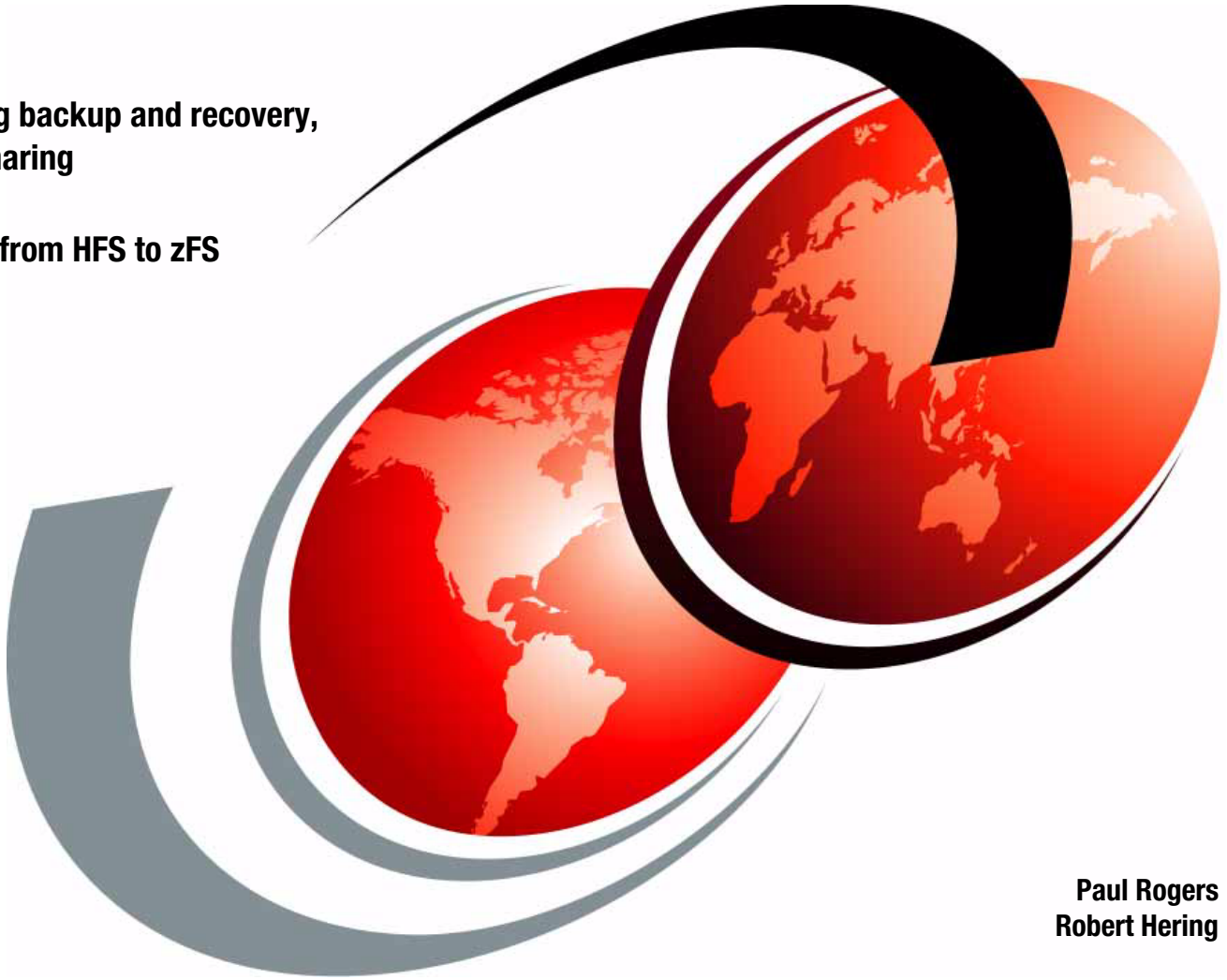
IBM

z/OS Distributed File Service zSeries File System Implementation z/OS V1R13

- Defining and installing a zSeries file system
- Performing backup and recovery, sysplex sharing
- Migrating from HFS to zFS



Paul Rogers
Robert Hering

Redbooks

ibm.com/redbooks

International Technical Support Organization

z/OS Distributed File Service zSeries File System Implementation z/OS V1R13

October 2012

SG24-6580-05

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

Sixth Edition (October 2012)

This edition applies to version 1 release 13 modification 0 of IBM z/OS (product number 5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

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

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

Domino®
IBM®
iNotes®
MVS™
OS/390®
RACF®
Redbooks®
Redpaper™
Redbooks (logo) ®
RMF™
S/390®
z/OS®
zSeries®

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Preface

The z/OS® Distributed File Service zSeries® File System (zFS) is a z/OS UNIX file system that can be used like the Hierarchical File System (HFS). zFS file systems contain files and directories, including Access Control Lists (ACLs), that can be accessed with the z/OS HFS 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, and so on). zFS does not replace HFS, but it is the z/OS UNIX strategic file system and IBM® recommends migrating HFS file systems to zFS. Beginning with z/OS V1R7, there are no restrictions for file system structures that should be kept as HFS instead of zFS.

This IBM Redbooks® publication helps you to install, tailor, and configure new zFS file systems. This information can be used by system administrators who work with the zFS component of the IBM z/OS Distributed File Service base element.

The book provides a broad description of the new architecture of the zFS file system. You can use it as a reference when converting HFS file systems to zFS file systems. It will help you to create a solution for migrating to zFS file systems, and to understand the performance differences between HFS file systems and zFS file systems.

The team who wrote this book

This book was produced by a team working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, JES3, Infoprint Server, and z/OS UNIX. Before joining the ITSO 23 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England providing OS/390® and JES support for IBM EMEA and the Washington Systems Center in Gaithersburg, Maryland.

Robert Hering is an IT Specialist at the ITS Technical Support Center, Mainz, Germany. He advises clients about z/OS and UNIX System Services-related questions and problems. He has participated in several ITSO residencies since 1988, writing about UNIX-related topics. Before providing support on OS/390 and z/OS, he worked with VM and all its different flavors (VM/370, VM/HPO, VM/XA, and VM/ESA®) for many years.

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

Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

Richard Conway
International Technical Support Organization, Poughkeepsie Center

Joe Bostian
IBM Poughkeepsie

All previous editions of this book were produced by the following specialists working at the International Technical Support Organization, Poughkeepsie Center:

Paul Rogers

Robert Hering

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an email to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

zFS file systems

The z/OS® Distributed File Service zSeries® File System (zFS) is a z/OS UNIX System Services (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 z/OS UNIX application programming interfaces (APIs). These file systems can support access control lists (ACLs). 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 and NFS).

This chapter introduces zFS file systems and describes the following topics:

- zFS overview
- Application programming interfaces
- zFS physical file system
- zFS colony address space
- zFS file system aggregates
- Metadata cache
- zFS file system clones
- zFS logs
- zFS recovery

1.1 zSeries File System introduction

The z/OS Distributed File Service (DFS) zSeries File System (zFS) is a z/OS UNIX file system that can be used instead of or in addition to the Hierarchical File System (HFS). zFS provides significant performance gains in accessing files approaching 8 K in size that are frequently accessed and updated. The access performance of smaller files is equivalent to that of HFS.

zFS provides reduced exposure to loss of updates by writing data blocks asynchronously and not waiting for a sync interval. zFS is a so-called “journaling” file system. It logs metadata updates and if a system failure occurs, zFS replays the log when it comes back up to ensure that the file system is consistent.

zFS is a Physical File System (PFS) that is started normally by z/OS UNIX System Services during IPL. A physical file system is the part of the operating system that handles the actual storage and manipulation of data on a storage medium.

1.2 Application programming interfaces

zFS file systems contain files and directories that can be accessed with the z/OS hierarchical file system application programming interfaces (APIs) on the z/OS operating system as follows:

- An application interface composed of C interfaces, some of which are managed within the C Run-Time Library (RTL), and others that access kernel interfaces to perform authorized system functions on behalf of the unauthorized caller
- An interactive z/OS shell interface by shell users

The PFS interface is a set of protocols and calling interfaces between the logical file system (LFS) and the PFSs that are installed on z/OS UNIX, as shown in Figure 1-1. In a UNIX System Services environment, UNIX programs and UNIX users access their files through these interfaces. PFSs mount and unmount file systems and perform other file operations.

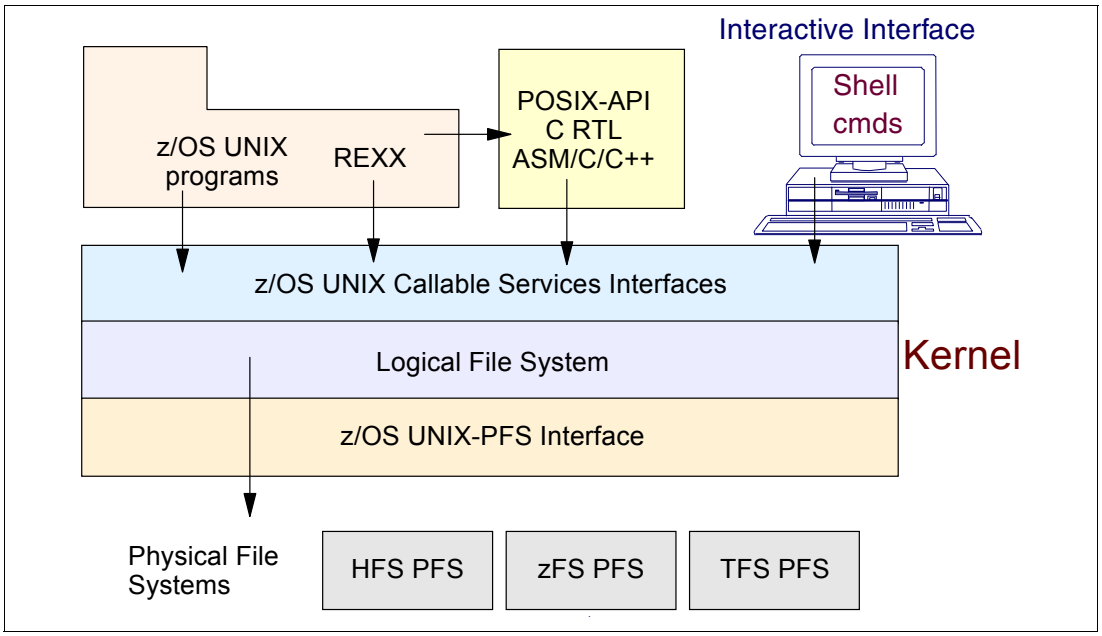


Figure 1-1 z/OS UNIX System Services

1.3 zFS physical file system

z/OS UNIX System Services (z/OS UNIX) allows you to install virtual file system servers (VFS servers) and PFSs.

A VFS server makes requests for file system services on behalf of a client. It is similar to a POSIX program that reads and writes files, except that it uses the lower-level VFS callable services API instead of the POSIX C-language API. An example of a VFS server is the Network File System (NFS). The corresponding client NFS PFS is shown in Figure 1-2.

A PFS controls access to data. PFSs receive and act upon requests to read and write files that they control. The format of these requests is defined by the PFS interface.

In a UNIX System Services environment, the physical file systems are defined in the BPXPRMxx PARMLIB member. zFS, as a physical file system, is also to be defined in the PARMLIB member. Figure 1-2 shows all the physical file systems that can be defined in a UNIX System Services environment.

The logical file system (LFS) is called by POSIX programs, non-POSIX z/OS UNIX programs, and VFS servers.

The PFS interface is a set of protocols and calling interfaces between the LFS and the PFSs that are installed on z/OS UNIX. PFSs mount and unmount file systems and perform other file operations.

There are two types of PFSs, those that manage files and those that manage sockets:

- File management PFSs, such as HFS and zFS, deal with objects that have path names and that generally follow the semantics of POSIX files.
- Socket PFSs deal with objects that are created by the socket() and accept() functions and that follow socket semantics.

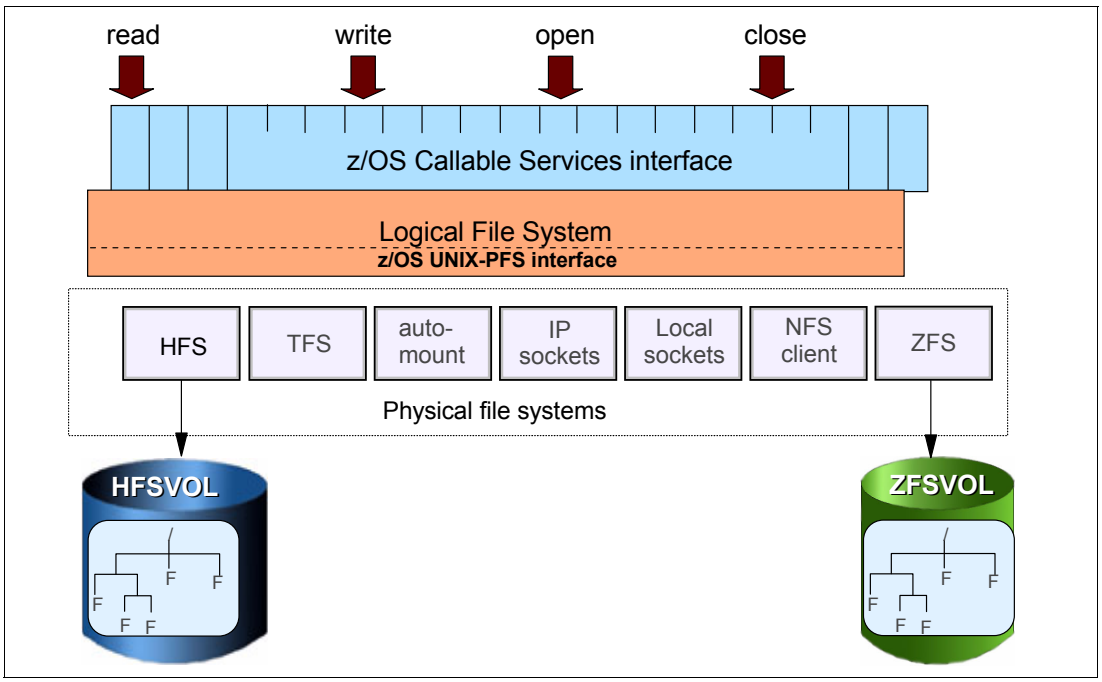


Figure 1-2 UNIX System Services physical file systems

zFS does not replace HFS; it can be considered to be complementary to HFS.

1.4 zFS colony address space

zFS runs in a UNIX System Services colony address space. A *colony address space* is an address space that is separate from the UNIX System Services address space. HFS runs inside the UNIX System Services address space and zFS runs in its own address space, as shown in Figure 1-3.

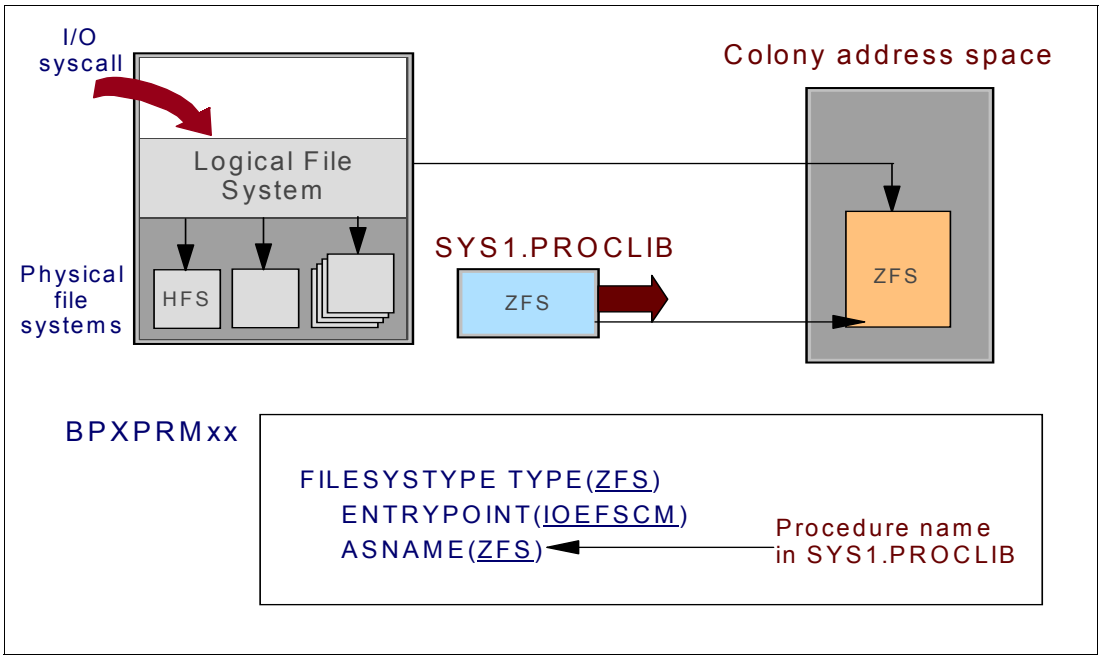


Figure 1-3 zFS executes in a colony address space

1.5 zFS supports z/OS UNIX ACLs

To provide better granularity of access control for z/OS UNIX files and directories, access control lists were introduced with z/OS V1R3. You can use access control lists (ACLs) to control access to files and directories by individual UIDs and GIDs. This provides the means to allow specific users and groups to have access to a file or directory.

To manage an ACL for a file, you must have one of the following security access controls:

- Be the file owner
- Have superuser authority (UID=0)
- Have READ access to SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class

Beginning with z/OS V1R3, ACLs are supported by the HFS and zFS file systems. You must also know whether your security product supports ACLs and what rules are used when determining file access.

ACL support works similar to the way access to MVS™ data sets is permitted, although the implementation is different. The ACL is a part of the File Security Packet (FSP), which is maintained by the PFS.

1.6 zFS file system aggregates

A zFS aggregate is a data set that contains zFS file systems. The aggregate is a VSAM Linear Data Set (VSAM LDS) and is a container that can contain one or more zFS file systems. An aggregate can only have one VSAM LDS, but it can contain an unlimited number of file systems. The name of the aggregate is the same as the VSAM LDS name.

Sufficient space must be available on the volume or volumes, as multiple volumes may be specified on the DEFINE of the VSAM LDS. DFSMS decides when to allocate on these volumes during any extension of a primary allocation. VSAM LDSs greater than 4 GB may be specified by using the extended format and extended addressability capability in the data class of the data set. For more information about allocating VSAM data sets for zFS aggregates, see 2.6.1, “zFS aggregate definitions” on page 22.

After the aggregate is created, formatting of the aggregate is necessary before any file systems can exist in it. A zFS file system is a named entity that resides in a zFS aggregate. It contains a root directory and can be mounted into the UNIX System Services hierarchy. Although the term “file system” is not a new term, a zFS file system resides in a zFS aggregate, which is different from an HFS file system.

zFS aggregates come in two types:

- Compatibility mode aggregates
- Multifile system aggregates

1.6.1 Compatibility mode aggregates

A compatibility mode aggregate can contain only one zFS file system, making this type of aggregate more like an HFS file system. The name of the file system is the same as the name of the aggregate, which is the same as the VSAM LDS cluster name. The file system size (called a quota) in a compatibility mode aggregate is set to the size of the aggregate. Compatibility mode aggregates are more like an HFS data set, except that they are VSAM linear data sets instead of HFS data sets. We recommend that you start using compatibility mode aggregates first, because they are more like the familiar HFS data sets. Figure 1-4 shows a compatibility mode aggregate.

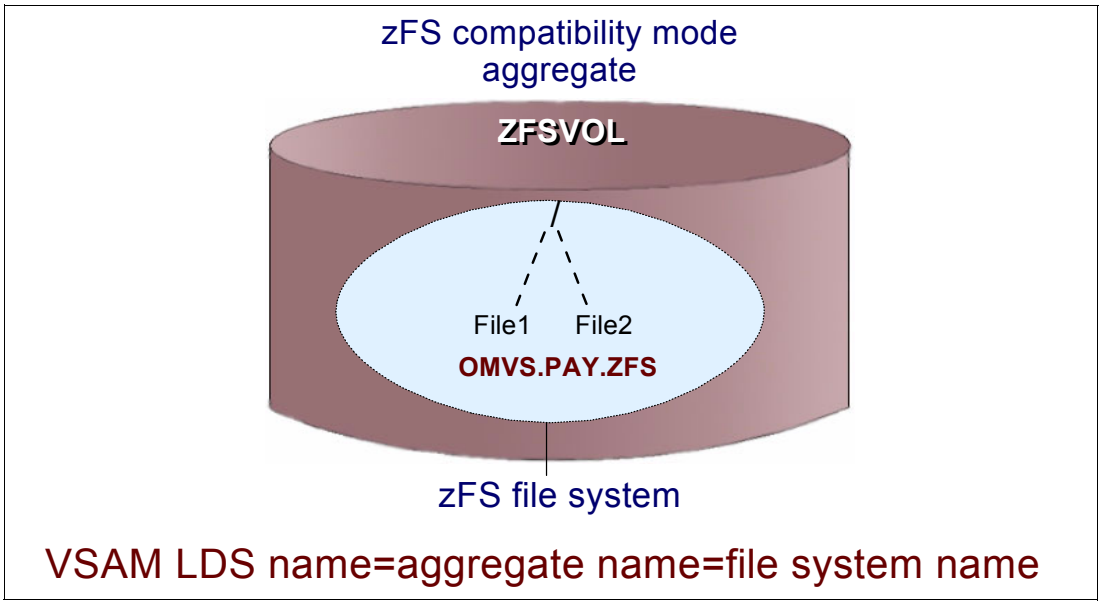


Figure 1-4 Compatibility mode aggregate

1.6.2 Multifile system aggregates

A multifile system aggregate allows the administrator to define multiple zFS file systems in a single aggregate. This allows *space sharing*.

Space sharing

Space sharing means that if you have multiple file systems in a single data set, and files are removed from one of the file systems, which frees DASD space, another file system can use that space when new files are created. This new type of file system is called a “multifile system aggregate”.

The multiple file system aggregate OMVS.MUL02.ZFS, shown in Figure 1-5, can contain multiple zFS file systems. This makes it possible to do space sharing between the zFS file systems within the aggregate.

The multiple file system aggregate has its own name. This name is assigned when the aggregate is created. It is always the same as the VSAM LDS cluster name. Each zFS file system in the aggregate has its own file system name. This name is assigned when the particular file system in the aggregate is created. Each zFS file system also has a predefined maximum size, called the *quota*.

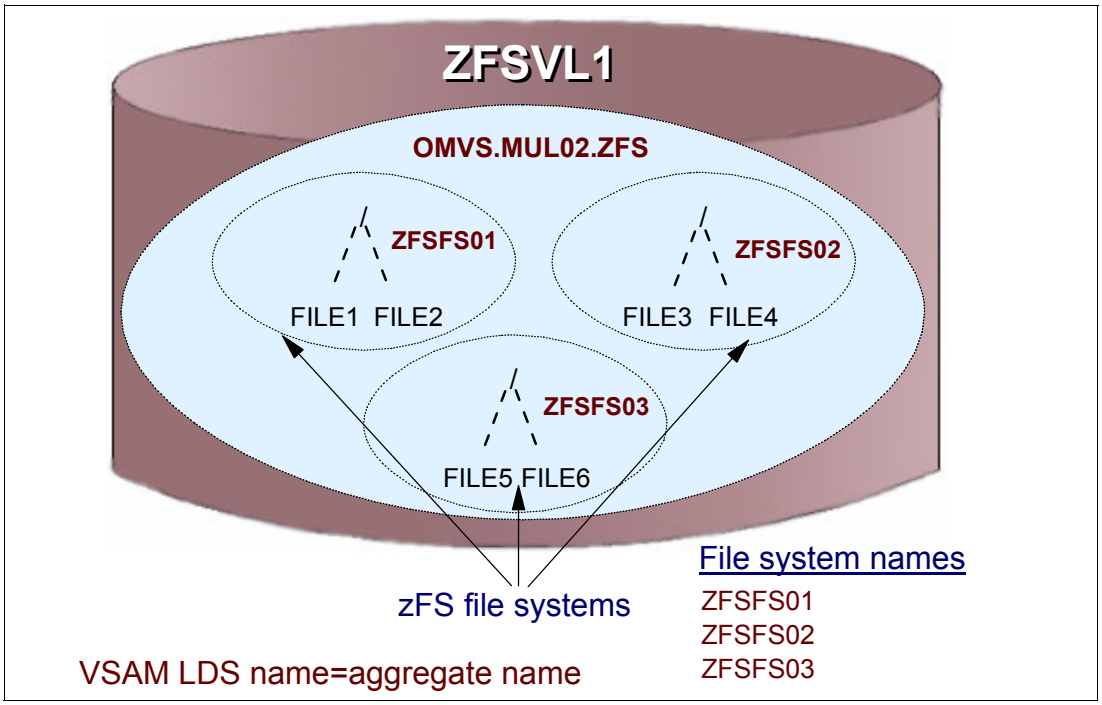


Figure 1-5 Multifile system aggregate

Important: In 1.11, “zFS statement of direction” on page 9 it is mentioned that IBM plans to withdraw support for zFS multifile system aggregates with the next release after z/OS V1R13. When this support is withdrawn, only zFS compatibility mode aggregates will be supported.

“Removal of support for multifile system aggregates” on page 204 provides details about the fact that in a UNIX System Services sysplex file system sharing environment, z/OS V1R7 is the last release to allow mounting zFS file systems contained in multifile system aggregates,

and that z/OS V1R10 is planned to be the last release to allow attaching a multifile system aggregate.

1.7 Metadata cache

The zFS file system has a cache for file system metadata, which includes directory contents and the data of files that are smaller than the aggregate block size. The setting of this cache size is important to performance because zFS references the file system metadata frequently. Synchronous reads of metadata increase I/O rates to disk and server response times. Metadata consists of things like owner, permission bit settings, and data block pointers.

The metadata cache is stored in the zFS primary address space; its default size is 32 MB. Because the metadata cache only contains metadata and small files, it normally does not need to be nearly as large as the user file cache.

Note: With z/OS V1R4, a new backing cache for metadata provides an extension to the meta cache and resides in a data space. Specify the following in the IOEFSPRM file:

```
metaback_cache_size=64M,fixed
```

The values allowed are 1 MB to 2048 MB. It is used as a “paging” area for metadata and allows a larger meta cache for workloads that need large amounts of metadata. This cache is only needed if the meta cache is constrained

1.8 zFS file system clones

zFS allows an administrator to make a read-only clone of a file system in the same aggregate. 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 is copied.

When a file system is cloned, a copy of it is created in the same aggregate, as shown in Figure 1-6 on page 8. There must be physical space available in the aggregate for the clone to be successful. For the clone to be used, it must be mounted.

Important: In 1.11, “zFS statement of direction” on page 9 it is mentioned that IBM plans to withdraw support for zFS clones with the next release after z/OS V1R13. When this support is withdrawn, you cannot use this function any longer.

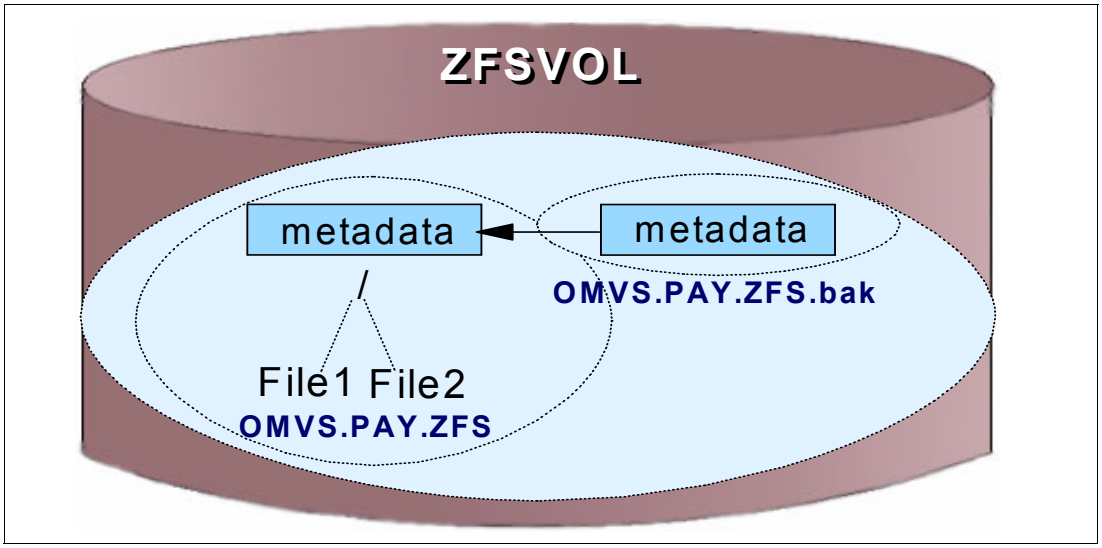


Figure 1-6 zFS file system clone

Note: Because the clone has to be mounted, this creates two mounted file systems in the compatibility mode aggregate. Therefore, in a sysplex sharing environment the clone cannot participate in an AUTOMOVE and must be mounted NOAUTOMOVE. See “Mount the clone” on page 171 for more information about this topic.

1.8.1 Backup file system

The zFS file system that is the result of the clone operation is called the backup file system. The backup file system is a read-only file system and can only be mounted as read-only. For additional information about cloning, see 4.1, “Backup file system” on page 170.

1.9 zFS log files

Every zFS aggregate contains a log file that is created when the aggregate is formatted. This log is used to record transactions describing changes to the file system structure. The default for the log file is 1% of the aggregate size, but it can be changed during the format of the aggregate. Usually, 1% is sufficient for most aggregates. However, very large aggregates might need less than 1%. Very small aggregates might need more than 1% if a high degree of parallel update activity occurs for the aggregate.

1.10 zFS recovery

zFS provides a recovery mechanism that uses a zFS file system log to verify or correct the structure of an aggregate. This recovery mechanism can also be invoked by a utility program named **IOEAGSLV**.

When you perform a system restart, a recovery program known as the *salvager* uses the zFS file system log to return consistency to a file system by running recovery on the aggregate on which the file system resides. Recovery consists of reading the log that contains all the changes made to metadata as a result of the operations done to the aggregate, 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 damaged, then the IOEAGSLV utility can be used to verify or repair the structure of the aggregate.

Note: For information about how to implement the recovery options, see Chapter 4, “Backup and recovery” on page 169.

1.11 zFS statement of direction

Please take into account the following important “Statement of Direction” in R13.

Important: z/OS V1R13 is planned to be the last release to support multifile system zSeries File System (zFS) aggregates, including zFS clones. Support for the **`zfsadm`** clone command and mount support for zFS file system data sets containing a cloned (.bak) file system will be removed. IBM recommends that you use copy functions such as pax and DFSMSdss to back up z/OS UNIX file systems to separate file systems. Support for zFS compatibility mode aggregates will remain.

- So, zFS multifile system aggregates are (finally) being removed.
- Also, the zFS clone function is being removed.

1.12 zFS sample scenarios

All sample scenarios and command sequences provided in this book always depend on the local environment and its actual release and service level when they were performed.

Important: If you replay scenarios described in this book within your own system environment, some results could be different.

1.13 Layout and history of this zFS book

Based on the long history of this book, please note the following statement.

Note: This will be the last version of the zFS book that contains the complete history of zFS. Next time it is intended to remove everything that is no longer supported, superseded or not really important anymore.

2

Installing and using zFS

This chapter describes the installation and customization of zFS file systems and discusses the following topics:

- Installing zFS
- Customization steps for the Distributed File Service
- Customization steps for zFS
- zFS RACF definitions
- BPXPRMxx definitions
- zFS procedure
- Allocating zFS aggregates
- Formatting zFS aggregates
- zFS configuration file
- Attaching an aggregate
- Create a zFS file system
- zFS file system mounts
- Accessing zFS files
- Increasing the size of an aggregate
- Accessing zFS files

2.1 Installing zFS

zFS, as part of the Distributed File Service base element, can be customized without using the DFS Client/Server or SMB File/Print Server components of the element. The zFS function became available for the first time with z/OS Version 1 Release 2 (V1R2). For the enablement of the zFS file system for General Availability with V1R2, install the following APARs/PTFs:

- OW50850 / UW82925
- OW51563 / UW83377

The support for zFS on Releases OS/390® V2R10 and z/OS V1R1 is provided by APAR OW51780.

