file system is then mounted into the z/OS UNIX hierarchy.

Figure 1 shows an example of a job that creates a compatibility mode file system.

```
//USERIDA  JOB ,'Compatibility Mode',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//AMSDUMP  DD     SYSOUT=H
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=PRV000
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(OMVS.PRV.COMPAT.AGGR001) -
            VOLUMES(PRV000) -
            LINEAR CYL(25 0) SHAREOPTIONS(2))
/*
//CREATE   EXEC   PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.PRV.COMPAT.AGGR001 -compat')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 1. Job to create a compatibility mode file system

Note, the **-compat** parameter in the CREATE step. That is what tells **IOEAGFMT** to create a compatibility mode file system. The result of this job is a VSAM LDS that is formatted as a zFS aggregate and contains one zFS file system. The zFS file system has the same name as the zFS aggregate (and the VSAM LDS). The size of the zFS file system (that is, its quota) is based on the size of the aggregate.

There are several other options that can be used when creating a compatibility mode file system that set the owner, group, and the permissions of the root directory. The **-owner** option specifies the owner of the root directory. The **-group** option specifies the group of the root directory. The **-perms** option specifies the permissions on the root directory. Refer to “Chapter 11. zFS commands” on page 43 for more information on **IOEAGFMT**.

The zFS file system can now be mounted into the z/OS UNIX hierarchy. This is accomplished with the TSO/E **MOUNT** command. Here is an example of mounting the compatibility mode file system that was just created:

```
MOUNT FILESYSTEM('OMVS.PRV.COMPAT.AGGR001') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/mountpt1')
```

This assumes that the directory `/usr/mountpt1` exists and is available to become a `mountpoint`. Note that the `TYPE` parameter of the **MOUNT** command specifies ZFS. This is required for any zFS file system. All other forms of the mount function are also supported (for example, the **/usr/sbin/mount** command, the automount facility, etc.). Once the zFS file system is mounted, applications and commands can be executed and files and directories can be accessed in zFS just as in HFS.

Growing a compatibility mode aggregate

If a compatibility mode aggregate becomes full, the administrator can grow the aggregate (that is, cause secondary allocations to occur and format them to be part of the aggregate). This is accomplished with the **zfsadm grow** command. The aggregate's VSAM Linear Data Set must have secondary allocation specified and space on the volume(s) must be available. The size specified on the **zfsadm grow** command must be larger than the current size of the aggregate.

For example, suppose a 2 cylinder (primary allocation, 3390) aggregate has a total of 179 8K blocks and a (potential) secondary allocation of 1 cylinder. 179 8K blocks is 1432K bytes. A **zfsadm aggrinfo** command for this aggregate might show 1296K with 136K reserved. This is a total of 1432K. A **zfsadm grow** command would need to specify a size greater than 1432 to actually grow the aggregate. **zfsadm grow** does this by calling DFSMS to allocate the additional DASD space. You may need to specify a few blocks larger than the current size before a secondary allocation occurs because DFSMS may require some number of reserved blocks. For example, you may need to specify a size of 1448 before the secondary allocation actually occurs. See the following example:

```
zfsadm aggrinfo omvs.prv.aggr003.lds0003

OMVS.PRV.AGGR003.LDS0003 (R/W COMP): 1150 K free out of total 1296 (136 reserved)
zfsadm grow omvs.prv.aggr003.lds0003 1447

IOEZ00175E Error 8 received growing aggregate OMVS.PRV.AGGR003.LDS0003
OMVS.PRV.AGGR003.LDS0003 (R/W COMP): 1150 K free out of total 1296 (136 reserved)
zfsadm grow omvs.prv.aggr003.lds0003 1448

IOEZ00173I Aggregate OMVS.PRV.AGGR003.LDS0003 successfully grown
OMVS.PRV.AGGR003.LDS0003 (R/W COMP): 1798 K free out of total 1944 (208 reserved)
```

The aggregate now has a total size of 2152K bytes (1944 + 208). The file system quota has also been increased based on the new aggregate size. Aggregates cannot be made smaller.

Note: There is no automatic grow mechanism in zFS.

Chapter 5. Sysplex considerations

zFS supports the ***shared HFS*** sysplex environment. That is, users in a sysplex can access zFS data that is ***owned*** by another system in the sysplex. zFS file systems are automoved on system failure. zFS file systems can be automounted. For full support, however, zFS must be running on all systems in the sysplex and all zFS file systems must be compatibility mode file systems (that is, they cannot be file systems in multi-file system aggregates).

The following considerations apply when using zFS in a sysplex in shared HFS mode:

- Only systems running zFS see zFS file systems. The file system hierarchy appears different when viewed from systems with zFS mounted file systems than it does from those systems not running zFS. Pathname traversal through zFS mountpoints have different results in such cases since the zFS file system is not mounted on those systems not running zFS.
- zFS file systems owned by another system is accessible from a member of the sysplex that is running zFS.
- zFS compatibility mode file systems can be automoved and automounted. A zFS compatibility mode file system can only be automoved to a system where zFS is running.
- File systems in multi-file system aggregates are not fully supported in a shared HFS environment. We recommend that you only use compatibility mode aggregates in a sysplex. Refer to “Multi-file system aggregates and shared HFS” on page 14 for more information about multi-file system aggregates and shared HFS.
- The **IOEFSPRM** file cannot be shared across systems in a sysplex when the file contains:
  - A multi-file system aggregate specification, or
  - A **trace_dsn** specification.

  In this case you should use the **&SYSNAME** system variable in the **IOEZPRM DD** of the ZFS PROC to specify different **IOEFSPRM** files for different systems.

  If you are only using compatibility mode aggregates (and file systems), and you are not specifying a **trace_dsn**, and you use the same options for all ZFS PFSs on all systems, you can share the same **IOEFSPRM** file across systems.

The following describes z/OS UNIX considerations that relate to the level of z/OS or OS/390 running on the members of the sysplex:

- When all members of the sysplex are at z/OS Version 1 Release 2 and some or all systems are running zFS:
  - All systems running zFS see zFS file systems. The file system hierarchy appears differently when viewed from systems with zFS mounted file systems than it does from those systems not running zFS. Pathname traversal through zFS mountpoints have different results in such cases since the zFS file system is not mounted on those systems not running zFS.
  - If a system running zFS is brought down,
    - zFS compatibility mode file systems owned by the system that can be automoved are automoved to another system running zFS. If this function fails to find another owner, the file system becomes unowned.
    - zFS file systems that are noautomove, become unowned.
    - File systems which are unowned are not visible in the file system hierarchy, but can be seen from a **D OMVS,F** operator command. To recover a file system that is mounted and unowned, the file system must be unmounted.
  - If zFS is brought down on one system in the sysplex,

- zFS compatibility mode file systems owned by the system that can be automoved are automoved to another system running zFS. If this function fails to find another owner, the file system and all file systems mounted under it are unmounted in the sysplex.
- zFS file systems that are noautomove and all file systems mounted under them are unmounted in the sysplex.

- When all members of the sysplex are **not** at z/OS Version 1 Release 2 and some or all systems are running zFS:
  - Only systems running zFS see zFS file systems. The file system hierarchy appears differently when viewed from systems with zFS mounted file systems than it does from those systems not running zFS. Pathname traversal through zFS mountpoints have different results in such cases since the zFS file system is not mounted on those systems not running zFS.
  - If a system running zFS is brought down:
    - zFS compatibility mode file systems owned by the system that can be automoved are automoved to another system running zFS. If this function fails to find another owner, the file system becomes unowned.
    - zFS file systems owned by the system that are noautomove become unowned.
    - File systems which are unowned are not visible in the file system hierarchy, but can be seen from a **D OMVS,F** operator command. To recover a file system that is mounted and unowned, the file system must be unmounted.
  - If zFS is brought down on a system:
    - All zFS file systems owned by any of the systems unmount even if this zFS does not own any zFS file system.

      Note: This means that terminating zFS on any one system causes all zFS file systems in the sysplex to be unmounted. Because of this, it is not recommended at this time to allow zFS file systems to be shared between systems in a shared HFS environment when all systems are not at a z/OS Version 1 Release 2 level of support or higher. In a sysplex environment where all members are not at the z/OS Version 1 Release 2 level of support, the following configuration choices are recommended:
      - Restrict zFS usage to one member of the shared HFS sysplex group, or,
      - Restrict zFS usage to images not participating in a shared HFS sysplex group.

      Alternatively, you should restrict zFS usage to a multi-system shared HFS sysplex where all systems are at the z/OS Version 1 Release 2 level of support or higher.

Multi-file system aggregates and shared HFS

Multi-file system aggregates have several restrictions and limitations in a shared HFS environment. We recommend that you only use compatibility mode aggregates in a shared HFS environment.

If you want to access data that resides on a file system in a multi-file system aggregate on a sysplex, here are the known restrictions and limitations.

- A file system in a multi-file system aggregate must be mounted **NOAUTOMOVE**. This is because z/OS UNIX is not aware of the relationship between zFS file systems in the same aggregate. The file systems cannot be moved by z/OS UNIX on a system failure. One of the file systems cannot be moved by itself. If you were to attempt to move a zFS file system (in a multi-file system aggregate) to another system (by using, for example the **chmount** command), the move would fail. This is because the (multi-file system) aggregate would need to be ″moved″ (attached) to the other system and all file systems in the aggregate would need to be moved to the other system together. This is not supported with an automatic mechanism. If you want to move ownership of a file system in a multi-file system aggregate, you must manually unmount each file system in the aggregate, detach the aggregate, attach the aggregate on the other system, and then mount all the file systems on the other system. This is a disruptive procedure to applications accessing data on those file systems.

- A zFS file system in a multi-file system aggregate should not be automounted since an automounted file system is always mounted automove.
- If you stop zfs, all the file systems in the multi-file system aggregates attached to that system will be unmounted because they cannot be moved.

Chapter 6. Backing up zFS

This chapter describes how to back up a zFS aggregate using a DFSMSdss logical dump. To do this procedure, the aggregate should be attached. File systems in the aggregate may or may not be mounted. The following job consists of three steps:

1. Quiesces the aggregate (this drains any activity and suspends any new requests)
2. Backs up the aggregate (and all the file systems)
3. Unquiesces the aggregate (allowing zFS activity to continue).

The size of the target sequential data set should be at least as big as the aggregate being backed up. Figure 2 shows an example of a job backing up a zFS aggregate.

```
//ZFSBKUP1 JOB (OS390),'PROGRAMMER',CLASS=A,
//             MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//* THIS JOB QUIESCES A ZFS AGGREGATE, DUMPS IT, THEN UNQUIESCES IT.
//*-------------------------------------------------------------------
//*-------------------------------------------------------------------
//* THIS STEP QUIESCES THE AGGREGATE.
//*-------------------------------------------------------------------
//QUIESCE  EXEC   PGM=IOEZADM,REGION=0M,
// PARM=('quiesce -aggregate OMVS.PRV.AGGR004.LDS0004')
//STEPLIB   DD    DISP=SHR,DSN=G1ZFS12.Z1222.LOAD
//*
//SYSPRINT  DD    SYSOUT=H
//STDOUT    DD    SYSOUT=H
//STDERR    DD    SYSOUT=H
//SYSUDUMP  DD    SYSOUT=H
//CEEDUMP   DD    SYSOUT=H
//*
//*-------------------------------------------------------------------
//* THIS STEP DUMPS THE AGGREGATE.
//*-------------------------------------------------------------------
//DUMP    EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//OUT      DD  DSN=SUIMGUR.HIGHRISK.ZFS.DUMP1,
//             DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(5,1),RLSE)
//SYSIN    DD  *
 DUMP DATASET(INCLUDE(OMVS.PRV.AGGR004.LDS0004)) -
 OUTDD(OUT) TOL(ENQF)
/*
/*
//*-------------------------------------------------------------------
//* THIS STEP UNQUIESCES THE AGGREGATE.
//*-------------------------------------------------------------------
//UNQUIES  EXEC   PGM=IOEZADM,REGION=0M,
// PARM=('unquiesce -aggregate OMVS.PRV.AGGR004.LDS0004')
//STEPLIB   DD    DISP=SHR,DSN=G1ZFS12.Z1222.LOAD
//*
//SYSPRINT  DD    SYSOUT=H
//STDOUT    DD    SYSOUT=H
//STDERR    DD    SYSOUT=H
//SYSUDUMP  DD    SYSOUT=H
//CEEDUMP   DD    SYSOUT=H
//*
//
```

Figure 2. Job to back up a zFS aggregate

The zFS aggregate can be restored using DFSMSdss logical restore. It is restored into a new aggregate (in this case, OMVS.PRV.AGGR005.LDS0005) if the original aggregate (in this case, OMVS.PRV.AGGR004.LDS0004) still exists. Figure 3 is an example of a restore job.