2.1.1 Changes to zFS since the initial release

The initial release of zFS was at the z/OS V1R2 level and the initial release of this book applied to Version 1 Release 3 of z/OS Distributed File Service for zFS.

zFS and z/OS V1R3

The changes to zFS for z/OS V1R3 are as follows:

- zFS file systems support UNIX ACLs. See 2.15.1, “Access control lists (ACLs)” on page 79.
- MOUNT commands for zFS file systems can now be placed in BPXPRMxx.
- Use the UNIXPRIV class profile, SUPERUSER.FILESYS.PFSCTL, to protect the use of **`zfsadm`** commands.
- Use data spaces for user data cache.

zFS and z/OS V1R4

The changes to zFS for z/OS V1R4 are as follows:

- Dynamic configuration of the IOESPRM member.
- The -grow option.
- Metadata backing cache.
- Log file cache.
- zFS mounts.
- Configuration changes can be made to zFS without stopping and restarting zFS.
- A zFS aggregate can be dynamically extended.
- Different zFS file systems can have the same name if they are in different multifile system aggregates.
- System symbols can be used in the IOEFSPRM file.
- There are several new pfsctl APIs.
- Creation of zFS aggregates using the ISHELL.

zFS and z/OS V1R5

The changes to zFS for z/OS V1R5 are as follows:

- zFS multilevel security support - In a multilevel-secure environment, you must use zFS file systems.

- Listing of information about zFS aggregates and file systems using the ISHELL.
- Dynamic creation of zFS file systems with automount.
- Using zFS and HFS file systems in parallel in the same automount policy.

zFS and z/OS V1R6

The changes to zFS for z/OS V1R6 are as follows:

- Logical parmlib search.
- Enhancements in abend handling and for hang conditions.
- Performance monitoring APIs.
- New LFS support for zFS on PFS termination in UNIX System Services sysplex sharing.

zFS and z/OS V1R7

The changes to zFS for z/OS V1R7 are as follows:

- With completion of zFS EOM support, zFS is now the preferred z/OS UNIX file system.
- Sysplex awareness of APIs and the zfsadm command interface.
- Various new **`zfsadm`** commands and options.
- Removal of zFS aggregate and file system name restrictions.
- Further enhancements to LFS support for zFS.
- New directory cache parameter.
- New performance monitoring APIs.

zFS and z/OS V1R8

The changes to zFS for z/OS V1R8 are as follows:

- UNIX System Services LFS support for zFS shutdown.
- Support of **`bpxmtext`** command for zFS reason codes (xEFxxnnnn).

zFS and z/OS V1R9

The changes to zFS for z/OS V1R9 are as follows:

- zFS auditid uniqueness.
- zFS read-only mount recovery.
- IOEAGFMT and IOEAGSLV authorization enhancements.

zFS and z/OS V1R10

The changes to zFS for z/OS V1R10 are as follows:

- Tool for migrating the sysplex root file system to zFS
- Man pages for **`zfsadm`** commands
- zFS and z/OS V1R11
- R11 ZFS Migration Health Checks
- zFS Admin support
- zFS Sysplex file support‘

2.2 Customization steps for the Distributed File Service

Our installation was done based on a ServerPac. With the Distributed File Service installed, if you only require zFS, the post-installation and configuration steps for the DFS client, DFS server, and the SMB server are not required to be performed. Also, the DFS client, DFS server, or SMB server do not need to be started to run zFS.

Note: The installation of DCE on the system is not required for zFS.

We ensured that the load library for the Distributed File Service is in the link list and is APF-authorized. Make sure that the symbolic link for the **`zfsadm`** shell command has been created (see Figure 2-1 on page 16).

Note: In this book, we use IOE as the hlq for all DFS data sets.

The library is therefore named IOE.SIOELMOD and all the samples supplied are located in IOE.SIOESAMP.

2.3 Customization steps for zFS

To begin using the zFS file system, you must install the z/OS Distributed File Service base element because it is required for zFS. If this element is already installed and configured on your system, then you only need to customize the new zFS file system.

zFS executables

zFS executables consisting of six PDS load modules must reside in an APF-authorized library and there is one APF-authorized HFS executable.

The new zFS-specific executables in IOE.SIOELMOD are:

IOEZM001 Aggregate format utility program (alias is IOEAGFMT)

IOEZM002 Aggregate salvage utility program (alias is IOEAGSLV)

IOEZM003 Removed in z/OS 1.3 (alias was IOEFSADM).

IOEZM004 Initialization and administration request handler and also primary file request handler, which runs in the zFS colony address space (alias is IOEFSCM)

IOEZM005 An administration command interface (alias is IOEZADM). There is also an APF-authorized HFS executable named zfsadm that can be used to perform the same functions.

IOEZM006 zFS trace formatting program for use under the direction of IBM in a client installation (alias is IOETRFMT).

IOEZM007 This module is new with z/OS 1.3 and runs in the zFS colony address space. It deals with “worker threads” (alias is IOEFSTHD). The worker threads are used to handle mount, unmount, and pfsctl.

2.3.1 Customization utilities and commands

zFS provides utility programs and z/OS UNIX commands to assist in the customization of the aggregates and file systems. These utilities and commands are to be used by system administrators. Examples of their use are shown in the following chapters of this book. For

more detailed information about options and usage, see *z/OS Distributed File Service zSeries File System Administration,* SC24-5989.

zFS commands

The **zfsadm** command and the IOEZADM utility program can be used to manage file systems and aggregates.

The **zfsadm** command can be run as a UNIX shell command from:

- A z/OS UNIX system services shell (OMVS) or a (z/OS UNIX) telnet session
- A batch job using the BPXBATCH utility program
- TSO foreground or in batch mode using z/OS UNIX APIs (SYSCALL commands, Callable Services)

Both the command and the utility program require the zFS colony address space to be up and running.

zFS file systems can be created using JCL, shown in Figure 2-14 on page 27, or by using the **zfsadm** command. Table 2-1 shows the **zfsadm** command with its subcommands.

Table 2-1 The zfsadm command and its subcommands

Command/subcommand	Command description	IOEFSPRM	SU mode
zfsadm aggrinfo	Obtain information about attached aggregate	Read	No
zfsadm apropos	Display first line of help entry	Read	No
zfsadm attach	Attach an aggregate	Read	Yes
zfsadm clone	Clone a file system	Read	Yes
zfsadm clonesys	Clone multiple file systems	Read	Yes
zfsadm create	Create a file system	Read	Yes
zfsadm define R3	Define a VSAM linear data set	Read	No
zfsadm delete	Delete a file system	Read	Yes
zfsadm detach	Detach an aggregate	Read	Yes
zfsadm format R3	Format a VSAM LDS as an aggregate	Alter	Yes
zfsadm grow	Grow an aggregate	Read	Yes
zfsadm help	Get help on commands	Read	No
zfsadm lsaggr	List all currently attached aggregates	Read	No
zfsadm lsfs	List all file systems on a aggregate or all	Read	No
zfsadm lsquota	Show quotas for file systems and aggregates	Read	No
zfsadm lssys R7	Shows the members in a sysplex	Read	No
zfsadm quiesce	Quiesce an aggregate and all file systems	Read	Yes
zfsadm rename	Rename a file system	Read	Yes
zfsadm setquota	Set the quota for a file system	Read	Yes
zfsadm unquiesce	Make aggregates and file systems available	Read	Yes

Command/subcommand	Command description	IOEFSPRM	SU mode
zfsadm config R4	Modify current configuration options	Read	Yes
zfsadm configquery R4	Display current configuration options	Read	No
zfsadm query R6	Query or reset the performance counters	Read	No

Note: The **zfsadm define** and **zfsadm format** commands are new with z/OS V1R3, **zfsadm config** and **zfsadm configquery** are new with z/OS V1R4, **zfsadm query** is new with z/OS V1R6, **zfsadm lssys** is new with z/OS V1R7.

To issue the commands shown in Table 2-1 on page 15, you must have RACF® authorization, as follows:

IOEFSPRM Command issuer must have access to a RACF data set profile to the IOEFSPRM data set.

SU mode Command issuer must have superuser authority.

Important: See 2.4.1, “Authority for zFS commands” on page 18 and 2.4.2, “UNIXPRIV class and BPX.SUPERUSER profile” on page 18 for the authorization needed to issue these commands.

Symbolic link for the zfsadm command

Verify that during your z/OS Distributed File Service installation a symbolic link was created for access to the **zfsadm** command. If for some reason this was not done, issue the following command to create a symbolic link in the /bin directory:

```
#> ln -s /usr/lpp/dfs/global/bin/zfsadm /bin/zfsadm
```

Figure 2-1 Creating the symbolic link for the zfsadm command

Attention: Whenever you see commands starting with #>, it means the command is run from a UNIX shell environment on superuser mode. A prompt string of $> indicates that the UID is not 0.

zFS utility programs

The following utility programs are provided:

IOEAGFMT A utility program to format an aggregate.

IOEAGSLV A utility program to scan an aggregate and report inconsistencies.

IOEZADM A utility program that allows **zfsadm** commands to be issued using JCL. It is also supported to be run in a TSO/E environment.

Note: Although zfsadm and IOEZADM are physically different modules, they contain identical code. Whenever we reference zfsadm as a zFS administration command in this book, we mean both **zfsadm** and IOEZADM. Therefore, IOEZADM can be used as a TSO/E command; it performs the same functions as the **zfsadm** command.

2.3.2 Steps needed to define file systems

Perform the following steps to define zFS file systems and then run zFS as a z/OS UNIX physical file system:

1. Create RACF definitions.
2. Define zFS to UNIX System Services with the BPXPRMxx parmlib statement using the FILESYSTYPE statement.

   Customize the zFS procedure.
3. Allocate a zFS aggregate.
4. Format a zFS aggregate.
5. Define the zFS parameter file for configuration options for aggregates - IOEFSPRM.
6. Attach a zFS multifile system aggregate (multifile systems only).

   Use the **zfsadm** command, or the IOEFSPRM member, or both in combination.
7. Create a zFS file system in an aggregate.
8. Mount a zFS file system in the z/OS UNIX directory hierarchy.
   a. Mount compatibility mode aggregates.

      Use the TSO MOUNT command, /usr/sbin/mount shell command, or the automount facility.
   b. Mount a zFS file system in a multifile system aggregate.

      Create a directory mount point; use the MOUNT command.
9. Access a zFS file system in the same way an HFS file system is accessed.

2.4 zFS RACF definitions

To define DFS to RACF you must create the following definitions with these exact names. Even if you use none of the functions of DFS, you can use the following DFS definitions for zFS:

- Define DFSGRP as a group.
- Define DFS as a user.
- Define zFS as a started task.

For RACF, use the commands shown in Figure 2-2.

```
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 ZFS.** STDATA(USER(DFS) GROUP(DFSGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

Figure 2-2 RACF definitions for zFS

Note: A user ID other than DFS can be used to run the zFS started task if it is defined with the same RACF characteristics shown for the DFS user ID.

2.4.1 Authority for zFS commands

To be able to use the zFS commands and utilities, users must be authorized with either READ or ALTER access via RACF profiles to the IOEFSPRM data set for certain commands. The access required for each command is shown in Table 2-1 on page 15.

2.4.2 UNIXPRIV class and BPX.SUPERUSER profile

To be able to issue the **zfsadm** commands shown in Table 2-1 on page 15, you may choose to assign a UID of 0 to multiple RACF user IDs. However, if you choose to minimize the assignment of superuser authority in your installation, you can accomplish this by managing superuser privileges by the following:

- Providing access to the FACILITY profile BPX.SUPERUSER, which allows a user to switch to superuser mode when this is needed
- UNIXPRIV profiles

BPX.SUPERUSER profiles

To make use of this profile, it must be available and the user must be permitted READ access, as shown in Figure 2-3.

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(HERING) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 2-3 BPX.SUPERUSER profile

UNIXPRIV profiles

zFS with z/OS V1R3 supports the SUPERUSER.FILESYS.PFSCTL profile of the UNIXPRIV class. This makes it possible for a zFS administrator to have just READ authority to this UNIXPRIV profile resource, SUPERUSER.FILESYS.PFSCTL, rather than requiring a UID of 0 for **zfsadm** commands that modify zFS file systems or aggregates. The same is true for the other zFS commands and utilities. Therefore, zFS administrators do not need a UID of 0.

To allow the zFS administrator to mount and unmount file systems, permit update access to another profile, SUPERUSER.FILESYS.MOUNT, in class UNIXPRIV.

Note: UPDATE access is needed if the user needs to **mount**, **chmount**, or **unmount** file systems with the setuid option; otherwise, READ access is sufficient.

UNIXPRIV authorization is invoked by creating the needed resources in the UNIXPRIV class and then giving users READ authority to it, as shown in Figure 2-4.

```
SETROPTS CLASSACT(UNIXPRIV)
SETROPTS RACLIST(UNIXPRIV)
RDEFINE UNIXPRIV SUPERUSER.FILESYS.PFSCTL UACC(NONE)
PERMIT SUPERUSER.FILESYS.PFSCTL CLASS(UNIXPRIV) ID(ROGERS) ACCESS(READ)
RDEFINE UNIXPRIV SUPERUSER.FILESYS.MOUNT UACC(NONE)
PERMIT SUPERUSER.FILESYS.MOUNT CLASS(UNIXPRIV) ID(ROGERS) ACCESS(UPDATE)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

Figure 2-4 UNIXPRIV RACF profiles

Other UNIXPRIV profiles

If a user also needs to perform other administration tasks and is not a superuser or BPX.SUPERUSER, it may be necessary to provide access to other UNIXPRIV profiles, as shown in Table 2-2.

Table 2-2 UNIXPRIV profile access control

Task	Profile	Access
User needs read and write access to any local file and directory	SUPERUSER.FILESYS	CONTROL
User needs to be able to change the permission bits of any file or directory	SUPERUSER.FILESYS.CHANGEPERMS	READ
User needs to be able to change the ownership of any file or directory	SUPERUSER.FILESYS.CHOWN	READ
User needs to be able to issue quiesce and unquiesce commands to any file system	SUPERUSER.FILESYS.QUIESCE	UPDATE

Note: You can define a generic profile called SUPERUSER.FILESYS.** to assign all file system privileges. This is allowed for resources in the UNIXPRIV class.

2.4.3 Access to the zFS configuration file

We need to define the data set containing the zFS configuration and parameter file to RACF. We are using a PDS called ZFS.SC43.IOEFSZFS, as shown in Figure 2-5.

```
ADDSD 'ZFS.SC43.IOEFSZFS' GENERIC UACC(NONE)
PERMIT 'ZFS.SC43.IOEFSZFS' CLASS(DATASET) GENERIC ID(DFSGRP) ACCESS(READ)
PERMIT 'ZFS.SC43.IOEFSZFS' CLASS(DATASET) GENERIC ID(HERING) ACCESS(ALTER)
PERMIT 'ZFS.SC43.IOEFSZFS' CLASS(DATASET) GENERIC ID(ROGERS) ACCESS(ALTER)
```

Figure 2-5 Defining the IOEFSPRM PDS data set to RACF

See 2.8, “zFS configuration file” on page 34 for more information about IOEFSPRM.

2.5 BPXPRMxx definitions

For zFS, you can specify the following statements in the z/OS UNIX BPXPRMxx member:

- Define the colony address space using the FILESYSTYPE statement.
- Beginning with z/OS V1R3, you can specify MOUNT statements to mount zFS file systems during the IPL or restart of OMVS.

2.5.1 zFS colony address space

Figure 2-6 shows a sample FILESYSTYPE statement for the zFS physical file system.

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

Figure 2-6 zFS FILESYSTYPE statement

It includes an ASNAME parameter specifying the zFS procedure to be started. It is required for PFSes that run in a colony address space.

zFS colony address space

zFS runs in a z/OS UNIX colony address space. As previously mentioned, a colony address space is an address space that is separate from the z/OS UNIX address space, as shown in Figure 2-7. HFS runs inside the z/OS UNIX address space.

Whether or not a PFS runs in a colony address space is controlled by the ASNAME parameter of the FILESYSTYPE statement for the PFS in the BPXPRMxx member of the SYS1.PARMLIB concatenation.

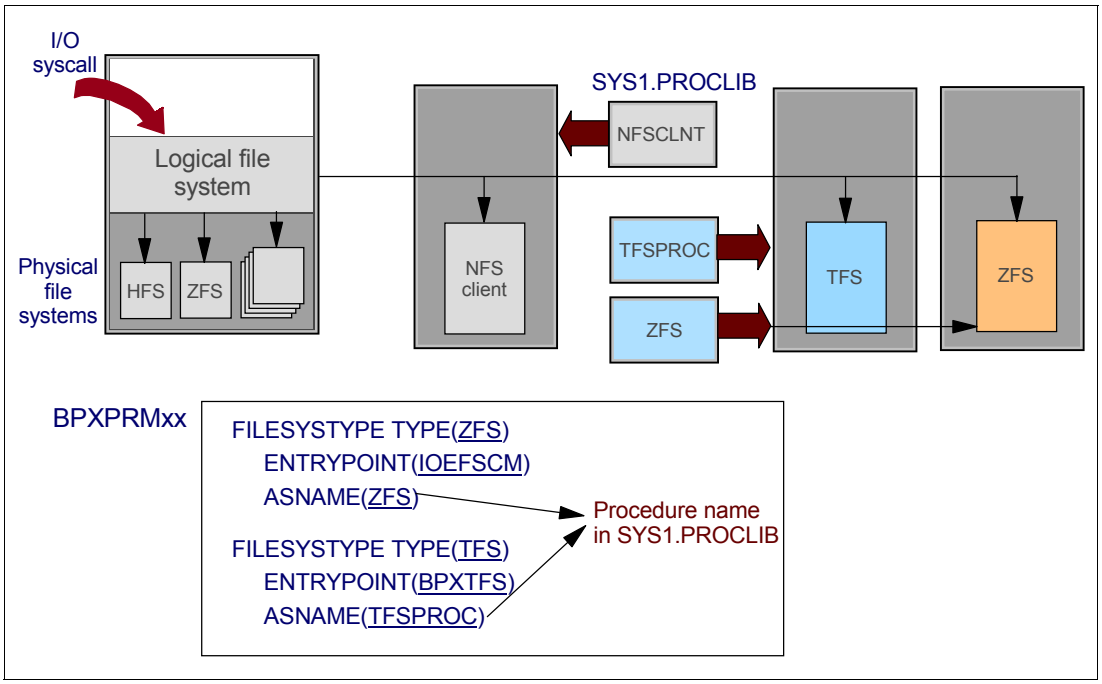


Figure 2-7 zFS executes in a colony address space

2.5.2 zFS procedure

The following JCL is required to start zFS. The file system runs in a colony address space. The JCL must be in a PROCLIB member, as shown in Figure 2-8 on page 21, with the member name that is referenced by the ASNAME parameter the BPXPRMxx FILESYSTYPE for the zFS, as shown in 2.5, “BPXPRMxx definitions” on page 19.

You can copy the zFS sample procedure, shown in Figure 2-8 on page 21, from IOE.SIOEPROC to SYS1.PROCLIB. You can also copy member IOEFSPRM from IOE.SIOESAMP to SYS1.PARMLIB, and then modify the zFS procedure accordingly.

IOEZPRM DD statement

The IOEZPRM DD statement can be omitted from the JCL procedure, or the data set can exist with no parameters. It can be a sequential data set or a PDS member.

```
//ZFS      PROC REGSIZE=0M
//*
//ZFZGO    EXEC PGM=BPXVCLNY,REGION=&REGSIZE,TIME=1440
//*
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD              <--ZFS LOADLIB
//IOEZPRM  DD DISP=SHR,                              <--ZFS PARM FILE
//            DSN=ZFS.&SYSNAME..IOEFSZFS(IOEFSPRM)
//*
```

Figure 2-8 zFS proclib member to start zFS

Although the zFS parameter file defined in the JCL DD statement IOEZPRM is optional, we recommend that you use it during testing to identify the parameter file in use when testing zFS.

Note: In z/OS V1R6, the zFS parameter file can be specified in any data set of the logical parmlib. See 2.8.4, “Logical parmlib search” on page 36.

2.5.3 Starting zFS

The first time that we started zFS, the filesystype statement for zFS had not been in the BPXPRMxx parmlib member during OMVS initialization. We put the statement (see Figure 2-10) in the BPXPRMxx member and also into a separate parmlib member, BPXPRMZS. For immediate activation we then used the SETOMVS RESET command, as shown in Figure 2-9).

```
SETOMVS RESET=(ZS)
 IEE252I MEMBER BPXPRMZS FOUND IN SYS1.PARMLIB
 ...
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
 Version 01.03.00 Service Level 0W53579.
 Created on Wed Feb 27 12:02:18 EST 2002.
 IOEZ00178I ZFS.SC43.IOEFSZFS(IOEFSPRM) is the
 configuration dataset currently in use.
 ...
 IOEZ00055I zFS kernel: initialization complete.
 BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
```

Figure 2-9 Dynamically starting zFS

Note: The MVS system command D OMVS,PFS can be used to verify that the physical file system zFS is active.

2.5.4 Colony address space outside of JES control

This release of z/OS UNIX allows a Physical File System (PFS) client to be started outside of the control of JES. This allows for a planned shutdown of individual systems in a shared HFS sysplex using NFS. You can start zFS outside of the control of JES; see Figure 2-10.

```
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS,'SUB=MSTR')
```

Figure 2-10 zFS colony address space outside of JES control

This support can be installed on previous releases with APAR OW48709, as follows:

- OS/390 V2R9 - UW82164
- OS/390 V2R10 - UW82162
- z/OS V1R2 - UW82163

APAR OW48709

In APAR OW48709, the capability was added to UNIX System Services to start its colony address spaces outside of JES control and to specify any additional START command parameters that may be useful. The PFSs that support this are the NFS client, zFS file system, and TFS file system. At this time, the DFS client does not support starting its colony address spaces outside of JES control.

In the procedures for the zFS and NFS file systems, the DD SYSOUT= statements must be replaced with either DD DSN= (keep the information in the MVS data set); DD PATH= (keep the information in the UNIX System Services file system); or DD DUMMY (throw the information away).

2.6 Allocating zFS aggregates

A multifile system or compatibility mode aggregate must be defined before a zFS file system can be created in the aggregate.

A zFS aggregate is created by defining a VSAM Linear Data Set (LDS) and then formatting it as an aggregate. This is done once for each zFS aggregate. You cannot assign more than one VSAM LDS per aggregate. An aggregate can contain one or more zFS file systems. A zFS file system is equivalent to an HFS file system.

2.6.1 zFS aggregate definitions

Aggregates are defined as VSAM linear data sets. VSAM data sets must be cataloged. VSAM data sets can be defined using the following options:

- Access method services ALLOCATE or DEFINE CLUSTER commands
- ISPF option 3.2
- **zfsadm** command: **zfsadm define**

IDCAMS must be used to define an LDS; the above options all use IDCAMS.

Linear data sets

A linear data set contains data that can be accessed as byte-addressable strings in virtual storage. It is a VSAM data set with a control interval (CI) size of 4096 bytes. An LDS has no imbedded control information in its CI, that is, no RDFs and CIDFs. All LDS bytes are data bytes. Logical records must be blocked and deblocked by the application program, but records do not exist from the point of view of VSAM.

For z/OS UNIX use, the VSAM data set must be linear. Therefore, allocations done by IDCAMS end up in space allocated in tracks, cylinders, or records. Specifying space to IDCAMS in bytes is permitted.

Note: You can specify space explicitly for VSAM linear data sets in units of records, tracks, cylinders, or megabytes. To maintain device independence, specify records. The amount of space you allocate depends on the size of your file system and the index options you select.

When a linear data set is defined, the catalog forces the block size to 4096 bytes. For VSAM data sets in general you can specify a control interval size of 4096 to 32,768 bytes in increments of 4096 bytes. There are further rules regarding how specified values are rounded. Using a variable value for the CI size is not allowed for zFS aggregates; otherwise, you receive an error message when trying to start the format processing.

Attention: For zFS aggregates the CI size must be 4 K, which is also the default if not specified on the IDCAMS DEFINE command.

zFS aggregates and VSAM LDS

As mentioned, the CI size of a zFS aggregate is always 4 K. zFS always reads and writes an 8 K block (two CIs) at a time. zFS can write multiple small files into an 8 K block by writing fragments. Fragments are always 1 k bytes. Thus, two 1 k files could be contained in a single 8 k block. However, up to eight 1 K files can fit into a single 8 K block. Also, a 7 K file and a 1 K file can fit into a single 8 K block. If the file is larger than 7 K, it takes up one or more full 8 K blocks.

A zFS aggregate can contain up to 4 GB 1K blocks for a maximum of 4 TB.

Restriction: zFS does not support the use of a striped VSAM Linear Data Set as a zFS aggregate. If you attempt to mount a compatibility mode file system that had previously been formatted and is a striped VSAM LDS, it will only mount as read-only.

2.6.2 Allocating an aggregate

The VSAM LDS is allocated with the VSAM utility program IDCAMS, as shown in Figure 2-11 on page 24. Allocation is done the same way for both types of aggregates.

Note: Multiple volumes may be specified when you define the aggregate.

Aggregate and file system naming

When creating aggregates and file systems, the choice of names is extremely important. This is especially true if you choose to use an automount policy to mount your zFS file systems. Some rules for choosing names are:

- zFS file system names are case sensitive. The name specified when you create a file system using the **`zfsadm create`** command is not folded to uppercase.
- Do not name any file systems in multifile system aggregates with the same name as any of your compatibility mode aggregates.
- Choose a file system name that allows all file system names to be used in a single automount policy map file. For example, suppose your map file has the file system as:

  ```
  OMVS.<uc_name.ZFS
  ```

  Then all your file system names should use this format to be eligible for automount, including the aggregate name for a compatibility mode aggregate, because that becomes the file system name; see Figure 2-62 on page 52.

  Note: A file system name can be a single qualifier, such as ZFSFS01.

```
//ZFSJOB   JOB ,'ZFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//DASDO    DD     DISP=OLD,UNIT=3390,VOL=SER=TOTZF1
//SYSIN    DD     *
  DEFINE CLUSTER (NAME(OMVS.MUL01.ZFS) VOLUMES(TOTZF1)  -
   LINEAR MEGABYTES(20 10) SHAREOPTIONS(3))
...
//*
```

Figure 2-11 Defining aggregates, JCL part one created from the sample named IOEAGFMT

Figure 2-12 shows the resulting job output.

```
IDCAMS  SYSTEM SERVICES

  DEFINE CLUSTER (NAME(OMVS.MUL01.ZFS) VOLUMES(TOTZF1)  -
   LINEAR MEGABYTES(20 10) SHAREOPTIONS(3))
IDC0508I DATA ALLOCATION STATUS FOR VOLUME TOTZF1 IS 0
IDC0512I NAME GENERATED-(D) OMVS.MUL01.ZFS.DATA
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 2-12 Output from an aggregate definition

Secondary allocations

You can allocate space for a multivolume data set the same as for a single volume data set. DASD space is initially allocated on the first volume only. When the primary allocation of space is filled, space is allocated in secondary storage amounts (if specified, as shown in Figure 2-11). The extents can be allocated on other volumes.

Suggestion: The secondary allocation is important for when you want to grow an aggregate in size. See 2.13, “Increasing the physical size of an aggregate” on page 60 for an example of making use of the secondary extent value.

2.6.3 Multiple volume aggregates

Figure 2-13 shows a sample of how to define a multiple volume aggregate without using SMS from the TSO foreground command line.

```
ioezadm define -aggregate OMVS.MUL02.ZFS -volumes TOTZF1 TOTZF2 -megabytes 20 10
```

Figure 2-13 Multiple volume aggregate allocation

If a guaranteed space storage class (STORAGECLASS parameter) is assigned to the data set and volume serial numbers are specified, primary space is allocated on all specified volumes if the following conditions are met:

- All volumes specified belong to the same storage group.
- The storage group to which these volumes belong is in the list of storage groups selected by the ACS routines for this allocation.

Secondary allocations

When the primary amount on the first volume is used up, a secondary allocation amount is allocated on that volume. Each time a new record does not fit in the allocated space, the system allocates more space in the secondary space amount. This can be repeated until the volume is out of space or the extent limit is reached.

When you allocate space for your data set, you can specify both a primary and a secondary allocation. A primary space allocation is the initial amount of space allocated. A secondary allocation is the amount of space allocated each time space is used up. It is possible for the storage administrator to specify whether to use primary or secondary allocation amounts when extending to a new volume (only valid for non-striped VSAM data sets). This is done with an SMS data class parameter for VSAM attributes.

You can expand the space for a VSAM data set defined in a catalog to a maximum of 251 to 255 extents. The system reserves the last four extents for extending a data set when the system cannot allocate the last extent in one piece.

2.6.4 Managing space on volumes

The VSAM LDS used for a zFS aggregate can be defined to multiple volumes. If the VSAM LDS is extended to additional volumes after the aggregate is initially formatted, the aggregate must be formatted to extend to the additional volume using the **`zfsadm grow`** command. 2.13, “Increasing the physical size of an aggregate” on page 60 describes a sample of how to use the command.

Important: zFS aggregates can be greater than 4 GB. You can create an aggregate greater than 4 GB only by using an extended format and addressability in the data class of the data set. Therefore, these aggregates need to be SMS-managed.

2.7 Formatting zFS aggregates

The VSAM linear data set must be formatted to be used as a zFS aggregate. Two options are available for formatting an aggregate:

- The IOEAGFMT format utility
- The **`zfsadm`** command

Both options use the same parameters, as follows:

Format parameters

```
zfsadm format -aggregate name [-initialempty blocks] [-size blocks] [-logsize
blocks] [-overwrite] [-compat] [-owner {uid | name}] [-group {group_id | name}]
[-perms decimal | octal | hex_number] [-level] [-help]
```

-aggregate name This specifies the aggregate name of the aggregate to be formatted. This becomes the name of the zFS aggregate that is formatted.

-initialempty blocks This specifies the number of 8 K blocks that are left empty at the beginning of the aggregate.

Default=1. This option is normally not specified.

We highly recommend that you use a value of 1 or do not use this parameter; take the default. You should never use a value of zero (0).

-size blocks This specifies the number of 8 K blocks that should be formatted to form the zFS aggregate.

The default is the number of blocks that fit in the primary allocation of the VSAM LDS. If a number less than the default is specified, it is rounded up to the default. If a number greater than the default is specified, a single extend of the VSAM LDS is attempted after the primary allocation is formatted. Space must be available on the volume.

-logsize blocks This specifies the number of 8 K blocks reserved for the aggregate log.

Default=1% of the size of the aggregate. This is normally sufficient.

-overwrite This specifies that an existing zFS aggregate should be overlaid and all existing data is lost.

Use this option with caution because this is not usually specified.

-compat This specifies that the zFS aggregate should be formatted as a compatibility mode aggregate and should be always be used, because multifile system aggregates are no longer suggested to be used.

-owner {uid|name} This specifies the owner of the root directory of the file system.

The default is the uid of the issuer of the **`zfsadm format`** command. Use this option with the **`-compat`** option, otherwise it is ignored. It may be specified as a z/OS user ID or as a uid.

-group {uid|name} This specifies the group owner of the root directory of the file system.

The default is the gid of the issuer of the **`zfsadm format`** command. This is used with the **`-compat`** option, otherwise it is ignored. It may be specified as a z/OS group ID or as a gid. If only `owner` is specified, the group is that owner's default group.

-perms number This specifies the permissions of the root directory of the file system.

The default= 0755. This is used with the **`-compat`** option; otherwise it is ignored. Specify as follows: **`-perms 0755`**, **`-perms x1ED`**, or **`-perms 493`**. For more information, see "Format parameter options" on page 26.

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

Format parameter options

The zFS aggregate format parameters are shown in "Format parameters" on page 25. The following parameter options should be considered when formatting the aggregates, as they are significantly different than when defining HFS file systems.

-perms With HFS, when file systems are created, the default permission bit settings are 700. With zFS compatibility mode file systems, this default is 755. You have the opportunity to set your installation settings during the format of the aggregate for compatibility mode aggregates.

-owner With HFS, when file systems are created, the owner is the user ID of the creator.

-group With zFS compatibility mode file systems, you can specify an `owner` or `group` during the create, which occurs during the end of format processing and as shown in Figure 2-17 on page 28.

-size After you have allocated the space for an aggregate, the default size is the number of 8 K blocks that fit into the primary allocation. You can specify a `-size` option giving the number of 8 K blocks for the aggregate. If you specify a number that is less than (or equal to) the number of blocks that fit into the primary allocation, the primary allocation size is used. If you specify a number that is larger than the number of 8 K blocks that fit into the primary allocation, the VSAM LDS is extended to the size specified. This occurs during its initial formatting.

2.7.1 Formatting using the IOEAGFMT utility

The zFS product provides this formatting utility, which can be used to format an HFS compatibility mode aggregate or a multifile system aggregate.

This utility formats the primary allocation and, if requested by using the **-size** parameter (with a value greater than the primary allocation), a single *extension*. An extension is a single call to the Media Manager to extend the data set to the size specified in the **-size** parameter. It is completely independent of the number of cylinders specified in the secondary allocation when the data set was defined (the secondary allocation could have been 0).

If your initial VSAM LDS allocation is for multiple volumes, the initial formatting of the zFS aggregate is limited to the primary allocation (in this case, the first volume), and one extension (in this case, the second volume). After that, you must use multiple **zfsadm grow** commands to grow to each new volume.

Formatting a multifile system aggregate

Figure 2-14 shows the JCL for the format of a multifile aggregate.