```
//ZFSREST1 JOB (OS390),'PROGRAMMER',CLASS=A,
//             MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//* THIS JOB RESTORES A ZFS AGGREGATE.
//*-------------------------------------------------------------------
//*-------------------------------------------------------------------
//* THIS STEP RESTORES THE AGGREGATE.
//*-------------------------------------------------------------------
//ZFSREST EXEC PGM=ADRDSSU,REGION=0M
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//INDS     DD  DISP=SHR,DSN=SUIMGUR.HIGHRISK.ZFS.DUMP1
//SYSIN    DD  *
  RESTORE DATASET(INCLUDE(**)) -
          CATALOG -
          RENAMEU( -
           (OMVS.PRV.AGGR004.LDS0004, -
              OMVS.PRV.AGGR005.LDS0005) -
                 ) -
          WRITECHECK -
          INDD(INDS)
```

Figure 3. Job to restore a zFS aggregate

After the aggregate is restored, you need to do the following steps for a compatibility mode aggregate:

1. Unmount the original aggregate (in this case, OMVS.PRV.AGGR004.LDS0004) if it still exists (this also detaches it).
2. Attach the restored aggregate (in this case, OMVS.PRV.AGGR005.LDS0005).
3. Rename the read-write file system in the restored aggregate to the same name as the aggregate name (in this case, OMVS.PRV.AGGR005.LDS0005).
4. Detach the restored aggregate.
5. Mount the file system in the restored aggregate.

After the aggregate is restored, you need to do the following steps for a multi-file system aggregate:

1. Detach the original aggregate (in this case, OMVS.PRV.AGGR004.LDS0004) if it still exists.
2. Attach the restored aggregate (in this case, OMVS.PRV.AGGR005.LDS0005).
3. Mount the file systems in the restored aggregate.

Another example of a logical restore of a zFS aggregate using DFSMSdss by replacing the existing aggregate is shown. The backup is restored into the original aggregate (in this case, OMVS.PRV.AGGR004.LDS0004). The aggregate cannot be mounted (or attached) during the restore operation. Figure 4 on page 19 is an example of a restore replace job.

```
//ZFSREST2 JOB (OS390),'PROGRAMMER',CLASS=A,
// MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------
//* THIS JOB RESTORES A ZFS AGGREGATE.
//*-------------------------------------------------
//*-------------------------------------------------
//* THIS STEP RESTORES THE AGGREGATE.
//*-------------------------------------------------
//ZFSREST EXEC PGM=ADRDSSU,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//INDS DD DISP=SHR,DSN=SUIMGUR.HIGHRISK.ZFS.DUMP1
//SYSIN DD *
  RESTORE DATASET(INCLUDE(OMVS.PRV.AGGR004.LDS0004)) -
          CATALOG -
          REPLACE  -
          WRITECHECK -
          INDD(INDS)
```

Figure 4. Job to restore a zFS aggregate with replace

Chapter 7. Migrating data from HFS to zFS

This chapter discusses how to migrate data from HFS to zFS.

Using the OMVS pax command

You can copy data from an HFS file system to a zFS file system by using the OMVS **pax** command with or without using an intermediate archive file. Refer to the *z/OS: UNIX System Services Command Reference*, SA22-7802, for more information on the **pax** command. When the data is being copied, the file system(s) being accessed must be mounted.

Using an intermediate archive file

Use the **pax** command to copy the source (HFS) file system into an intermediate archive file and then use the **pax** command to copy from the archive file into the target (zFS) file system. This archive file can be a z/OS UNIX file or it can be an MVS data set.

Suppose you have an HFS file system mounted at **/etc/dfs**. You want to copy this into an empty zFS file system mounted at **/etc/dce/testzfs1**. You issue the following commands from OMVS:

1. Position to the source (HFS) file system mounted at **/etc/dfs**

   ```
   cd /etc/dfs
   ```

2. Create a z/OS UNIX archive file called **/tmp/zfs1.pax** that contains the HFS file system mounted at **/etc/dfs**

   ```
   pax -wvf /tmp/zfs1.pax
   ```

3. Position to the target (zFS) file system mounted at **/etc/dce/testzfs1**

   ```
   cd /etc/dce/testzfs1
   ```

4. Read the archive file into the zFS file system mounted at **/etc/dce/testzfs1**

   ```
   pax -rvf /tmp/zfs1.pax
   ```

Without using an intermediate archive file

Use the **pax** command to copy the source (HFS) file system to the target (zFS) file system, without an intermediate archive file.

Suppose you have an HFS file system mounted at **/etc/dfs**. You want to copy this into an empty zFS file system mounted at **/etc/dce/testzfs1**. You issue the following commands from OMVS:

1. Position to the source (HFS) file system mounted at **/etc/dfs**

   ```
   cd /etc/dfs
   ```

2. Copy the (HFS) file system mounted at /etc/dfs to the (zFS) file system mounted at **/etc/dce/testzfs1**

   ```
   pax -rwv . /etc/dce/testzfs1
   ```

Chapter 8. Multi-file system aggregates

This chapter discusses multi-file system aggregates, however, it is recommended that you use compatibility mode aggregates first, refer to “Chapter 1. zSeries File System (zFS) overview” on page 3. They are more like HFS file systems. Compatibility mode aggregates have a single file system in the aggregate and are fully supported in a sysplex (shared HFS) environment. Refer to “Chapter 4. Creating and managing zFS file systems using compatibility mode aggregates” on page 11 for information on zFS compatibility mode aggregates and file systems.

If you are ready to use multi-file system aggregates, note that multi-file system aggregates are not supported in a sysplex shared HFS environment. Refer to “Chapter 5. Sysplex considerations” on page 13 for information on shared HFS and zFS file systems.

Multi-file system aggregates allow the administrator to create multiple file systems in a single aggregate. This allows space sharing between different file systems in the same aggregate. Therefore, if files are being deleted from one file system, another file system (in the same aggregate) can use that physical space for creating new files.

Creating a multi-file system aggregate

A multi-file system aggregate is a VSAM Linear Data Set (LDS) that can contain multiple zFS file systems. The multi-file system aggregate and the zFS file systems that are contained in the aggregate are created separately. First, the multi-file system aggregate is created using the zFS **ioeagfmt** utility. The aggregate must be attached and then one or more zFS file systems are created in the aggregate using one or more **zfsadm create** commands. Creating a zFS multi-file system aggregate is a two step process:

1. Create a VSAM LDS using IDCAMS
2. Format the VSAM LDS as a multi-file system aggregate using **ioeagfmt**.

The VSAM LDS and the zFS multi-file system aggregate both have the same name and that name is equal to the VSAM LDS cluster name.

Figure 5 shows an example of a job that creates and formats a zFS multi-file system aggregate.

```
//USERIDA  JOB ,'Multi-File System',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//AMSDUMP  DD     SYSOUT=H
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=PRV000
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(OMVS.PRV.MULTI.AGGR002) -
            VOLUMES(PRV000) -
            LINEAR CYL(25 0) SHAREOPTIONS(2))
/*
//CREATE   EXEC   PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.PRV.MULTI.AGGR002')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 5. Job to create a multi-file system aggregate

After the multi-file system aggregate is formatted, it has zero zFS file systems contained in it. The size of the aggregate is reported by **ioeagfmt** as the number of 8K blocks that fit into the primary allocation. The multi-file system aggregate must then be attached on a system before any **zfsadm** commands can be issued against it.

When you attach a multi-file system aggregate on a system, the ZFS Physical File System (PFS) must be active on that system. Refer to “Chapter 3. Managing zFS processes” on page 9 for information on starting and stopping ZFS. You can attach in one of the following ways:

- The **zfsadm attach** command can be issued on that system, or
- A **define_aggr** statement can be placed in the **IOEFSPRM** file for that system and the ZFS PFS can be started (or restarted).

After a new multi-file system aggregate is defined and formatted, the zFS administrator attaches it by issuing the **zfsadm attach** command and then adds a **define_aggr** statement for the aggregate in that system's **IOEFSPRM** file so that the aggregate is automatically attached each time the ZFS PFS is subsequently started (or restarted). The **define_aggr** statement does not have to be added to the **IOEFSPRM** file but then the aggregate would need to be attached (by using the **zfsadm attach** command) each time the ZFS PFS is started (or restarted).

The OMVS **zfsadm attach** command for the multi-file system aggregate just created is show in the following example:

```
zfsadm attach -aggregate omvs.prv.multi.aggr002
```

A **define_aggr** statement in the **IOEFSPRM** file for the multi-file system aggregate just created is shown in the following example:

```
define_aggr cluster(omvs.prv.multi.aggr002)
```

Note: zFS aggregate names are case insensitive. Since they are always VSAM LDS names, they are always folded to upper case.

After the multi-file system aggregate is attached, the administrator can now create zFS file systems in the aggregate. This is accomplished using the OMVS **zfsadm create** command. The following example shows the creating a file system in the aggregate you just created and attached:

```
zfsadm create -filesystem OMVS.PRV.FS1 -aggregate omvs.prv.multi.aggr002 -size 5000
```

The previous example creates a zFS file system (named `OMVS.PRV.FS1`) in the OMVS.PRV.MULTI.AGGR002 aggregate. The file system has a maximum size of 5000 1K blocks.

Note: zFS file system names are case sensitive. The file system name specified on the **zfsadm create** command is not folded to upper case. If you create a zFS file system using lower case letters, it must be mounted using these same lower case letters. This can be accomplished by using the TSO/E **MOUNT** command and surrounding the file system name with a pair of three single quotes. Refer to the *z/OS: UNIX System Services Command Reference* for information on the TSO/E **MOUNT** command.

The maximum size of the file system you just created is known as its ***quota***. This is a logical number that is compared against each time additional blocks are allocated to the file system. When the quota is reached, the file system indicates that it is full (even if there are more physical blocks available in the aggregate). A quota can be smaller than the space available in the aggregate (this is typical), it can be equal or it can be larger. If the quota is larger than the space available in the aggregate, or more typically, if the sum of the quotas for all file systems in an aggregate is larger than the space available in the aggregate, the file system can run out of physical space before it reaches its quota.

The quota of a file system can be displayed by using the **zfsadm lsquota** command and it can be increased by using the **zfsadm setquota** command. The quota of a file system can also be decreased (as

long as the usage has not exceeded the new quota) by using the **zfsadm setquota** command. The quota is the number used when determining if a message to the operator is required due to the **FSFULL** parameter of the **MOUNT** command. Refer to “MOUNT” on page 50 for more information.

File system names must be unique across all attached aggregates on a system. If you attempt to create a file system with a duplicate name, it will be denied. If you attempt to attach an aggregate that contains a duplicate file system name, the attach is successful but an error message for the duplicate file system name occurs and you are not able to use the duplicate named file system.

After creating a zFS file system in a multi-file system aggregate, the file system can be mounted. Following is an example of a TSO/E **MOUNT** command for the zFS file system just created:

```
MOUNT FILESYSTEM('OMVS.PRV.FS1') MOUNTPOINT('/etc/mountpt2') TYPE(ZFS) MODE(RDWR) NOAUTOMOVE
```

The previous example assumes that the directory `/etc/mountpt2` exists and is available to become a mount point. Note that the **TYPE** parameter of the **MOUNT** command specifies **ZFS**. This is required for any zFS file system. Once the zFS file system is mounted, applications and commands can be executed and files and directories can be accessed in zFS just as in HFS. Refer to “Chapter 5. Sysplex considerations” on page 13 for the reason that **NOAUTOMOVE** is specified for file systems in multi-file system aggregates.

When multiple file systems are created in an aggregate, this allows the possibility of space sharing between those file systems. That is, physical DASD space that is made available by erasing files in one file system (A), is potentially available to another file system (B) in the same aggregate (assuming that the other file system (B) is not at its quota limit).

Growing a multi-file system aggregate

If the sum of the quotas of all the file systems in an aggregate is greater than the physical space available in the aggregate, it is possible for a file system to run out of physical space before exceeding its quota. If this occurs, the application gets ENOSPC as a return code (the same return code it would get for exceeding its quota). The administrator can grow the aggregate (that is, cause secondary allocations to occur and format them to be part of the aggregate). This is accomplished with the **zfsadm grow** command. The aggregate’s VSAM LDS must have secondary allocation specified and space on the volume(s) must be available. The size specified on the **zfsadm grow** command must be larger than the current size of the aggregate.

For example, suppose a 2 cylinder (primary allocation, 3390) aggregate has a total of 179 8K blocks and a (potential) secondary allocation of 1 cylinder. 179 8K blocks is 1432K bytes. A **zfsadm aggrinfo** command for this aggregate might show 1296K with 136K reserved. This is a total of 1432K. A **zfsadm grow** command would need to specify a size greater than 1432 to actually grow the aggregate. **zfsadm grow** does this by calling DFSMS to allocate the additional DASD space. You may need to specify a few blocks larger than the current size before a secondary allocation occurs because DFSMS may require some number of reserved blocks. For example, you may need to specify a size of 1448 before the secondary allocation actually occurs. Refer to the following example:

```
zfsadm aggrinfo omvs.prv.aggr004.lds0004

OMVS.PRV.AGGR004.LDS0004 (R/W MULT): 1159 K free out of total 1296 (136 reserved)
zfsadm grow omvs.prv.aggr004.lds0004 1447

IOEZ00175E Error 8 received growing aggregate OMVS.PRV.AGGR004.LDS0004
OMVS.PRV.AGGR004.LDS0004 (R/W MULT): 1159 K free out of total 1296 (136 reserved)
zfsadm grow omvs.prv.aggr004.lds0004 1448

IOEZ00173I Aggregate OMVS.PRV.AGGR004.LDS0004 successfully grown
OMVS.PRV.AGGR004.LDS0004 (R/W MULT): 1807 K free out of total 1944 (208 reserved)
```

The aggregate now has a total size of 2152K bytes (1944 + 208). The size of the aggregate is rounded up to the allocation size. File systems that have not exceeded their quota can now use the additional physical space that is available. (If necessary, a file system quota can be increased with the **zfsadm setquota** command.) Aggregates cannot be made smaller.

Note: There is no automatic grow mechanism in zFS.

Cloning a file system

The ability to clone a file system is another capability provided in zFS. This is accomplished with the **zfsadm clone** command. When a file system is cloned, a copy of the file system is created in the same aggregate. There must be physical space available in the aggregate for the clone to be successful. This copy of the file system is read-only and is called a ***backup file system***. The file system name of the backup file system is the same as the original (read-write) file system with **.bak** (in lower case) appended to the file system name. This means that you need to limit the length of a file system name to 40 characters if you want to clone it.