```
//ZFSJOB   JOB,'ZFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//* Format VSAM Linear Data Set as ZFS Multiple File System Aggregate
//FORMAT   EXEC   PGM=IOEAGFMT,REGION=0M,
//         PARM=('-aggregate OMVS.MUL01.ZFS')
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD    <--LOADLIB FOR ZFS
//SYSPRINT DD   SYSOUT=*
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CEEDUMP  DD   SYSOUT=*
//*
```

Figure 2-14 Formatting aggregates using the IOEAGFMT utility

Note: The sample JCL was run using all the defaults. See “Format parameters” on page 25 for the default values.

Figure 2-15 on page 28 shows the resulting job output.

```
IOEZ00004I Loading dataset 'OMVS.MUL01.ZFS'.
IOEZ00005I Dataset 'OMVS.MUL01.ZFS' loaded successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 2609
*** Defaulting to 26 log blocks(maximum of 2 concurrent transactions).
Done.  /dev/lfs1/OMVS.MUL01.ZFS is now a zFS aggregate.
```

Figure 2-15 Output from aggregate formatting

Formatting a compatibility mode aggregate

When you format a compatibility mode aggregate (the JCL shown in Figure 2-16), the processing shown in Figure 2-17 is performed, as explained here:

- The aggregate is attached, allowing creation of a file system.
- A file system is created for the aggregate with the same name as the aggregate.

```
//ZFSJOB   JOB ,'ZFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//* Format VSAM Linear Data Set as ZFS Multiple File System Aggregate
//FORMAT   EXEC   PGM=IOEAGFMT,REGION=0M,
//         PARM=('-aggregate omvs.cmp01.zfs -compat')
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD    <--LOADLIB FOR ZFS
//SYSPRINT DD   SYSOUT=*
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CEEDUMP  DD   SYSOUT=*
//*
```

Figure 2-16 Formatting a compatibility mode aggregate using the IOEAGFMT utility

Figure 2-17 shows the output of the job.

```
IOEZ00004I Loading dataset 'omvs.cmp01.zfs'.
IOEZ00005I Dataset 'omvs.cmp01.zfs' loaded successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 17999
*** Defaulting to 179 log blocks(maximum of 19 concurrent transactions).
Done.  /dev/lfs1/omvs.cmp01.zfs is now an zFS aggregate.
IOEZ00071I Attaching aggregate OMVS.CMP01.ZFS to create hfs-compatible file system
IOEZ00074I Creating file system of size 140487K, owner id 0, group id 0, permissions
x1ED
IOEZ00048I Detaching aggregate OMVS.CMP01.ZFS
IOEZ00077I HFS-compatibility aggregate OMVS.CMP01.ZFS has been successfully created
```

Figure 2-17 Messages from the format of a compatibility mode aggregate

Important: The IOEAGFMT utility can be run to format an aggregate even if the zFS PFS is not active on the system.

2.7.2 Formatting using the zfsadm command

Figure 2-18 shows a **zfsadm format** command example of how to define, format, and mount a compatibility mode aggregate from the OMVS shell command line. Defining the aggregate does not require the issuing user to be a superuser. However, formatting and mounting the aggregate requires superuser authority, as shown by the $> and #> prompt characters in Figure 2-18.

```
$> zfsadm define -a OMVS.CMP01.ZFS -volumes totzf1 -cylinders 10 1
IOEZ00248E VSAM linear dataset OMVS.CMP01.ZFS successfully created.
$> su
#> zfsadm format -a OMVS.CMP01.ZFS -compat -owner 316 -p o755
IOEZ00077I HFS-compatibility aggregate OMVS.CMP01.ZFS has been successfully created
#> /usr/sbin/mount -f OMVS.CMP01.ZFS -t ZFS /u/hering/test
```

Figure 2-18 Defining, formatting, and mounting a compatibility mode aggregate

Note: The **zfsadm define** command is new starting with z/OS V1R3.

Formatting with previous releases

The **zfsadm format** command is not available on releases prior to z/OS V1R3. Therefore, if you want to use a command to format aggregates, you can do the following:

- To use the IOEAGFMT utility from the z/OS UNIX shell, you must define an external link to IOEAGFMT in the z/OS UNIX file structure, as follows:

  ```
  #> ln -e IOEAGFMT /u/hering/bin/zfsformat
  ```

  If the specified directory, /u/hering/bin, is part of the PATH chain, you can use the new command to format aggregates in superuser mode from a z/OS UNIX shell, as shown in Figure 2-19.

```
#> export PATH=/u/hering/bin:/bin
#> tso -t "DEFINE CLUSTER (NAME('OMVS.SC43.COMPTEST.ZFS') VOLUMES(TOTZF1) LINEAR
CYLINDERS(1 0) SHAREOPTIONS(3))"
DEFINE CLUSTER (NAME('OMVS.SC43.COMPTEST.ZFS') VOLUMES(TOTZF1) LINEAR CYLINDERS(1 0)
SHAREOPTIONS(3))
IDC0508I DATA ALLOCATION STATUS FOR VOLUME TOTZF1 IS 0
IDC0512I NAME GENERATED-(D) OMVS.SC43.COMPTEST.ZFS.DATA
#> zfsformat -aggregate OMVS.SC43.COMPTEST.ZFS -size 360 -compat -owner 888 -perms 755
IOEZ00004I Loading dataset 'OMVS.SC43.COMPTEST.ZFS'.
IOEZ00005I Dataset 'OMVS.SC43.COMPTEST.ZFS' loaded successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 360
*** Defaulting to 13 log blocks(maximum of 1 concurrent transactions).
Done.  /dev/lfs1/OMVS.SC43.COMPTEST.ZFS is now a zFS aggregate.
IOEZ00071I Attaching aggregate OMVS.SC43.COMPTEST.ZFS to create hfs-compatible file
system
IOEZ00074I Creating file system of size 5039K, owner id 888, group id 0, permissions
x2F3
IOEZ00048I Detaching aggregate OMVS.SC43.COMPTEST.ZFS
IOEZ00077I HFS-compatibility aggregate OMVS.SC43.COMPTEST.ZFS has been successfully
created
```

Figure 2-19 Defining and formatting an aggregate using an external link

In this example, the extension was a single 3-cylinder extent because the total size specified was 360 and the primary allocation was 1 cylinder (1 cylinder = 90 8 K blocks; 360-90 = 270; 270/90 = 3 cylinders more than what DFSMS managed to find in a single extent).

Note: If your initial VSAM LDS allocation is for multiple volumes, the initial formatting of the zFS aggregate is limited to the primary allocation (in this case, the first volume), and one extension (in this case, the second volume). After that, you must use multiple **`zfsadm grow`** commands to grow to each new volume; see 2.13, “Increasing the physical size of an aggregate” on page 60 for more information about this topic.

2.7.3 Formatting aggregates with the -grow option

Beginning with z/OS V1R4, a new option was created to allow control of the amount of space that is formatted after an aggregate is defined.

When an aggregate is initially formatted using the IOEAGFMT format utility or the **`zfsadm format`** command, the formatting takes place as follows:

- The default size formatted is the number of blocks that will fit in the primary allocation of the VSAM LDS.
- Using the **-size** parameter, if the number of blocks to be formatted is less than the default, it is rounded up to the default.
- If a number greater than the default is specified, a single extend of the VSAM LDS is attempted after the primary allocation is formatted.

Note: The size of the secondary extension is proportional to the number of blocks to be formatted and rounded up to the next multiple of the secondary size specified in the VSAM LDS definition.

The **`-size`** parameter specified can only be allocated in the primary allocation and one extension. Thus, if you wanted to format a three-volume aggregate with the IOEAGFMT utility, this would not be possible because it requires a primary and at least two extensions. Therefore, a new **`-grow`** option is provided with z/OS V1R4 for the IOEAGFMT utility and the **`zfsadm format`** command to allow specification of the increment that can be used for extension of the aggregate when **`-size`** is larger than the primary allocation, as shown in Figure 2-20. This allows extension by the **`-grow`** amount until **`-size`** is satisfied.

-grow This specifies the number of 8 K blocks that zFS uses as the increment for an extension when the **`-size`** option specifies a size greater than the primary allocation.

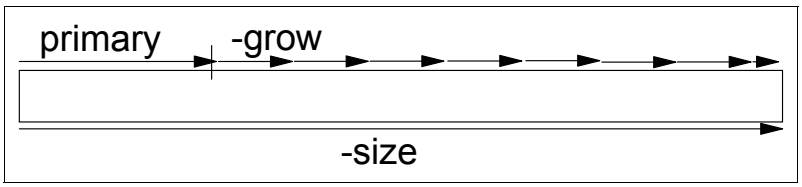


Figure 2-20 New -grow option

Using -grow, example 1

This example illustrates how to format a 3-volume aggregate using the **`-grow`** option with the **`zfsadm format`** command. Each volume has 300240 8 K blocks available for formatting in the

allocation, as shown in Figure 2-21. This allocation of the aggregate contains one primary and two secondaries.

Using the **zfsadm format** command to format the primary and two secondary allocations, specify:

```
zfsadm format -size 900720 -grow 300240
```

Where:

-size 900720 is the total size to be formatted in 8 K blocks.

-grow 300240 is the size of each secondary allocation, which is the number of 8 K blocks that zFS uses as the increment for an extension.

The formatting of the 3-volume aggregate continues until the **-size** value is formatted. The **-grow** value is the size of a secondary extension that allows the formatting to continue until the **-size** value is reached; see Figure 2-21.

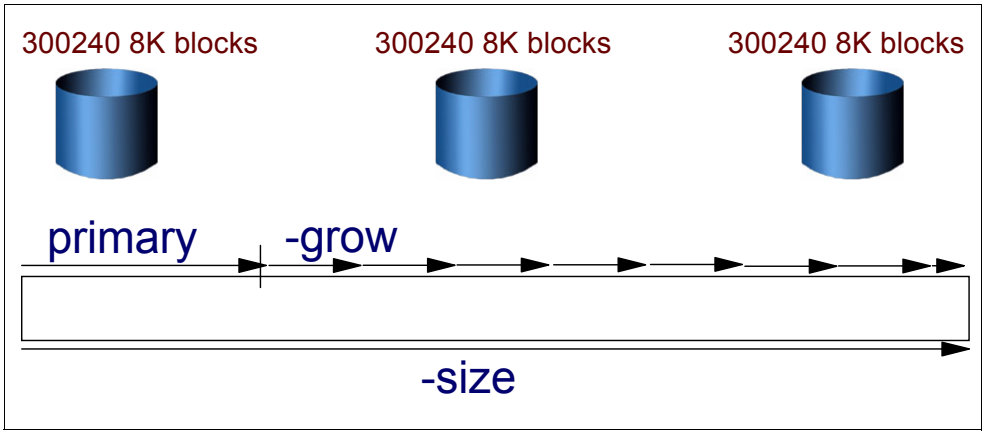


Figure 2-21 A 3-volume aggregate being formatted using the -grow option

Using -grow, example 2

To illustrate the before and after cases of using the **-grow** option, the example shown in Figure 2-22 has a VSAM LDS defined with 2 cylinders of primary space and 1 cylinder of secondary space.

```
//AYVIVAR2 JOB CLASS=J,MSGCLASS=A,NOTIFY=AYVIVAR
/*JOBPARM S=SC65
//PO10 EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER(NAME(OMVS.TESTA.ZFS) VOLUMES(SBOX43) -
  LINEAR CYL(2,1) SHAREOPTIONS(3))
//
```

Figure 2-22 Allocating a VSAM LDS for a zFS aggregate

Format the aggregate

The VSAM LDS is formatted as a compatibility mode aggregate, shown in Figure 2-23 on page 32, using a **-size** value of 276 8 K blocks.

```
//AYVIVAR2 JOB CLASS=J,MSGCLASS=V,NOTIFY=AYVIVAR
/*JOBPARM S=SC65
//FORMAT EXEC PGM=IOEAGFMT,REGION=0M,
//       PARM=('-aggregate OMVS.TESTA.ZFS -size 276 -compat')
//SYSPRINT DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//*
```

Figure 2-23 Formatting the aggregate

Listcat output of aggregate

The listcat output, displayed in Figure 2-24, shows that the file has two extents. The first extent corresponds to the primary space specified in the define process (30 tracks). The second extent has 30 tracks.

```
ALLOCATION
  SPACE-TYPE------CYLINDER     HI-A-RBA---------2949120
  SPACE-PRI--------------2     HI-U-RBA---------2949120
  SPACE-SEC--------------1
VOLUME
  VOLSER------------SBOX44     PHYREC-SIZE---------4096     HI-A-RBA---------2949120     EXTENT-NUMBER----------2
  DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12     HI-U-RBA---------2949120     EXTENT-TYPE--------X'40'
S  SYSTEM SERVICES                                          TIME: 19:31:08
  VOLFLAG------------PRIME     TRACKS/CA-------------15
  EXTENTS:
  LOW-CCHH-----X'02080000'     LOW-RBA----------------0 TRACKS----------------30
  HIGH-CCHH----X'0209000E'     HIGH-RBA---------1474559
  LOW-CCHH-----X'020A0000'     LOW-RBA----------1474560 TRACKS----------------30
  HIGH-CCHH----X'020B000E'     HIGH-RBA---------2949119
```

Figure 2-24 Listcat of the VSAM LDS aggregate

Using the -grow option for example 2

With the new **-grow** option, you can specify the increment that will be used for an extension size larger than the primary allocation. After the primary space is allocated, multiple extensions of the amount specified by the **-grow** parameter, which are rounded up to a multiple of the secondary space defined, are attempted until the total number of blocks specified by the **-size** parameter is satisfied, as shown in Figure 2-20 on page 30.

Replacing the example shown in Figure 2-23, we used the **-grow** parameter on the format process, as shown in Figure 2-25.

```
//AYVIVAR2 JOB CLASS=J,MSGCLASS=V,NOTIFY=AYVIVAR
/*JOBPARM S=SC65
//FORMAT EXEC PGM=IOEAGFMT,REGION=0M,
//       PARM=('-aggregate OMVS.TESTA.ZFS -size 276 -grow 90  -compat')
//SYSPRINT DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Figure 2-25 Format of VSAM LDS aggregate using -grow

Listcat output after using -grow

The listcat output, displayed in Figure 2-26, shows that now the VSAM LDS has three extensions; the first one corresponding to the primary space specified, and the next ones by the **-grow** amount.

```
ALLOCATION
  SPACE-TYPE------CYLINDER     HI-A-RBA---------2949120
  SPACE-PRI--------------2     HI-U-RBA---------2949120
  SPACE-SEC--------------1
VOLUME
  VOLSER------------SBOX15     PHYREC-SIZE---------4096     HI-A-RBA---------2949120     EXTENT-NUMBER----------3
  DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12     HI-U-RBA---------2949120     EXTENT-TYPE--------X'40'
S  SYSTEM SERVICES                                         TIME: 19:42:56
  VOLFLAG------------PRIME     TRACKS/CA-------------15
  EXTENTS:
  LOW-CCHH-----X'003F0000'     LOW-RBA----------------0     TRACKS--------------30
  HIGH-CCHH----X'0040000E'     HIGH-RBA---------1474559
  LOW-CCHH-----X'00410000'     LOW-RBA----------1474560     TRACKS--------------15
  HIGH-CCHH----X'0041000E'     HIGH-RBA---------2211839
  LOW-CCHH-----X'00420000'     LOW-RBA----------2211840     TRACKS--------------15
  HIGH-CCHH----X'0042000E'     HIGH-RBA---------2949119
```

Figure 2-26 Listcat output using the -grow option

If you prefer to not specify an absolute value in 8 K blocks and a corresponding **-grow** setting, we provide another means when discussing growing zFS aggregates that are initially formatted already; see 2.13.5, “Growing zFS aggregates in a loop” on page 69 for more information.

2.7.4 Display quota information

The maximum size of the compatibility mode file system just created is known as its *quota*. This is a logical number that is checked each time additional blocks are allocated to the file system. Figure 2-27 shows the file system quota to be 6335 for the 10 cylinders that were defined for the compatibility mode aggregate.

```
#> exit
$> zfsadm lsquota -filesystem OMVS.CMP01.ZFS
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.CMP01.ZFS        6335        9     0     2 = 154/6480 (zFS)
```

Figure 2-27 Display quota information about file systems and aggregates

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 for multifile systems), 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.

“Defining file systems by quota” on page 143 shows a situation where the single quotas of the file systems in the multifile system aggregate are smaller than the current amount of space available in the aggregate, but the sum of all the quotas is larger.

2.8 zFS configuration file

A configuration file is provided in IOE.SIOESAMP(IOEFSPRM) and is shown in Figure A-1 on page 360. This zFS file is optional. You need to specify parameters only if you want to override the defaults for the zFS parameters. This file can be used for the following purposes:

- To change the processing options of the zFS PFS
- To specify definition options for multifile aggregate attaches

2.8.1 Create a data set for IOEFSPRM

Because this file is optional, if you create an empty IOEFSPRM member in a PDS, the IOEFSPRM member should have a single line in it that is a comment by placing an asterisk (*) in column 1. Update the IOEZPRM DD statement in zFS PROC to contain the name of the IOEFSPRM member, as shown in Figure 2-8 on page 21.

Note: Sysplex considerations regarding the use of the configuration file are provided in “IOEFSPRM file considerations” on page 203.

Using the IOEFSPRM file, you can:

- Run zFS without an IOEFSPRM file.
- Run zFS with an IOEFSPRM defined in a flat file.
- Run with IOEFSPRM defined as a member of a PDS.

Running zFS without IOEFSPRM

We performed the following tests for running without an IOEFSPRM file:

- With no IOEZPRM DD statement
- With a DD DUMMY statement

Figure 2-28 illustrates using a DD DUMMY statement.

```
//IOEZPRM  DD DUMMY                                    <--ZFS PARM FILE
```

Figure 2-28 DD DUMMY statement for IOEZPRM in the zFS procedure

The result was that we received a huge number of messages in both of the tested situations at the startup of zFS in z/OS V1R3. This has been changed in z/OS V1R4. Figure 2-29 shows the zFS startup messages that are displayed without an IOEFSPRM DD statement.

```
IEF196I IEC130I IOEZPRM  DD STATEMENT MISSING
IEC130I IOEZPRM  DD STATEMENT MISSING
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.04.00 Service Level OA06893.
Created on Fri Mar 12 09:34:21 EST 2004.
IOEZ00055I zFS kernel: initialization complete.
```

Figure 2-29 zFS startup messages if no IOEFSPRM DD statement in zFS STC

Figure 2-30 on page 35 shows the zFS startup messages with a dummy IOEFSPRM DD statement.

```
IEF196I IEF236I ALLOC. FOR ZFS ZFS
IEF196I IEF237I DMY  ALLOCATED TO IOEZPRM
IEF196I IEC141I 013-10,IGG019AV,ZFS,ZFS,SYS00001,,,NULLFILE
IEC141I 013-10,IGG019AV,ZFS,ZFS,SYS00001,,,NULLFILE
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.04.00 Service Level OA06893.
Created on Fri Mar 12 09:34:21 EST 2004.
IOEZ00178I NULLFILE is the configuration dataset currently in use.
```

Figure 2-30 zFS startup messages with a dummy IOEFSPRM DD statement in ZFS STC

When using the **zfsadm** command to obtain information regarding the defined aggregates, some messages in z/OS V1R3 were displayed with the command output and the same messages that appeared during zFS startup were seen in the syslog. Beginning with z/OS V1R4, no further messages are displayed when running commands.

Important: We strongly recommend that if you do not want to place parameters in the IOEFSPRM file, then define at least a one-line IOEFSPRM file with an asterisk (*) in column 1 to avoid problems.

2.8.2 Running zFS with IOEFSPRM in a flat file

Running zFS with IOEFSPRM in a flat file has the disadvantage that you cannot update the contents while the zFS PFS is active. If you try to edit the file you get the ISPF message `Data set in use`. This is as expected because zFS still has read access to the file, as shown in Figure 2-31.

```
D GRS,RES=(SYSDSN,ZFS.SC43.IOEFSPRM)
 ISG343I 15.19.54 GRS STATUS 230
 S=SYSTEMS SYSDSN   IOE.IOEFSPRM
 SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
 SC43      ZFS                01F9       007FFBF8   SHARE      OWN
```

Figure 2-31 Displaying resource information for IOEFSPRM

Unfortunately, specifying DD parameter FREE=CLOSE for IOEFSPRM in the zFS procedure was not a solution in z/OS V1R3 because the data set is opened several times during processing, and beginning with the second try to open the file you get the `IOEZPRM DD STATEMENT MISSING` message, as shown in Figure 2-29 on page 34. Using FREE=CLOSE is still not usable in z/OS V1R4; the file is not opened several times but it now never gets closed. The same is true for z/OS V1R5 and z/OS V1R6 (if the IOEFSPRM file is used).

2.8.3 IOEFSPRM as a member of a PDS data set

The best and only reasonable option is to define IOEFSPRM as a member of a PDS data set. This allows updates while the zFS PFS is active.

The following two **define_aggr** statements are added to the IOEFSPRM member, as shown in Figure 2-32 on page 36. These statements are used to attach multifile aggregates automatically during startup of zFS and **zfsadm** commands. See 2.9, “Attaching an aggregate” on page 42 for more information. The changes made are available immediately, as shown in Figure 2-45 on page 43.

```
# ------------------------------------------------------------------ #
# List of multi-file system aggregates definitions to be attached    #
# when ZFS is started or restarted                                   #
# ------------------------------------------------------------------ #

define_aggr R/W attach nbs aggrfull(85,5) cluster(OMVS.MUL01.ZFS)
define_aggr R/W attach cluster(OMVS.MUL02.ZFS)
```

Figure 2-32 Define_aggr statements in IOEFSPRM

2.8.4 Logical parmlib search

In z/OS V1R6, the IOEPRMxx file can be placed in the parmlib. With this new support, a logical parmlib search is used. The logical parmlib concatenation is a set of up to 10 partitioned data sets defined by PARMLIB statements in the LOADxx member used at IPL time. This support includes the following:

- Member names are in the form IOEPRMxx.
- Multiple members can be specified.
- Allows an installation to have a common parmlib member that is shared among members of the sysplex, and also have an additional member that is unique.
- A list of members (suffixes) is specified in the FILESYSTYPE statement for zFS in a BPXPRMxx member.
- System symbols (that result in a two-character suffix) can be used in the PARM parameter of the FILESYSTYPE statement.

Using system symbols

Figure 2-33 displays static system symbols. Symbol &OMVSPARM. is used as a release-specific OMVS level identifier.

```
 D SYMBOLS
 IEA007I STATIC SYSTEM SYMBOL VALUES
...
         &SYSCLONE. = "65"
         &SYSNAME.  = "SC65"
...
         &OMVSPARM. = "6B"
```

Figure 2-33 Displaying static system symbols

Figure 2-34 shows an example using three levels of parmlib suffixes and pointing to specific parameter files by using static system symbols.

00 This member reflects all the desired general parameter settings.

&OMVSPARM. This member is used for version- or release-specific settings.

&SYSCLONE. This member can be used to add system-specific settings.

```
FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM)
            ASNAME(ZFS,'SUB=MSTR')
            PARM('PRM=(&SYSCLONE.,&OMVSPARM.,00)')
```

Figure 2-34 zFS FILESYSTYPE statement with three levels of IOEPRMxx parmlib files specified

Rules for parmlib specifications

Note the following rules for specifying the PARM parameter on the FILESYSTYPE statement:

- The PARM string is case sensitive. You must enter it in upper case.
- Alphanumeric characters are allowed (A to Z and 0 to 9).
- Up to 32 members (suffixes) can be specified. The default is IOEPRM00.
- Columns 73-80 are ignored in an IOEPRMxx member.
- If the IOEPRMxx member does not exist, the next one is searched for.

For information about specifying a list of members spreading over more than one line, see *z/OS Distributed File Service zSeries File System Administration,* SC24-5989.

Important: Keep the following points in mind:

- Parmlib search in z/OS V1R6 is only used if no IOEZPRM DD statement is placed into the zFS PROC.
- If an option is specified more than once, its value is taken from the first member in the list where it is found. To allow a more specific setting to always override the others, specify the list of suffixes from most specific to general, just as done in the example.

Additional IEF285I messages

Beginning with z/OS V1R5, when zFS is starting you receive additional IEF285I messages referencing the zFS parameter file or the data set containing it, as shown in Figure 2-35.

```
IEF196I IEF285I   SYS1.TEST.IOEFSZFS                          KEPT
IEF196I IEF285I   VOL SER NOS= ZO3DL2.
```

Figure 2-35 Additional IEF285I messages

You may see more than 100 pairs of such messages. When performing specific zFS PFS operations such as attaching aggregates, you receive more of the messages every time. If you start zFS outside of JES control by using in the following FILESYSTYPE statement, then these messages go to the syslog.

```
ASNAME(ZFS,'SUB=MSTR')
```

If zFS runs under the control of JES, the messages are found in the SYSOUT data set of the zFS started task. In all the sample output, these high numbers of IEF285I messages are not referenced.

Running without an IOEZPRM DD statement

When starting zFS in z/OS V1R6 without this DD statement in the zFS started task, and there is no member IOEPRMxx in any parmlib data set, you will receive a huge number of new messages stating that the default parmlib member has not been found. These messages are shown in Figure 2-36 on page 38. Instead of the IEC130I IOEZPRM DD STATEMENT MISSING message, you now receive these IEF764I messages over and over again.

Note: We also used a different zFS started task named zFSR6 to be able to stay with using the IOEZPRM DD statement in started task zFS for the z/OS V1R5 systems within the sysplex.

If you run in a mixed z/OS environment with releases of z/OS before z/OS V1R6, you can take one of the following actions:

- Wait to remove the IOEZPRM DD statement in the zFS started task.
- Use a different started task for zFS for the z/OS V1R6 systems, as we did.

Figure 2-36 shows starting zFS when no IOEPRM00 parmlib member is available.

```
 IEF764I ZFSR6 ZFSR6 SYS00001 IOELBRM PARMLIB READ FAILED - MEMBER IOEPRM00 NOT FOUND.
...
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
 Version 01.06.00 Service Level OA07700 - HZFS360.
 Created on Fri Jun 11 13:31:26 EDT 2004.
 IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
 IEF764I ZFSR6 ZFSR6 SYS00017 IOELBRM PARMLIB READ FAILED - MEMBER IOEPRM00 NOT FOUND.
...
 IOEZ00055I zFS kernel: initialization complete.
 BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
...
```

Figure 2-36 Starting zFS when no IOEPRM00 parmlib member is available

Running with a default member

If the default parmlib member IOEPRM00 exists in the parmlib search support, the behavior is the same whether or not you have “00” in the PARM specification of the FILESYSTYPE statement of zFS. Excerpts of the messages displayed during zFS startup are shown in Figure 2-37.

```
...
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
 Version 01.06.00 Service Level OA07700 - HZFS360.
 Created on Fri Jun 11 13:31:26 EDT 2004.
 IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
 IEE252I MEMBER IOEPRM00 FOUND IN CPAC.PARMLIB
...
 IOEZ00055I zFS kernel: initialization complete.
...
```

Figure 2-37 Starting zFS when IOEPRM00 parmlib member exists

On additional zFS operations (such as attaching of an aggregate) you may again see message IEE252I, as shown in Figure 2-38.

```
 IEE252I MEMBER IOEPRM00 FOUND IN CPAC.PARMLIB
...
 IOEZ00044I Aggregate OMVS.TEST.MULTIFS.ZFS attached successfully.
```

Figure 2-38 Additional IEE252I messages on further zFS operations

Exploiting the parmlib search using several IOEPRMxx members

When setting up the FILESYSTYPE statement for zFS, as shown in Figure 2-33 on page 36 and Figure 2-34 on page 36, this works as desired.

Figure 2-39 on page 39 shows a command to perform a remount for a zFS file system.

```
#> /usr/sbin/chmount -w -d SC65 test
```

Figure 2-39 Changing owner and mode for a zFS file system

During this remount processing, the following aggregate detaching message is displayed for all systems in the UNIX System Services sysplex sharing environment:

```
IOEZ00048I Detaching aggregate OMVS.HERING.ZFS
```

The message is followed by a block of IEE252I messages, as shown in Figure 2-40.

```
IEE252I MEMBER IOEPRM00 FOUND IN CPAC.PARMLIB
IEE252I MEMBER IOEPRM6B FOUND IN SYS1.PARMLIB
IEE252I MEMBER IOEPRM65 FOUND IN SYS1.PARMLIB
```

Figure 2-40 Block of IEE252I messages displayed

This flood of console messages like IEE252I, which was reported originally with APAR OA08537 in z/OS V1R6 and again with APAR OA17269 in z/OS V1R7 (for zFS releases 370 and 380), is no longer displayed if a certain service or release level is installed.

Important: If APAR OA20070 is applied in z/OS V1R7 and z/OS V1R8, then extra console messages such as IEE252I are no longer displayed.

2.8.5 Dynamic configuration

zFS has an IOEFSPRM configuration file that specifies the processing options for the zFS PFS, as well as definitions for multifile system aggregates. Prior to z/OS V1R4, to change any configuration file parameters, you had to perform the following tasks:

1. Modify the IOEFSPRM file.
2. Shut down and restart the zFS PFS.

However, this causes unmounts and potential moves of zFS file systems in a sysplex environment, which could be disruptive to applications and is administratively complex.

zFS provides utility programs and z/OS UNIX commands to assist in the customization of the aggregates and file systems. These utilities and commands are to be used by system administrators. For more detailed information about **zfsadm** command options and use, see *z/OS Distributed File Service zSeries File System Administration,* SC24-5989.

New zfsadm commands

Two new **zfsadm** commands were added with z/OS V1R4 to change configuration values dynamically without a shutdown, restart the zFS PFS, and display the current value of the zFS configuration options. Table 2-1 on page 15 shows the **zfsadm** command with its subcommands. To issue the commands shown in Table 2-1 on page 15, you must have RACF authorization, as follows:

IOEFSPRM The command issuer must have access to a RACF data set profile to the IOEFSPRM data set.

SU mode The command issuer must have superuser authority.

Table 2-3 The new zfsadm command and subcommands for z/OS V1R4

Command/subcommand	Command description	IOEFSPRM	SU mode
zfsadm config	Modify current configuration options	Read	Yes
zfsadm configquery	Display current configuration options	Read	No

Important: If the parmlib search is used (for IOEPRMxx members), which means that no IOEZPRM DD statement is in the zFS STC JCL, then there is no need for authorization to an IOEFSPRM file.

zfsadm config command

The **zfsadm config** command changes the value of zFS configuration options in memory that were specified in the IOEFSPRM file (or defaulted).

The format of the **zfsadm config** command is shown in Figure 2-41.

```
zfsadm config [-admin_threads number]
         [-user_cache_size number]
         [-meta_cache_size number]
         [-log_cache_size number]
         [-sync_interval number]
         [-vnode_cache_size number]
         [-nbs {on|off}][-fsfull threshold,increment]
         [-aggrfull threshold,increment]
         [-trace_dsn PDSE_dataset_name]
         [-tran_cache_size number]
         [-msg_output_dsn Seq_dataset_name]
         [-user_cache_readahead {on|off}]
         [-metaback_cache_size number]
         [-fsgrow increment,times]
         [-aggrgrow {on|off}]
         [-allow_dup_fs {on|off}]
         [-system system_name] R7
         [-level]
         [-help]
```

Figure 2-41 zfsadm config command parameters

The issuer must have READ authority to the data set that contains the IOEFSPRM file and must be a superuser or have READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the UNIXPRIV class.

Note: Option **-system system_name** is new with z/OS V1R7.