A backup file system takes up a relatively small amount of space because only the metadata is copied, not the user data. The backup file system's data block pointers point to the same data blocks that the read-write file system's data block pointers point to. After a clone operation, if the read-write file system user data is updated, zFS ensures that new physical blocks are allocated to hold the updates, while maintaining the backup file system's data pointers to the original data. The backup file system remains a point-in-time read-only copy in the face of updates to the read-write file system. This backup file system can be mounted (read-only) so that users of the read-write file system can have an online backup of that file system available without administrative intervention. That is, if a user accidently erases a file from the read-write file system, they can simply copy the file from the backup into the read-write file system to restore the file to the time the backup was created.

The read-write file system can be cloned again (reclone). When a backup file system already exists, it is replaced during the clone operation. One backup file system can exist for each read-write file system. Backup file systems cannot be mounted during the clone operation.

You can clone (or reclone) a set of file systems (on a single system) with the **zfsadm clonesys** command. This can be specified in terms of file systems with a file system name prefix or file systems in an aggregate or both.

Comparing compatibility mode aggregates and multi-file system aggregates

The difference between a compatibility mode aggregate and a multi-file system aggregate is simply the number of read-write file systems in the aggregate, the name of the file system, and whether the aggregate has been explicitly attached or not. There is no special bit stored on the disk that indicates whether an aggregate is compatibility mode or multi-file system.

A compatibility mode aggregate has exactly one read-write file system in the aggregate whose file system name is the same as the aggregate name and it is not attached before being mounted. It is only mounted and unmounted. (An implicit attach occurs during the mount; an implicit detach occurs during the unmount.) The decision as to whether to treat an aggregate as a compatibility mode aggregate is made at mount time or explicit attach time. If no attach has been done and the mount is successful, the aggregate is treated as a compatibility mode aggregate. The mount will only be successful if there is exactly one read-write file system in the aggregate and its name is the same as the aggregate name.

If an explicit attach is done before any mounting, the aggregate is treated as a multi-file system aggregate. If you want, you can cause zFS to treat an aggregate that has been formatted with the **-compat** option, as a multi-file system aggregate by attaching it before you mount the file system. You only do this if you

wanted to rename the file system contained in the aggregate to match the aggregate name. Then you could detach the aggregate and then mount it (and it would be treated as a compatibility mode aggregate).

You can always query an (attached) aggregate to determine if it is compatibility mode or multi-file system using the **zfsadm aggrinfo** command. COMP indicates compatibility mode; MULT indicates multi-file system.

Chapter 9. Performance and debugging

This chapter discusses performance tuning techniques and what should be done if a problem occurs that requires IBM service assistance.

Performance tuning

zFS performance is dependent on many factors. zFS provides performance information to help the administrator determine bottlenecks. The **IOEFSPRM** file contains many tuning options that can be adjusted. The output of the operator modify query commands provide feedback about the operation of zFS. This section describes those **IOEFSPRM** options and the operator commands that relate to performance.

zFS 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 there are no volumes or channels that are pushed beyond their capacity. The following describes some of the considerations when tuning zFS performance.

User file cache

The user file cache is used to cache all ″regular″ files. It caches any file no matter what its size and performs write-behind and asynchronous read-ahead for files. It performs I/O for all files that are 7K or larger. For files smaller than 7K, I/O is normally performed through the metadata cache.

The user file cache is allocated in the zFS primary address space. Its size by default is 256M and can be tailored to meet your performance needs based on your overall system memory and the sizes chosen for the other zFS caches. The general rule for any cache is to ensure a good hit ratio and additionally, for a user file cache it is good to have it large enough to allow write-behind activity to occur (if the cache is too small you need to recycle buffers more frequently and it could degrade write-behind performance). The operator **MODIFY ZFS,QUERY,ALL** command output shows the cache hit ratio. (Actually, it shows the ″Fault Ratio″. To get the hit ratio subtract the fault ratio from 100%).

In general you should have hit ratios of at least 80% or more, over 90% usually gives good performance. However, the desired hit ratio is very much workload dependent. For example, a zFS file system exported exclusively to SMB clients by using the SMB server would likely have a low hit ratio since the SMB client and the SMB server caches data, making the zFS cache achieve a low hit ratio in this case. That is expected and is not considered a problem.

NOREADAHEAD option

For sequential file access, read-ahead provides an overlap of I/O with processing that can result in smaller response time for file read requests. However for random file access, read-ahead can degrade performance. zFS generally attempts to first determine if a file's access pattern is sequential or random before it decides if read-ahead should be performed for that file. Since zFS really has no knowledge of an applications future requests sometimes zFS can make the wrong guess. For file systems that have random access patterns, which are typical for database systems such as Lotus Notes, sequential access is rare and the administrator can disable read-ahead for that file system by specifying the **NOREADAHEAD** option for the **MOUNT** command. This ensures that zFS never performs read-ahead for any files in that file system and avoids any overhead due to unnecessary read-aheads.

Metadata cache

The metadata cache is used to contain all file system metadata which includes all directory contents, file status information (such as atime, mtime, size, permission bits, and so on), file system structures and additionally, it also caches data for files smaller than 7K. Essentially, zFS stores a file by using one of the following three methods:

inline If the file is smaller than 48 bytes, its data is stored in the structure that contains the status information for the file.

fragmented If the file is less than 7K it is stored in blocks on disk that could be shared with other files, hence multiple files are stored in the same physical disk block. Physical disk blocks are always 8K in size.

blocked Files larger than 7K are stored in multiple blocks, blocked files are only stored in the user file cache, and all I/O is performed directly to or from user file cache buffers.

Since inline files are stored in the status block, files that are stored on disk by using the inline method are stored in the metadata and hence are cached in the metadata cache (and also in the user file cache). Since the metadata cache is the only component that knows about multiple files sharing the same disk blocks, small fragmented files are stored in the metadata cache (and also in the user file cache) and I/O is performed directly to or from the metadata cache for these small user files.

Generally metadata is referred to and updated very frequently for most zFS file operations, hence achieving a good hit ratio is often essential to good performance for most workloads. A good hit ratio might be considered to be 90% or more depending on your workload.

The metadata cache is stored in the primary address space and its default size is 32M. Since the metadata cache only contains metadata and small files it normally does not need to be nearly as large as the user file cache. The operator **MODIFY ZFS,QUERY,ALL** command output shows statistics for the metadata cache including the cache hit ratio and I/O rates from the metadata cache.

Log files

Every zFS aggregate contains a log file used to record transactions describing changes to the file system structure. This log file is, by default, 1% of the aggregate size but is tailorable by the administrator on the **ioeagfmt** command. Usually, 1% is sufficient for most aggregates, especially large aggregates might need less than 1% while very small aggregates might need more than 1% if a high degree of parallel update activity occurs for the aggregate.

Log file cache

The log file cache is a pool of 8K buffers used to contain log file updates. Log file buffers are always written asynchronously to disk and normally only need to be waited upon when the log is becoming full, or if a file is being fsync'ed.

The log file cache is stored in the primary address space and its default is 64M. The log file cache is grown dynamically by adding one 8K buffer for each attached aggregate. This ensures each aggregate always has one 8K buffer to use to record its most recent changes to file system metadata. Since log files are written asynchronously, the cache essentially allows write-behind of log files and since the cache is shared among all aggregates, aggregates that have a higher write rate use more buffers in the cache using a least-recently-used (LRU) algorithm.

The log file cache is a write-only cache, so a read hit ratio is not relevant, however the operator **MODIFY ZFS,QUERY,ALL** command does show log file I/O rates and I/O waits. Its desirable to make the log file cache large enough so that log file I/O waits do not occur too frequently compared to the log file I/O rates, however, every workload is different. For example, workloads that issued fsync operations force zFS to sync the log file more frequently.

Transaction cache

Every change to zFS file system metadata is bounded by a transaction describing its changes by using records written to the log file. The transaction cache is a cache of data structures representing transactions.

The transaction cache is stored in the zFS primary address space and its default is 2000. zFS dynamically increases the size of this cache based on the number of concurrent pending transactions (transactions that have not been fully committed to disk) in the zFS file system. Therefore, the administrator does not have to tailor the transaction cache size, but the **MODIFY ZFS,QUERY,ALL** output shows how large the transaction cache is at any given time.

Vnode cache

Every object in the zFS file system is represented by a data structure called a vnode in memory. zFS keeps a cache of these and recycles these vnodes in an LRU fashion. Every operation in zFS requires a vnode and z/OS UNIX keeps pointers to zFS vnodes. Since z/OS UNIX keeps references to zFS vnodes, zFS may be forced to dynamically increase the size of this cache to meet the demands of z/OS UNIX. To create a zFS vnode for a newly referenced file or a newly created file for a user requires the pathlength to initialize the structure and obtain its status information from the metadata cache. If the file's status is not in the metadata cache then a disk I/O may also be required.

The vnode cache is stored in the zFS primary address space and the default number of vnodes is 16384. As with any cache a good hit ratio is desirable and the operator **MODIFY ZFS,QUERY,ALL** command shows the vnode cache hit ration. Since the vnode cache is essentially backed by the metadata cache, if the vnode hit ratio is low but the metadata cache hit ratio is high your performance may not suffer too much since a vnode cache miss only requires some pathlength to initialize the vnode structures.

Fixed storage

By default, zFS does not fix any pages in any of the caches except when an I/O is pending to or from the cache buffers. The administrator can permanently page fix the user file cache, the metadata cache, and/or the log file cache by choosing the **fixed** option for the cache. This ensures the cache experiences no paging and avoids the overhead of page fixing for each I/O but comes at the expense of using real storage for the given cache which means the real storage is not available for other applications.

If your file system performance is critical and you have enough real memory to support it, the **fixed** option may be useful. Otherwise, you should not set it.

I/O balancing

Any file system's performance is heavily dependent on DASD I/O performance. If any channel(s) or DASD volume(s) are overloaded, then excessive I/O waits could occur for that DASD.

Performance products such as RMF will show DASD performance.

zFS operator **MODIFY ZFS,QUERY,ALL** commands also provide reports that show I/O rates per aggregate, and file system request rates per aggregate and per file system. This information, along with DASD performance information from RMF or performance products similar to RMF can be used to easily balance I/O among your DASD. For example, the QUERY output can be used to show which file systems could be moved to different DASD to achieve a better balance among disks.

Total cache size

Currently, the ZFS address space is restricted to 2GB of total storage. Therefore, the total storage size for all of the caches must be less than 2GB. Since storage is needed in addition to the cache storage to process file requests and for products zFS might use, generally you should restrict total zFS cache storage to approximately 1.5GB. The zFS operator **MODIFY ZFS,QUERY,ALL** command shows total zFS storage allocated which includes storage allocated for all the caches and everything else zFS might need. zFS terminates during initialization if it cannot obtain all the storage for the caches as directed by the zFS parameters file.

Debugging

If a problem occurs in zFS that requires IBM service support, it is important that appropriate problem determination information be obtained so that the problem can be resolved quickly.

One of the most important aspects of zFS problem determination is its tracing capability. zFS has an internal (wrap around) trace table that is always tracing certain events. The size of this trace table is controlled by the **IOEFSPRM trace_table_size** option. The trace table can be reset (that is, set to empty) to minimize the amount of information generated if you are recreating a problem. This is accomplished with the operator **MODIFY ZFS,TRACE,RESET** command. The trace table can be formatted and sent to a trace output data set by using the operator **MODIFY ZFS,TRACE,PRINT** command. The trace output data set must be preallocated and its name must be specified in the **IOEFSPRM trace_dsn** option. A separate trace output data set is required for each member of a sysplex. (This requires separate **IOEFSPRM** files. Refer to “Chapter 5. Sysplex considerations” on page 13.) It should be allocated as a PDSE, RECFM=VB, LRECL=133 with a primary allocation of at least 50 cylinders and a secondary allocation of 30 cylinders. Each trace output is created as a new member with a name of ZFSKNT*nn*. *nn* starts at 01 and is incremented for each trace output until zFS is restarted. After restart, when the next trace output is sent to the trace output data set, ZFSKNT01 is overlayed. You should not be accessing the trace output data set while a trace is being sent to the trace output data set. The space used by a particular trace depends on how large the **trace_table_size** is and how recently the trace was reset. As an example, a 32M **trace_table_size** can generate a trace output member of 100 cylinders of 3390. It is important that the trace output data set be large enough to hold the trace output since ZFS terminates if it runs out of room sending the trace to the trace output data set.

Another important source of information is a ZFS dump. Any time a ZFS failure occurs, you should check the system log to see if ZFS has dumped. ZFS also sends the trace to the trace output data set when a ZFS dump occurs. Note that when ZFS abends, other application failures may occur (since ZFS is unavailable during this time). For problem determination, these failures are not as important as the original ZFS failure and dump. ZFS attempts to restart after the dump is taken. If it is successful, the administrator must remount any zFS file systems.

If a failure of a zFS operation occurs (other than a user error), but zFS does not dump, you should get a trace of the failure, if possible. You can perform the following steps:

1. Issue the operator **MODIFY ZFS,TRACE,RESET** command
2. Recreate the failing scenario
3. Issue the operator **MODIFY ZFS,TRACE,PRINT** command
4. Capture the ZFSKNT*nn* member from the trace output data set (for example, copy it to a sequential data set) so that it can be sent to IBM service.

You can also obtain a dump of the ZFS address space by issuing the operator **MODIFY ZFS,DUMP** command.

IBM service may need more events to be traced. Additional tracing can be specified in two ways:

- The events that are traced can be added to by specifying ioedebug statements in a data set that is read when ZFS is started (or restarted). The data set name is specified in the **IOEFSPRM debug_settings_dsn** option. It is a PDS member with an LRECL of at least 80. IBM specifies the exact statements needed in the data set.
- The events that are traced can be added dynamically by issuing operator **MODIFY ZFS,IOEDEBUG** commands. IBM specifies the exact statements needed.

If a hang condition occurs, the operator **MODIFY ZFS,QUERY,THREADS** command can be used to determine if any zFS threads are hanging and why. You can also issue the operator **MODIFY ZFS,ABORT** command to cause ZFS to send the trace to the trace output data set and to issue a dump. This also causes ZFS to terminate and attempt to restart.

If you have a ZFS dump but could not capture the trace, the trace can be obtained from the dump.

The service level of the ZFS physical file system can be determined by examining the first messages that occur on the operator's console when zFS initializes as shown in the following example.

```
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.02.00 Service Level OW51563.
Created on Thu Oct  4 12:22:41 EDT 2001.
```

In addition, the service level of the **zfsadm** command can be determined by using the **-level** option of the **zfsadm** command. For example:

```
zfsadm -level
```

```
IOEZ00020I zfsadm: z/OS    zSeries File System
Version 01.02.00 Service Level OW51563.
Created on Thu Oct  4 12:22:41 EDT 2001.
```

Part 2. zFS administration reference

This part of the book discusses the zSeries File System (zFS) reference information.

Chapter 10. z/OS system commands

This chapter introduces you to the following z/OS system commands:

- **MODIFY**, a system command which enables you to query internal counters and values. It also allows you to initiate or gather debugging information.
- **SETOMVS RESET**, a system command that starts the ZFS Physical File System (PFS) if it has not been started at IPL or if it has been stopped and the BPXF032D message has been responded to with a reply of i.
- **STOP**, a system command that enables you to stop the ZFS PFS program (**IOEFSCM**).

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

modify zfs process

Purpose

Enables you to query internal ZFS counters and values. They are displayed on the system log. It also allows you to initiate or gather debugging information. The ZFS PFS must be running to use this command.

Format

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

modify *procname*,**query**,**{all** | **settings** | **storage** | **threads}**

modify *procname*,**reset**,**{all** | **storage}**

modify *procname*,**trace**,**{reset** | **print}**

modify *procname*,**abort**

Parameters

procname The name of the ZFS PFS PROC. The default procname is **ZFS**.

command The action that is performed on the ZFS PFS. This parameter can have one of the following values:

query Displays ZFS counters or values.

all Displays all the ZFS counters.

settings
Displays the ZFS configuration settings. These are based on the **IOEFSPRM** file and defaults.

storage
Displays the ZFS storage values.

threads
Displays the threads running in the ZFS address space.

reset Resets ZFS counters to zero.

all Resets all the ZFS counters to zero.

storage
Resets the ZFS storage counters to zero.

trace Resets or prints the internal ZFS trace table.

reset Resets the internal (wrap around) trace table to empty.

print Formats and sends the current trace table to the data set specified in the **IOEFSPRM** file **trace_dsn** entry. This data set must be preallocated as a PDSE with RECFM FB and LRECL 133. It must be large enough to hold the formatted trace table. Refer to “Chapter 9. Performance and debugging” on page 29 for more information of the trace output data set.

abort Causes the ZFS PFS to abnormally terminate and dump. The internal trace table is also printed to the data set specified in the **IOEFSPRM** file **trace_dsn** entry.

Usage

The **modify zfs** *command* command is used to display ZFS counters or values and to initiate or gather debugging information.

Privilege Required

This command is a z/OS system command.

Examples

The following example queries all the ZFS counters:

```
modify zfs,query,all
```

The following example resets the ZFS storage counters:

```
modify zfs,reset,storage
```

The following example formats and sends the trace table to the data set specified in the **IOEFSPRM** file **trace_dsn** entry:

```
modify zfs,trace,print
```

The following example causes the ZFS PFS to dump and terminate:

```
modify zfs,abort
```

Related Information

File:

IOEFSPRM

setomvs reset

Purpose

Can be used to start the ZFS PFS if it has not been started at IPL or to restart it if it has been terminated by replying **i** to the BPXF032D operator message (after stopping the ZFS PFS).

Format

setomvs reset=(*xx***)**

Parameters

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

Usage

The **setomvs reset** command can be used to start the ZFS PFS.

Privilege Required

This command is a z/OS system command.

Examples

The following command starts the ZFS Physical File System if the BPXPRMSS member of the PARMLIB contains the ZFS FILESYSTYPE statement:

```
setomvs reset=(ss)
```

Related Information

File:

IOEFSPRM

stop zfs

Purpose

Stops the ZFS PFS. All zFS file systems are unmounted and all zFS aggregates are detached. After the ZFS PFS stops, z/OS UNIX displays the message BPXF032D. A response to this message can restart the ZFS PFS (**r nn,r**) or remove the message (**r nn,i**) and leave the ZFS PFS terminated.

Format

stop *procname*

Parameters

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

Usage

The STOP command stops the ZFS PFS (**ioefscm**) and the process controlled by the **ioefscm** process. This process is:

ioefsadm The **ioefsadm** process. The **ioefsadm** process is the process that handles ZFS mount and pfsctl requests.

Privilege Required

This command is a z/OS system command.

Examples

The following command stops the ZFS PFS:

```
stop zfs
```

Related Information

File:

IOEFSPRM

stop zfs

Chapter 11. zFS commands

This chapter provides a description of the relevant zFS commands.

ioeagfmt

Purpose

Creates an HFS compatibility mode aggregate or a multi-file system aggregate.

Format

```
ioeagfmt -aggregate name [-logsize blocks] [-overwrite] [-compat] [-owner {uid|name}] [-group {gid|name}]
                         [-perms {number}] [-parmfile name] [-help]
```

Options

-aggregate *name*

Specifies the name of the data set to format. This is also the aggregate name. The aggregate name is always translated to upper case. The following characters can be included in the name of an aggregate:

- All uppercase and lowercase alphabetic characters (a to z, A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name can be no longer than 44 characters. If this is a compatibility mode aggregate (see the **-compat** option), and you intend to clone the file system (see the **zfsadm clone** command), you may want to limit the aggregate name to 40 characters.

-logsize *blocks*

Specifies the size in 8K blocks of the log. The default is 1% of the aggregate size (which is sufficient).

-overwrite Required if you are reformatting an existing aggregate. Use this option with caution, since it destroys any existing data. This option is not usually specified.

-compat Indicates that a compatibility mode aggregate should be created. This means that in addition to formatting the VSAM Linear Data Set (LDS) as a zFS aggregate, a zFS file system by the same name (the aggregate name) is created and its quota is set to the size of the available blocks on the aggregate. This option should normally be specified unless you want to create a multi-file system aggregate. Refer to “Chapter 8. Multi-file system aggregates” on page 23 for more information on multi-file system aggregates.

-owner *uid* **|** *name*

Specifies the owner for the root directory of the file system. This is used with the **-compat** option, otherwise it is ignored. It may be specified as a z/OS user ID or as a *uid*. The default is the *uid* of the issuer of **ioeagfmt**.

-group *gid* **|** *name*

Specifies the group owner for the root directory of the file system. This is used with the **-compat** option, otherwise it is ignored. It may be specified as a z/OS group name or as a *gid*. The default is the *gid* of the issuer of **ioeagfmt**. If only **-owner** is specified, the group is that owner’s default group.

-perms *number*

Specifies the permissions for the root directory of the file system. This is used with the **-compat** option, otherwise it is ignored. The number can be specified as octal (for example, o755), as hexadecimal (for example, x1ED), or as decimal (for example, 493). The default is o755 (owner read/write/execute, group read/execute, other read/execute).

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

Usage

The **ioeagfmt** utility is used to format an existing VSAM LDS as a zFS aggregate. All zFS aggregates must be formatted before use (including HFS compatibility mode aggregates). **ioeagfmt** can be run even if the ZFS PFS is not active on the system. The size of the aggregate is as many 8K blocks as will fit in the primary allocation of the VSAM LDS. To extend it to its secondary allocation (assuming it has a secondary allocation specified), use the **zfsadm grow** command. If **-overwrite** is specified, all existing primary and secondary allocations will be formatted and the size will include all of that space.

Privilege Required

The user must have ALTER authority to the VSAM LDS.

Examples

Figure 6 shows an example of a job that creates a compatibility mode aggregate and file system.

```
//USERIDA  JOB ,'Compatibility Mode',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//AMSDUMP  DD     SYSOUT=H
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=PRV000
//SYSIN    DD     *
     DEFINE CLUSTER (NAME(OMVS.PRV.COMPAT.AGGR001) -
            VOLUMES(PRV000) -
            LINEAR CYL(25 0) SHAREOPTIONS(2))
/*
//CREATE   EXEC   PGM=IOEAGFMT,REGION=0M,
// PARM=('-aggregate OMVS.PRV.COMPAT.AGGR001 -compat')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 6. Job to create a compatibility mode aggregate and file system

ioeagslv

Purpose

Scans an aggregate and reports inconsistencies. Aggregates can be verified, recovered (that is, the log is replayed), or salvaged (that is, the aggregate is repaired). This utility is known as the Salvager.

Note: This command is not normally needed. If a system failure occurs, the aggregate log is replayed automatically, the next time the aggregate is attached (or for compatibility mode aggregates, the next time the file system is mounted). This normally brings the aggregate (and all the file systems) back to a consistent state.

Format

ioeagslv -aggregate *name* **[-recoveronly] [-verifyonly | -salvageonly] [-force] [-verbose] [-help]**

Options

-aggregate *name*
Specifies the name of the aggregate to be verified, recovered, or salvaged.

-recoveronly Directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate. Refer to “Usage” for information about using and combining the command’s options.

-verifyonly Directs the Salvager to verify the specified aggregate. The Salvager examines the structure of the aggregate to determine if it contains any inconsistencies, reporting any that it finds. Refer to “Usage” for information about using and combining the command’s options.

-salvageonly Directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate. Refer to “Usage” for information about using and combining the command’s options.

-force Executes the Salvager in non-interactive mode. By default, the Salvager prompts for confirmation before proceeding in certain situations (for example, if it believes an aggregate on which it is run may be a non-zFS aggregate). Use this option to direct the Salvager to proceed with all operations without asking whether it should continue. Use this option with care; the Salvager’s changes can be unpredictable if this option is used with an invalid aggregate.

-verbose Directs the Salvager to produce detailed information about the aggregate as it executes. The information is useful primarily for debugging purposes. It is displayed on standard output (which can be redirected). Use this option alone or with any combination of the available options.

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

Usage

The **ioeagslv** command invokes the Salvager on the zFS aggregate specified with the **-aggregate** option. Following a system restart, the Salvager employs the zFS file system log mechanism to return consistency to a file system by running recovery on the aggregate on which the file system resides. Recovery is the replaying of the log on the aggregate; the log records all changes made to metadata as a result of operations such as file creation and deletion. If problems are detected in the basic structure of the aggregate, if the log mechanism is damaged, or if the storage medium of the aggregate is suspect, the **salvage** command must be used to verify or repair the structure of the aggregate.

Use the command's **-recoveronly**, **-verifyonly**, and **-salvageonly** options to indicate the operations the Salvager is to perform on the specified aggregate, as follows:

- Specify the **-recovery** option

  To run recovery on the aggregate without attempting to find or repair any inconsistencies found on it. Recovery is the replaying of the log on the aggregate. Use this option to quickly return consistency to an aggregate that does not need to be salvaged; this represents the normal production use of the Salvager. Unless the contents of the log or the physical structure of the aggregate is damaged, replaying the log is an effective guarantee of a file system's integrity.
- Specify the **-verifyonly** option

  To determine whether the structure of the aggregate contains any inconsistencies without running recovery or attempting to repair any inconsistencies found on the aggregate. Use this option to assess the extent of the damage to an aggregate. The Salvager makes no modifications to an aggregate during verification. Note that it is normal for the Salvager to find errors when it verifies an aggregate that has not been recovered; the presence of an unrecovered log on an aggregate makes the findings of the Salvager, positive or negative, of dubious worth.
- Specify the **-recoveronly** and **-verifyonly** options

  To run recovery on the aggregate and then analyze its structure without attempting to repair any inconsistencies found on it. Use these options if you believe replaying the log can return consistency to the aggregate, but you want to verify the consistency of the aggregate after recovery is run. Recovering an aggregate and then verifying its structure represents a cautious application of the Salvager.
- Specify the **-salvageonly** option

  To attempt to repair any inconsistencies found in the structure of the aggregate without first running recovery on it. Use this option if you believe the log is damaged or replaying the log does not return consistency to the aggregate and may in fact further damage it. In most cases, you do not salvage an aggregate without first recovering it.
- Omit the **-recoveronly**, **-verifyonly**, and **-salvageonly** options

  To run recovery on the aggregate and then attempt to repair any inconsistencies found in the structure of the aggregate. Because recovery eliminates inconsistencies in an undamaged file system, an aggregate is typically recovered before it is salvaged. In general, it is good first to recover and then to salvage an aggregate if a system goes down or experiences a hardware failure.

  Omit these three options if you believe the log should be replayed before attempts are made to repair any inconsistencies found on the aggregate. (Omitting the three options is equivalent to specifying the **-recoveronly** and **-salvageonly** options.)

The following rule summarizes the interaction of the **-recoveronly**, **-verifyonly**, and **-salvageonly** options: The salvage command runs recovery on an aggregate and attempts to repair it unless one of the three salvage options is specified; once one of these options is specified, you must explicitly request any operation you want the Salvager to perform on the aggregate.

The basic function of the Salvager is similar to that of the z/OS UNIX **fsck** program. The Salvager recovers a zFS aggregate and repairs problems it detects in the structure of the aggregate. It does not verify or repair the format of user data contained in files on the aggregate. If it makes changes, the Salvager displays the pathnames of the files affected by the modifications, when the pathnames can be determined. The owners of the files can then verify the files' contents, and the files can be restored from backups if necessary

The Salvager verifies the structure of an aggregate by examining all of the anodes, directories, and other metadata in each file system on the aggregate. An ***anode*** is an area on the disk that provides information used to locate data such as files, directories, ACLs, and other types of file system objects. Each file system contains an arbitrary number of anodes, all of which must reside on the same aggregate. By following the links between the various types of anodes, the Salvager can determine whether the organization of an aggregate and the file systems it contains is correct and make repairs if necessary.

ioeagslv

Not all aggregates can be salvaged. In cases of extensive damage to the structure of the metadata on an aggregate or damage to the physical disk that houses an aggregate, the Salvager cannot repair inconsistencies. Also, the Salvager cannot verify or repair damage to user data on an aggregate. The Salvager cannot detect problems that modified the contents of a file but did not damage the structure of an aggregate or change the metadata of the aggregate.

Like the z/OS UNIX **fsck** command, the Salvager analyzes the consistency of an aggregate by making successive passes through the aggregate. With each successive pass, the Salvager examines and extracts a different type of information from the blocks and anodes on the aggregate. Later passes of the Salvager use information found in earlier passes to help in the analysis.

Unlike the **fsck** command, the Salvager does not normally prompt the issuer for additional information as it executes. It typically performs the requested operation without prompting for input or pausing to verify any changes before it makes them. It prompts the issuer for confirmation only in the following cases:

- It believes the specified aggregate does not contain a zFS aggregate. This can occur if it finds a non-zFS superblock whose creation time is more recent than the creation time of the zFS superblock.
- It finds that the size of the aggregate recorded in the zFS superblock exceeds the capacity of the VSAM LDS on which the aggregate resides.

At the prompt, the issuer can choose to cancel or continue the operation. If the operation is continued under either of these circumstances and the aggregate proves to be invalid, unpredictable results can ensue. The best response in either case is to cancel the operation and attempt to determine the cause of the problem.

If you are confident that you want the salvager to continue in any case, you can include the **-force** option with the command. This option directs the Salvager to perform the requested operation without prompting the issuer for confirmation. Exercise caution when using the **-force** option, as the Salvager can produce unpredictable results if this option is used with an invalid aggregate.

In general, the Salvager exits with an error code of at least 16 without analyzing a VSAM LDS that it is sure is not a zFS aggregate. It also exits with an error code of 16 if a file system on the aggregate to be recovered or salvaged is attached. (If necessary, you can use the **zfsadm detach** command to detach the aggregate.)

As the Salvager executes, it maintains a number of internal lists. Each list consists of anodes that failed verification in specific ways. When it initially scans an aggregate, the Salvager marks as ″unsafe″ anodes with which it encounters problems. The Salvager later attempts to determine the actual pathnames associated with these anodes to include the pathnames in the lists. When it has finished salvaging, the Salvager displays any non-empty lists.

Privilege Required

The privileges required depend on whether the **-recoveronly**, **-verifyonly**, or **-salvageonly** option is specified with the command:

- If just the **-verifyonly** option is included, the issuer needs only the READ authority for the specified VSAM LDS (aggregate).
- If the **-recoveronly** or **-salvageonly** option is included, or if all three of these options are omitted, the issuer must have ALTER authority for the specified VSAM LDS.

Examples

Figure 7 on page 49 shows an example of a job that invokes the **ioeagslv** utility.