The example shown in Figure 2-42 changes the value of the sync_interval to 60 seconds.

```
#>zfsadm config -sync_interval 60
IOEZ00300I Successfully set -sync_interval to 60
```

Figure 2-42 Command to change the value of the sync_interval

Note: To make the configuration specification permanent, update the IOEFSPRM file.

zfsadm configquery command

The **zfsadm configquery** command displays the current value of zFS configuration options retrieved from the zFS address space memory, rather than from the IOEFSPRM file.

The format of the **zfsadm configquery** command is shown in Figure 2-43.

```
zfsadm configquery [-adm_threads]
          [-aggrfull]
          [-aggrgrow]
          [-all]
          [-allow_dup_fs]
          [-auto_attach]
          [-cmd_trace]
          [-code_page]
          [-debug_dsn]
          [-fsfull]
          [-fsgrow]
          [-group] R7
          [-log_cache_size]
          [-meta_cache_size]
          [-metaback_cache_size]
          [-msg_input_dsn]
          [-msg_output_dsn]
          [-nbs]
          [-storage_details]
          [-sync_interval]
          [-sysplex_state] R7
          [-trace_dsn]
          [-trace_table_size]
          [-tran_cache_size]
          [-user_cache_readahead]
          [-user_cache_size]
          [-usercancel]
          [-vnode_cache_size]
          [-level]
          [-system system_name] R7
          [-help]
```

Figure 2-43 zfsadm configquery command parameters

Note: Options **-group**, **-sysplex_state** and **-system system_name** are new with z/OS V1R7. See 5.8, “Sysplex awareness of the zfsadm command interface” on page 234 and 5.8.2, “The zFS sysplex group” on page 238 for more information.

The example in Figure 2-44 on page 42 displays all z/OS V1R4 values of zFS configuration options.

```
$>zfsadm configquery -all
IOEZ00317I The value for config option -adm_threads is 10.
IOEZ00317I The value for config option -aggrfull is <no value>.
IOEZ00317I The value for config option -aggrgrow is OFF.
IOEZ00317I The value for config option -allow_dup_fs is OFF.
IOEZ00317I The value for config option -auto_attach is ON.
IOEZ00317I The value for config option -cmd_trace is OFF.
IOEZ00317I The value for config option -debug_dsn is <no value>.
IOEZ00317I The value for config option -fsfull is <no value>.
IOEZ00317I The value for config option -fsgrow is <no value>.
IOEZ00317I The value for config option -log_cache_size is 64M.
IOEZ00317I The value for config option -meta_cache_size is 32M.
IOEZ00317I The value for config option -metaback_cache_size is <no value>.
IOEZ00317I The value for config option -msg_input_dsn is <no value>.
IOEZ00317I The value for config option -msg_output_dsn is <no value>.
IOEZ00317I The value for config option -nbs is OFF.
IOEZ00317I The value for config option -sync_interval is 30.
IOEZ00317I The value for config option -trace_dsn is <no value>.
IOEZ00317I The value for config option -trace_table_size is 256K.
IOEZ00317I The value for config option -tran_cache_size is 2000.
IOEZ00317I The value for config option -user_cache_readahead is ON.
IOEZ00317I The value for config option -user_cache_size is 256M.
IOEZ00317I The value for config option -vnode_cache_size is 8192.
```

Figure 2-44 Display of all the IOEFSPRM values in a z/OS V1R4 system

2.9 Attaching an aggregate

Compatibility mode aggregates do not need to be attached. When the format processing for the compatibility aggregate is finished, the aggregate is attached and the file system is created; see Figure 2-17 on page 28.

A multifile system aggregate must be attached before issuing a **mount** command for it and before a zFS file system can be created in the aggregate. There are three different ways to attach a multifile system aggregate, as follows:

- The attach can be done on the zFS colony address space startup by having an entry in the IOEFSPRM file.
- Use the IOEZADM utility program in a submitted job.
- Use the **zfsadm attach** command.

Note: You do not have to attach an HFS compatibility mode zFS aggregate. Simply mount or automount the file system.

2.9.1 Attach using the IOEFSPRM member

If you created and formatted a zFS multifile system aggregate, you may add an entry in the IOEFSPRM member for the aggregate. This causes the multifile system aggregate to be attached when zFS is started. Add the following lines to the IOEFSPRM member:

```
define_aggr R/W attach cluster(OMVS.MUL01.ZFS)
define_aggr R/W attach cluster(OMVS.MUL02.ZFS)
```

Attach without IPL

If you add those statements while zFS is already running, you can use the command shown in Figure 2-45 to attach the aggregates immediately without having to do an IPL.

```
zfsadm attach -all
IOEZ00117I Aggregate OMVS.MUL01.ZFS attached successfully
IOEZ00118I Aggregate OMVS.MUL02.ZFS is already attached
```

Figure 2-45 Attaching multifile aggregates known from IOEFSPRM define_aggr statements

2.9.2 Attach using the IOEZADM program

If a zFS multiple file system aggregate is created after zFS is started, you must attach the zFS aggregate to tell the zFS PFS about it. Use the **zfsadm** command, shown in 2.9.3, “Attach using the zfsadm command” on page 43, or use the JCL shown in Figure 2-46 to attach an aggregate.

A sample of the JCL to run PGM=IOEZADM is supplied in IOE.SIOESAMP(IOEZADM). You may also use IOEZADM from the TSO foreground as a command.

```
//ZFSATTA JOB ,'ZFS Attach Aggregate',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//ZFSADMA  EXEC   PGM=IOEZADM,REGION=0M,
// PARM=('attach -aggregate OMVS.MUL02.ZFS')
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD
//SYSPRINT DD   SYSOUT=T
//STDOUT   DD   SYSOUT=T
//STDERR   DD   SYSOUT=T
//SYSUDUMP DD   SYSOUT=T
//CEEDUMP  DD   SYSOUT=T
//*
```

Figure 2-46 JCL to attach a multifile system aggregate

2.9.3 Attach using the zfsadm command

You need a symbolic link named zfsadm in the /bin directory that points to a real UNIX file executable (see Figure 2-1 on page 16) to be able to use the command just by its name.

The command shown in Figure 2-47 can be used from the OMVS shell to attach a multifile aggregate.

```
#> zfsadm attach -aggregate OMVS.MULTUSER.ZFS -aggrfull 90,5 -nbs
IOEZ00117I Aggregate OMVS.MULTUSER.ZFS attached successfully
```

Figure 2-47 Attaching a multifile aggregate

Note: The aggregate name specified in the command is not case sensitive; it will be translated to upper case if entered in lower case.

Which aggregates are attached

To determine which aggregates are attached, issue the **zfsadm lsaggr** command as shown in Figure 2-48 on page 44.

```
ROGERS @ SC43:/>zfsadm lsaggr
IOEZ00106I A total of 3 aggregates are attached
OMVS.MULTUSER.ZFS                          (id=100002)
OMVS.MUL02.ZFS                             (id=100001)
OMVS.MUL01.ZFS                             (id=100000)
```

Figure 2-48 Listing aggregates that are attached

2.10 Create a zFS file system

There are two types of aggregates that zFS file systems can be defined in, as explained here:

Compatibility mode There can be only one zFS file system in a compatibility mode aggregate. The name of the file system is the same as the aggregate name. The file system size is the size of the VSAM LDS aggregate. Use this file system name to mount the file system.

> **Note:** The file system for a compatibility mode aggregate is created following the format of the aggregate, as shown in Figure 2-17 on page 28.

Multifile system A multifile system aggregate can contain multiple zFS file systems. The aggregate has a name different from the file system names. Each file system has a predefined size. A multifile aggregate must be attached before a zFS file system can be defined in it. All file systems in the aggregate share the space allocated to the aggregate.

2.10.1 Create a multifile file system

After an aggregate has been attached, a zFS file system can be created in the aggregate. The zFS colony address space must be running. There are three methods to define a zFS file system, as follows:

- Use the IOEZADM utility program using JCL.
- Use the IOEZADM command from TSO.
- Use the **zfsadm** command from the OMVS shell.

> **Important:** The user must have superuser authority to use the IOEZADM program or the proper authority for the **zfsadm** command, as shown in Table 2-1 on page 15.

The IOEZADM program using JCL

To define a zFS file system in an aggregate, use the IOEZADM program with the subcommand CREATE. When defining the parm= keyword, as shown in Figure 2-49 on page 45, specify the following:

- The file system name, which is case sensitive
- The aggregate name where the file system is to be defined
- The size of the file system in 1 K blocks, which is a required parameter

Note: There is no default logical size for a zFS file system. The zFS file system name is case sensitive, and must be in upper case letters.

```
//ZFZADM   JOB ,'ZFS Create Filesys',NOTIFY=ROGERS,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//*
//ZFSADM   EXEC   PGM=IOEZADM,REGION=0M,
//         PARM=('create -filesystem OMVS.MO2A.ZFS
//             -aggregate omvs.mul02.zfs -size 5000')
//STEPLIB DD DISP=SHR,DSN=IOE.SIOELMOD
//SYSPRINT DD   SYSOUT=T
//STDOUT   DD   SYSOUT=T
//STDERR   DD   SYSOUT=T
//SYSUDUMP DD   SYSOUT=T
//CEEDUMP  DD   SYSOUT=T
//*
```

Figure 2-49 Defining a zFS file system in a specific aggregate

The IOEZADM command using TSO

The IOEZADM program is also supported from a TSO foreground environment by issuing the **ioezadm** command, as shown in Figure 2-50.

```
ioezadm create -filesystem OMVS.MO2A.ZFS -size 5000 -aggregate OMVS.WEB.ZFS
```

Figure 2-50 Creating a file system using IOEZADM from TSO

The zfsadm command using the OMVS shell

If you prefer to use the OMVS shell to define the zFS file system using the **zfsadm** command, the command shown in Figure 2-51 is equivalent to the preceding IOEZADM examples (the message displayed is also seen in the IOEZADM examples).

```
#> zfsadm create -filesystem OMVS.MO2A.ZFS -size 5000 -aggregate OMVS.WEB.ZFS
IOEZ00099I File system OMVS.MO2A.ZFS created successfully
```

Figure 2-51 Creating a file system using zfsadm from a UNIX shell environment

Important: When you create a file system and give it a file system name for multifile system aggregates, you specify the aggregate name on the create. As a result, when you mount the file system, the mount processing knows on which aggregate the file system exists.

2.10.2 Duplicate file system names

Prior to z/OS V1R4, file system names were required to be unique among all attached aggregates on a system. Although it is possible to create the same file system name on two different aggregates, you need to ensure that they are not attached at the same time. If a second aggregate is attached, an error message is issued and the duplicate file system becomes unavailable.

With z/OS V1R4, a new IOEFSPRM configuration file option allows duplicate file system names to be in different aggregates. The new option is:

```
allow_duplicate_filesystems=on
```

For commands that specify zFS file systems, you can now specify an aggregate name to specify the aggregate in which the file system name exists. However, if a file system name is ambiguous (meaning that the name is a duplicate and an aggregate name is not specified), then the command fails.

If you specify **allow_duplicate_filesystems=off** (which is the default) and try to create a duplicate file system name, the request will be denied and the following error message will be issued:

```
IOEZ00097E File system ZFSA already exists.
```

In previous versions (before the new option in z/OS V1R4), it was possible to create the same file system name on two different aggregates by not having them attached at the same time. When the second aggregate is attached, the attach is successful but an error message is issued for the duplicate file system name and it becomes unavailable, as follows:

```
IOEZ00314E The file system name ZFSA is not unique. Its aggregate name must
also be specified.
```

2.11 zFS file system mounts

After creating your zFS file systems, to make them available you must mount them at a mount point off the root directory. First decide where in the root file system to create the starting mount point.

There are two types of aggregates that contain file systems to be mounted, as follows:

Compatibility mode This is a zFS file system in a compatibility mode aggregate can be mounted, using the **mount** command, at an installation-created mount point. A zFS file system in a compatibility mode aggregate can also be AUTOMOVEed or automounted using the automount facility.

Multifile system This is a multifile system aggregate must be attached before a zFS file system can be created. See 2.9, “Attaching an aggregate” on page 42. After creating a directory for the mount point, you may use the **mount** command for the mount.

2.11.1 Mounting zFS file systems

After the file systems have been created, they must be mounted at a directory mount point so that users can access them. During the startup of z/OS UNIX, zFS file systems can be automatically mounted during the IPL by specifying them as follows:

- In the /etc/rc file using an OMVS shell mount command, **/usr/sbin/mount**, as shown in Figure 2-58 on page 50
- In the BPXPRMxx member beginning with z/OS V1R3, as shown in Figure 2-59 on page 50

Important: When allocating zFS aggregates, choose a useful naming convention for the aggregate and file system names. 2.11.4, “Automount for zFS file systems” on page 50 describes a useful method for mounting all zFS file systems.

2.11.2 Mounting user file systems

A preferred place to mount all user HFS data sets is a user directory under the /u user directory. In this book we describe the methods for mounting zFS file systems, because some installations may already be using direct mount while others may be using the automount facility.

z/OS UNIX suggests you use one of the following methods:

- Direct mount

  See 2.11.3, “Direct mount” on page 47
- Automount facility using zFS

  See 2.11.4, “Automount for zFS file systems” on page 50
- Automount facility using zFS for users

  For this automount facility example, see 3.3, “Automount facility for zFS” on page 141.

2.11.3 Direct mount

Using direct mount requires the allocation of an intermediate HFS or zFS data set, as shown in Figure 2-53 on page 48. We named this HFS data set OMVS.USERS.HFS. It is to be mounted at the /u directory between the root file system and all user file systems.

Note: To mount OMVS.USERS.HFS every time an IPL occurs, you need to add the HFS data set name and its mount point to the BPXPRMxx member of parmlib.

Creating zFS mount points

Create a mount point in the OMVS.USERS.HFS data set by using the **mkdir** command as shown in Figure 2-52.

```
#> cd /u
#> mkdir zfs
```

Figure 2-52 Creating directory zfs under /u

You can now either add other new directories for each zFS file system in OMVS.USERS.HFS, or add them off of the zfs directory mount point as shown in Figure 2-57 on page 49.

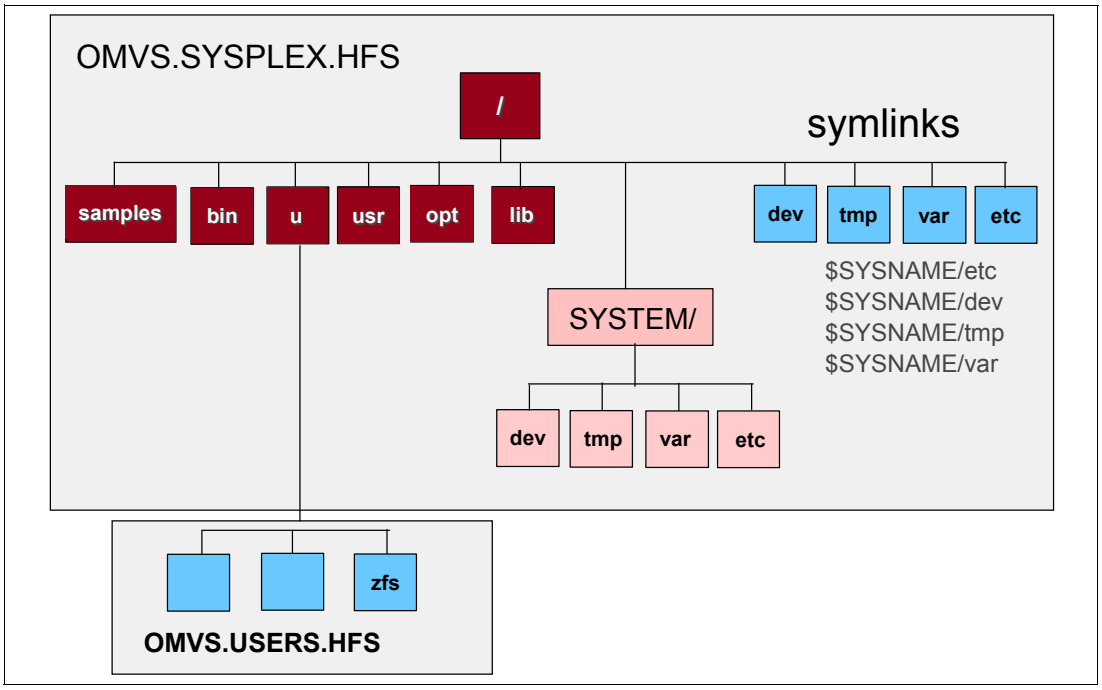


Figure 2-53 Creating the zfs directory in the root file system

Direct mount example

In the example in Figure 2-57 on page 49, we show multiple zFS file systems mounted at specified mount points. For the multifile system aggregates, we are using single-qualifier file system names.

Important: In this direct mount example, we are using a different naming convention for the aggregate names and file system names. Also, the creation of the aggregates and file systems is not shown.

Figure 2-54 illustrates how to create the directories and issue the mounts.

```
#> cd zfs
#> mkdir cmp01
#> mkdir m02a
#> mkdir m02b
#> mkdir m01a
```

Figure 2-54 Creating mount points in the zfs directory

The OMVS shell command **/usr/sbin/mount** or the TSO/E MOUNT command may be used to mount your file system. The compatibility mode zFS file system can also be mounted using the **AUTOMOVE** option on the **mount** command.

Mount commands

After your zFS file systems have been defined, you decide how to mount them:

- You may IPL the system and use the startup of z/OS UNIX to have them mounted, as shown in 2.11.1, “Mounting zFS file systems” on page 46.

- You may use the MOUNT command from TSO/E or the **mount** command from the OMVS shell.

Note: There are also other possible ways to mount file systems; for example, you can use option 3 in the File_systems action menu of the ISHELL.

You can issue the **MOUNT** command from TSO/E for the compatibility mode file system as shown in Figure 2-55.

```
MOUNT FILESYSTEM('OMVS.CMP01.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/u/zfs/cmp01')
```

Figure 2-55 Mounting a compatibility mode file system

Issue the **MOUNT** command from TSO/E for the multifile file system, as shown in Figure 2-56.

```
MOUNT  FILESYSTEM(' OMVS.M02A.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/u/zfs/m02a')
MOUNT  FILESYSTEM(' OMVS.M02B.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/u/zfs/m02b')
MOUNT  FILESYSTEM(' OMVS.M01A.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/u/zfs/m01a')
```

Figure 2-56 Mounting multifile file systems

Note: zFS multifile file system names can be any number of qualifiers.

As shown in Figure 2-57, there are two multiple file aggregates: one with one file system in it (OMVS.M01A.ZFS), and the other with two file systems (OMVS.M02A.ZFS and OMVS.M02B.ZFS).

The compatibility mode aggregate has only one file system, which is the same as the aggregate name.

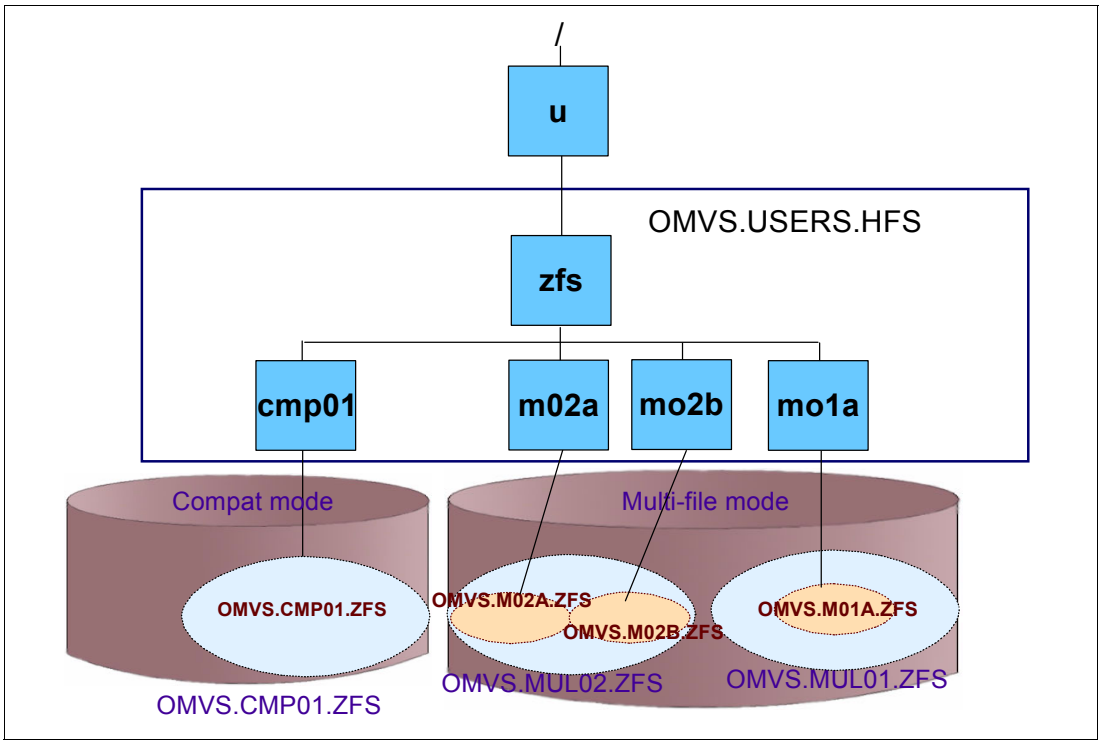


Figure 2-57 Mounted zFS file systems

Mounting during IPL

You can place an OMVS shell **mount** command in the /etc/rc file to mount a zFS file system when UNIX System Services is started, as shown in Figure 2-58.

```
# Initialization shell script, pathname = /etc/rc
...
/usr/sbin/automount

/usr/sbin/mount -t ZFS -f OMVS.CMP01.ZFS /u/zfs/cmp01
/usr/sbin/mount -t ZFS -f OMVS.M01A.ZFS /u/zfs/m01a
/usr/sbin/mount -t ZFS -f OMVS.M02A.ZFS /u/zfs/m02a
/usr/sbin/mount -t ZFS -f OMVS.M02B.ZFS /u/zfs/m02b

echo /etc/rc script executed, `date`
```

Figure 2-58 /usr/sbin/mount statements in /etc/rc

Beginning with z/OS V1R3, you may use mount statements in the BPXPRMxx member in the same way as done for HFS file systems, as shown in Figure 2-59.

```
MOUNT FILESYSTEM('OMVS.CMP01.ZFS') MOUNTPOINT('/u/zfs/cmp01') TYPE(ZFS) MODE(RDWR)
MOUNT FILESYSTEM('OMVS.M01A.ZFS') MOUNTPOINT('/u/zfs/m01a') TYPE(ZFS) MODE(RDWR)
MOUNT FILESYSTEM('OMVS.M02A.ZFS') MOUNTPOINT('/u/zfs/m02a') TYPE(ZFS) MODE(RDWR)
MOUNT FILESYSTEM('OMVS.M02B.ZFS') MOUNTPOINT('/u/zfs/m02b') TYPE(ZFS) MODE(RDWR)
```

Figure 2-59 MOUNT statements in BPXPRMxx

Important: A multifile aggregate must be attached to allow mounting of the file systems that it contains. If you want to exploit mounting multifile file systems from the BPXPRMxx member, place a define_aggr statement into the IOEFSPRM file because there is no other way that would allow an attach of a multifile aggregate early enough during the IPL. BPXPRMxx mount statements are processed before the startup of the BPXOINIT address space, and therefore before /etc/rc or other UNIX System Services scripts may be performed.

2.11.4 Automount for zFS file systems

If your installation already has an existing automount policy using /u, you can consider the implementation we used for the mounting of zFS file systems into the z/OS UNIX hierarchy.

Important: If you are at a z/OS level that supports HFS and zFS within the same automount policy, the following additional effort is not needed. See 3.3.3, “Automount assistance for HFS-to-zFS migration” on page 144 for details about this topic.

The first step is to create a new directory in the root. In our example this is directory z, shown in Figure 2-60.

```
#> cd /
#> mkdir z
```

Figure 2-60 Create a new directory in the root

One of the main concerns when creating zFS file systems is *where* they should be mounted in the z/OS UNIX hierarchy.

If you choose to add to an existing automount policy, you can modify it as shown in Figure 2-62 on page 52 by following these steps:

1. Add a new entry in the auto.master file.
2. Create a new map file for the new mount point in the auto.master file.

Map file file system name

The file system name in the z.map file specifies the file systems that are to be mounted by the automount facility. In our previous examples, we created the multifile aggregate OMVS.MUL02.ZFS. We defined the following file systems in the aggregate as shown in Figure 2-61.

```
ROGERS @ SC43:>zfsadm lsfs -a OMVS.MUL02.ZFS -fast
OMVS.MO2A.ZFS
OMVS.MO2D.ZFS
OMVS.MO2B.ZFS
OMVS.mO2c.ZFS
```

Figure 2-61 List of file systems in aggregate OMVS.MUL02.ZFS

Important: The file systems shown in Figure 2-61 are eligible to be mounted by the automount map file shown in Figure 2-62 on page 52. However, the last file system in the list has a lower case qualifier for the <uc_name>. This file system will not be mounted by automount. See 2.11.6, “Mixed-case multifile file system names” on page 56, for more information about this topic.

Using <uc_name>

The <uc_name> variable is used to convert the name being looked up to upper case. Whenever this variable is encountered it is replaced by the name being looked up. A directory with the looked-up name is created, as shown in Figure 2-64 on page 53, and used as a mount point for the file system to be mounted. When creating a map file, the <uc_name> variable can be used to replace any level qualifier in the data set name.

Note: If you want to use upper case and lower case qualifiers, you can use the <asis_name> variable to represent the name as is instead of <uc_name>.

2. /etc/auto.master

```
/u          /etc/u.map
/z          /etc/z.map
```

3. /etc/z.map

```
### ZFS automount map file for mount point /z ###
name       *
type       ZFS
filesystem OMVS.<uc_name>.ZFS
mode       rdwr
duration   nolimit
delay      10
```

Figure 2-62 Automount policy changed to add zFS file systems

Automount a file system

A file system is mounted by the automount facility when any user runs the program shown in Figure 2-63.

```
ROGERS @ SC43:>cd /z/m02a
ROGERS @ SC43:>/z/m02a>ls -al
total 336
drwxr-xr-x   9 HERING   SYS1         736 Apr  2 01:43 .
drwxr-xr-x   9 HERING   SYS1         736 Apr  2 01:43 ..
-rwxr-xr-x   1 HERING   SYS1        1869 Mar 14 16:27 .profile
-rwxr-xr-x   1 HERING   SYS1         689 Mar 14 16:29 .setup
-rw-------   1 HERING   SYS1        3033 Apr  1 15:29 .sh_history
drwxr-xr-x   2 HERING   SYS1         256 Mar 15 16:08 bin
drwx------   4 RC43     SYS1        1088 Apr  4 00:35 test
-r--r--r--   1 RC43     SYS1         147 Mar 17 14:41 test.extattr
drwx------   2 RC43     SYS1         256 Mar 31 15:29 test1
lrwxrwxrwx   1 RC43     SYS1          15 Apr  4 00:35 test1.sl -> /u/hering/test
1
drwxr-xr-x   2 HERING   SYS1         256 Mar 14 14:58 test2
drwxr-xr-x   2 HERING   SYS1         256 Mar 19 00:43 test3
```

Figure 2-63 Automount of the file system

After all the file systems have been accessed, *AMD/z is shown as managing the /z directory; see Figure 2-64 on page 53.

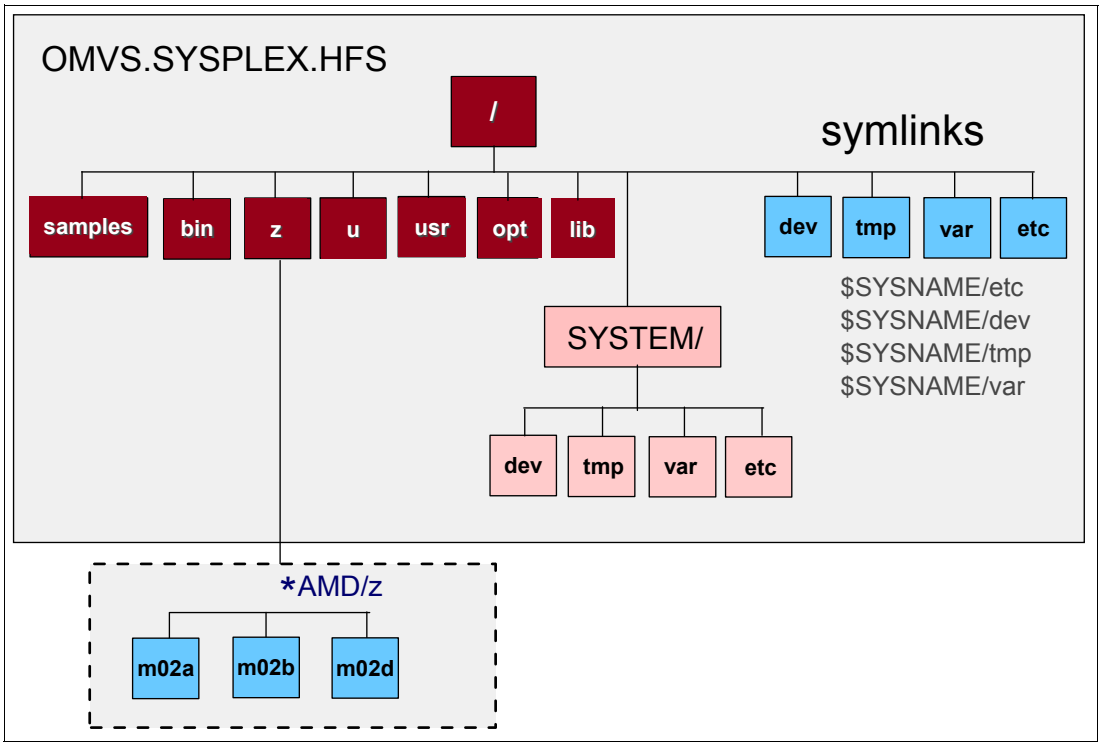


Figure 2-64 Automount using the /z directory for zFS file systems

2.11.5 Dynamic creation of automounted zFS file systems

Beginning with z/OS V1R5, it is now possible to have automount policies that allow for dynamic creation of automounted zFS file systems that do not exist. This was possible for HFS in previous releases. The dynamically created file system is a compatibility mode zFS aggregate.

Important: Automount allocation specifications for zFS should not be used in a UNIX System Services file system sharing environment unless all systems are at V1R5 or later.

The two keywords in the automount policy for creating a zFS compatibility mode aggregate are listed here:

- **allocany** allocation-spec

  This specifies the allocation parameters when using automount to allocate HFS or zFS data sets. **allocany** will cause an allocation if the data set does not exist for any name looked up in the automount-managed directory.

- **allocuser** allocation-spec

  This specifies the allocation parameters when using automount to allocate HFS or zFS data sets. **allocuser** will cause an allocation to occur only if the name looked up matches the user ID of the current user.

Note that allocation-spec is a string that specifies allocation keywords. The following keywords can be specified in the string:

```
space(primary-alloc[,secondary alloc])
cyl | tracks |  block(block size)
vol(volser[,volser]...)
```

```
maxvol(num-volumes)
unit(unit-name)
storclas(storage-class)
mgmtclas(management-class)
dataclas(data-class)
```

Restriction: For non-HFS compatibility mode aggregates, the aggregates must be attached. Because this assumes HFS-compatible zFS file systems, do not use this support if file systems in general aggregates are mounted with the same automount policy. If you do, unused data sets may be created. For more details, see *z/OS Distributed File Service zSeries File System Administration,* SC24-5989.

Examples for dynamic creation of automounted file systems

The following examples demonstrate how this works. Figure 2-65 shows the contents of the following files of the automount facility:

- auto.master.zfs
- auto.map.zfs

The auto.map.zfs file contains the option allocuser that can now be used by zFS. Specifying allocuser causes an allocation to occur only if the name looked up matches the user ID of the current user.

The command to start the automount facility by a superuser is:

```
/usr/sbin/automount /etc/auto.master.zfs
```

```
$> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.05.00
$> cat /etc/auto.master.zfs
/u              /etc/auto.map
/z              /etc/auto.map.zfs
$> cat /etc/auto.map.zfs
name              *
type              ZFS
filesystem        OMVS.<uc_name>.ZFS
mode              rdwr
allocuser         space(1,1) cyl storclas(openmvs)
duration          1440
delay             360
parm              aggrgrow
$> su
#> /usr/sbin/automount /etc/auto.master.zfs
FOMF0107I Processing file /etc/auto.map
FOMF0107I Processing file /etc/auto.map.zfs
FOMF0108I Managing directory /u
FOMF0108I Managing directory /z
#> exit
$>
```

Figure 2-65 Activating an automount policy exploiting dynamic creation of zFS aggregates

Example 1 - using allocuser

Now assume the zFS aggregates OMVS.PAUL.ZFS and OMVS.RICH.ZFS already exist, but OMVS.HERING.ZFS and OMVS.HERITST.ZFS do not exist.