```
//USERIDA  JOB ,'Salvage',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//SALVAGE  EXEC   PGM=IOEAGSLV,REGION=0M,
// PARM=('-aggregate OMVS.PRV.COMPAT.AGGR001 -verifyonly')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 7. Job to verify a zFS aggregate

MOUNT

Purpose

Mounts a file system into the z/OS UNIX hierarchy. This section only documents MOUNT options that are unique to zFS. For additional information on this command, refer to the *z/OS: UNIX System Services Command Reference*.

Format

MOUNT TYPE(*file_system_type***) [PARM(***parameter_string***)]**

Options

TYPE(*file_system_type***)**
Specifies the file system type. For zFS, this must be specified as **ZFS**.

PARM(*parameter_string***)**
Specifies a parameter string to be passed to zFS. Parameters are case sensitive and separated by a comma. Enclose the parameter string in quotes.

FSFULL(*threshold,increment***)**
Specifies the threshold and increment for reporting file system quota error messages to the operator. The default is the **fsfull** specification in the **IOEFSPRM** file.

AGGRFULL(*threshold,increment***)**
Specifies the threshold and increment for reporting aggregate full error messages to the operator. The default is the **aggrfull** specification in the **IOEFSPRM** file. This parameter only applies to compatibility mode aggregates/file systems.

NBS | NONBS
Specifies the new block security processing for this aggregate. The default is the **nbs** specification in the **IOEFSPRM** file. This parameter only applies to compatibility mode aggregates/file systems.

NOREADAHEAD
Specifies that this file system will not be accessed sequentially and that the zFS read ahead processing normally done should be disabled. This should be used for file systems that have random access patterns (for example, for file systems that are used like a database). The default is to do read ahead processing.

Usage

The **MOUNT** command mounts a zFS file system.

Examples

The following TSO/E example mounts a zFS file system and specifies a threshold and increment to display a message when the file system becomes almost full:

```
MOUNT FILESYSTEM('OMVS.PRV.AGGR004.LDS0004') MOUNTPOINT('/etc/zfscompat1') TYPE(ZFS) MODE(RDWR) PARM('FSFULL(90,5)')
```

Related Information

Command:

UNMOUNT (For information on this command, refer to the *z/OS: UNIX System Services Command Reference*.

File:

IOEFSPRM

zfsadm

Purpose

Introduction to the **zfsadm** command suite.

Command Syntax

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

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

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

zfsadm detach {-all | -aggregate *name* **} [-help]**

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

- **Command** - A command consists of the command suite (**zfsadm** in the previous example) and the command name (**detach**). 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 bold type in the text, are always preceded by a **-** (dash), and are often followed by arguments. In the previous example, **-aggregate** is an option, with *name* as its argument. An option and its arguments tell the program which entities to manipulate when executing the command (for example, which aggregate, or which file system). In general, the issuer should provide the options for a command in the order detailed in the documentation. The { | } (braces separated by a vertical bar) indicate that the issuer must enter either one option or the other (**-all** or **-aggregate** in the previous example).
- **Arguments** - Arguments for options always appear in italic type in the text. The { | } indicate that the issuer must enter either one argument or the other (**-all** or **-aggregate** in the preceding example). The ... (ellipsis) indicates that the issuer can enter multiple arguments.
- **Optional** information - Some commands have optional, as well as required, options and arguments. Optional information is enclosed in [ ] (brackets). All options except **-all** or **-aggregate** in the previous example are optional.

Options

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

-filesystem *name*
: Specifies the file system to use with the command.

-aggregate *name*
: Specifies the aggregate name of the aggregate to use with the command.

-size *kbytes*
: Specifies the size in K-bytes for the *kbytes* argument.

-help
: Prints the online help for this command. All other valid options specified with this option are ignored. For complete details about receiving help, refer to “Receiving Help” on page 52.

Usage

Most **zfsadm** commands are administrative-level commands used by system administrators to manage file systems and aggregates. They apply to multi-file system aggregates although several apply to compatibility mode aggregates, too (for example, **zfsadm grow** and **zfsadm quiesce**/**unquiesce**). They can be issued from OMVS or as a batch job. The decriptions of the **zfsadm aggrinfo** and the **zfsadm attach** commands show examples of issuing them as a batch job. The other **zfsadm** commands can be run as a batch job in a similar manner.

zfsadm commands only work on zFS file systems and aggregates.

Receiving Help

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

$ zfsadm help
Displays a list of commands in a command suite.

$ zfsadm help -topic *command*
Displays the syntax for one or more commands.

$ zfsadm apropos -topic *string*
Displays a short description of any commands that match the specified *string*.

Privilege Required

zfsadm commands that query information (for example, **lsfs**, **aggrinfo**) can be issued by any user that has READ authority to the data set hat contains the **IOEFSPRM** file. **zfsadm** commands that modify (for example, **setquota**, **create**) additionally require that the issuer be logged in as **root**. Specific privilege information is listed with each command's description.

Related Information

Commands:

- **zfsadm aggrinfo**
- **zfsadm apropos**
- **zfsadm attach**
- **zfsadm clone**
- **zfsadm clonesys**
- **zfsadm create**
- **zfsadm delete**
- **zfsadm detach**
- **zfsadm grow**
- **zfsadm help**
- **zfsadm lsaggr**
- **zfsadm lsfs**
- **zfsadm lsquota**
- **zfsadm quiesce**
- **zfsadm rename**
- **zfsadm setquota**
- **zfsadm unquiesce**

Files:

- **IOEFSPRM**

zfsadm aggrinfo

Purpose

Displays information about an aggregate, or all attached aggregates, if there is no specific aggregate specified.

Format

zfsadm aggrinfo [-aggregate *name***] [-level] [-help]**

Options

-aggregate *name*
: Specifies the name of an aggregate about which information is to be displayed. The aggregate must be attached. The aggregate name is not case sensitive. It is translated to upper case. If this option is omitted, information is provided about all of the attached aggregates on the system. Compatibility mode aggregates are implicitly attached when they are mounted.

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

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

Usage

The **zfsadm aggrinfo** command lists information about the total amount of disk space and the amount of disk space currently available on attached aggregates. The **-aggregate** option can be used to specify a single aggregate about which information is to be displayed. If this option is omitted, information about all aggregates that are attached on the system is displayed. Compatibility mode aggregates are implicitly attached when they are mounted.

This command displays a separate line for each aggregate. Each line displays the following information:

- The aggregate name.
- Whether the aggregate is read-write (R/W) or read-only (R/O) and whether it is a compatibility mode aggregate (COMP) or a multi-file system aggregate (MULT).
- The amount of space available in K-bytes.
- The total amount of space in the aggregate in K-bytes. (To grow an aggregate, you would specify a number larger than the sum of this number and the next number.)
- The amount of space reserved for other processing in K-bytes.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

Examples

The following OMVS example displays information about the disk space available on all aggregates attached on the system:

```
zfsadm aggrinfo

OMVS.PRV.AGGR004.LDS0004 (R/W COMP): 1149 K free out of total 1296 (136 reserved)
OMVS.PRV.AGGR003.LDS0001 (R/W MULT): 2344983 K free out of total 2369344 (23928 reserved)
OMVS.PRV.AGGR002.LDS0002 (R/W MULT): 1159 K free out of total 1296 (136 reserved)
OMVS.PRV.AGGR001.LDS0001 (R/W MULT): 176119 K free out of total 177992 (2000 reserved)
```

Figure 8 on page 54 shows the same example as a job that invokes **zfsadm aggrinfo**.

zfsadm aggrinfo

```
//USERIDA  JOB ,'Zfsadm Aggrinfo',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//AGGRINFO EXEC   PGM=IOEZADM,REGION=0M,
// PARM=('aggrinfo')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 8. Job to display aggregate information

Related Information

Command:

zfsadm lsaggr

File:

IOEFSPRM

zfsadm apropos

Purpose

Shows each help entry containing a specified string.

Format

zfsadm apropos -topic *string* **[-level] [-help]**

Options

-topic Specifies the keyword string for which to search. If it is more than a single word, surround it with double quotes (″″) or other delimiters. Type all strings for **zfsadm** commands in all lowercase letters.

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

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

Usage

The **zfsadm apropos** command displays the first line of the online help entry for any **zfsadm** command containing the string specified by **-topic** in its name or short description.

To display the syntax for a command, use the **zfsadm help** command.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

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 **zfsadm** command where the string specified by **-topic** is part of the command name or first line.

Examples

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