Figure 2-66 on page 55 shows that referencing structures associated to data sets that already exist causes these file systems to be mounted. Because directory HERITST is different from

the current user HERING, the missing data set is not created because allocuser was used. In the case of switching to directory HERING, this results in the new zFS aggregate to be built and mounted, as shown by the **zfsadm lsaggr** command in Figure 2-66.

```
$> id
uid=888(HERING) gid=2(SYS1) groups=1047(USSTEST)
$> cd /z
$> ls -En
total 0
$> cd heritst && cd ..
cd: heritst: EDC5129I No such file or directory.
$> cd paul && cd ..
cd: paul: EDC5111I Permission denied.
$> cd rich && cd ..
$> cd hering && cd ..
$> ls -En
total 6
drwxr-x---        2 888      2            256 Aug 30 20:09 hering
drw-------        2 0        2            256 Sep 20  2002 paul
drwxr-xr-x        2 0        2            256 Mar  3  2003 rich
$> zfsadm lsaggr
IOEZ00106I A total of 8 aggregates are attached
OMVS.HERING.ZFS                                (id=100009)
OMVS.RICH.ZFS                                  (id=100008)
TWS.TWSABIN.RVA.ZFS                            (id=100003)
TWS.TWSABIN.ZFS                                (id=100002)
OMVS.PAUL.ZFS                                  (id=100007)
OMVS.HERING.TEST.ZFS                           (id=100006)
TWS.TWSAWRK.ZFS                                (id=100001)
TWS.TWSAWRK.RVA.ZFS                            (id=100000)
```

Figure 2-66 Creating and mounting a new zFS aggregate by referencing the directory name

Example 2 - using allocany

Using the allocany statement in the automount policy causes an allocation if the data set does not exist for any name looked up in the automount-managed directory. The difference compared to allocuser is shown in Figure 2-67 on page 56.

This also shows that the newly created aggregates and file systems always get the z/OS UNIX UID value of the user referencing the structure.

```
$> cd /z
$> cd heritst && cd ..
$> ls -En
total 8
drwxr-x---        2 888      2            256 Aug 30 20:09 hering
drwxr-x---        2 888      2            256 Aug 30 20:23 heritst
drw-------        2 0        2            256 Sep 20  2002 paul
drwxr-xr-x        2 0        2            256 Mar  3  2003 rich
$> zfsadm lsaggr
IOEZ00106I A total of 9 aggregates are attached
OMVS.HERING.ZFS                                  (id=100009)
OMVS.RICH.ZFS                                    (id=100008)
TWS.TWSABIN.RVA.ZFS                              (id=100003)
TWS.TWSABIN.ZFS                                  (id=100002)
OMVS.HERITST.ZFS                                 (id=100010)
OMVS.PAUL.ZFS                                    (id=100007)
OMVS.HERING.TEST.ZFS                             (id=100006)
TWS.TWSAWRK.ZFS                                  (id=100001)
TWS.TWSAWRK.RVA.ZFS                              (id=100000)
```

Figure 2-67 Creating and mounting a file system when using allocany in the policy definition

2.11.6 Mixed-case multifile file system names

Although we cannot recommend creating multifile file system names with mixed case, this is allowed. We created a file system named OMVS.m02c.ZFS as a test case. There is no problem in using the following BPXPRMxx mount statement for a mixed-case name, as shown in Figure 2-68.

```
MOUNT FILESYSTEM('OMVS.m02c.ZFS')
      MOUNTPOINT('/u/zfs/m02c')
      TYPE(ZFS) MODE(RDWR)
```

Figure 2-68 Mixed-case multifile file system aggregate mount statement in BPXPRMxx

Rules for mixed-case names

The rules for mixed-case file system names are:

- Mixed case is supported for TSO mounts only as follows:

  If you want to do the mount from TSO, you need to specify triple quotes, as shown in Figure 2-69 on page 57. Otherwise, the file system name is folded to uppercase, which will cause a mount failure.

- Mixed case is supported for the mount in BPXPRMxx.

  See the example in Figure 2-68.

- Mixed case is supported for the UNIX mount command **/usr/sbin/mount**.

  This may be used in /etc/rc statements, for example.

- The ISHELL does not support mixed-case file system names before z/OS V1R4 and always switches the name to upper case first before trying to perform the mount; this causes a mount failure.

  See 2.11.7, “Mounting zFS file systems using the ISHELL” on page 57 for an explanation of how this is supported for the ISHELL starting in z/OS V1R4.

```
mount filesystem('''OMVS.m02c.ZFS''') mountpoint('/zfs/m02c') type(zfs)
unmount filesystem('OMVS.m02c.ZFS')
RETURN CODE 00000079, REASON CODE 0588002E.  THE UNMOUNT FAILED FOR FILE SYSTEM
OMVS.M02C.ZFS.....
```

Figure 2-69 Successful mount and unsuccessful unmount for a mixed-case multifile file system

2.11.7 Mounting zFS file systems using the ISHELL

Mounting file systems using the ISHELL was provided with the additional service of automatically changing the file system name entered to uppercase letters before doing the mount. When mounting zFS file systems living in a zFS multifile system aggregate. it may happen that you have file systems with names containing lowercase letters.

It is now necessary to keep file system names as they are entered or provide another means for mounting of this new file system type in parallel. The solution in z/OS V1R4 is that entering the name without enclosing leading and trailing quotes works as in the past. If quotes are provided, the file system name is used exactly as given, with the quotes removed.

In the examples shown in Figure 2-70 and Figure 2-71 on page 58 we assume that the mount point used is free, the file system exists, and mount authorization is OK. Figure 2-70 shows the ISHELL mount panel with the error message that you receive when no quotes are used.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
 ┌──────────────────────────────────────────────────────────────────────┐
 │                          Mount a File System                         │
 │                                                                      │
 │ Mount point:                                                         │
 │                                                          More:     + │
 │   /u/zfs/m03/                                                        │
 │   ____________________________________________________________       │
 │                                                                      │
 │ File system name   OMVS.SC43.MULTIF02.m03.ZFS                        │
 │ File system type   ZFS       New owner  . . . . . ________           │
 │ Owning system  . . ________  Character Set ID . . _____              │
 │                                                                      │
 │ Select additional mount options:                                    │
 │ _  Read-only file system         _  Set automove attribute...        │
 │ _  Ignore SETUID and SETGID      _  Text conversion enabled          │
 │ _  Bypass security                                                   │
 │                                                                      │
 │ Mount parameter:                                                     │
 │ ____________________________________________________________         │
 │                                                                      │
 └──────────────────────────────────────────────────────────────────────┘
 ┌──────────────────────────────────────────────────────────────────────┐
 │ Errno=81x No such file or directory exists; Reason=EF096055.  Press Enter │
 │ to continue.                                                         │
 └──────────────────────────────────────────────────────────────────────┘

Command ===> o
```

Figure 2-70 No success when mounting mixed-case named zFS file system without using quotes

However, when you press Enter in the situation shown in Figure 2-71 on page 58, the mount is successful.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
 ┌──────────────────────────────────────────────────────────────────────┐
 │                          Mount a File System                         │
 │                                                                      │
 │ Mount point:                                                         │
 │                                                            More:   + │
 │   /u/zfs/m03/                                                        │
 │   ____________________________________________________________       │
 │                                                                      │
 │ File system name   'OMVS.SC43.MULTIF02.m03.ZFS'                      │
 │ File system type   ZFS       New owner  . . . . . ________           │
 │ Owning system  . . ________  Character Set ID . . _____              │
 │                                                                      │
 │ Select additional mount options:                                    │
 │ _  Read-only file system         _  Set automove attribute...        │
 │ _  Ignore SETUID and SETGID      _  Text conversion enabled          │
 │ _  Bypass security                                                   │
 │                                                                      │
 │ Mount parameter:                                                     │
 │ ____________________________________________________________         │
 │                                                                      │
 └──────────────────────────────────────────────────────────────────────┘
Command ===> o
```

Figure 2-71 Successfully mounting mixed-case named zFS file system when using quotes

2.12 zFS file systems in sysplex sharing

Chapter 5, “Sysplex considerations” on page 201 explains the possibilities and restrictions regarding how to use zFS file systems in a sysplex sharing environment.

2.12.1 Write protection implementation

Currently, UNIX System Services file systems cannot be mounted R/W on more than one system at the same time. Even UNIX System Services sysplex R/W sharing does not violate this rule and is implemented as a function within the logical file system interface. This is implemented within the same resource serialization complex (or sysplex) and is not a problem if set up correctly.

However, for systems that are completely independent and access a file system at the same time accidentally, this can cause problems and be harmful for the file system. Therefore, a physical file system may implement a function called “write protection” that prevents this situation. This is done by writing an identifier into the file system when accessing it that contains, among other things, the sysplex and system name and a time stamp.

When an HFS file system is mounted in system A, an identifier is written into the HFS data set that is checked every time data is physically put into the file system. If system B then also accesses the file system, it is not obvious that another system has mounted the file system R/W already and it overwrites the identifier of system A.

The next time system A wants to update the HFS data set, the problem is recognized and system A stops writing to the file system to avoid destroying the internal structure.

This solution is not perfect because system A is the first one accessing the data set and still loses its R/W access.

With z/OS V1R6, zFS has implemented a different technique to guarantee a correct state for zFS aggregates that are accessed R/W in a system, as explained here:

- The identifier is cleared when a compatibility mode aggregate is successfully unmounted or a multifile system aggregate is detached.
- A daemon process updates the time stamp twice a minute.

These two actions are sufficient to ensure that a second system trying to get access in read/write mode can recognize whether there is another system writing to the aggregate already. This implementation with zFS assures that the first system always keeps its read/write access as desired.

2.12.2 Mounting zFS file systems copied outside the sysplex

If you have a specific system located in another sysplex that is used for preparing and servicing UNIX System Services file systems that are type zFS, then the first mount may take at least 65 seconds. This is caused by the write protection implementation in the following situation:

- The aggregate used in the service system is still mounted read-write after adding the service.
- You use ADRDSSU to copy the aggregate into a new aggregate to be used in the production environment afterwards, or dump it first and restore it later for use.

As long as the aggregate is attached read-write, the identifier is active in block zero (0). As a result, on ADRDSSU COPY or DUMP processing, it is copied as well.

When you mount or attach the aggregate in the target production system, zFS sees this identifier. Because it does not reflect the sysplex environment, zFS waits 65 seconds for an update of the time stamp value within the identifier. If the aggregate is used read-write outside of the sysplex, then this time stamp should get updated twice a minute, as mentioned earlier.

If no update occurs, this shows that the identifier is there without a reason. In this case, after the 65 seconds, zFS attaches and mounts the aggregate.

Avoiding the unnecessary wait on the first access to the aggregate

To avoid the 65 seconds wait time on access to the aggregate in the production environment, perform *one* of the following actions:

- Unmount or detach the source aggregate in the service system before using **ADRDSSU COPY** or **ADRDSSU DUMP**.
- Switch the source aggregate to read-only access, using the ISHELL modify interface from the mount table panel or the **/usr/sbin/chmount** shell command. Then copy the contents of the aggregate. This switch removes the write protection identifier, as well.
- If these actions are not feasible and you used **ADRDSSU COPY**, mount the aggregate read-write in this maintenance system and unmount it again. There is no delay in this situation because the identifier fits to the current environment.

2.13 Increasing the physical size of an aggregate

To make the physical size of an aggregate larger, use the **zfsadm grow** command. Looking at a LISTCAT output before and after running the command proves this. Figure 2-72 shows a LISTCAT output before issuing a **grow** subcommand. It shows the primary and secondary space in cylinders and the number of tracks that are currently allocated.

```
CLUSTER ------- OMVS.MUL02.ZFS
     IN-CAT --- CATALOG.TOTICF1.VTOTCAT
...
       DATA-----OMVS.MUL02.ZFS.DATA
   DATA ------- OMVS.MUL02.ZFS.DATA
     IN-CAT --- CATALOG.TOTICF1.VTOTCAT
...
     ALLOCATION
       SPACE-TYPE------CYLINDER     HI-A-RBA--------21381120
       SPACE-PRI-------------29     HI-U-RBA--------21381120
       SPACE-SEC-------------15
     VOLUME
       VOLSER------------TOTZF1     PHYREC-SIZE---------4096     HI-A-RBA--------21381120
       DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12     HI-U-RBA--------21381120
       VOLFLAG------------PRIME     TRACKS/CA-------------15
       EXTENTS:
       LOW-CCHH-----X'00400000'     LOW-RBA----------------0     TRACKS---------------435
       HIGH-CCHH----X'005C000E'     HIGH-RBA--------21381119
     VOLUME
       VOLSER------------TOTZF2     PHYREC-SIZE------------0     HI-A-RBA---------------0
       DEVTYPE------X'3010200F'     PHYRECS/TRK------------0     HI-U-RBA---------------0
       VOLFLAG--------CANDIDATE     TRACKS/CA--------------0
```

Figure 2-72 LISTCAT output for an aggregate

Figure 2-73 shows the **zfsadm grow** command to grow the aggregate size where the secondary allocation size is used. Specifying a size of 0 indicates to use the secondary allocation.

```
#> zfsadm grow -aggregate OMVS.MUL02.ZFS -size 0
IOEZ00173I Aggregate OMVS.MUL02.ZFS successfully grown
OMVS.MUL02.ZFS (R/W MULT): 29311 K free out of total 29672 (2000 reserved)
```

Figure 2-73 Grow the size of an aggregate

Figure 2-74 on page 61 shows a LISTCAT following the grow of the aggregate. You can now see the secondary allocation of tracks in the LISTCAT output.

```
CLUSTER ------- OMVS.MUL02.ZFS
     IN-CAT --- CATALOG.TOTICF1.VTOTCAT
...
       DATA-----OMVS.MUL02.ZFS.DATA
   DATA ------- OMVS.MUL02.ZFS.DATA
     IN-CAT --- CATALOG.TOTICF1.VTOTCAT
...
     ALLOCATION
       SPACE-TYPE------CYLINDER     HI-A-RBA--------32440320
       SPACE-PRI-------------29     HI-U-RBA--------32440320
       SPACE-SEC-------------15
     VOLUME
       VOLSER------------TOTZF1     PHYREC-SIZE---------4096     HI-A-RBA--------32440320
       DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12     HI-U-RBA--------32440320
       VOLFLAG------------PRIME     TRACKS/CA-------------15
       EXTENTS:
       LOW-CCHH-----X'00400000'     LOW-RBA----------------0     TRACKS---------------435
       HIGH-CCHH----X'005C000E'     HIGH-RBA--------21381119
       LOW-CCHH-----X'00740000'     LOW-RBA---------21381120     TRACKS---------------225
       HIGH-CCHH----X'0082000E'     HIGH-RBA--------32440319
     VOLUME
       VOLSER------------TOTZF2     PHYREC-SIZE------------0     HI-A-RBA---------------0
       DEVTYPE------X'3010200F'     PHYRECS/TRK------------0     HI-U-RBA---------------0
       VOLFLAG--------CANDIDATE     TRACKS/CA--------------0
```

Figure 2-74 LISTCAT output after a grow of the aggregate

2.13.1 Adding additional candidate volumes

If a zFS aggregate or HFS data set has used up all its candidate volumes that are known to zFS or HFS at mount time and the last volume has been filled up completely, then you need to add new candidate volumes.

Note: The same is true for HFS data sets, and you can use the same or similar techniques to add and use new volumes.

Figure 2-75 illustrates a situation when growing a zFS aggregate is not successful because all the space on the last volume is used up and there are no further candidate volumes available for an extension.

Message IEC161I with return code = 104 says that no more volumes are available on which to allocate space.

```
#> /usr/sbin/mount -f OMVSM.HERING.TEST.ZFS -t ZFS /u/hering/testfs
#> zfsadm grow -aggregate OMVSM.HERING.TEST.ZFS -size 0
IOEZ00445E Error extending OMVSM.HERING.TEST.ZFS.
See MVS system message IEC161I with return code = 104 and PDF code = 204.
```

Figure 2-75 Unsuccessfully growing a zFS aggregate when no space is left

Add a volume

Use the **ALTER** command to provide additional volumes for the data set, as shown in Figure 2-76 on page 62.

For the **ALTER** command used in Figure 2-76, DFSMS is choosing the particular candidate volumes because in place of a specific volser, an asterisk (*) is used. To add multiple volumes, specify them by the number of asterisks.

```
ALTER 'OMVSM.HERING.TEST.ZFS.DATA' ADDVOLUMES(*)
IDC0526I ALTERED ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0531I ENTRY OMVSM.HERING.TEST.ZFS.DATA ALTERED
```

Figure 2-76 Using ALTER from TSO to add a candidate volume for a zFS aggregate

In the case of a mounted zFS aggregate, this further means that it must be unmounted and mounted again so that the new volumes put in the catalog can be used. This is the same situation as with HFS or other data sets that are open. This restriction is demonstrated in Figure 2-77, showing commands run after the **ALTER** command.

```
#> zfsadm grow -aggregate OMVSM.HERING.TEST.ZFS -size 0
IOEZ00445E Error extending OMVSM.HERING.TEST.ZFS.
See MVS system message IEC161I with return code = 210 and PDF code = 213.
#> /usr/sbin/unmount /u/hering/testfs
#> /usr/sbin/mount -f OMVSM.HERING.TEST.ZFS -t ZFS /u/hering/testfs
#> zfsadm grow -aggregate OMVSM.HERING.TEST.ZFS -size 0
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 85606 K free out of total 86400
```

Figure 2-77 Successfully growing a zFS aggregate after ALTER, unmount and new mount

Adding necessary candidate volumes while a file system is in use

Unmounting file systems in a production environment may be undesirable. Therefore, using the following approach for a UNIX System Services sysplex file system sharing environment is a much better alternative compared to the standard technique for making new volumes or candidate volumes known.

The UNIX System Services return code in the message 133 (=85x) shown in Figure 2-78 is ENOSPC, and this means that there is no space left on the device.

```
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#>
```

Figure 2-78 Errors on extending a zFS aggregate

Therefore, a new candidate volume needs to be added, as shown in Figure 2-79.

```
#> tsocmd "ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)"
ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)
ALTERED ALLOCATION STATUS FOR VOLUME * IS 0
ENTRY OMVS.TEST.ZFS.DATA ALTERED
READY
END
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#>
```

Figure 2-79 After adding a candidate volume, extending still does not work

Notice that the result shown in Figure 2-79 on page 62 is the same, however, because the candidate volume is not yet known.

You can force the necessary new open of the data set by performing *one* of the following actions:

- Perform a double move (set a new owing system and move back the file system to the original system).
- Or, perform a double switch (switch to read-only and then back to read-write).

Afterwards you can successfully add an extension, as shown in Figure 2-80. This is because the MVS data sets are accessed and opened just as on a normal MOUNT.

```
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00173I Aggregate OMVS.TEST.ZFS successfully grown
OMVS.TEST.ZFS (R/W COMP): 250462 K free out of total 252000
```

Figure 2-80 Successfully extending the aggregate

Important notes:

- When moving the ownership of the file system, this technique is transparent for applications using the file systems.
- Beginning with z/OS V1R11, when a read-write zFS file system is mounted sysplex-aware, you can no longer use moving an aggregate to force a re-open of the data set. However, if all systems are running on level R11 at least, you can use the “remount samemode” function instead.
- Use “remount samemode” always as it is even more efficient because no double move is needed.
- Instead of candidate volumes you can also use a data class with a high Dynamic Volume Count (DVC) because this is supported with zFS and does not need space in the catalog (in comparision to a candidate volume) entry when a volume is not yet used.

2.13.2 Dynamic aggregate extension

Before this change in z/OS V1R4, if aggregates became full they could only be grown by using the **`zfsadm grow`** command. You needed to specify a larger size or specify zero (0) for the size to get a secondary allocation size extension.

z/OS V1R4 introduced the possibility to dynamically grow an aggregate if it becomes full. The aggregate is extended automatically when an operation cannot complete because the aggregate is full. If the extension is successful, the operation continues transparent to the application.

Important: To dynamically grow an aggregate when it becomes full, the VSAM LDS must have a secondary allocation and have space on the volumes.

Implementing a dynamic aggregate extension

A dynamic aggregate extension can be enabled in the following ways:

- In the IOEFSPRM configuration file, you can dynamically extend an aggregate when it becomes full by doing one of the following:
  - You can specify a new option: `aggrgrow=on | off`. The default value is off.

Note: The option specified here is the default if none of the following ways of specifying the `aggrgrow | noaggrgrow` options are used.

- You can specify either **`aggrgrow`** or **`noaggrgrow`** as a suboption on the **`define_aggr`** option for a multifile system aggregate, as shown in the following definition:

```
define_aggr R/W attach aggrgrow cluster(OMVS.TEST.ZFS)
```

- Using the **`mount`** command, in the PARM keyword you can specify either **`aggrgrow`** or **`noaggrgrow`**, as shown in the following example

```
mount filesystem('omvs.test.zfs') mountpoint('/tmp/test') type(zfs) mode
(rdwr) parm('aggrgrow')
```

Note: This **`aggrgrow | noaggrgrow`** option can only be used with compatibility mode aggregates.

- Using the **`zfsadm attach`** command for attaching a multifile system aggregate, you can specify either the **`-aggrgrow`** or **`-noaggrgrow`** option, as shown in the following example.

```
zfsadm attach -aggregate OMVS.TEST.ZFS -aggrgrow
```

- Using the **`zfsadm config`** command, you can dynamically change the configuration file option **`aggrgrow on | off`**. This becomes the new default if no other option specification is in use.

Aggregate extension processing

When an aggregate fills and dynamic aggregate extension has been specified using one of the options, the aggregate is extended using secondary allocation extensions, the extensions taken are formatted, and it becomes available transparently to the application. The messages that are issued indicating the process are shown in Figure 2-81.

```
IOEZ00312I Dynamic growth of aggregate OMVS.TEST.ZFS in progress, (by user AYVIVAR).
IOEZ00329I Attempting to extend OMVS.TEST.ZFS by a secondary extent.
IOEZ00324I Formatting to 8K block number 360 for secondary extents of OMVS.TEST.ZFS
IOEZ00309I Aggregate OMVS.TEST.ZFS successfully dynamically grown (by user AYVIVAR).
```

Figure 2-81 Messages issued when dynamically growing an aggregate

More detailed examples forcing dynamic aggregate extension

We performed tests with UNIX commands and utilities running copy processing to a zFS aggregate that did not have enough space left to keep all the new data without the need of extending it.

Assume we have an HFS data set OMVS.HERING.TEST mounted at /u/hering/testfs with about 16 cylinders of real space used for data. In all tests we define a new zFS aggregate that is too small but has set automatic extension on, as shown in Figure 2-82.

```
#> zfsadm define -aggregate OMVS.HERING.TEST.ZFS -storageclass OPENMVS -cylinders 1 1
#> zfsadm format -aggregate OMVS.HERING.TEST.ZFS -compat -owner 888 -perms o755
IOEZ00077I HFS-compatibility aggregate OMVS.HERING.TEST.ZFS has been successfully
created
#> /usr/sbin/mount -f OMVS.HERING.TEST.ZFS -t ZFS -o aggrgrow /u/hering/testfs.target
```

Figure 2-82 Creating and mounting a small zFS aggregate with automatic extension set on

Now we use the **copytree** utility to copy the contents of the HFS data set OMVS.HERING.TEST into the newly defined zFS aggregate. As it turns out, this is done without any further problems, as shown in Figure 2-83 on page 65.

```
#> /samples/copytree -a /u/hering/testfs /u/hering/testfs.target
Copying /u/hering/testfs to /u/hering/testfs.target
Scanning for file nodes...
Skipping mountpoint: /u/hering/testfs/..
Processing 378 nodes
Creating directories
Creating other files
Setting file attributes

******************

Copy complete.  Error count= 0
Directory errors:   0
Directories copied: 31
File errors:        0
Files copied:       328
Symlink errors:     0
Symlinks copied:    19
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
ACL count:          4
```

Figure 2-83 Copying data to a small zFS aggregate using “copytree”

The same is true when using the **pax** command to do the same thing; see Figure 2-84.

```
#> cd /u/hering/testfs && pax -pe -rwX . /u/hering/testfs.target
#>
```

Figure 2-84 Copying data to a small zFS aggregate using the pax utility

In the syslog you will find the messages indicating 15 automatic grows that are necessary to enlarge the zFS aggregate in parallel step by step. Excerpts are shown in Figure 2-85 for reference.

```
IOEZ00312I Dynamic growth of aggregate OMVS.HERING.TEST.ZFS in progress, (by user
HERING).
IOEZ00329I Attempting to extend OMVS.HERING.TEST.ZFS by a secondary extent.
IOEZ00324I Formatting to 8K block number 180 for secondary extents of
OMVS.HERING.TEST.ZFS
IOEZ00309I Aggregate OMVS.HERING.TEST.ZFS successfully dynamically grown (by user
HERING).
...
IOEZ00312I Dynamic growth of aggregate OMVS.HERING.TEST.ZFS in progress, (by user
HERING).
IOEZ00329I Attempting to extend OMVS.HERING.TEST.ZFS by a secondary extent.
IOEZ00324I Formatting to 8K block number 1440 for secondary extents of
OMVS.HERING.TEST.ZFS
IOEZ00309I Aggregate OMVS.HERING.TEST.ZFS successfully dynamically grown (by user
HERING).
```

Figure 2-85 Syslog messages indicating automatic grows

2.13.3 Dynamic file system quota increase

The maximum size of a file system is known as its *quota*. This is a logical number that is checked against each time additional blocks are allocated to the file system. A quota can be smaller than, equal to, or larger than the space available in the aggregate. When the quota is reached, the file system indicates that it is full.

z/OS V1R4 introduces several ways to dynamically increase the file system quota if the file system becomes full for multifile system aggregates. Dynamic file system quota increase can be specified in the following ways:

- A new option in the IOEFSPRM configuration file, fsgrow=(increment,times), specifies whether file systems in a multifile aggregate can have their quota dynamically extended. The value that is specified in this file becomes the default if it is not changed in some other way, where:

  increment The number of k-bytes to increase the quota. The maximum value that can be specified is 2147483647.

  times The number of times to extend the quota

- Using the **mount** command, in the PARM keyword you can specify the file system quota extension of 500 KB a maximum of four times, as fsgrow(increment,times), as shown in the following example:

  ```
  mount filesystem(zfstest) mountpoint('/tmp/zfstest') type(zfs) mode(rdwr)
  parm('fsgrow(500,4)') noautomove
  ```

- Using the **zfsadm config** command, you can dynamically change the configuration file option by specifying, -fsgrow increment times, as shown in the following example

  ```
  zfsadm config -fsgrow 500,4
  ```

  For example, fsgrow(500,4) means grow the quota by 500 K bytes up to 4 times.

Displaying the quota

Figure 2-86 shows the file system quota to be 1159, which was defined for a compatibility mode aggregate.

```
$> zfsadm lsquota -filesystem OMVS.TEST.ZFS
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.TEST.ZFS         1159       9     0    11 = 146/1296 (zFS)
```

Figure 2-86 Display quota information about file systems and aggregates

Compatibility mode aggregates

For a compatibility mode aggregate, only the **aggrgrow** specification is used to extend an aggregate size. The **fsgrow** option is ignored if it is specified. The quota increases by the size of the extension. For compatibility mode aggregates, the file system quota is dynamically increased based on a new aggregate size once it becomes full, as shown in Figure 2-87 on page 67.

```
Before dynamic extension:
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.TEST.ZFS         1159       9     0    11 = 146/1296 (zFS)
After dynamic extension:
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.TEST.ZFS         1807    1577    87    88 = 1714/1944 (zFS)
```

Figure 2-87 Compatibility mode aggregate dynamic extension

Multifile system aggregates

For a multifile system aggregate, there may be multiple file systems in the aggregate. Therefore, if a file system becomes full, equal to its quota, an **fsgrow** option must be in place for the file system to then use the additional physical space that is available in the aggregate, following the dynamic increase of the individual quota for the file system.

2.13.4 Displaying dynamic aggregate and quota extensions

Values set for dynamic aggregate or quota extensions can be displayed if you want to know which values have been assigned in each case.

Global parameters defined in IOEFSPRM or modified later with the **zfsadm config** command can be displayed with the **zfsadm configquery** command. As shown in Figure 2-88, you can obtain the current values for both the `aggrgrow` and `fsgrow` parameters.

```
@ SC65:/>zfsadm configquery -aggrgrow -fsgrow
IOEZ00317I The value for config option -aggrgrow is ON.
IOEZ00317I The value for config option -fsgrow is (500,4).
```

Figure 2-88 Display the current aggrgrow and fsgrow options

Specific values that are set by using the **mount** command for a zFS file system can be displayed with the **df -v** command, as shown in Figure 2-89. Alternatively, you can use the z/OS **d omvs,f** command, as shown in Figure 2-90 on page 68.

```
@ SC65:/>df -v /tmp/zfs3
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/tmp/zfs3 (ZFS3)                    2310/5000      4294967291 Available
ZFS, Read/Write, Device:178, ACLS=Y
fsgrow(500,4)
File System Owner : SC65        Automove=N      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.TEST7.ZFS

@ SC65:/>df -v /tmp/test8
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/tmp/test8 (OMVS.TEST8.ZFS)          460/3614       4294967291 Available
ZFS, Read/Write, Device:170, ACLS=Y
aggrgrow
File System Owner : SC65        Automove=N      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.TEST8.ZFS
```

Figure 2-89 Display the current values of the aggrgrow and fsgrow options

```
d omvs,f
..............
ZFS           170 ACTIVE                      RDWR
  NAME=OMVS.TESTC.ZFS
  PATH=/SC65/tmp/testc
  AGGREGATE NAME=OMVS.TESTC.ZFS
  MOUNT PARM=aggrgrow
  OWNER=SC65     AUTOMOVE=N CLIENT=N
ZFS           178 ACTIVE                      RDWR
  NAME=ZFSTEST
  PATH=/SC65/tmp/zfstest
  AGGREGATE NAME=OMVS.TEST.ZFS
  MOUNT PARM=fsgrow(500,4)
  OWNER=SC65     AUTOMOVE=N CLIENT=N
.................
```

Figure 2-90 Display options using the d omvs,f z/OS command

The **f bpxoinit,filesys=display,filesystem=filesystem** command also shows information about the **aggrgrow** and **fsgrow** parameters; see Figure 2-91.

```
f bpxoinit,filesys=display,filesystem=zfsa
BPXMO27I COMMAND ACCEPTED.
BPXFO35I 2002/05/24 14.26.12 MODIFY BPXOINIT,FILESYS=DISPLAY
-------------NAME-------------------------        DEVICE  MODE
ZFSA                                                198  RDWR
  AGGREGATE NAME=OMVS.TESTA.ZFS
  PATH=/SC65/tmp/testa
  PARM=fsgrow(500,4)
  STATUS=ACTIVE                LOCAL STATUS=ACTIVE
  OWNER=SC65        RECOVERY OWNER=SC65     AUTOMOVE=N PFSMOVE=Y
  TYPENAME=ZFS      MOUNTPOINT DEVICE=      72
  MOUNTPOINT FILESYSTEM=/SC65/TMP
  ENTRY FLAGS=90060000  FLAGS=40000010  LFSFLAGS=00000000
  LOCAL FLAGS=40000010  LOCAL LFSFLAGS=20000000
BPXFO40I MODIFY BPXOINIT,FILESYS PROCESSING IS COMPLETE.
```

Figure 2-91 Display aggrgrow and fsgrow options using the f bpxoinit command

2.13.5 Growing zFS aggregates in a loop

When growing zFS aggregates using the secondary extent value specified, you may want to add several extents. To do so easily, we provide a small UNIX REXX script named zfsgrow that allows you simply to specify the number of grow runs to perform, together with the name of the zFS aggregate. The contents of the procedure are provided in B.12, “UNIX script zfsgrow” on page 409.

Figure 2-92 on page 69 shows how to use the command, provided the directory of file zfsgrow is referenced by your current PATH setting. The secondary extent value of the zFS aggregate has a value of 4.

```
#> zfsgrow 5 OMVSM.HERING.TEST2.ZFS
Performing grow run 1...
IOEZ00173I Aggregate OMVSM.HERING.TEST2.ZFS successfully grown
OMVSM.HERING.TEST2.ZFS (R/W COMP): 5606 K free out of total 5760
Performing grow run 2...
IOEZ00173I Aggregate OMVSM.HERING.TEST2.ZFS successfully grown
OMVSM.HERING.TEST2.ZFS (R/W COMP): 8486 K free out of total 8640
Performing grow run 3...
IOEZ00173I Aggregate OMVSM.HERING.TEST2.ZFS successfully grown
OMVSM.HERING.TEST2.ZFS (R/W COMP): 11366 K free out of total 11520
Performing grow run 4...
IOEZ00173I Aggregate OMVSM.HERING.TEST2.ZFS successfully grown
OMVSM.HERING.TEST2.ZFS (R/W COMP): 14246 K free out of total 14400
Performing grow run 5...
IOEZ00173I Aggregate OMVSM.HERING.TEST2.ZFS successfully grown
OMVSM.HERING.TEST2.ZFS (R/W COMP): 17126 K free out of total 17280
```

Figure 2-92 Growing a zFS aggregate in a loop using REXX utility zfsgrow

This technique may also be useful when you want to format more than just the primary extent before using the aggregate when initially formatting bigger zFS aggregates. For this purpose it may be an alternate possibility to 2.7.3, “Formatting aggregates with the -grow option” on page 30, if you do not like calculating the absolute 8 K value and an appropriate grow setting. Furthermore, the elapsed times needed for both approaches are similar.

Formatting an aggregate using -grow

Figure 2-93 shows the definition and formatting of a zFS aggregate using the **-grow** option. To request a total size of 4000 cylinders, request a size of 360000 8 K blocks and use a **-grow** value of 36000 8 K blocks, which is 400 cylinders.

```
#> zfsadm define -aggregate OMVSM.HERING.TEST.ZFS -dataclass ZFSTEST -storageclass \
> ZFSTEST -cylinders 400 400
IOEZ00248E VSAM linear dataset OMVSM.HERING.TEST.ZFS successfully created.
#> zfsadm format -aggregate OMVSM.HERING.TEST.ZFS -compat -owner 888 -perms o755 \
> -size 360000 -grow 36000
IOEZ00077I HFS-compatibility aggregate OMVSM.HERING.TEST.ZFS has been successfully
created
#> /usr/sbin/mount -f OMVSM.HERING.TEST.ZFS -t ZFS /u/hering/testfs
#> zfsadm aggrinfo -aggregate OMVSM.HERING.TEST.ZFS
OMVSM.HERING.TEST.ZFS (R/W COMP): 2850710 K free out of total 2880000
```

Figure 2-93 Initially formatting a zFS aggregate using the -grow option

Formatting an aggregate using the zfsgrow REXX script

Figure 2-94 on page 71 shows the commands used when exploiting the zfsgrow REXX script function. This time you simply format the aggregate with no size specification and this request grows the aggregate by 9 secondary extents of 400 cylinders.

```
#> zfsadm define -aggregate OMVSM.HERING.TEST.ZFS -dataclass ZFSTEST -storageclass \
> ZFSTEST -cylinders 400 400
IOEZ00248E VSAM linear dataset OMVSM.HERING.TEST.ZFS successfully created.
#> zfsadm format -aggregate OMVSM.HERING.TEST.ZFS -compat -owner 888 -perms o755
IOEZ00077I HFS-compatibility aggregate OMVSM.HERING.TEST.ZFS has been successfully
created
#> /usr/sbin/mount -f OMVSM.HERING.TEST.ZFS -t ZFS /u/hering/testfs
#> zfsgrow 9 OMVSM.HERING.TEST.ZFS
Performing grow run 1...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 572966 K free out of total 576000
Performing grow run 2...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 860926 K free out of total 864000
Performing grow run 3...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 1148886 K free out of total 1152000
Performing grow run 4...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 1436846 K free out of total 1440000
Performing grow run 5...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 1724806 K free out of total 1728000
Performing grow run 6...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 2012766 K free out of total 2016000
Performing grow run 7...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 2300726 K free out of total 2304000
Performing grow run 8...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 2588686 K free out of total 2592000
Performing grow run 9...
IOEZ00173I Aggregate OMVSM.HERING.TEST.ZFS successfully grown
OMVSM.HERING.TEST.ZFS (R/W COMP): 2876646 K free out of total 2880000
```

Figure 2-94 Initially formatting a zFS aggregate using the zfsgrow REXX script utility

2.13.6 zFS aggregates with a size greater than 4 GB

zFS file systems that are greater than 4 GB (about 5825 cylinders of a 3390), or which may be expected to grow over this limit, must be defined with a data class that includes extended addressability:

```
Data Set Name Type=EXTENDED REQUIRED
Extended Addressability=YES
```

2.14 Working with zFS aggregates using the ISHELL

In the File_systems pull-down menu of the ISHELL shown in Figure 2-95 on page 72, there are two zFS-related choices available to create zFS aggregates, as follows:

- Option 4 - New zFS was introduced with z/OS V1R4.

  This option allows the administrator to create zFS aggregates.

- Option 5 - zFS aggregates were introduced with z/OS V1R5.

  This option allows an administrator to do the following:

  – The aggregate list shows all zFS aggregates that are currently attached explicitly or implicitly as an HFS-compatible aggregate. The amounts of free space and total space are also shown in units of kilobytes.

  – You can select an option or one or more aggregates with an action code. The valid action codes are:

    A Show the attributes for the aggregate.

    L List the file systems in the aggregate. The file system list will allow you to perform actions on file systems.

  This option is useful because you do not need to know the syntax of the **zfsadm** commands.

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
                                        5  1. Mount table...
                      UNIX System Serv     2. New HFS...
Command ===> ____________________________  3. Mount(O)...
                                           4. New ZFS...
Enter a pathname and do one of these:      5. ZFS aggregates...
    - Press Enter.
    - Select an action bar choice.
    - Specify an action code or command on the command line.

Return to this panel to work with a different pathname.
                                                              More:     +
   /u
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________

EUID=0
```

Figure 2-95 ISHELL options for working with zFS aggregates

2.14.1 Creating zFS aggregates

This function has been added to ISHELL with z/OS V1R4. As for HFS, you can now create compatibility mode aggregates. Figure 2-96 on page 73 shows the File_systems pull-down menu with option 4 already entered.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
_____________________________________ _____________________ ________
                     UNIX System Serv | 4  1. Mount table...    |
                                      |    2. New HFS...        |
Enter a pathname and do one of these: |    3. Mount(0)...       |
                                      |    4. New zFS...        |
    - Press Enter.                    |    5. zFS aggregates... |
    - Select an action bar choice.    |_________________________|
 - Specify an action code or command on the command line.
Return to this panel to work with a different pathname.
                                                          More:     +
   /u/hering/test
   ____________________________________________________________
   ____________________________________________________________
   ____________________________________________________________

EUID=0

Command ===> ____________________________________________________________
```

Figure 2-96 File_systems pull-down menu with the new zFS options

On the panel displayed you can enter the settings for the new compatibility mode aggregate and the file system, as shown in Figure 2-97.

Note: Instead of using option 4 in the File_systems menu, you can use the command **mkzfs** on the command line of the main ISHELL panel.

```
File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
s  ____________________________________________________________________
   |                Create a zFS Aggregate and File System              |
   |                                                                    |
E  | Enter the fields as required then press Enter.                     |
   |                                                                    |
   | Aggregate name  . . . . . omvs.hering.testish.zfs_____________    |
   | Owning User . . . . . . . hering__  (Number or user name)          |
   | Owning Group  . . . . . . sys1____  (Number or group name)         |
   | Permissions . . . . . . . 755   (3 digits, each 0-7)               |
R  | Primary cylinders . . . . 10______                                 |
   | Secondary cylinders . . . 5_______                                 |
   | Storage class . . . . . . openmvs_                                 |
   | Management class  . . . . ________                                 |
   | Data class  . . . . . . . multvsam                                 |
   | Volume names  . . . . . . ______  ______  ______  ______  ______  |
   |                           ______  ______  ______  ______  ______   |
E  |                           ______  ______  ______  ______  ______   |
   |                                                                    |
   |____________________________________________________________________|

Command ===> MKZFS
```

Figure 2-97 Creating a new zFS aggregate and file system

When you press Enter for the panel in Figure 2-97, this creates and formats the compatibility mode zFS aggregate. The file system can then be mounted.

2.14.2 Support for managing zFS aggregates and file systems

This function has been added to the ISHELL with z/OS V1R5. To make use of it, select option **5** in the File_systems menu, and an aggregate list is displayed as shown in Figure 2-98.

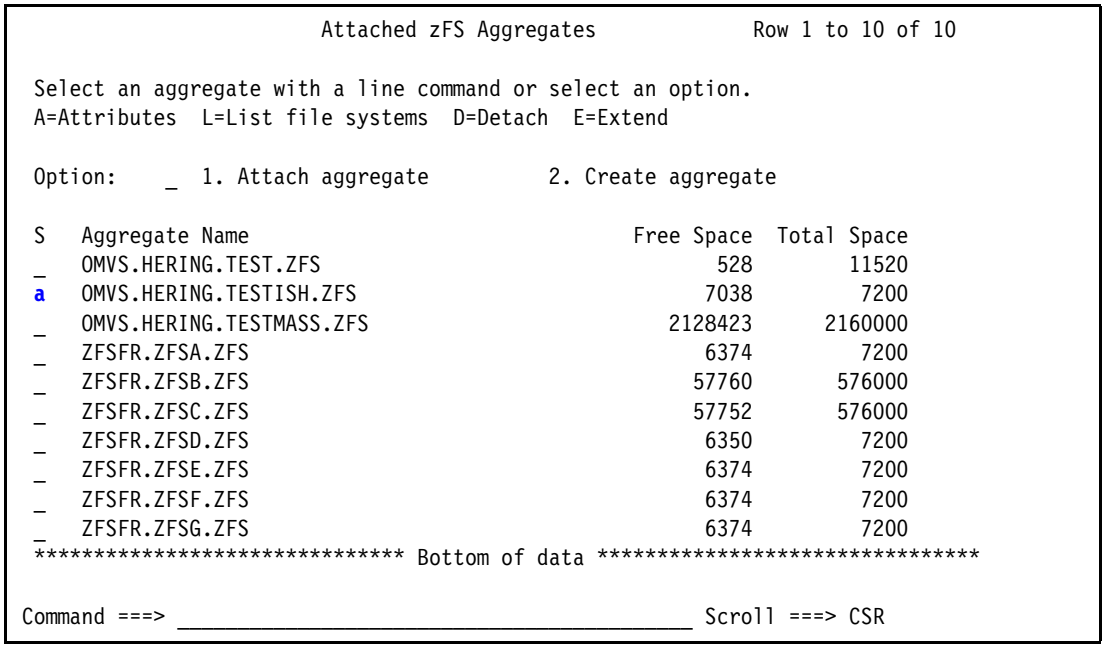


Figure 2-98 Listing zFS aggregates

Listing aggregate attributes

After selecting an aggregate with option **A**, you get the list of this aggregate's attributes as shown in Figure 2-99 on page 75.

```
Attached zFS Aggregates             Row 1 to 10 of 10

S ┌──────────────────────────── Aggregate Attributes ──────────────────────────┐
A │                                                                            │
  │ Aggregate name  . . . . : OMVS.HERING.TESTISH.ZFS                          │
O │ Attach mode . . . . . . : Read/write                                       │
  │ Monitored for full  . . : Disabled                                         │
S │ New block security  . . : Enabled                                          │
_ │ HFS compatibility . . . : Enabled                                          │
A │ Auto-extend . . . . . . : Enabled                                          │
_ │ Number of file systems  :          1                                       │
_ │ Threshold . . . . . . . :          0                                       │
_ │ Increment . . . . . . . :          0                                       │
_ │ Number of fragments . . :       7200                                       │
_ │ Fragment size . . . . . :       1024                                       │
_ │ Block size  . . . . . . :       8192                                       │
_ │ Blocks available  . . . :       7200                                       │
_ │ Maximum fragments . . . :       7038                                       │
* │ Minimum fragments . . . :          0                                       │ *
  └────────────────────────────────────────────────────────────────────────────┘

Command ===> ____________________________________________ Scroll ===> CSR
```

Figure 2-99 Listing aggregate attributes

Listing zFS file systems

Using option **L** instead of option A on the aggregate list in Figure 2-98 on page 74, you can list the file systems contained in an aggregate. Figure 2-100 shows the R/W file system of the compatibility mode aggregate that we defined in Figure 2-97 on page 73.

```
                          Attached zFS Aggregates             Row 1 to 10 of 10
 ┌──────────────────────────────────────────────────────────────────────────┐
S │                            File System List              Row 1 to 1 of 1 │
A │                                                                          │
  │ Select a file system with a line command or select an option.            │
O │ A=Attributes  C=Clone  D=Delete  Q=Set Quota  R=Rename                    │
  │                                                                          │
S │ Option:    _  1. Create file system                                      │
_ │                                                                          │
L │ S   File System Name                              Space Used       Quota │
_ │ a   OMVS.HERING.TESTISH.ZFS                                9        7047 │
_ │ ************************** Bottom of data ***************************** │
  │ Command ===> ____________________________________________ Scroll ===> CSR  │
  └──────────────────────────────────────────────────────────────────────────┘

Command ===> ____________________________________________ Scroll ===> CSR
```

Figure 2-100 File system list of an aggregate

Listing file system attributes

Using option **A** on the File System List panel shown in Figure 2-100 results in listing a file system's attributes, as shown in Figure 2-101 on page 76.

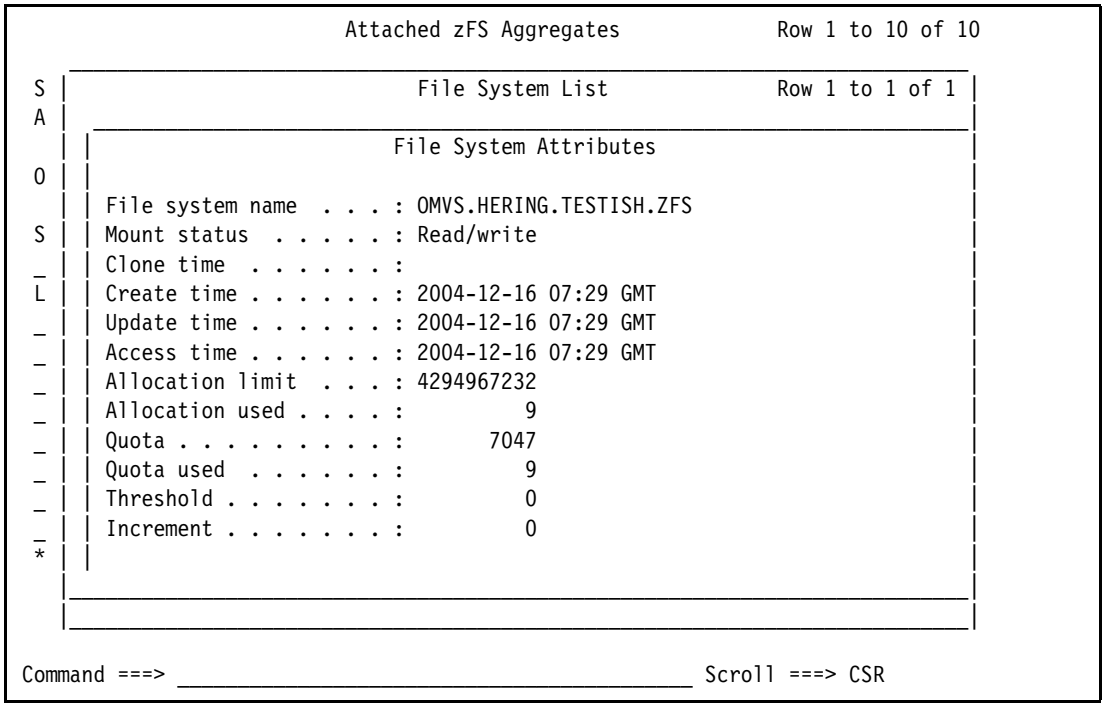


Figure 2-101 Listing a file system's attributes

Creating, replacing, and deleting a clone for a file system

You can clone the file system on the panel displayed in Figure 2-100 on page 75 by entering c in the option column. The result is shown in Figure 2-102.

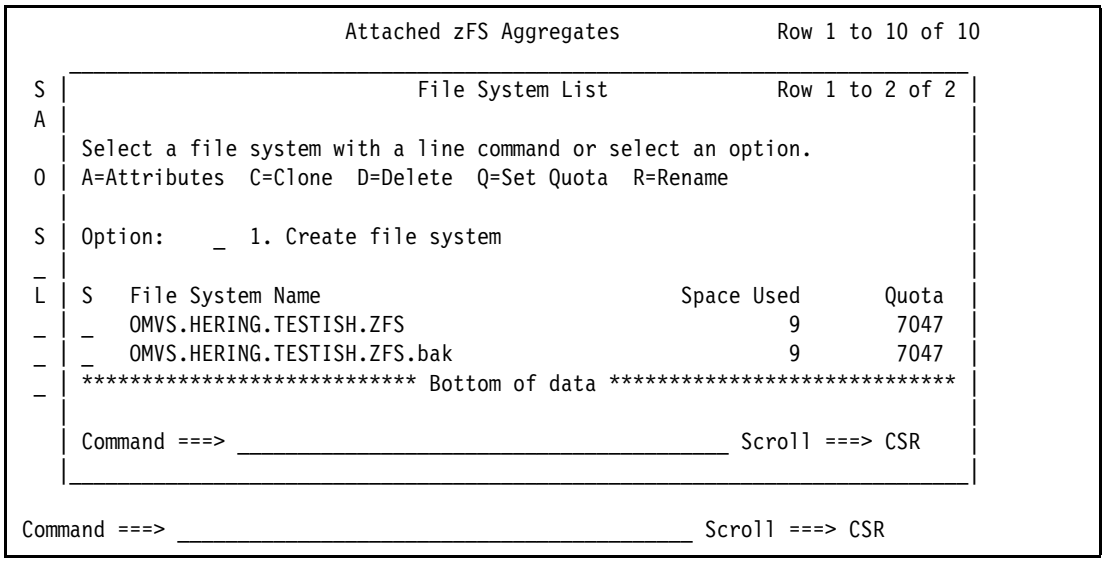


Figure 2-102 File system list with a new clone added

Working with clones

Using c to clone the clone will not work. You get the following error message in your ISHELL session:

```
Errno=81x ENOENT: No such file, directory, or IPC member exists.
Reason=EF1A6288x.  Press Enter to continue.
```

The zFS reason code 6288x means `File system not read-write`. The file system to be cloned is not a read-write file system, it is a backup file system.

Using option `c` again for the R/W file system will replace the clone with a new version. When you enter `d`, the clone is deleted again. Figure 2-103 shows the prompt panel displayed in this situation.

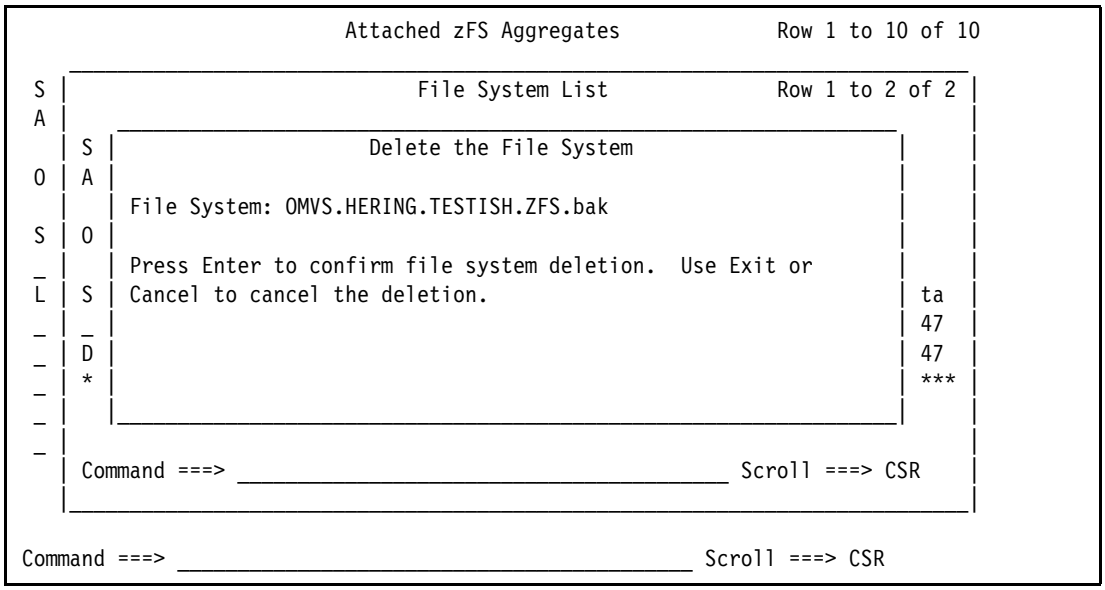


Figure 2-103 Prompt displayed when deleting the clone

Errors with options in specific situations

Many options available on these aggregate and file system panels are designed especially for the case where you deal with multifile system aggregates. For example, when a compatibility mode file system is mounted, you cannot rename it using option `R` and you receive the following message:

```
Errno=72x EBUSY: The resource is busy.  Reason=EF1A6225x.  Press Enter to
continue.
```

The zFS reason code 6225x simply says `R/W or backup file system is mounted`. You must unmount the file system and its backup file system before renaming the file system.

If the R/W file system is mounted R/O and you try to clone it, you receive the following message:

```
Errno=72x EBUSY: The resource is busy.  Reason=EF1A6285x.  Press Enter to
continue.
```

Here zFS reason code 6285x means `Aggregate is read only`. The aggregate must be attached in read-write mode before a file system in the aggregate can be cloned.

Growing an aggregate

The initial format done when creating the zFS aggregate (shown in Figure 2-97 on page 73) only formatted the primary extent specified. The 7200 fragments of 1 K (shown in Figure 2-99 on page 75) are exactly 10 cylinders. (There are 90 8 K blocks per cylinder.)

Using option **E** for an aggregate on the aggregate list displays a panel with the actual size of the aggregate. It allows you to specify the new total size in KB, as shown in Figure 2-104.

```
                        Attached zFS Aggregates            Row 1 to 10 of 10
 ____________________________________________________________________________
S |              Enter the New Aggregate Size in Kilobytes                     |
A |                                                                            |
  | Aggregate  . . : OMVS.HERING.TESTISH.ZFS                                   |
O | New size . . . . 7200                                                      |
  |                                                                            |
S |                                                                            |
_ |                                                                            |
E |                                                                            |
_ |                                                                            |
_ |____________________________________________________________________________|
_   ZFSFR.ZFSC.ZFS                                      57752       576000
_   ZFSFR.ZFSD.ZFS                                       6350         7200
_   ZFSFR.ZFSE.ZFS                                       6374         7200
_   ZFSFR.ZFSF.ZFS                                       6374         7200
_   ZFSFR.ZFSG.ZFS                                       6374         7200
****************************** Bottom of data ********************************

Command ===> ____________________________________________ Scroll ===> CSR
```

Figure 2-104 Growing the size of an aggregate

In this example, entering a value of 90000 worked to extend the aggregate to this new size, as shown in Figure 2-105.

```
                        Attached zFS Aggregates            Row 1 to 10 of 10

Select an aggregate with a line command or select an option.
A=Attributes  L=List file systems  D=Detach  E=Extend

Option:    _  1. Attach aggregate          2. Create aggregate

S   Aggregate Name                                 Free Space  Total Space
_   OMVS.HERING.TEST.ZFS                                  528        11520
_   OMVS.HERING.TESTISH.ZFS                             89830        90000
_   OMVS.HERING.TESTMASS.ZFS                          2128423      2160000
_   ZFSFR.ZFSA.ZFS                                       6374         7200
_   ZFSFR.ZFSB.ZFS                                      57760       576000
_   ZFSFR.ZFSC.ZFS                                      57752       576000
_   ZFSFR.ZFSD.ZFS                                       6350         7200
_   ZFSFR.ZFSE.ZFS                                       6374         7200
_   ZFSFR.ZFSF.ZFS                                       6374         7200
_   ZFSFR.ZFSG.ZFS                                       6374         7200
****************************** Bottom of data ********************************

Command ===> ____________________________________________ Scroll ===> CSR
```

Figure 2-105 Resulting aggregate display after the extension

Note: In a situation where an error occurs that is not obvious from the information displayed by the ISHELL, look for syslog entries for any messages. The messages in the syslog often provide more detailed information.

2.15 Accessing zFS files

zFS files in zFS file systems are accessed exactly as HFS files are accessed. Therefore, shell users and UNIX programs require no changes in the way access is done to files. HFS file systems can be copied into empty zFS file systems, as explained in 3.1, “Creating zFS file systems” on page 128.

2.15.1 Access control lists (ACLs)

To provide a better granularity of access control for z/OS UNIX files and directories, access control lists (ACLs) were introduced with z/OS V1R3. You can use access control lists to control access to files and directories by individual UIDs and GIDs. This provides the means to allow specific users and groups different types of access.

To manage an ACL for a file, you must have one of the following security access controls:

- Be the file owner
- Have superuser authority (UID=0)
- Have READ access to SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class

Beginning with z/OS V1R3, ACLs are supported by the HFS and zFS file systems. You must also know whether your security product supports ACLs and what rules are used when determining file access.

ACL works similar to the way access to MVS data sets is permitted, although the implementation is different. The ACL is a part of the File Security Packet (FSP), which is maintained by the Physical File System (PFS).

RACF class FSSEC

To activate the use of ACLs in z/OS UNIX file authority checks, the following RACF command needs to be run to activate the new RACF class FSSEC:

```
SETROPTS CLASSACT(FSSEC)
```

ACL shell commands

As with other RACF general resource classes, the FSSEC class does not need to be active to create profiles, or in this case, ACLs. The new shell commands are **`setfacl`** and **`getfacl`**, and a changed command **`getconf`** can be used to set and query ACLs as follows:

setfacl This sets an ACL definition for a file or directory.

getfacl This obtains an ACL for a file or directory.

getconf This changed command returns configuration values associated with the file at the specified pathname, as follows:

_PC_ACL This indicates whether an access control mechanism is supported by the file system owning the file specified by “pathname”. A value of 1 indicates that it is supported, as shown in Figure 2-106 on page 80. A value of zero (0) indicates it is not supported.

_PC_ACL_ENTRIES_MAX
Maximum number of entries in an ACL for a file or directory, as shown in Figure 2-106, which indicates a value of 1024.

getconf command example

You can use the **getconf** command to display the configuration values for your zFS file systems, as shown in Figure 2-106.

```
ROGERS @ SC43:/>getconf _PC_ACL /u/zfs/mO2c
1
ROGERS @ SC43:/>getconf _PC_ACL_ENTRIES_MAX /u/zfs/mO2c
1O24
```

Figure 2-106 Examples of the getconf command

Defining ACLs from OMVS

The z/OS UNIX **setfacl** command is used to define ACLs. Activating the RACF FSSEC class causes the ACLs to be used during access checking. To set or modify ACLs, the same requirements are needed as for changing the permission bits.

In Figure 2-107, user ALAN is given read and write access to the file testfile. The **getfacl** command displays the file access information and permission bit settings for the testfile file.

```
HERING:/u/hering:$> touch testfile
HERING:/u/hering:$> setfacl -m user:alan:rw- testfile
HERING:/u/hering:$> getfacl testfile
#file:  testfile
#owner: HERING
#group: SYS1
user::rw-
group::r--
other::r--
user:ALAN:rw-
```

Figure 2-107 Setting and displaying ACLs

From the OMVS command line, you can determine which files have an ACL associated with them. The ACL indication is a plus (+) following the permission bits, as shown in Figure 2-108 on page 81 for the file named testfile.

```
ROGERS @ SC43:/u/hering>ls -aE
total 200
drwxr-xr-x        9 HERING   SYS1        8192 Apr 10 19:28 .
drwxr-xr-x       17 OMVSKERN SYS1        8192 Mar 28 14:38 ..
-rwxr-xr-x  --s-  1 HERING   SYS1        1869 Mar 14 16:27 .profile
-rwxr-xr-x  --s-  1 HERING   SYS1         689 Mar 14 16:29 .setup
-rw-------  --s-  1 HERING   SYS1        3336 Apr 10 19:54 .sh_history
drwxr-xr-x        2 HERING   SYS1        8192 Mar 15 16:08 bin
drwx------        4 OMVSKERN SYS1        8192 Mar 27 16:29 test
-r--r--r--  ----  1 OMVSKERN SYS1         147 Mar 17 14:41 test.extattr
drwx------        2 OMVSKERN SYS1        8192 Mar 31 15:29 test1
lrwxrwxrwx        1 OMVSKERN SYS1          15 Mar 31 13:59 test1.sl -> /u/hering
/test1
drwxr-xr-x        2 HERING   SYS1        8192 Mar 14 14:58 test2
drwxr-xr-x        2 HERING   SYS1        8192 Mar 19 00:43 test3
drwxr-xr-x        2 OMVSKERN SYS1        8192 Apr  2 01:15 test4
drwxr-xr-x        2 HERING   SYS1        8192 Apr  2 01:43 test5
-rw-r--r--+ --s-  1 HERING   SYS1          23 Apr  9 15:00 testfile
-rwxr-xr-x  ----  1 HERING   SYS1        2481 Mar 15 15:48 xzfs.save
-rw-------  --s-  1 HERING   SYS1         309 Mar 25 21:33 ztest1047XXX
```

Figure 2-108 Displaying the existence of ACLs using the ls command

Defining ACLs from the ISHELL

You can use the ISHELL to display, add, delete, and modify ACLs, as shown beginning in Figure 2-109. There, the plus sign (+) in column one for testfile indicates that the file has associated ACLs.

```
  File  Directory  Special_file  Commands  Help
--------------------------------------------------------------------------------
                              Directory List

Select one or more files with / or action codes.  If / is used also select an
action from the action bar otherwise your default action will be used.  Select
with S to use your default action.  Cursor select can also be used for quick
navigation.  See help for details.
EUID=0   /u/hering/
  Type  Perm  Changed-EST5EDT   Owner      ------Size  Filename   Row 1 of 17
_ Dir    755  2002-04-05 00:22  HERING           8192  .
_ Dir    755  2002-03-28 14:38  OMVSKERN         8192  ..
_ File   755  2002-03-14 16:27  HERING           1869  .profile
_ File   755  2002-03-14 16:29  HERING            689  .setup
_ File   600  2002-04-05 00:31  HERING           3090  .sh_history
_ Dir    755  2002-03-15 16:08  HERING           8192  bin
_ Dir    700  2002-03-27 16:29  OMVSKERN         8192  test
_ File   444  2002-03-17 14:41  OMVSKERN          147  test.extattr
_ File  +644  2002-04-09 15:00  HERING              23  testfile
_ Dir    700  2002-03-31 15:29  OMVSKERN         8192  test1
_ Syml   777  2002-03-31 13:59  OMVSKERN           15  test1.sl
_ Dir    755  2002-03-14 14:58  HERING           8192  test2
_ Dir    755  2002-03-19 00:43  HERING           8192  test3
_ Dir    755  2002-04-02 01:15  OMVSKERN         8192  test4
_ Dir    755  2002-04-02 01:43  HERING           8192  test5
_ File   755  2002-03-15 15:48  HERING           2481  xzfs.save
_ File   600  2002-03-25 21:33  HERING            309  ztest1047XXX
```

Figure 2-109 ISHELL display of the testfile file

By placing an `a` action code for testfile the file attribute panel is displayed, as shown in Figure 2-110 on page 82. This display shows that there is one entry in the ACL and user ALAN was entered, as shown in Figure 2-107 on page 80.