```
zfsadm apropos list

lsaggr: list aggregates
lsfs: list filesystem information
lsquota: list filesystem and aggregate space usage
```

Related Information

Command:

zfsadm help

zfsadm attach

Purpose

Attaches an aggregate to zFS as a multi-file system aggregate.

Note: Do not explicitly attach compatibility mode aggregates. They are implicitly attached when the file system is mounted.

.

Format

zfsadm attach {-all | -aggregate *name***} [-aggrfull** *threshold,increment***] [-R/O] [-nbs] [-nonbs] [-level] [-help]**

Options

-all Specifies that all aggregates listed in the **IOEFSPRM** file that are not currently attached are to be attached. Use this option or use **-aggregate**.

-aggregate *name*
Specifies the name of the aggregate to be attached. The aggregate name is not case sensitive. It is translated to upper case. This aggregate does not need an entry in the **IOEFSPRM** file. If the aggregate is not contained in the **IOEFSPRM** file, it needs to be attached again if it is a multi-file system aggregate and the ZFS PFS is restarted.

Note: Compatibility mode aggregates do not need to be attached with the **zfsadm attach** command, nor do they need to be contained in the **IOEFSPRM** file. Compatibility mode aggregates are automatically attached on **MOUNT** of the compatibility mode file system.

-aggrfull *threshold,increment*
Specifies the threshold and increment for reporting aggregate full error messages to the operator. Both numbers must be specified. The first number is the threshold percentage and the second number is the increment percentage. For example, if 90,5 were specified, the operator would be notified when the aggregate became 90% full, then again at 95% full and again at 100% full. This overrides the **aggrfull** option in the **define_aggr** entry for this aggregate in the **IOEFSPRM** file and the global **aggrfull** entry in the **IOEFSPRM**. The default is the global **aggrfull** entry of the **IOEFSPRM** file.

-R/O Specifies that the aggregate should be opened in read-only mode. A read-only aggregate means that all file systems are read-only and can only be mounted as read-only. The default is read-write unless **R/O** is specified in the **define_aggr** entry for the aggregate in the **IOEFSPRM** file.

-nbs Specifies whether New Block Security is used for file systems in this aggregate. New block security refers to the guarantee made when a system crashes. If **-nbs** is specified then we guarantee that at the time of a crash, if a file was being extended or new blocks were being allocated for the file, but the user data has not yet made it to the disk when the crash occurred then we show the newly allocated blocks as all binary 0's and not whatever was on disk in those blocks at time of failure. The default for this is **-nonbs** since **-nbs** degrades performance and it is rare that you would have a case where the metadata updates are committed but the corresponding user updates are not on disk. However, in the case of high security requirements, this specification provides this guarantee.

-nonbs Specifies that the New Block Security guarantee is not required. This is the default. See the explanation of **-nbs** for a description of the New Block Security guarantee.

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

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

Usage

The **zfsadm attach** command attaches zFS aggregates on this system. File systems on attached aggregates are available to be mounted on the system.

If the **-all** option is provided, the command attaches all aggregates listed in the **IOEFSPRM** file. If the **-aggregate** option is provided, only the aggregate specified is attached. The specified name need not be listed in the **IOEFSPRM** file.

When **zfsadm attach -all** executes, it reads the **IOEFSPRM** file to determine the aggregates to be attached. All aggregates will be attached. If an aggregate is already attached, this will be indicated. If the attach fails because log recovery is unsuccessful, you can run the **ioeagslv** command with the **-verifyonly** option on the aggregate to determine if there is an inconsistency. If this is the case, use the **ioeagslv** command to recover the aggregate that caused the failure and reissue the **zfsadm attach** command.

The **zfsadm lsaggr** command can be used to display a current list of all aggregates attached on this system.

For multi-file system aggregates, **define_aggr** entries are generally included in the **IOEFSPRM** file for them rather than issuing **zfsadm attach** commands at the keyboard. Once included in the **IOEFSPRM** file, all aggregates listed in the **IOEFSPRM** file are attached whenever ZFS is started (or restarted) and **auto_attach=on** in the **IOEFSPRM** file.

Compatibility mode aggregates do not need to be separately attached since they are attached during **MOUNT** processing. Therefore, compatibility mode aggregates do not need **define_aggr** entries in **IOEFSPRM**, nor do they need to be attached with the **zfsadm attach** command.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and is requiredbe logged in as **root** .

Examples

The following command attaches all of the aggregates that have entries in the system's **IOEFSPRM** file.

```
zfsadm attach -all
```

The following command attaches an aggregate. No entry is needed in the system's **IOEFSPRM** file.

```
zfsadm attach -aggregate OMVS.PRV.AGGR001.LDS0001
```

Figure 9 on page 58 shows the same example as a job that invokes **zfsadm attach**.

zfsadm attach

```
//USERIDA  JOB ,'Zfsadm Attach',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//AGGRINFO EXEC   PGM=IOEZADM,REGION=0M,
// PARM=('attach -aggregate OMVS.PRV.AGGR001.LDS0001')
//SYSPRINT DD     SYSOUT=H
//STDOUT   DD     SYSOUT=H
//STDERR   DD     SYSOUT=H
//SYSUDUMP DD     SYSOUT=H
//CEEDUMP  DD     SYSOUT=H
//*
```

Figure 9. Job to attach an aggregate

zFS, by default, attaches aggregates listed in the **IOEFSPRM** file at start-up (or restart). This is based on the **auto_attach** option (default is **on**) of the **IOEFSPRM** file. The **zfsadm attach** command would be used if you had created and formatted a multi-file system aggregate after starting ZFS and you did not want to restart ZFS. A **define_aggr** entry for this multi-file system aggregate may be placed in the **IOEFSPRM** file so that it is attached the next time ZFS is started.

Related Information

Commands:

zfsadm create
zfsadm lsaggr

File:

IOEFSPRM

zfsadm clone

Purpose

Creates a backup version of a specific file system.

Format

zfsadm clone -filesystem *name* **[-level] [-help]**

Options

-filesystem *name*
Specifies the file system name of the read-write source file system.

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

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

Usage

This command creates a backup version, or clone, of the indicated read-write zFS file system. It names the new backup version by adding a **.bak** extension to the name of its read-write source file system. It places the backup version on the same aggregate as the read-write version. The aggregate that the read-write file system is contained in must be attached. The read-write file system may or may not be mounted when the clone operation is issued. The backup file system cannot be mounted when the clone operation is issued. After the clone operation, the backup file system can be mounted read-only. The **zfsadm clone** command *cannot* clone non-zFS file systems.

If a backup version already exists, the new clone replaces it. If the read-write file system name is longer than 40 characters, the clone fails.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be **root**.

Examples

The following command creates a backup version of the file system **OMVS.PRV.FS1**:

```
$ zfsadm clone OMVS.PRV.FS1
IOEZ00026I Clonesys starting for aggregate id 100000, file system OMVS.PRV.FS1
IOEZ00031I Clonesys ending for aggregate id 100000 (Total: 0, Failed: 0, Time 0.3)
```

Related Information

Command:

zfsadm clonesys

zfsadm clonesys

Purpose

Creates backup versions of all indicated file systems.

Format

zfsadm clonesys [-prefix *string***] [-aggregate** *name***] [-level] [-help]**

Options

-prefix *string* Specifies a character string of any length. Every file system with a name matching this string is cloned. Include field separators (such as periods) if appropriate. This option can be combined with **-aggregate**. Omit all options to back up all file systems on the system. The prefix name is case sensitive.

-aggregate *name*
Specifies the aggregate name of the aggregate where the read-write source file systems are stored. Omit all options to back up all file systems on the system. The aggregate name is not case sensitive. It is translated to upper case.

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

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

Usage

The **zfsadm clonesys** command creates a backup version, or clone, of each indicated read-write zFS file system. The file systems must be in aggregates that are attached. The read-write file systems may or may not be mounted when the clonesys operation is issued. The backup file systems cannot be mounted when the clonesys operation is issued. The command names each backup version by adding a **.bak** extension to the name of its read-write source file system. It places each backup version in the same aggregate as its read-write version. The **zfsadm clonesys** command *cannot* backup non-zFS file systems.

If a backup version of a file system already exists, the new clone replaces it.

By combining the **-prefix** and **-aggregate** options, you can create backup copies of different subsets of read-write file systems. To back up:

- All file systems on the system, specify no options
- All file systems on the system with a name beginning with the same character string (for example, **sys.** or **user.**), specify the string with the **-prefix** option
- File systems on a specific aggregate on the system, specify the **-aggregate** option
- File systems with a certain prefix on a specific aggregate on the system, specify the **-prefix** and **-aggregate** options.

Use the **zfsadm clone** command to back up a single read-write zFS file system.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file must be **root**.

Examples

The following example creates a backup version of each zFS file system on the system in aggregate OMVS.PRV.AGGR001.LDS0001:

```
$ zfsadm clonesys -aggregate omvs.prv.aggr001.lds0001

IOEZ00026I Clonesys starting for aggregate id 100000, file system * all filesystems *
IOEZ00031I Clonesys ending for aggregate id 100000 (Total: 3, Failed: 0, Time 0.148)
```

Related Information

Command:

zfsadm clone

File:

IOEFSPRM

zfsadm create

Purpose

Creates a read-write zFS file system in an aggregate. This is for multi-file system aggregates only.

Format

```
zfsadm create -filesystem name -aggregate name -size kbytes [-owner name | uid] [-group name | gid]
              [-perms permbits] [-level] [-help]
```

Options

-filesystem *name*

Specifies a name for the read/write file system. The file system name is case sensitive. That is, if you specify the file system name in lower case on the **zfsadm create** command, you must specify the file system name in lower case when you **MOUNT** it. The TSO/E **MOUNT** command translates the file system name to upper case even if it is within quotes. You can avoid this translation to upper case if you specify the file system name on the TSO/E **MOUNT** command within triple quotes. For example, if you specify FILESYSTEM('''lower.case.example'''), the file system name is not translated to upper case. However, you may find it simpler to specify the file system name in upper case on the **zfsadm create** command. The name must be unique within the system, and it should be indicative of the file system's contents. The following characters can be included in the name of a file system:

- All uppercase and lowercase alphabetic characters (a to z, A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name can be no longer than 44 characters. This includes the **.bak** extension, which is added automatically when a backup version of the file system is created (for example, by using **zfsadm clone**). If you intend to clone this file system, you may want to limit the file system name to 40 characters. Note that the **.bak** extension is reserved for use with backup file systems so you cannot specify a file system name that ends with this extension.

-aggregate *name*

Specifies the name of the aggregate where the read-write file system is to be stored. The aggregate name is not case sensitive. It is translated to upper case.

-size *kbytes* Specifies the initial maximum quota for the file system in K-bytes.

-owner *name* **|** *uid*

Specifies the owner of the root directory of the file system. This can be specified as a z/OS user ID or as a numeric *uid*. The default is the *uid* of the issuer of the command.

-group *name* **|** *gid*

Specifies the group of the root directory of the file system. This can be specified as a z/OS group ID or a numeric *gid*. The default is the group of the issuer of the command. If only **-owner** is specified, the group is the owner's default group.

-perms *permbits*

Specifies the permissions for the root directory of the file system. The number can be specified as octal (for example, o755), as hexadecimal (for example, x1ED), or as decimal (for example, 493). The default is o755 (owner read/write/execute, group read/execute, other read/execute).

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

Usage

The **zfsadm create** command creates a read-write zFS file system, names it as specified by **-filesystem**, and places it in the multi-file system aggregate specified by **-aggregate**. The aggregate must be attached. (This is accomplished by issuing the **zfsadm attach** command or by placing a **define_aggr** entry for the aggregate in the **IOEFSPRM** file and starting (or restarting) ZFS.)

If this command succeeds, the file system can be made available for use by **MOUNT**ing it into the z/OS UNIX hierarchy. The command creates an empty root directory in the file system, which becomes visible when the file system is mounted.

Note: Do not create another file system in a compatibility mode aggregate. This will change the compatibility mode aggregate into a multi-file system aggregate.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be **root**.

Examples

The following command creates the read-write file system **OMVS.USER.PAT**, with an initial quota of 5000 1K blocks in aggregate OMVS.PRV.AGGR001.LDS0001.

```
$ zfsadm create OMVS.USER.PAT omvs.prv.aggr001.lds0001 5000

IOEZ00099I File system OMVS.USER.PAT created successfully
```

Related Information

Commands:

zfsadm delete
zfsadm lsfs

File:

IOEFSPRM

zfsadm delete

Purpose

Removes a file system.

Format

zfsadm delete -filesystem *name* **[-level] [-help]**

Options

-filesystem *name*
: Specifies the name of the read-write or backup file system to be removed. Include the **.bak** extension if specifying the name of a backup file system. The file system name is case sensitive.

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

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

Usage

The **zfsadm delete** command removes the read-write or backup zFS file system indicated by the **-filesystem** option from its aggregate. The aggregate containing the file system to be deleted must be attached. Read-write file systems and backup file systems are related during removal as follows:

- Removing a read-write file system automatically removes its associated backup version (if the backup version exists).
- Removing a backup file system does not remove the read-write file system.

If the zFS file system to be removed is also mounted, you must unmount it before you delete it. The **zfsadm delete** command cannot be used to delete a file system that is mounted. You can delete a compatibility mode file system (and its aggregate) by using the IDCAMS DELETE operation. This deletes the VSAM Linear Data Set.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be **root**.

Examples

The following command deletes the read-write file system named **OMVS.USER.PAT** and its backup version (if it exists) from its aggregate:

```
zfsadm delete OMVS.USER.PAT

Destroying file system 100000:OMVS.USER.PAT (100000,,11)
IOEZ00105I File system OMVS.USER.PAT deleted successfully
```

Related Information

Commands:

zfsadm create
zfsadm lsfs

File:

IOEFSPRM

zfsadm detach

Purpose

Detaches one or more aggregates from zFS. This makes any file systems contained in the aggregate unavailable to zFS.

Format

zfsadm detach {-all | -aggregate *name***} [-level] [-help]**

Options

-all Specifies that all attached aggregates are to be detached. Use this option or use **-aggregate** but not both.

-aggregate *name*
Specifies the aggregate name of the aggregate to be detached. Use this option or use **-all**, but not both. The aggregate name is not case sensitive. It is always translated to upper case.

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

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

Usage

The **zfsadm detach** command is used to detach an aggregate. Detaching an aggregate makes it unavailable to the system. To detach one or more aggregates, use the **-all** or the **-aggregate** option to specify the aggregates to be detached.

Before detaching an aggregate, all file systems in the aggregate must be unmounted.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following is an example of a **zfsadm detach** command that detaches the aggregate OMVS.PRV.AGGR001.LDS0001.

```
zfsadm detach -aggregate omvs.prv.aggr001.lds0001

IOEZ00122I Aggregate OMVS.PRV.AGGR001.LDS0001 detached successfully
```

Related Information

Commands:
zfsadm attach

Files:
IOEFSPRM

zfsadm grow

Purpose

Makes the physical size of an aggregate larger.

Format

zfsadm grow -aggregate *name* **-size** ***kbytes*** **[-level] [-help]**

Options

-aggregate *name*
: Specifies the aggregate name of the aggregate to be grown. The aggregate name is not case sensitive. It is always translated to upper case.

-size *kbytes*
: Specifies the new total size in kilobytes of the aggregate after the grow operation. The size is rounded up to a secondary allocation boundary.

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

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

Usage

The **zfsadm grow** command increases the size of an aggregate. The aggregate must be attached. The aggregate is quiesced before the grow and is unquiesced after the grow completes. This means that any current operations against the aggregate are completed and no new operations begin until after the grow completes. Any mounted file systems are quiesced. An aggregate cannot be made smaller than its current size. The current size of an aggregate can be determined by using the **zfsadm aggrinfo** command. A grow operation fails if no secondary allocation is specified for the VSAM Linear Data Set or if no space is available on the volume(s).

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following command displays the online help entry for the **zfsadm grow** command:

```
zfsadm grow -help

Usage: zfsadm grow -aggregate <name> -size <size in K bytes> [-level] [-help]
```

Related Information

Command:

zfsadm aggrinfo

zfsadm help

Purpose

Shows syntax of specified **zfsadm** commands or lists functional descriptions of all **zfsadm** commands.

Format

zfsadm help [-topic *command*...**] [-level] [-help]**

Options

-topic *command*
: Specifies each command whose syntax is to be displayed. Provide only the second part of the command name (for example, **lsfs**, not **zfsadm lsfs**). Multiple topic strings can be specified. If this option is omitted, the output provides a short description of all **zfsadm** commands.

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

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

Usage

The **zfsadm help** command displays the first line (name and short description) of the online help entry for every **zfsadm** command if **-topic** is not provided. For each command name specified with **-topic**, the output lists the entire help entry.

The online help entry for each **zfsadm** 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 **zfsadm apropos** command to show each help entry containing a specified string.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

Examples

The following command displays the online help entry for the **zfsadm lsfs** command and the **zfsadm lsaggr** command:

```
zfsadm help -topic lsfs lsaggr

zfsadm lsfs: list filesystem information
Usage: zfsadm lsfs [-aggregate <aggregate name>] [{-fast | -long}] [-level] [-help]
zfsadm lsaggr: list aggregates
Usage: zfsadm lsaggr [-level] [-help]
```

Related Information

Command:

zfsadm apropos

zfsadm lsaggr

Purpose

Lists all currently attached aggregates for zFS.

Format

```
zfsadm lsaggr [-level] [-help]
```

Options

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

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

Usage

The **zfsadm lsaggr** command displays information about all attached aggregates.

This command displays a separate line for each aggregate. Each line displays the following information:

- The aggregate name
- The aggregate ID number.

You can use the **zfsadm aggrinfo** command to display information about the amount of disk space available on a specific aggregate or on all aggregates on a system.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

Examples

The following example shows that five aggregates are attached to the system:

```
zfsadm lsaggr

OMVS.PRV.AGGR004.LDS0004                    (id=100004)
OMVS.PRV.AGGR003.LDS0002                    (id=100003)
OMVS.PRV.AGGR003.LDS0001                    (id=100002)
OMVS.PRV.AGGR002.LDS0002                    (id=100001)
OMVS.PRV.AGGR001.LDS0001                    (id=100000)
```

Related Information

Command:

zfsadm aggrinfo

File:

IOEFSPRM

zfsadm lsfs

Purpose

Lists all the file systems on a given aggregate or all attached aggregates.

Format

zfsadm lsfs [-aggregate *name* **] [-fast | -long] [-level] [-help]**

Options

-aggregate *name*
: Specifies an aggregate name that is used to retrieve file system information. The aggregate name is not case sensitive. It is always translated to upper case. If this option is not specified, the command displays information for all attached aggregates.

-fast
: Causes the output of the command to be shortened to display only the aggregate name if it contains one or more file systems or a message indicating that there are no file systems contained in the aggregate.

-long
: Causes the output of the command to be extended to five lines for each file system found in an aggregate.

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

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

Usage

The **zfsadm lsfs** command displays information about file systems in an aggregate. The file systems do not need to be mounted to use this command.

The **zfsadm lsfs** command displays the following information for a specified aggregate or all attached aggregates on a system:

- The total number of file systems contained in the aggregate.
- The file system's name (with a **.bak** extension, if appropriate).
- The type (RW for read-write, or BK for backup).
- If it is mounted or not.
- The allocation usage and the quota usage, in kilobytes.
- If the file system is on-line or not.
- The total number of file systems on-line, off-line, busy, and mounted appear at the end of the output for all file systems.

If **-fast** is specified, it only displays the file system names.

If **-long** is specified, the following is displayed:

- The total number of file systems contained in the aggregate.
- The file system's name.
- The file system's ID.
- The type (RW for read-write, or BK for backup).
- If it is mounted or not.
- The state vector of the file system.

- If the file system is on-line or not.
- The allocation limit and allocation usage.
- The quota limit and quota usage.
- The day, date, and time when the file system was created (backed up for a backup file system).
- The day, date, and time when the contents of the file system were last updated (same as the creation time for a backup file system).
- The total number of file systems on-line, off-line, busy and mounted appears at the end of the output for all file systems.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

Examples

The following example displays information for the aggregate **OMVS.PRV.AGGR001.LDS0001**:

```
zfsadm lsfs -aggregate omvs.prv.aggr001.lds0001 -long
IOEZ00129I Total of 2 file systems found for aggregate OMVS.PRV.AGGR001.LDS0001
OMVS.PRV.FS1 100000,,5 RW (Not Mounted)      states 0x10010005 On-line
        Infinite alloc limit;          9 K alloc usage
         25000 K quota limit;          9 K quota usage
    Creation Thu Aug  9 17:17:03 2001
    Last Update Thu Aug  9 17:17:03 2001

OMVS.PRV.FS2 100000,,6 RW (Not Mounted)      states 0x10010005 On-line
        Infinite alloc limit;          9 K alloc usage
         45000 K quota limit;          9 K quota usage
    Creation Thu Aug  9 17:26:54 2001
    Last Update Thu Aug  9 17:26:54 2001

Total file systems on-line 2; total off-line 0; total busy 0; total mounted 0
```

Related Information

Commands:

zfsadm create
zfsadm clone

zfsadm lsquota

Purpose

Shows quota information about file systems and aggregates.

Format

zfsadm lsquota -filesystem *name* **[-level] [-help]**

Options

-filesystem *name*
: Specifies the name of the file system about which quota and usage information is to be displayed. The file system name is case sensitive. If it was specified in upper case when the file system was created, it must be specified in upper case here.

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

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

Usage

The **zfsadm lsquota** command displays quota and usage information about a file system. The command also provides usage information on the aggregate in which the file system resides. The file system does not need to be mounted to use this command. The aggregate containing the file system must be attached.

The **zfsadm lsquota** command displays the name of the file system, the quota and the quota used (in kilobytes) of the file system, and the percentage of the quota in use. It also displays the information about the percentage of the aggregate in use, the number of kilobytes in use on the aggregate and the number of available kilobytes on the aggregate in which the file system resides. It also reports that the file system is zFS.

The size of a compatibility mode file system is equal to the size of the aggregate on which it resides. Therefore, the size and usage information displayed for the aggregate in the output of the **zfsadm lsquota** command equals the quota and quota usage information of the file system in the aggregate.

This command displays the following information about each specified file system:

- The name of the file system.
- The quota, in kilobytes, of the file system.
- The number of kilobytes of the quota currently in use on the file system.
- The percentage of the quota currently in use on the file system.
- The percentage of available disk space currently in use on the aggregate on which the file system resides.
- The number of kilobytes of disk space in use on the aggregate and the total number of kilobytes on the aggregate on which the file system resides.
- The file system type of the aggregate (zFS).

If the file system quota usage rises above 90% or the aggregate usage rises above 97%, the appropriate percentage is indicated with << and the message <<**WARNING** is displayed after the aggregate usage information at the end of the output line. (The 90% and the 97% are not related to the **FSFULL** and **AGGRFULL** options on **MOUNT** and in the **IOEFSPRM** file. Those are used to determine when to report to the operator.)

zfsadm lsquota

Note: Because each compatibility mode aggregate contains a single file system, the information displayed for a compatibility mode aggregate applies to the single file system it houses.

The **zfsadm aggrinfo** command can be used to display the total disk space on an aggregate and the amount currently available.

Every newly created zFS file system has a quota specification. The **zfsadm setquota** command can be used to increase or decrease the quota of a zFS file system. Because the quota of a zFS file system does not represent the amount of physical data space allocated to the file system, it can be larger than the size of the aggregate on which the file system resides. Similarly, the combined quotas of all file systems on an aggregate can be larger than the size of the aggregate. It cannot be changed to smaller than the usage of the file system.

Privilege Required

READ authority to the data set that contains the **IOEFSPRM** file is required.

Examples

The command that follows lists quota and usage information for the file system **OMVS.PRV.FS1**. It also displays the size and usage information for the aggregate that contains this file system.

```
zfsadm lsq OMVS.PRV.FS1

Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.PRV.FS1          25000       9     0     1 = 1891/177992 (zFS)
```

The following command lists quota and usage information for the zFS file system named **OMVS.PRV.AGGR004.LDS0004**, and size and usage information for the aggregate on which the file system resides. The <<**WARNING** message directs the issuer's attention to the fact that the percentage of the quota in use on the indicated file system is above the warning level of 90% or the aggregate usage is above 97%.

```
zfsadm lsq -f OMVS.PRV.AGGR004.LDS0004

Filesys Name              Quota    Used  Percent Used  Aggregate
OMVS.PRV.AGGR004.LDS0004   1300    1266    97<< 100<< = 1412/1412 (zFS)    <<WARNING
```

Related Information

Commands:

- **zfsadm aggrinfo**
- **zfsadm lsfs**
- **zfsadm setquota**

zfsadm quiesce

Purpose

Specifies that an aggregate and all the file systems contained in it should be quiesced.

Format

zfsadm quiesce -aggregate *name* **[-level] [-help]**

Options

-aggregate *name*
: Specifies the name of the aggregate that is to be quiesced. The aggregate name is not case sensitive. It is always translated to upper case. An aggregate must be attached to be quiesced. All current activity against the aggregate is allowed to complete but no new activity is started. Any mounted file systems are quiesced.

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

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

Usage

The **zfsadm quiesce** command is used to temporarily drain activity to the aggregate. During this time:

- No file systems in the aggregate can be created, deleted, renamed, or cloned
- No quotas for file systems contained in the aggregate can be modified
- The aggregate cannot be detached, or grown
- No activity can occur against mounted file systems.

The aggregate can be the target of **lsaggr**, **aggrinfo**, **lsfs** (file systems are indicated as busy).

The aggregate would normally be quiesced prior to backing up the aggregate. After the backup is complete, the aggregate can be unquiesced.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following command quiesces the aggregate **OMVS.PRE.AGGR001.LDS0001**.

```
zfsadm quiesce -aggregate omvs.prv.aggr001.lds0001

IOEZ00163I Aggregate OMVS.PRV.AGGR001.LDS0001 successfully quiesced
```

Related Information

Commands:

zfsadm unquiesce

zfsadm rename

Purpose

Renames a file system.

Format

zfsadm rename -oldname *oldname* **-newname** *newname* **[-level] [-help]**

Options

-oldname *oldname*

Specifies the current name of the read-write file system. It is case sensitive.

-newname *newname*

Specifies the new name for the read-write file system. The name must be unique within the sysplex (or system, if not in a sysplex), and it should be indicative of the file system's contents. The following characters can be included in the name of a file system:

- All uppercase and lowercase alphabetic characters (a to z, A to Z)
- All numerals (0 to 9)
- The . (period)
- The - (dash)
- The _ (underscore).

The name can be no longer than 44 characters. This length includes the **.bak** extension, which is added automatically when a read-only or backup version of the file system is created. If you intend to clone this file system, you may want to limit the file system name to 40 characters. Note that the **.bak** extensions are reserved for use with backup zFS file systems, so you cannot specify a file system name that ends with that extension.

Note: The file system name is case sensitive. That is, if you specify the file system name in lower case as the **-newname** on the **zfsadm rename** command, you must specify the file system name in lower case when you mount it. The TSO/E **MOUNT** command translates the file system name to upper case even if it is within quotes. It is not translated to upper case if you specify the file system name on the TSO/E **MOUNT** command within triple quotes. For example, you can specify FILESYSTEM('''lower.case.example''') and the file system name is not translated to upper case. However, you may find it simpler to specify the file system name in upper case on the **zfsadm rename** command.

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

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

Usage

The **zfsadm rename** command changes the name of the read-write file system specified with **-oldname** to the name specified with **-newname**. The name of the read-write file system's backup copy, if any, automatically changes to match. The aggregate that the file system is contained in must be attached. The file system cannot be mounted.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following command changes the file system name **OMVS.PRV.FS2** to the file system name **OMVS.PRV.FS9**:

```
zfsadm rename -oldname OMVS.PRV.FS2 -newname OMVS.PRV.FS9

IOEZ00108I File system OMVS.PRV.FS2 renamed to OMVS.PRV.FS9
IOEZ00108I File system OMVS.PRV.FS2.bak renamed to OMVS.PRV.FS9.bak
```

Related Information

Commands:

- **zfsadm create**
- **zfsadm clone**

zfsadm setquota

Purpose

Sets the quota for a filesystem.

Format

zfsadm setquota -filesystem *name* **-size** *kbytes* **[-level] [-help]**

Options

-filesystem *name*
: Specifies the file system name of the read-write file system whose quota is to be set. The file system name is case sensitive.

-size *kbytes*
: Specifies the maximum amount of disk space that all of the files and directories in the read-write file system can occupy. This includes files and directories in the read-write version of the file system that are actually pointers to disk blocks in the backup version of the file system. Specify the value in 1-kilobyte blocks. (A value of 1024 kilobytes is 1 megabyte.)

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

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

Usage

The **zfsadm setquota** command sets the quota limit for a read-write zFS file system. (It cannot be used to set the quota for a non-zFS file system or for a backup zFS file system.) The file system whose quota is to be set is indicated by specifying the file system name with the **-filesystem** option.

Quota refers to the amount of disk space occupied by all of the files and directories in the read-write version of the file system. This includes files and directories in the read-write version of the file system that are actually pointers to disk blocks in the backup version of the file system. Do not confuse quota with allocation; the latter identifies the amount of disk space occupied by the data that a file system actually houses; excluding those files and directories that are pointers to disk blocks in the backup version of the file system.

This command increases or decreases a file system's quota to be the number of kilobytes specified with the **-size** option. Because it does not represent the amount of physical data the file system contains, a file system's quota can be larger than the size of the aggregate on which it resides. Similarly, the sum of the quotas of all file systems on an aggregate can exceed the size of the aggregate.

The **zfsadm lsfs** and **zfsadm lsquota** commands display, among other things, the current quota for a file system.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following command sets the quota for the file system named **OMVS.PRV.FS1** to be 15,000 kilobytes:

```
zfsadm setquota -filesystem OMVS.PRV.FS1 -size 15000
```

```
zfsadm lsquota OMVS.PRV.FS1
```

```
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.PRV.FS1          15000       9     0     1 = 1907/177992 (zFS)
```

Related Information

Commands:

zfsadm lsfs
zfsadm lsquota

zfsadm unquiesce

Purpose

Makes an aggregate (and all the file systems contained in the aggregate) available to be accessed.

Format

zfsadm unquiesce -aggregate *name* **[-level] [-help]**

Options

-aggregate *name*
: Specifies the name of the aggregate that is to be unquiesced. The aggregate name is not case sensitive. It is always translated to upper case. An aggregate must be attached to be unquiesced. All current activity against the aggregate is allowed to resume. Any mounted file systems is unquiesced.

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

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

Usage

The **zfsadm unquiesce** command allows activity that has been suspended by **zfsadm quiesce**, to be resumed.

The aggregate would normally be quiesced prior to backing up the aggregate. After the backup is complete, the aggregate can be unquiesced.

Privilege Required

The issuer must have READ authority to the data set that contains the **IOEFSPRM** file and must be logged in as **root**.

Examples

The following command unquiesces the aggregate **OMVS.PRV.AGGR001.LDS0001**.