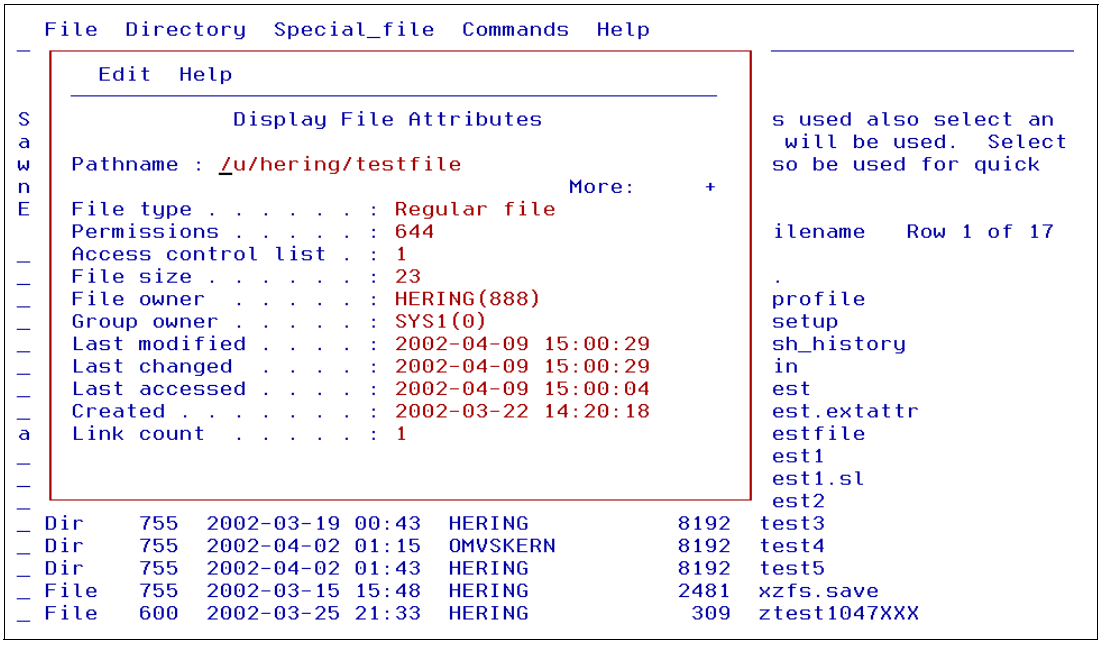


Figure 2-110 Display file attributes for the testfile file

Placing the cursor under the Edit pull-down menu and pressing Enter, select option **8**; Figure 2-111 is displayed.

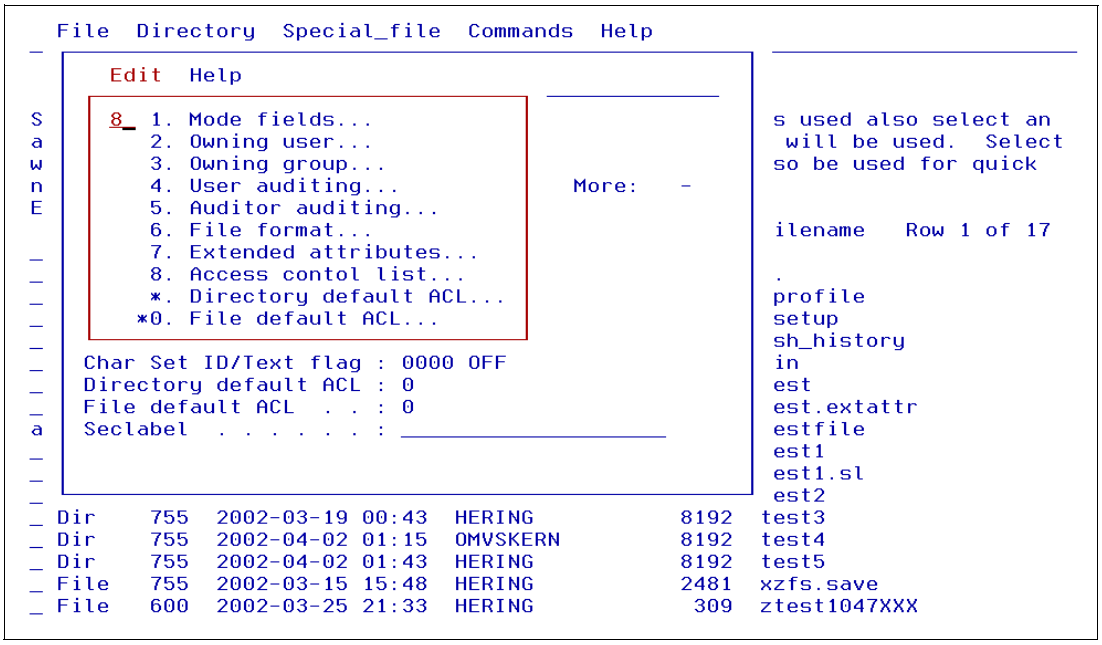


Figure 2-111 Selecting option 8 to access ACL information

When you press Enter, you now have access to modify, add, or delete ACL entries, as shown in Figure 2-112 on page 83

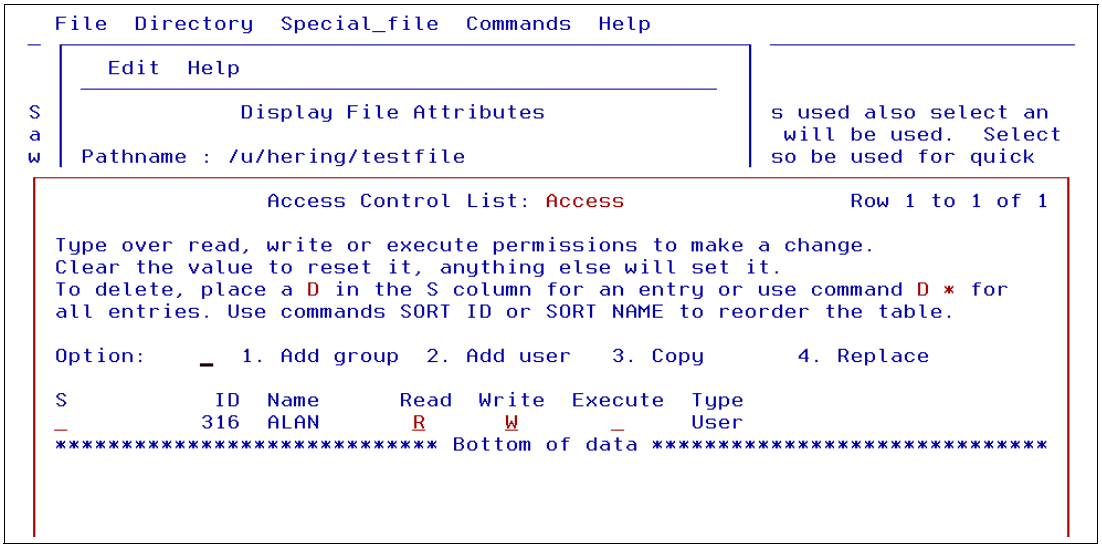


Figure 2-112 Access control list panel to change ACL definitions

Note: Using ACLs must be supported by the file system that the file or directory belongs to. It is supported in z/OS V1R3 by zFS and HFS. ACLs are not supported currently for a temporary file system (TFS) in z/OS V1R3.

REXX programming interfaces

New services in REXX are provided to get, create, update, replace, or delete an ACL for a file or directory.

For more information about ACLs, see *z/OS UNIX System Services Planning,* GA22-7800 or *z/OS UNIX System Services Command Reference,* SA22-7802.

2.16 Displaying aggregate information and stopping zFS

To stop zFS you can use the zFS **stop** (**p**) command. All zFS file systems are unmounted and all zFS aggregates are detached, as shown in Figure 2-121 on page 88.

To determine if any aggregates are currently attached, use the commands shown in Figure 2-114 on page 84 from the OMVS command line.

Note: You can use either the **zfsadm** command from the OMVS shell or the IOEZADM command from the TSO/E command line.

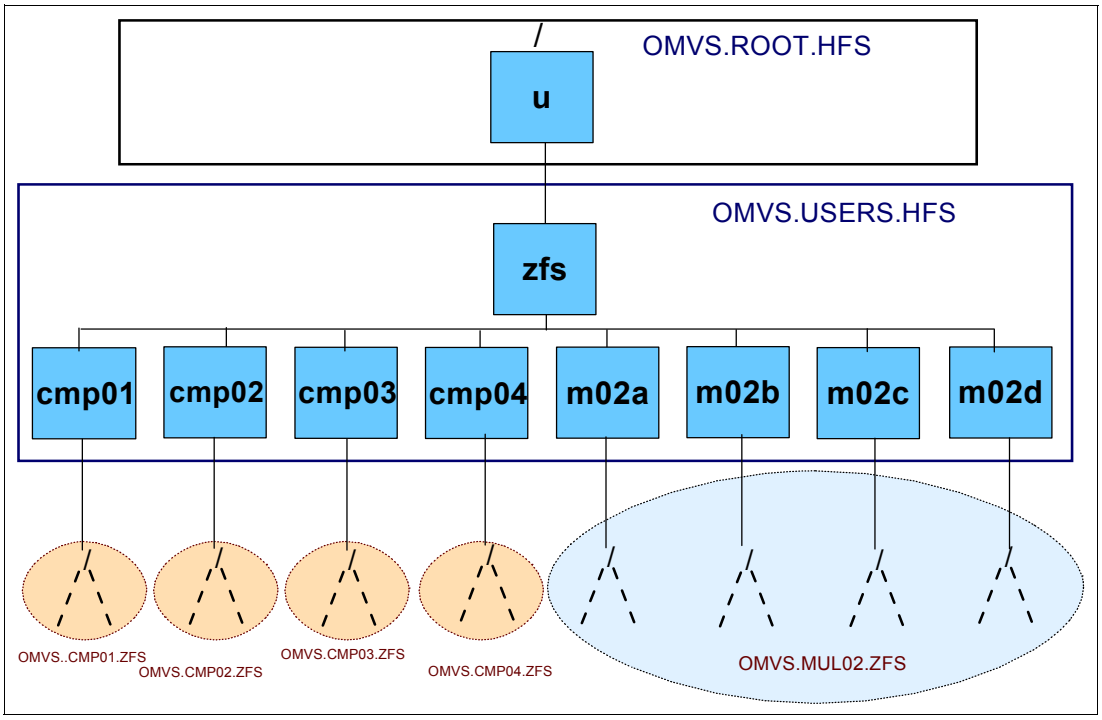


Figure 2-113 Direct-mounted zFS file systems

Note: The file system names for the multifile aggregate are not shown in Figure 2-113, but can be seen in the command output shown in Figure 2-115 on page 85 and Figure 2-116 on page 85.

2.16.1 Displaying aggregates and file systems

Table 2-1 on page 15 lists the **zfsadm** commands that are to be used by system administrators to manage the file systems and aggregates. Examples of several of these commands are used here to illustrate the information they provide.

The **zfsadm lsaggr** command, shown in Figure 2-114, lists the number of aggregates that are currently attached. Each aggregate is assigned an aggregate id when it is attached, beginning with id 0. (This output format has been changed in z/OS V1R6, as shown in Figure 2-118 on page 86.)

```
$> zfsadm lsaggr
IOEZ00106I A total of 6 aggregates are attached
OMVS.MUL01.ZFS                          (id=100000)
OMVS.CMP04.ZFS                          (id=100005)
OMVS.CMP03.ZFS                          (id=100004)
OMVS.CMP02.ZFS                          (id=100003)
OMVS.CMP01.ZFS                          (id=100002)
OMVS.MUL02.ZFS                          (id=100001)
```

Figure 2-114 Display attached aggregates

To list all the mounted file systems for the six aggregates that are currently attached, issue the **zfsadm lsfs** command, as shown in Figure 2-115 on page 85.

```
$> zfsadm lsfs
IOEZ00127I No file systems found for aggregate  OMVS.MUL01.ZFS

IOEZ00129I Total of 1 file systems found for aggregate  OMVS.CMP04.ZFS
OMVS.CMP04.ZFS   RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 1; total off-line 0; total busy 0; total mounted 1

IOEZ00129I Total of 1 file systems found for aggregate  OMVS.CMP03.ZFS
OMVS.CMP03.ZFS   RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2

IOEZ00129I Total of 1 file systems found for aggregate  OMVS.CMP02.ZFS
OMVS.CMP02.ZFS   RW (Mounted R/W)      58 K alloc     58 K quota On-line
Total file systems on-line 3; total off-line 0; total busy 0; total mounted 3

IOEZ00129I Total of 1 file systems found for aggregate  OMVS.CMP01.ZFS
OMVS.CMP01.ZFS   RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 4; total off-line 0; total busy 0; total mounted 4

IOEZ00129I Total of 4 file systems found for aggregate  OMVS.MUL02.ZFS
OMVS.m02c.ZFS RW (Mounted R/W)       9 K alloc      9 K quota On-line
OMVS.M02A.ZFS RW (Mounted R/W)      75 K alloc     75 K quota On-line
OMVS.M02B.ZFS RW (Mounted R/W)      76 K alloc     76 K quota On-line
OMVS.M02D.ZFS RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 8; total off-line 0; total busy 0; total mounted 8
```

Figure 2-115 List all mounted file systems for all aggregates

To issue a short version of the list of all the mounted file systems, use the **-fast** option, as shown in Figure 2-116.

```
$> zfsadm lsfs -fast
IOEZ00127I No file systems found for aggregate OMVS.MUL01.ZFS
OMVS.CMP04.ZFS
OMVS.CMP03.ZFS
OMVS.CMP02.ZFS
OMVS.CMP01.ZFS
OMVS.M02A.ZFS
OMVS.M02B.ZFS
OMVS.m02c.ZFS
OMVS.M02D.ZFS
```

Figure 2-116 Short list of the mounted file systems for all aggregates

As a further option, **-long** shows more details; see Figure 2-117 on page 86.

```
$> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.07.00
$> zfsadm lsfs OMVS.HERING.TEST.ZFS -long
IOEZ00129I Total of 1 file systems found for aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS 100005,,5 RW (Mounted R/W)      states 0x10010005 On-line
    4294967232 K alloc limit;      10847 K alloc usage
         11375 K quota limit;      10847 K quota usage
           112 K Filesystem Inode Table           21 file requests

    version 1.4
    Creation Tue Aug 31 20:37:56 2004
    Last Update Tue Aug 31 20:49:30 2004

Total file systems on-line 1; total off-line 0; total busy 0; total mounted 1
```

Figure 2-117 Detailed listing for a compatibility mode aggregate

Note: The aggregate version, Filesystem Inode Table, and file requests information in Figure 2-117 is new with z/OS V1R7.

Change to zfsadm lsaggr command

In z/OS V1R6 the output format of the **zfsadm lsaggr** command is changed to include the system name and whether the aggregate is attached R/W or R/O. This is shown in Figure 2-118.

Note: This was a first step to make the **zfsadm** commands and interfaces sysplex-aware. This functionality has been added with z/OS V1R7.

Furthermore, note that command **uname** is used to list the level of z/OS that the commands are run on.

```
$> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.06.00
$> zfsadm lsaggr
IOEZ00106I A total of 10 aggregates are attached
OMVS.HERING.ZFS                                SC65        R/O
ZFSFR.ROOT.ZFS                                 SC65        R/W
ZFSFR.ZFSG.ZFS                                 SC65        R/W
ZFSFR.ZFSF.ZFS                                 SC65        R/W
ZFSFR.ZFSE.ZFS                                 SC65        R/W
ZFSFR.ZFSD.ZFS                                 SC65        R/W
ZFSFR.ZFSC.ZFS                                 SC65        R/W
ZFSFR.ZFSB.ZFS                                 SC65        R/W
ZFSFR.ZFSA.ZFS                                 SC65        R/W
ZFSFR.ROOT1.ZFS                                SC65        R/W
```

Figure 2-118 New output format of zfsadm lsaggr in z/OS V1R6

Listing information about aggregates using zfsadm aggrinfo

The command shown in Figure 2-119 on page 87 lists all zFS aggregates that are owned locally.

```
$> zfsadm aggrinfo
IOEZ00370I A total of 4 aggregates are attached.
ZFSFR.ZFSB.ZFS (R/O COMP): 57760 K free out of total 576000
ZFSFR.ZFSA.ZFS (R/O COMP): 6374 K free out of total 7200
OMVS.HERING.TEST01.ZFS (R/W COMP): 528 K free out of total 11520
OMVS.TEST.MULTIFS.ZFS (R/W MULT): 20597 K free out of total 20880
```

Figure 2-119 Listing all aggregates owned by the local system

In z/OS V1R7 the new option **-long** has been introduced for **zfsadm aggrinfo**, which in particular provides information about the version of the aggregate and the number of 8 K blocks and 1 K fragments that are free.

```
$> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.07.00
$> zfsadm aggrinfo -aggregate OMVS.HERING.TEST.ZFS -long
OMVS.HERING.TEST.ZFS (R/W COMP): 528 K free out of total 11520
version 1.4

            52 free 8k blocks;        112 free 1K fragments
           112 K log file;             24 K filesystem table
             8 K bitmap file
```

Figure 2-120 Listing additional information with zfsadm aggrinfo -long

Beginning with z/OS V1R9, the output of **zfsadm aggrinfo -long** includes the auditfid. See 2.25, “zFS auditid uniqueness” on page 107 for more information about zFS auditids. See 2.25.2, “New format of zFS auditids” on page 109 and 2.25.3, “Changing zFS auditids” on page 109 for details about these auditids.

2.16.2 Stop zFS command

This section describes the initial method of stopping and restarting a zFS physical file system.

For an explanation of the new and enhanced method of stopping and restarting a zFS physical file system, as introduced in z/OS V1R8, see 2.16.3, “LFS support for zFS shutdown” on page 88.

To stop the zFS physical file system, you can use the stop command from an MVS console (**P ZFS**) to unmount all file systems and detach all aggregates, as shown in Figure 2-121 on page 88.

Note: Whenever you see a description or sample in this book that references the **P ZFS** command, keep in mind that this was done in an environment prior to z/OS V1R8.

```
 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate OMVS.MUL02.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system OMVS.MUL02.ZFS.
 IOEZ00048I Detaching aggregate OMVS.CMP04.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP04.ZFS.
 IOEZ00048I Detaching aggregate OMVS.CMP03.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP03.ZFS.
 IOEZ00048I Detaching aggregate OMVS.CMP02.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP02.ZFS.
 IOEZ00048I Detaching aggregate OMVS.CMP01.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP01.ZFS.
 IOEZ00057I zFS kernel program IOEFSCM is ending
*070 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO IGNORE.
```

Figure 2-121 Stopping the zFS PFS

Replying `R` to the BPXF032D message at any time restarts zFS, as shown in Figure 2-122.

```
R 70,R
IEE600I REPLY TO 071 IS;R
...
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.03.00 Service Level OW53579.
Created on Wed Feb 27 12:02:18 EST 2002.
IOEZ00178I ZFS.SC43.IOEFSZFS(IOEFSPRM) is the
configuration dataset currently in use.
...
IOEZ00055I zFS kernel: initialization complete.
```

Figure 2-122 Restarting the zFS PFS

Figure 2-123 shows a sample with reply `I` after stopping zFS. This leaves the zFS PFS terminated and you need to use the SETOMVS command to start zFS again, as shown in Figure 2-9 on page 21.

```
 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00057I zFS kernel program IOEFSCM is ending
*071 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO IGNORE.
 R 71,I
 IEE600I REPLY TO 071 IS;I
```

Figure 2-123 Stop zFS PFS and reply “I”

2.16.3 LFS support for zFS shutdown

In prior releases of z/OS, there was no way to stop the zFS address space in a normal way. The only way to stop the zFS PFS was by using the **P ZFS** command or by canceling the address space. However, this caused problems after remounting the zFS file systems. A file system would not dismount correctly and applications might become unstable.

Beginning with z/OS V1R8, a new command is introduced to stop the zFS address space:

```
F OMVS,STOPPFS=ZFS
```

The new operator command allows z/OS UNIX to move file system ownership (nondisruptively) to another system before the PFS is terminated. This allows applications that are accessing file systems owned on the system where the zFS PFS is being terminated to proceed without I/O errors in a shared file system environment.

Figure 2-124 shows sample output from stopping the zFS file system. After receiving the stop command, the system invokes a sync to all locally mounted zFS file systems. Afterward, all zFS file systems are unmounted in a single system.

When you are in a shared file system environment, all AUTOMOVE-defined file systems will be moved to another system. Before the file system move starts, a new message, BPXI068D, will appear to confirm the file system shutdown.

```
 F OMVS,STOPPFS=ZFS
*010 BPXI078D STOP OF ZFS REQUESTED. REPLY 'Y' TO PROCEED. ANY OTHER REPLY WILL
 CANCEL THIS STOP
 R 10,Y
 IEE600I REPLY TO 010 IS;Y
 IOEZ00048I Detaching aggregate LUTZ.ZFS
 IOEZ00048I Detaching aggregate PELEG.ZFS
 IOEZ00048I Detaching aggregate DAURCES.ZFS
 IOEZ00048I Detaching aggregate TEMEL.ZFS
 IOEZ00048I Detaching aggregate VAINI.ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00057I zFS kernel program IOEFSCM is ending
 IEF352I ADDRESS SPACE UNAVAILABLE
 IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=0057.
*011 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART.
```

Figure 2-124 Stopping a zFS file system

Restriction: The **P ZFS** command no longer functions in z/OS V1R8. If you use the **P ZFS** command, you see the following message:

```
IOEZ00523I zFS no longer supports the stop command. Issue f omvs,stoppfs=zfs
```

Remember to remove the command from all procedures used for automated shutdown. If the address space is already down or you give a wrong file system type, the new message BPXI077I appears, as shown in Figure 2-125.

```
F OMVS,STOPPFS=HFS
BPXI077I THE PFS NAME IS INVALID OR THE PFS DOES NOT SUPPORT STOPPFS OR IS
ALREADY STOPPED.
```

Figure 2-125 Wrong file system name that does not support shutdown

2.17 Alternate sysplex root support

Several clients experienced a significant number of issues involving the system root being a single point of failure in a sysplex. If a sysplex root gets corrupted and becomes unavailable and unrecoverable, it could lead to a multi-system outage situation (MSO) and require a sysplex-wide IPL.

In a sysplex configuration, the alternate sysplex root file system becomes a standby for the sysplex root file system. The alternate sysplex root file system can be used to replace the current sysplex root file system if the sysplex root file system becomes unavailable. The alternate sysplex root file system is established by using the ALTROOT statement in the BPXPRMxx parmlib member during OMVS initialization, or by using the **SET OMVS** command.

This support provides the capability to dynamically replace a failed or failing sysplex root automatically or manually using an external command.

Using alternate sysplex root support, you can:

- Replace the failed sysplex root automatically.
- Or, replace the failed or failing sysplex root manually.

This support provides the following advantages:

- It eliminates the single point of failure (SPOF) for the sysplex root file system.
- It allows for a dynamic replacement of a failed or failing sysplex root.
- It prevents a single point sysplex root failure from causing a multi-system outage.

2.17.1 Using the automatic replacement

To set up an alternate sysplex root file system, perform the following steps:

- The alternate sysplex root file system must be allocated and the mount points and symlinks must be set up exactly as in the current sysplex root (refer to Figure 2-127 on page 92 for an example).
  - You can also use the **pax** shell command instead of the copytree utility to populate the alternate sysplex root.
  - The ALTROOT mount point must be created before an alternate sysplex root file system can be established via the BPXPRMxx parmlib member. There is no default mount point.
- The alternate sysplex root must be established using the new ALTROOT statement in a BPXPRMxx parmlib member, as shown here:

```
ALTROOT FILESYSTEM ('PLEX75.SYSPLEX.ROOTALT1.ZFS') MOUNTPOINT('/rootalt')
```

  - The ALTROOT file system should be mounted as read-only and with AUTOMOVE=Yes.
  - It can be established during OMVS initialization or using the **SET OMVS** command as shown in Figure 2-126 on page 91.

```
SET OMVS=(RA)
IEE252I MEMBER BPXPRMRA FOUND IN SYS1.PARMLIB
BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
IEF196I IEF285I   SYS1.PARMLIB
IEF196I IEF285I   VOL SER NOS= BH5CAT.
IEF196I IEF285I   CPAC.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z19CAT.
IEF196I IEF285I   SYS1.IBM.PARMLIB
IEF196I IEF285I   VOL SER NOS= Z1BRC1.
BPXF013I FILE SYSTEM PLEX75.SYSPLEX.ROOTALT1.ZFS 031
WAS SUCCESSFULLY MOUNTED.
```

Figure 2-126 Establishing the altroot sysplex root with command SET OMVS

Important: Make sure that the alternate sysplex root does not reside in the same volume, device, and control unit as the current sysplex root.

Figure 2-127 on page 92 shows an example of using the **copytree** utility to create the alternate sysplex root.

```
#> zfsadm define PLEX75.SYSPLEX.ROOTALT1.ZFS -volume BH5ST1 -cylinders 2 2
IOEZ00248I VSAM linear dataset PLEX75.SYSPLEX.ROOTALT1.ZFS successfully created.
#> zfsadm format PLEX75.SYSPLEX.ROOTALT1.ZFS -compat
IOEZ00077I HFS-compatibility aggregate PLEX75.SYSPLEX.ROOTALT1.ZFS has been successfully
created
#> /usr/sbin/chmount -w /
#> mkdir -m 700 /rootalt
#> /usr/sbin/chmount -r /
#> /usr/sbin/mount -f PLEX75.SYSPLEX.ROOTALT1.ZFS /rootalt
#> copytree / /rootalt
Copying / to /rootalt
From filesystem PLEX75.SYSPLEX.ROOT.ZFS
  To filesystem PLEX75.SYSPLEX.ROOTALT1.ZFS
Scanning for file nodes...
Skipping mountpoint: //u
Skipping mountpoint: //SC75
Skipping mountpoint: //SC74
Skipping mountpoint: //pp
Skipping mountpoint: //Z1BRC1
Skipping mountpoint: //rootalt
Processing 28 nodes
Creating directories
Creating other files
Setting file attributes
Creating mount points

********************

Copy complete.  Error count= 0
Directory errors:   0
Directories copied: 15
File errors:        0
Files copied:       1
Symlink errors:     0
Symlinks copied:    12
Char-spec errors:   0
Char-spec copied:   0
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
#> /usr/sbin/unmount /rootalt
#>
```

Figure 2-127 Allocating and filling the alternate root file system

ALTROOT mount status

When finished, you will have established an alternate sysplex root in the shared file system configuration. The alternate sysplex root is mounted in read-only mode at the specified mount point and designated as AUTOMOVE. When the alternate sysplex root becomes the current sysplex root, it is mounted in read-only mode and designated as AUTOMOVE regardless of the current sysplex root settings.

Note: Although the internal structure of the sysplex root is more complex, the processing works as desired.

Notes and comments

Consider the following information:

- The alternate sysplex root file system is a hot standby for the sysplex root file system that is used to replace the current sysplex root file system when the sysplex root file system becomes unowned or unrecoverable.
- You need to ensure that the alternate sysplex root does not reside in the same volume, device, and control unit as the current sysplex root.
- Validation on the mount points in the alternate root (as compared to the current root) will be done during replacement processing. If mount points are missing or incorrect, then the replacement fails.
- The sample BPXPRMxx in the SYS1.SAMPLIB data set provides a detailed explanation of ALTROOT statements and accepted keywords.
- The current sysplex root and the alternate sysplex root file system type do not need to be identical.

Restrictions for replacement

The following constraints must be met to allow for using the alternate sysplex root support:

- The system must be in a shared file system configuration.
- All systems in the shared file system environment must be at the minimum system level of z/OS V1R11.
- The ALTROOT mount point must not exceed 64 characters, and it must reside in the root directory of the current sysplex root file system.
- The UID, GID, and the permission bits of the root directory in the alternate sysplex root must match the root directory in the current sysplex root.
- If the SECLABEL class is active and the MLFSOBJ option is active, then the multilevel security label for the alternate sysplex root must be identical to the assumed multilevel security label of the current sysplex root.
- All systems in the shared file system configuration must have direct access to the new file system and must be able to locally mount it.
- The file system type for the alternate sysplex root and the current sysplex root must be either HFS or ZFS.
- The alternate sysplex root PFS must be active on all systems in the shared file system configuration.
- The real path name for the mount points in the current sysplex root must not exceed 64 characters in length.
- The sysplex root, or any directories in the sysplex root file system, must not be exported by the DFS or SMB server.

Important: If you make changes to the current sysplex root after the alternate sysplex root has been successfully established, you must make the *same* changes to the alternate sysplex root.

2.17.2 Displaying the alternate sysplex root

The output of **D OMVS,O** command will display the status of the alternate sysplex root.

- If the alternate sysplex root is established and active, you see the name of alternate root as shown in Figure 2-128.

```
D OMVS,O
BPXO043I 09.02.37 DISPLAY OMVS 445
OMVS     0010 ACTIVE             OMVS=(RA)
...
ALTROOT         = PLEX75.SYSPLEX.ROOTALT1.ZFS
```

Figure 2-128 Displaying the alternate sysplex root being active on D OMVS

- If the alternate sysplex root is inactive, you simply see an empty ALTROOT line in the output, as shown here.

```
ALTROOT =
```

Command **F BPXOINIT,FILESYS=DISPLAY,GLOBAL** also displays information about the status of the alternate sysplex root.

- If the alternate sysplex root is established in the sysplex, the name is displayed as shown in Figure 2-129.

```
F BPXOINIT,FILESYS=DISPLAY,GLOBAL
BPXM027I COMMAND ACCEPTED.
BPXF242I 2009/05/18 12.31.40 MODIFY BPXOINIT,FILESYS=DISPLAY,GLOBAL
SYSTEM   LFS VERSION ---STATUS--------------------- RECOMMENDED ACTION
SC74       1. 11.  0 VERIFIED                      NONE
SC75       1. 11.  0 VERIFIED                      NONE
CDS VERSION=  2        MIN LFS VERSION=  1. 11.  0
DEVICE NUMBER OF LAST MOUNT=      88
MAXIMUM MOUNT ENTRIES=     500   MOUNT ENTRIES IN USE=      40
MAXIMUM AMTRULES=           50   AMTRULES IN USE=            2
MAXSYSTEM=                   4
ALTROOT=  PLEX75.SYSPLEX.ROOTALT1.ZFS
BPXF040I MODIFY BPXOINIT,FILESYS PROCESSING IS COMPLETE.
```

Figure 2-129 Displaying the alternate sysplex root being active on F BPXOINIT,FILESYS

- If the alternate sysplex root is not active, you see the following line instead:

```
ALTROOT=  N/A
```

2.17.3 Disabling the alternate sysplex root

The alternate sysplex root file system is (or can be) disabled in any of the following situations:

- If you unmount the alternate sysplex root file system.
- If a down-level system, previous to z/OS V1R11, is joining the UNIX System Services shared file system configuration, you cannot have an alternate sysplex root. (All systems must be at the z/OS V1R11 level.)

- If you specify a BPXPRMxx parmlib member with a specific ALTROOT statement, as follows:

  ```
  ALTROOT NONE
  ```

  This is the recommended way to disable the alternate root, if needed for some reason. This command deletes any outstanding BPXF253E eventual operator messages.

  Note: In this case, the alternate sysplex root file system is left mounted in the sysplex as a regular file system.

2.17.4 New console messages

Figure 2-130 lists the new messages that have been introduced regarding alternate sysplex root support.

```
BPXF248I THE NEW SYSPLEX ROOT FILE SYSTEM IS MISSING THE FOLLOWING MOUNT POINT:
         NAME: filesysname PATH: path
BPXF249I THE MOUNT POINT PATH FOR THE FOLLOWING FILE  SYSTEM EXCEEDS THE
         MAXIMUM LENGTH:
         NAME: filesysname
BPXF252I ALTROOT FILE SYSTEM fsname WAS NOT MOUNTED.
         RETURN CODE = return_code REASON CODE = rsn_code
BPXF254I ALTROOT STATEMENT IN PARMLIB MEMBER ONLY VALID IN SHARED FILE SYSTEM
         ENVIRONMENT.
BPXF255I ALTROOT NONE PARMLIB STATEMENT SUCCESSFULLY PROCESSED ON THIS SYSTEM.
BPXF256I fsname IS NOW ACTIVE AS CURRENT SYSPLEX ROOT.
BPXF257I SYSPLEX ROOT REPLACEMENT FAILED:
         RETURN CODE = return_code REASON CODE = rsn_code
BPXF258I SYSPLEX ROOT REPLACEMENT FAILED:
         <text>
BPXF259I ALTROOT FAILED TO MOUNT ON THIS SYSTEM.
               RETURN CODE = return_code  REASON CODE = rsn_code
BPXF253E ALTROOT INACTIVE:
       - ALTROOT FILE SYSTEM IS NOT MOUNTED OR IS UNMOUNTED.
       - ALTROOT FILE SYSTEM IS CURRENTLY UNOWNED.
       - NOT ALL SYSTEMS ARE AT THE REQUIRED RELEASE.
       - ALTROOT IS NOW ACTIVE AS CURRENT SYSPLEX ROOT.
       - ALTROOT MOUNT FAILED ON SOME SYSTEMS.
```

Figure 2-130 Messages related to alternate sysplex root processing

2.17.5 Command replacement

Replacing a failed or failing sysplex root can be initiated manually by using the existing operator command **F OMVS,NEWROOT** with a new option **FORCE** on any system in the sysplex.