```
$ zfsadm unquiesce -aggregate omvs.prv.aggr001.lds0001

IOEZ00166I Aggregate OMVS.PRV.AGGR001.LDS0001 successfully unquiesced
```

Related Information

Command:

zfsadm quiesce

Chapter 12. zFS data sets

The following data sets are used during zFS processing.

IOEFSPRM

Purpose

This file lists the processing options for the ZFS PFS and the definitions of the multi-file system aggregates. There is no mandatory information in this file, therefore it is not required. The options all have defaults. Aggregates can all be compatibility mode aggregates (which do not need definitions). Multi-file system aggregates can be attached by using the **zfsadm attach** command. They do not need definitions in IOEFSPRM to be attached using **zfsadm attach**. However, if you need to specify any options (for tuning purposes, for example) or if you want to have any multi-file system aggregates automatically attached when ZFS is started, you need to have an IOEFSPRM file.

The location of the IOEFSPRM file is specified by the IOEZPRM DD statement in the ZFS PROC. The IOEFSPRM file is normally a PDS member, so the IOEZPRM DD might look like the following:

```
//IOEZPRM  DD  DSN=SYS4.PVT.PARMLIB(IOEFSPRM),DISP=SHR
```

If you need to have separate IOEFSPRM files and you want to share the ZFS PROC in a sysplex, you can use a system variable in the ZFS PROC so that it points to different IOEFSPRM files. The IOEZPRM DD might look like the following:

```
//IOEZPRM  DD  DSN=SYS4.PVT.&SYSNAME..PARMLIB(IOEFSPRM),DISP=SHR
```

Your IOEFSPRM file might reside in SYS4.PVT.SY1.PARMLIB(IOEFSPRM) on system SY1; in SYS4.PVT.SY2.PARMLIB(IOEFSPRM) on system SY2; etc.

Any line beginning with # or * is considered a comment. The text in the IOEFSPRM file is case insensitive. Any option or value can be upper or lower case. Blank lines are allowed.

Usage

The following options are used as processing options for the ZFS PFS:

adm_threads

Specifies the number of threads defined to handle pfsctl or mount requests.

Default Value	10
Expected Value	A number greater than 0.
Example	`adm_threads=20`

auto_attach

Controls whether aggregates defined and listed in the **IOEFSPRM** file are attached by default when ZFS is started (or restarted). When the value is on, you can add new multi-file system aggregates (with the define_aggr option) to the IOEFSPRM file and they are attached automatically the next time ZFS is started.

Default Value	On
Expected Value	On or off.
Example	`auto_attach=off`

user_cache_size

Specifies the size, in bytes, of the cache used to contain file data. You can also specify a **fixed** option which indicates that the pages are permanently fixed for performance. Note, the **fixed** option reserves real storage for usage by ZFS only.

Default Value	256M
Expected Value	A positive number. A 'K' or 'M' can be appended to the value to mean kilobytes or megabytes, respectively.
Example	`user_cache_size=64M,fixed`

meta_cache_size

Specifies the size of the cache used to contain metadata. You can also specify a fixed option which indicates that the pages are permanently fixed for performance. Note, the **fixed** option reserves real storage for usage by ZFS only.

Default Value 32M
Expected Value A positive number. A 'K' or 'M' can be appended to the value to mean kilobytes or megabytes, respectively.
Example `meta_cache_size=64M,fixed`

log_cache_size

Specifies the size of the cache used to contain buffers for log file pages. You can also specify a fixed option which indicates that the pages are permanently fixed for performance. Note, the **fixed** option reserves real storage for usage by ZFS only.

Default Value 64M
Expected Value A positive number. A 'K' or 'M' can be appended to the value to mean kilobytes or megabytes, respectively.
Example `log_cache_size=32M,fixed`

sync_interval

Specifies the number of seconds between syncs.

Default Value 30
Expected Value A positive number.
Example `sync_interval=60`

vnode_cache_size

Specifies the size of the internal zFS vnode cache.

Default Value 8192 (will grow if z/OS UNIX needs more than this number)
Expected Value A positive number.
Example `vnode_cache_size=16384`

nbs Controls whether new block security is globally on by default or off by default for any aggregate. New block security refers to the guarantee made when a system crashes.

Default Value Off
Expected Value On or off.
Example `nbs=on`

aggrfull

Specifies the threshold and increment for reporting aggregate full error messages to the operator.

Default Value Off
Expected Value Off or two numbers within parentheses separated by a comma.
Example `aggrfull(85,5)`

fsfull Specifies the threshold and increment for reporting file system quota full error messages to the operator.

Default Value Off
Expected Value Off or two numbers within parentheses separated by a comma.
Example `fsfull(85,5)`

tran_cache_size

Specifies the initial number of transactions in the transaction cache.

Default Value 2000
Expected Value A positive number.
Example `tran_cache_size=4000`

msg_input_dsn

Specifies the name of a data set containing translated zFS messages. It is specified when the installation uses non-English messages. (When you use English messages, you should not specify this option.) It is read when zFS is started (or restarted). Currently, Japanese messages are supported.

Default Value None

Expected Value The name of a data set containing translated zFS messages.
Example `msg_input_dsn=USERA.SIOEMJPN`

The following option is used to define multi-file system aggregates so that they are attached at ZFS start (or restart):

define_aggr
Defines a multi-file system aggregate, its corresponding data set name (which is the same as the aggregate name), and any processing options for that aggregate. Options include:

R/O or R/W **R/O** specifies that the aggregate should be opened in read-only mode. A **R/O** aggregate means that all file systems in the aggregate are read-only and can only be mounted read-only. This allows sharing of an aggregate among multiple systems. **R/W** specifies that the aggregate should be opened in read-write mode. **R/W** is the default.

attach or noattach Indicates whether this aggregate is attached automatically when ZFS is started (or restarted). Aggregates that are not automatically attached must be attached with the **zfsadm attach** command. The default is the global **auto_attach** option (refer to page80).

nbs or nonbs Indicates if new block security algorithms should be used for this aggregate. The default is the global **nbs** option (refer to page 81).

aggrfull Specifies the threshold and increment for reporting aggregate full messages for this aggregate to the operator. The default is the global **aggrfull** option (refer to page 81).

cluster Specifies the VSAM Linear Data Cluster name. This is also the aggregate name. This option is required.

Example
```
define_aggr R/W attach nonbs aggrfull(85,5)
cluster(OMVS.PRV.AGGR0001.LDS0001)
```

The following options are used during debugging of the ZFS PFS:

debug_settings_dsn
Specifies the name of a data set containing debug classes to enable when zFS starts up. It is read when zFS is started (or restarted).
Default Value None
Expected Value The name of a data set containing debug classes to enable.
Example `debug_settings_dsn=USERA.ZFS.DEBUG.INPUT(FILE1)`

trace_dsn
Contains the output of any operator **MODIFY ZFS,TRACE,PRINT** commands or the trace output if the ZFS PFS abends. Each trace output creates a member in the PDSE. Traces that come from the ZFS PFS kernel have member names of ZFSKNTnn. nn starts with 01 and increments for each trace output. nn is reset to 01 when ZFS is started (or restarted). Refer to “Chapter 9. Performance and debugging” on page 29. This is not a required parameter.
Default Value None
Expected Value The name of a PDSE data set.
Example `trace_dsn=USERA.ZFS.TRACE.OUT`

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 ZFS address space that is used for internal tracing that is always on. The trace can be sent to the **trace_dsn** by using the operator **MODIFY ZFS,TRACE,PRINT** command.
Default Value 512K
Expected Value A positive number. A ’K’ or ’M’ can be appended to the value to mean kilobytes or megabytes, respectively.

Example `trace_table_size=1M`

storage_details

Indicates whether or not the ZFS internal storage tracking mechanisms are active. The results can be sent to the **storage_details_dsn** with the operator **MODIFY ZFS,QUERY,STORAGE,DETAILS** command.

Default Value Off
Expected Value On or off.
Example `storage_details=on`

storage_details_dsn

Indicates where the storage map is written if **storage_details** is on and the operator **MODIFY ZFS,QUERY,STORAGE,DETAILS** command is run.

Default Value None
Expected Value The name of a data set.
Example `storage_details_dsn=usera.zfs.storage.output(file1)`

Examples

The following IOEFSPRM sample file lists every program option and includes some sample multi-file system aggregate definitions.

```
+ + + + Beginning of sample file + + + +

************************************************************************
* zSeries File System (zFS) Sample Parameter File:  ioefsprm
* For a description of these and other zFS parameters, refer to the
* zSeries File System Administration, SC24-5989.
* Notes:
*  1. The ioefsprm file and parameters in the file are optional but it
*     is recommended that the parameter file be created in order to be
*     referenced by the DDNAME=IOEZPRM statement the PROCLIB JCL for
*     the zFS started task.
*  2. An asterisk in column 1 identifies a comment line.
*  3. A parameter specification must begin in column 1.
************************************************************************
* The following msg_input_dsn parameter is ONLY required if the optional
* NLS feature (e.g J0H232J) is installed. The parameter specifies the
* message input data set containing the NLS message text which is
* supplied by the NLS feature. If this parameter is not specified or if
* the data set is not found, English language messages will be generated
* by zFS. You must delete the * from a line to activate the parameter.
************************************************************************
* msg_input_dsn=data.set.name
************************************************************************
* The following are examples of some of the optional parameters that
* control the sizes of caches, tuning options, and program operation.
* You must delete the * from a line to activate a parameter.
************************************************************************
*adm_threads=5
*auto_attach=ON
*user_cache_size=256M
*log_cache_size=12M
*sync_interval=45
*vnode_cache_size=5000
*nbs=off
*fsfull(85,5)
*aggrfull(90,5)
************************************************************************
* The following are examples of some of the options that control zFS
* debug facilities. These parameters are not required for normal
* operation and should only be specified on the recommendation of IBM.
* You must delete the * column from a line to activate a parameter.
************************************************************************
*trace_dsn=data.set.name
```

IOEFSPRM

```
*debug_settings_dsn=data.set.name(membername)
*trace_table_size=32M
*storage_details=ON
*storage_details_dsn=data.set.name
```

Part 3. Appendixes

Appendix. Notices

This information was developed for products and services offered in the USA. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing , to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT ANY 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road

Poughkeepsie, NY 12601-5400
USA
Attention: Information Request

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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains sample application programs in source language, which illustrates 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. 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 IBM application programming interfaces.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This *Distributed File Service zSeries File System Administration* documents information that is NOT intended to be used as Programming Interfaces of Distributed File Service.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

BookManager
IBM
IBMLink
OS/390
Resource Link
z/OS

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through The Open Group.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

This bibliography lists and provides a brief description of each publication in the z/OS Distributed File Service library.

- *z/OS: Distributed File Service DFS Administration*, SC24-5915

This book introduces the Distributed File Service concepts to system and network administrators and provides an in-depth understanding of the Distributed File Service, it uses and benefits. This book also provides reference information for the commands and files used by system and network administrators to work with the Distributed File Service.

- *z/OS: Distributed File Service Customization*, SC24-5916

This book helps system and network administrators configure the Distributed File Service.

- *z/OS: Distributed File Service Messages and Codes*, SC24-5917

This book provides detailed explanations and recovery actions for the messages, status codes, and exception codes issued by the Distributed File Service.

- *z/OS: Distributed File Service SMB Administration*, SC24-5918

This book provides guidance and reference information for system and network administrators to use when they work with the Server Message Block (SMB) support of the Distributed File Service base element of z/OS. SMB is a protocol for remote file/print access used by Windows.

Index

Special Characters

A

B

C

D

E

F

G

Program Number: 5694-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC24-5989-00

Spine information:

z/OS

Distributed File Service zFS Administration