Important: This method does not use the alternate sysplex root file system that is established using the BPXPRMxx parmlib member. Instead, it uses the file system that is specified on the command invocation.

To do so, first prepare a second file system that can replace the sysplex root, as described in Figure 2-127 on page 92. Now, assuming that this has been done and the new name is PLEX75.SYSPLEX.ROOTALT2.ZFS,use the command shown in Figure 2-131.

```
 F OMVS,NEWROOT=PLEX75.SYSPLEX.ROOTALT2.ZFS,COND=FORCE
*010 BPXI085D REPLACEMENT OF SYSPLEX ROOT IS REQUESTED. REPLY 'Y' TO
 PROCEED. ANY OTHER REPLY TO CANCEL.
 R 10,Y
 IEE600I REPLY TO 010 IS;Y
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING 967
 COMPLETED SUCCESSFULLY.
 IOEZ00048I Detaching aggregate PLEX75.SYSPLEX.ROOT.ZFS
```

Figure 2-131 Replacing a sysplex root manually using F OMVS,NEWROOT with option FORCE

A BPXI085D WTOR message is issued to the console to confirm the FORCE option.

2.17.6 Additional requirements

Consider the following requirements for this support:

- The new sysplex root must not be mounted, HSM migrated, or in use.
- All systems in the shared file system configuration must be at z/OS V1R11 to use the command option.
- Validation on the mount points will be done during the replacement processing. If mount points are missing or incorrect, then replacement will fail.

2.17.7 Migration and coexistence considerations

The alternate sysplex root file system support requires a minimum system level of z/OS V1R11.

If a down-level system (z/OSV1R10 and below) joins the shared file system configuration after the successful establishment of an alternate sysplex root file system, then the established alternate sysplex root file system is not considered an alternate sysplex root file system anymore.

Note: In this situation, the alternate sysplex root file system will be treated as a regular mounted file system in the shared file system configuration.

2.17.8 zFS abnormal termination

If zFS terminates abnormally, then automatic ownership movement occurs for both the z/OS UNIX owner and the zFS owner, if possible. zFS aggregate ownership moves unless the file system is unmounted by z/OS UNIX.

Applications with an open file on these file systems receive I/O errors until the file is closed. After zFS is restarted, the operator must remount any file systems that were locally mounted (that is, file systems that were owned by that system and were not moved). This can be done by using the following command:

```
F BPXOINIT,FILESYS=REINIT
```

This causes a remount for each file system that was mounted through a BPXPRMxx parmlib statement.

Restriction: The ALTROOT statement is ignored during processing of the **F BPXOINIT,FILESYS=REINIT** command. You will have to manually issue a **SET OMVS=(xx)** command where BPXPRMxx is the parmlib member containing the original ALTROOT statement.

2.18 Changing an aggregate mode

The chapter “Multifile system aggregates” in *z/OS Distributed File Service zSeries File System Administration,* SC24-5989, provides a comparison between these two zFS aggregate types. It describes how to make a multifile system aggregate from a compatibility mode aggregate because the **attach** command only applies to multifile aggregates.

We tested what happens when unmounting a compatibility mode file system and performing an attach of the aggregate itself. Figure 2-132 shows that, except for the number of file systems that can reside in the aggregate, there is basically no difference between the two types of zFS aggregates.

```
#> zfsadm aggrinfo -aggregate  OMVS.CMP04.ZFS
 OMVS.CMP04.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
#> /usr/sbin/unmount /u/zfs/c04
#> zfsadm attach -aggregate  OMVS.CMP04.ZFS
IOEZ00117I Aggregate  OMVS.CMP04.ZFS attached successfully
#> zfsadm aggrinfo -aggregate  OMVS.CMP04.ZFS
 OMVS.CMP04.ZFS (R/W MULT): 4962 K free out of total 5184 (568 reserved)
#> zfsadm detach -aggregate  OMVS.CMP04.ZFS
IOEZ00122I Aggregate  OMVS.CMP04.ZFS detached successfully
#> /usr/sbin/mount -f  OMVS.CMP04.ZFS -t ZFS /u/zfs/c04
#> zfsadm aggrinfo -aggregate  OMVS.CMP04.ZFS
 OMVS.CMP04.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
```

Figure 2-132 Attaching a compatibility mode aggregate

2.19 zFS application programming interfaces

The chapter “zFS application programming interfaces” in *z/OS Distributed File Service zSeries File System Administration,* SC24-5989, describes the zFS commands that are available with the pfsctl (BPX1PCT) API to manage zFS aggregates and file systems and to query and set configuration options.

The pfsctl application programming interface is used to send physical file system-specific requests to a physical file system. The description of the different zFS commands and subcommands allows you to write your own programs or procedures in C, Assembler, or REXX to exploit these interfaces.

We created two routines that list attached aggregate information by using the subcommands "List attached aggregate names (version 2)" / opcode 140 and "List aggregate status (Version 2)" / opcode 146 for the pfsctl **API ZFSCALL_AGGR** command.

2.19.1 Utility RXLSAGGR

The **rxlsaggr** utility may be used in TSO, or in a UNIX shell environment. Figure 2-133 illustrates how to use the tool from a UNIX shell session.

```
$> rxlsaggr \?
rxlsaggr           - lists one information line per aggregate
rxlsaggr -axd      - lists the aggregates with aggrgrow=on
rxlsaggr -nax      - lists the aggregates with aggrgrow=off
rxlsaggr -qsd      - lists the aggregates that are quiesced
rxlsaggr -f8k      - lists the number of free 8K blocks
rxlsaggr -all      - lists important data for all aggregates
rxlsaggr aggr_name - lists important data for given aggregate
$> rxlsaggr -qsd
OMVS.HERING.HFS.TEST
$> rxlsaggr -nax
$> rxlsaggr omvs.hering.hfs.test
OMVS.HERING.HFS.TEST                            SC70      R/O QUIESCE
Monitored for full  . . : Disabled
New block security  . . : Enabled
HFS compatibility . . . : Enabled
Auto-extend . . . . . . : Enabled
Number of fragments . . :       3600
8K Blocks available . . :        450
Aggregate free space KB :       2809
Number free 8K blocks . :        323
Number free 1K fragments:        225
Log file size in KB . . :        112
File system table in KB :         24
Bitmap file size in KB  :          8
Disk format version . . : 1.4

$>
```

Figure 2-133 Displaying aggregate information using rxlsaggr

rxlsaggr REXX procedure

The following tips help you to understand the functions supported by **rxlsaggr** when entered as a UNIX shell command.

- Specifying **rxlsaggr** without parameters provides the same information as using the **zfsadm lsaggr** command as stated in Figure 2-133.
- Listing all aggregates that are quiesced currently provides complementary information in the following console message:

  ```
  IOEZ00581E There are quiesced zFS aggregates.
  ```

  The command **rxlsaggr -qsd** issues the following:

  ```
  ROGERS @ SC70:/u/rogers>rxlsaggr -qsd
  OMVS.HERING.HFS.TEST
  ```

- Listing all aggregates with **aggrgrow=off** displays aggregates that might end up as completely full if not grown explicitly. Issue the command as follows:

  ```
  rxlsaggr -nax
  ```

- Listing the number of 8 K blocks that are still free is useful information. The **zfsadm aggrinfo ... -long** command displays that, along with other information for single aggregates, as follows:

  ```
  ROGERS @ SC70:/u/rogers>zfsadm aggrinfo -aggregate OMVS.HSMA110.ZFS -long
  ```

```
OMVS.HSMA110.ZFS (R/W COMP): 62897 K free out of total 72000
version 1.4
auditfid E2C2D6E7 F1D309F3 0000

          7853 free 8k blocks;         73 free 1K fragments
           720 K log file;             40 K filesystem table
            16 K bitmap file
```

Note: See Example B-12 on page 410 to view the contents of the **rxlsaggr** utility.

2.19.2 Utility RXZFSMON

The **rxzfsmon** utility provides even better free space planning and status information for aggregates.

Figure 2-134 shows how you can use it from UNIX, as well as what information is provided.

```
$> rxzfsmon "?"

rxzfsmon           - lists freespace information for all active aggregates
rxzfsmon aggr_name - lists freespace information for given aggregate

Information provided is the following:

Column 1= aggregate name
Column 2= total number of 8K blocks in the aggregate
Column 3= used number of 8K blocks in the aggregate
Column 4= percentage of the used number of 8K blocks
Column 5= X|- Q|- R|- (auto-eXtend Quiesced Read-only)

$> rxzfsmon

Aggregate name                              8K blk tot 8K blk use Percent  XQR
------------------------------------------- ---------- ---------- -------  ---
HFS.ZOSR19.Z19RC1.JAVA31V5                       36180      32882 = 90.9%  X-R
OMVS.DB2V9.SDSNMQLS.D070705                        900        216 = 24.0%  X--
HFS.ZOSR19.Z19RE1.SELAROOT                         594        564 = 94.9%  X-R
HFS.ZOSR19.Z19RE1.AAOPROOT                       18630      17099 = 91.8%  X-R
HFS.ZOSR19.Z19RC1.SHPJROOT                        2520       2500 = 99.2%  X-R
... (42 lines removed) ...
ZFSFR.ZFSB.ZFS                                   72000      64794 = 90.0%  X--
OMVS.HERING.TEST.FSFULL                            900        900 =100.0%  -Q-
OMVS.DB2V9.SDSNAHFS.D070705                       1800        942 = 52.3%  X--
OMVS.DB2V9.SDSNMQLS.D071020                        900        216 = 24.0%  X--
OMVS.DB2V9.SDSNWORF.D070705                        900        181 = 20.1%  X--
... (60 lines removed) ...
HFS.ZOSR19.Z19RE1.SIGYROOT                          42         41 = 97.6%  X-R
OMVS.SSRE70.HFS                                    900         24 =  2.7%  X--
OMVS.MIKEMOR.HFS                                   900         30 =  3.3%  X--
OMVS.HSMA110.ZFS                                  9000       1147 = 12.7%  X--

A total of 116 aggregates are attached.

$>
```

Figure 2-134 Displaying aggregate information using rxzfsmon

This information is useful for planning purposes. It can be retrieved whenever you want, and the compact form may become more desirable as the number of zFS file systems increases in your system. In addition, you can see related information such as whether the zFS would get automatically extended, whether it is used read-only (and then cannot get into trouble), and whether it is quiesced.

See Example B-13 on page 415 to view the contents of the **rxzfsmon** utility.

2.20 Rollback of zFS functions for releases before z/OS V1R6

All the zFS-related functions available in z/OS V1R6 have been rolled back to z/OS V1R4 and z/OS V1R5, with APAR OA11602; see Table 2-4.

Table 2-4 PTFs for APAR OA11602

z/OS release	zFS release	PTF
z/OS V1R4	340	UA18863
z/OS V1R5	350	UA18862

When OA11602 is installed in z/OS V1R4 or z/OS V1R5, all the zFS R6 functions are available; see the zFS R6 documentation.

Also, OA11958 for R6 puts zFS R6 at the same level of fixes as OA11602; see Table 2-5.

Table 2-5 PTF for APAR OA11958

z/OS release	zFS release	PTF
z/OS V1R6	360	UA18846

There is an additional APAR OA12704 for which the appropriate PTF needs to be installed. Without this code a problem is encountered on non-z/Architecture processors (for example, a 9672). In the original zFS R6 code, an instruction is used that is not supported in this situation. For this reason, the new level of code needed to have release variable logic to address the problem when running on older hardware; see Table 2-6.

Table 2-6 PTFs for APAR OA12704

z/OS release	zFS release	PTF
z/OS V1R4	340	UA20485
z/OS V1R5	350	UA20482
z/OS V1R6	360	UA20483
z/OS V1R7	370	UA20484

2.21 Performance improvement for the zFS mount function

In z/OS V1R7 a new function called zFS Fast Mount has been implemented to provide a performance improvement in the zFS mount function. This function is provided in APAR OA12519; see Table 2-7 on page 101.

Table 2-7 PTF for APAR OA12519

z/OS release	zFS release	PTF
z/OS V1R7	370	UA21177

In addition, a further APAR OA11573 has been created to allow z/OS V1R7 and prior releases to tolerate this function; see Table 2-8.

Table 2-8 PTFs for APAR OA11573

z/OS release	zFS release	PTF
z/OS V1R4	340	UA20485
z/OS V1R5	350	UA20482
z/OS V1R6	360	UA20483
z/OS V1R7	370	UA20484

OA11573 is an APAR that needs to be applied to all z/OS V1R7, z/OS V1R6, z/OS V1R5, and z/OS V1R4 systems. It must be applied before subsequent APAR OA12519 is applied to z/OS V1R7. APAR OA12519 is a new function APAR for z/OS V1R7 that improves the performance of mount processing for zFS aggregates and file systems. In particular, it solves problems regarding the movement of zFS file systems with millions of files (and more) to another system.

To support this improvement, the first time that a zFS aggregate is mounted on z/OS V1R7 after APAR OA12519 is applied, it is converted to a new, on-disk format called version 1.4, so that additional information for mount processing can be recorded in the aggregate. Note that all existing aggregates are version 1.3 aggregates.

After conversion, subsequent mount processing will occur more quickly. Previously, zFS needed to scan the aggregate and the file systems in the aggregate to retrieve required information for mount.

During the conversion, you see messages in the syslog such as:

```
IOEZ00500I Converting OMVS.HERING.TEST.ZFS for fast mount processing
IOEZ00518I Converting filesystem OMVS.HERING.TEST.ZFS to allow for fast mount
```

2.21.1 Special considerations for previous z/OS releases

APAR OA11573 supplies the code to process the new version 1.4 format aggregates for z/OS V1R7, z/OS V1R6, z/OS V1R5, and z/OS V1R4. APAR OA11573 must be installed on all coexisting systems before any aggregates are converted to version 1.4.

This allows the prior releases to correctly access the new version 1.4 structure for zFS aggregates. If you do not install toleration APAR OA11573 on prior releases, prior releases will not be able to correctly access the aggregates.

In this case, you can convert a zFS aggregate back to a version 1.3 structure so that it can be accessed. Apply APAR OA11573 as soon as possible. To convert a zFS aggregate back to a version 1.3 structure, use the zFS Salvager utility (IOEAGSLV). A new option, **-converttov3**, is provided to convert a version 1.4 zFS aggregate back to a version 1.3 zFS aggregate. For details, see 2.21.2, “Salvager utility changes” on page 102.

This version of the Salvager utility is provided with z/OS V1R7. IOEAGSLV is also installed in the MIGLIB PDS.

Note: IOEAGSLV can be executed from any supported release by referring to this MIGLIB in a steplib statement.

Figure 2-135 shows a sample job illustrating how to do this.

```
//ZFSJOB   JOB ,'ZFS Salvager',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//STEPLIB  DD DSN=hlq.MIGLIB,DISP=OLD
//SALVAGE  EXEC PGM=IOEAGSLV,REGION=0M,
// PARM=('-aggregate OMVS.HERING.TEST.ZFS -converttov3')
//SYSPRINT DD SYSOUT=H
//STDOUT   DD SYSOUT=H
//STDERR   DD SYSOUT=H
//SYSUDUMP DD SYSOUT=H
//CEEDUMP  DD SYSOUT=H
//*
```

Figure 2-135 Converting a zFS version 1.4 aggregate to version 1.3

You can also use the Salvager utility from a UNIX shell environment. See 4.2.2, “Using the Salvager utility” on page 174 for a description of how make it available with the name zfssalvage. Figure 2-136 shows how to refer to a specific STEPLIB library when using the Salvager from UNIX.

```
$> su
#> STEPLIB=SYS1.MIGLIB zfssalvage -aggregate OMVS.HERING.TEST.ZFS -converttov3
Converting OMVS.HERING.TEST.ZFS
IOEZ00543E Converting filesystem OMVS.HERING.TEST.ZFS to version 3
Done.  OMVS.HERING.TEST.ZFS converted to version 3.
#>
```

Figure 2-136 Converting a zFS version 1.4 aggregate to version 1.3 with a UNIX command

This allows you to use the Salvager utility as a BPX.SUPERUSER. In parallel, the following message is displayed in the syslog: `IOEZ00543E Converting filesystem OMVS.HERING.TEST.ZFS to version 3.`

2.21.2 Salvager utility changes

The Salvager utility in z/OS V1R7 has an option to convert a zFS aggregate that is in version 1.4 format back to a version 1.3 format. This function can be used if the 1.4 aggregate needs to be used on a system that does not have APAR OA11573 applied. A **`-converttov3`** option is added. The new syntax is shown in Figure 2-137.

```
ioeagslv -aggregate name
[-recoveronly]
[-converttov3 | -verifyonly | -salvageonly]
[-verbose]
[-level]
[-help]
```

Figure 2-137 New syntax of the Salvager utility

The conversion is only successful if all file systems and the aggregate are successfully converted. If the conversion is interrupted before completion, it must be run again to completion.

Important: An attempt to mount or attach an aggregate that has been partially converted will be denied.

The aggregate can also be formatted again with the **-overwrite** option.

Restriction: Formatting an aggregate with option **-overwrite** destroys any existing data.

When **-convertov3** is specified, the aggregate is recovered. This means the log is replayed, whether or not option **-recoveronly** is specified.

You can mount the version 1.3 aggregate NOAUTOMOVE until the toleration APAR is applied to all systems in the sysplex, or you can choose AUTOMOVE while excluding all z/OS V1R7 systems. Otherwise, the aggregate will be automatically converted to a version 1.4 aggregate if it is moved to a z/OS V1R7 system.

2.21.3 Concurrent log recovery

A new IOEFSPRM keyword, recovery_max_storage=, is available in z/OS V1R9 to indicate the maximum amount of zFS address space storage to use for concurrent log recovery during multiple concurrent aggregate mounts (attaches).

This allows multiple concurrent mounts to occur when sufficient storage is available for multiple concurrent log recovery processing. The specifications are as follows:

```
Default Value: 256M
Expected Value: A number in the range of 128M - 512M.
Example: recovery_max_storage=128M
```

2.22 Changed VSAM share options for zFS aggregates

In z/OS V1R7, the zFS started task may display messages on attach or mount processing of zFS aggregates such as the following:

```
IOEZ00410I Shareoptions for aggregate OMVS.HERING.SHROPTS.ZFS altered. New
value is (3,3).
```

This behavior was not documented in the migration manual, and the message was not documented.

Note: This message requires no action on the operator's part and can be ignored.

The share options are changed from 2 to 3 internally by zFS for the zFS VSAM linear data sets, in preparation for some future support.

The only documentation change made shows the IDCAMS examples with `SHAREOPTIONS(3)` in *zFS Administration*. If you use **zfsadm define**, in z/OS V1R7 this command creates a VSAM LDS with `SHAREOPTIONS(3,3)`. If you use IDCAMS to DEFINE a zFS aggregate with `SHAREOPTIONS(2)` or if you have existing zFS aggregates with share options 2, they are converted to `SHAREOPTIONS(3,3)` on the first mount or attach.

Documentation APAR OA12762 describes this change in the meantime, along with four new zFS messages in total, including the one with identifier IOEZ00410I.

2.23 IOEAGFMT and IOEAGSLV authorization

Prior to z/OS V1R9, to run the IOEAGFMT format utility to format a zFS aggregate, you need to be a superuser with either one of the following:

- A UID=0
- READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the UNIXPRIV class

With z/OS v1R9, the new check being made does not add any additional protection. The idea was to ensure that users who could format a zFS aggregate would also need some UNIX authority, because they could overwrite existing data. However, anyone who has UPDATE authority to the data set can do that anyway. Therefore, the UID=0 check during format is really not needed.

Furthermore, this behavior was not wanted because when creating an HFS data set, you need ALTER authority to the data set profile only.

The zFS behavior is now changed to make it behave more like HFS. The enhancement requires ALTER access for the authorization to use the IOEAGFMT and IOEAGSLV utilities. Table 2-9 lists the APARs needed to make this change available on previous releases.

Table 2-9 APAR numbers for IOEAGFMT and IOEAGSLV authorization enhancement

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R7 (370)	UA35890	OA18981
z/OS V1R8 (380)	UA35891	OA18981
z/OS V1R9 (390)	UA35368	OA20613

In the following section, these APARs are explained in more detail.

2.23.1 APARs OA18981 and OA20613

APAR OA18981 and OA20613 remove the authorization check for a UID using the IOEAGFMT and IOEAGSLV utilities, and the user does not need to have a UID=0.

- For the IOEAGFMT utility - the user must have ALTER authority to the VSAM LDS; or must be UID 0; or must have READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the RACF UNIXPRIV class. In fact, UPDATE authority to the VSAM LDS is sufficient for format, but zFS will not be able to set the zFS bit in the catalog unless the issuer has ALTER authority.
- For the IOEAGSLV utility - the user needs UPDATE authority for the specified VSAM LDS; or must be UID 0; or must have READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the RACF UNIXPRIV class.

Set the zFS bit in the catalog, which can be displayed with IDCAMS or with the TSO **LISTCAT ALL** command. Figure 2-138 on page 105 shows the catalog before setting the bit. Figure 2-140 on page 106 shows the catalog after setting the bit.

```
ioezadm define -aggregate omvs.testfmt.zfs -storageclass openmvs -cylinders 1 1
IOEZ00248I VSAM linear dataset omvs.testfmt.zfs successfully created.
listcat entries('omvs.testfmt.zfs') all
CLUSTER ------- OMVS.TESTFMT.ZFS
...
     ATTRIBUTES
       KEYLEN-----------------0     AVGLRECL---------------0     BUFSPACE------
       RKP--------------------0     MAXLRECL---------------0     EXCPEXIT------
       SHROPTNS(3,3)   RECOVERY     UNIQUE           NOERASE     LINEAR
       UNORDERED        NOREUSE     NONSPANNED
...
```

Figure 2-138 Defining a new zFS aggregate and listing LDS attributes

After defining the aggregate, you can format it using IOEAGFMT in JCL. In non-superuser mode, you can format it using an external link from UNIX System Services. This is shown in Figure 2-139. The parameters used in Figure 2-139 for **zfsformat** are the same as you would use with IOEAGFMT JCL.

Note: The hexadecimal value x1ED is the well-known octal number o755.

```
$> id
uid=888(HERING) gid=2(SYS1) groups=1047(USSTEST)
$> ln -e IOEAGFMT bin/zfsformat
$> zfsformat -aggregate omvs.testfmt.zfs -compat
IOEZ00004I Formatting to 8K block number 90 for primary extent of
OMVS.TESTFMT.ZFS.
IOEZ00005I Primary extent loaded successfully for OMVS.TESTFMT.ZFS.
IOEZ00535I *** Using initialempty value of 1.
*** Using 89 (8192-byte) blocks
*** Defaulting to 13 log blocks(maximum of 1 concurrent transactions).
IOEZ00327I Done.  OMVS.TESTFMT.ZFS is now a zFS aggregate.
IOEZ00048I Detaching aggregate OMVS.TESTFMT.ZFS
IOEZ00071I Attaching aggregate OMVS.TESTFMT.ZFS to create HFS-compatible file
system
IOEZ00074I Creating file system of size 720K, owner id 888, group id 2,
permissions x1ED
IOEZ00048I Detaching aggregate OMVS.TESTFMT.ZFS
IOEZ00077I HFS-compatibility aggregate OMVS.TESTFMT.ZFS has been successfully
created
```

Figure 2-139 Formatting the zFS aggregate without being authorized

Figure 2-140 on page 106 shows that IOEAGFMT is also set on the zFS bit in the catalog. Note that zFS will also try to set the bit during mount processing, if it is not already set.

```
listcat entries('omvs.testfmt.zfs') all
CLUSTER ------- OMVS.TESTFMT.ZFS
...
     ATTRIBUTES
       KEYLEN-----------------0     AVGLRECL---------------0     BUFSPACE------
       RKP--------------------0     MAXLRECL---------------0     EXCPEXIT------
       SHROPTNS(3,3)   RECOVERY     UNIQUE           NOERASE     LINEAR
       UNORDERED        NOREUSE     NONSPANNED           ZFS
...
```

Figure 2-140 Listcat output showing the LDS is marked as a zFS aggregate

2.23.2 zFS utility considerations

To summarize the zFS utility considerations:

- The IOEAGFMT utility now allows you to create and format a zFS aggregate in a job if you have ALTER access to the data set profile. With an external link like **zfsformat**, you can do the same from a UNIX System Services shell environment.
- The other zFS utility, IOEAGSLV, the Salvager, is also included in this change. IOEAGSLV can now be used successfully if you have UPDATE authority for the data set profile. You no longer need to run in superuser mode or have READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the UNIXPRIV class.

  Note, however, that even when you simply verify a zFS aggregate using option **-verifyonly**, IOEAGSLV still accesses the aggregate exclusively; therefore, the aggregate cannot be mounted.

Figure 2-141 displays a JCL sample of using IOEAGSLV in a job.

```
//SALVAGE  EXEC PGM=IOEAGSLV,REGION=0M,
// PARM=('-aggregate OMVS.TESTFMT.ZFS -verifyonly')
//SYSPRINT DD SYSOUT=*
```

Figure 2-141 Using IOEAGSLV in a job

Figure 2-142 shows the successful case.

```
Verifying OMVS.TESTFMT.ZFS
Processed 1 vols 6 anodes 1 dirs 0 files 0 acls
Done.  OMVS.TESTFMT.ZFS checks out as zFS aggregate.
```

Figure 2-142 Output of a successfully running Salvager verification

Figure 2-143 shows the unsuccessful case because the zFS aggregate OMVS.HERING.TEST.ZFS is mounted.

```
IKJ56225I DATA SET OMVS.HERING.TEST.ZFS ALREADY IN USE, TRY LATER+
IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IEF237I JES2 ALLOCATED TO SYSOUT
IOEZ00003E While opening minor device 1, could not open dataset
OMVS.HERING.TEST.ZFS.
```

Figure 2-143 Output in the case of the Salvager being unable to access the zFS aggregate

2.24 Terminology

The following definitions explain the terminology related to this topic:

Auditid This refers to a unique 16-byte number that can be used to map back to the original path name of the file. It can be retrieved with certain z/OS UNIX calls (for example, stat) and appears in certain (Type 80) SMF records.

Auditfid This refers to the part of the auditid that represents the file system that contains the file.

Inode This refers to a 4-byte number that represents a file in a file system while it exists. When the file is deleted (removed), the inode can be reused for another file.

Uniquifier This refers to a 4-byte number associated with an inode that is incremented each time an inode is reused. The combination of inode and uniquifier should represent a unique file. That is, you should be able to determine whether or not a particular inode and uniquifier pair represents an existing file.

2.25 zFS auditid uniqueness

As mentioned, an auditid is a 16-byte value that is associated with each z/OS UNIX file or directory. The auditid identifies a z/OS UNIX file or directory in an SMF audit record or in certain authorization failure messages (for example, RACF message ICH408I).

An auditid appears in Type 80 SMF records and in the output of certain z/OS UNIX APIs (for example, stat). Prior to z/OS V1R9, you could not use the auditid for a zFS file to uniquely identify the file or directory. In z/OS V1R9 and above, however, zFS allows the administrator to specify whether zFS uses a more unique auditid for a zFS file or directory, or continues to use the existing, non-unique, standard auditid.

Important: Do not begin to use the unique auditid capability until *all* sysplex members are at z/OS V1R9 with APAR OA20614 installed.

Regardless of the auditfid stored in the aggregate, zFS aggregates that are owned by or moved to a system running a release prior to z/OS V1R9 generate the standard zFS auditids. As a result, an auditid for a file can change, based on which system owns the aggregate.

Prior to this new support, zFS does not provide a unique auditid for zFS files. However, HFS does provide a unique auditid for HFS files. As a result, this may cause problems when you are analyzing authorization problems.

An IBM UNIX tool named **`auditid`** is available from the z/OS UNIX tools Web site. This tool allows you to list the UNIX System Services auditid or FID for a given pathname. More importantly, for a given auditid the tool searches for a pathname having set this specific file identifier (FID). The **`auditid`** tool is available for download at the following address:

`http://www-03.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty2.html`

The tool can be very useful when you are analyzing the causes of a RACF ICH408I message with an FID included. An example is shown in Figure 2-144 on page 108.

```
$> cd samples/subdirectory1/subdirectory2/subdirectory3
cd: samples/subdirectory1/subdirectory2/subdirectory3: EDC5111I Permission
denied.
```

Figure 2-144 Failing change directory command

The RACF message is displayed in Figure 2-145.

```
 ICH408I USER(HERING  ) GROUP(IBMDE#01) NAME(ROBERT HERING       ) 981
   samples/subdirectory1/subdirectory2/subdirectory3
   CL(DIRSRCH ) FID(01E5E2D4C9F0F500011A000008540000)
   INSUFFICIENT AUTHORITY TO LOOKUP
   ACCESS INTENT(--X)  ACCESS ALLOWED(GROUP      ---)
   EFFECTIVE UID(0000000888)  EFFECTIVE GID(0000000001)
```

Figure 2-145 ICH408I message corresponding to the failing cd command

In this case, a pathname is included in the ICH408I message. However, more often a pathname will *not* be displayed and you will only see the FID value on a ICH408I message.

Moreover, even though a pathname is displayed in this case, it is not the pathname that is causing the problem. The real pathname is identified by the FID only. So, using an auditid can show you the name. This is illustrated in Figure 2-146.

```
$> auditid 01E5E2D4C9F0F500011A000008540000
/u/hering/samples
```

Figure 2-146 Listing the pathname for a given auditid or FID

2.25.1 Old-style zFS auditids

The old-style zFS auditids for a pathname start with the 4-byte inode number followed by a 4-byte uniquifier and 8-byte of x'00'. This is illustrated in Figure 2-147.

```
$> df -v test | grep Aggregate
Aggregate Name : OMVS.HERING.TEST.ZFS
$> auditid test/testfile
/u/hering/test/testfile
01E2C2D6E7F0F2413C26000000000003 /
00000000000000000000000000000000 /u
01E2C2D6E7F1F4002D0C000000000003 /u/hering
00000001000000010000000000000000 /u/hering/test
00001200000011F0000000000000000 /u/hering/test/testfile
$> find /u/hering/test -xdev -inum $(rexx 'say x2d(120)')
/u/hering/test/testfile
```

Figure 2-147 Listing an old-style zFS auditid for a file within a zFS aggregate

This format has two significant drawbacks, as explained here:

- These old-style zFS auditids are unique only within the *same* zFS aggregate; they are not unique within the whole UNIX file structure.

- The old auditid tool only supports HFS. It does *not* support the most important UNIX System Services file system type of zFS; see Figure 2-148.

```
$> auditid 000001200000011F0000000000000000
Audit id 000001200000011F0000000000000000 not found
```

Figure 2-148 Utility auditid cannot find the pathname for an old-style zFS FID

2.25.2 New format of zFS auditids

As a result of these limitations, zFS has been changed to generate unique auditids similar to HFS.

Important: This change allows a zFS auditid to be mapped back to the original pathname of the file or directory, thus making zFS more like HFS.

Table 2-10 lists the associated APAR number.

Table 2-10 APAR number for the new zFS auditid uniqueness

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA37323	OA20614

The new zFS format auditid consists of:

- A 10-byte auditfid that includes:
  - A 6-byte volser
  - A 4-byte CCHH of the first extent of the aggregate
- A 4-byte for the inode of the entry
- Another 2-byte uniquifier

The auditfid is stored in the zFS aggregate.

Notes:

- The new auditfid is a fixed value for all entries within the zFS aggregate.
- The auditid or FID is unique for every single file system entry.

2.25.3 Changing zFS auditids

Be aware that switching to the new type of zFS auditids is not forced automatically, because there could be problems in mixed zFS release environments.

There are several ways to change the zFS auditids and the format used.

- There is a zFS IOEPRMxx configuration option to control conversion to the new auditid format on mount or attach operations:
  - `convert_auditfid=on | off`
- IOEAGFMT and **`zfsadm format`** can store the new auditfid in the aggregate by using the option **`-newauditfid`**.
- The **`zfsadm setauditfid`** command can be used to change the aggregate auditfid after format. The help information is shown in Figure 2-149 on page 110.

```
$> zfsadm setauditfid -help
IOEZ00243I Usage: zfsadm setauditfid -aggregate <aggregate name> [{-force |
-old}] [-level] [-help]
```

Figure 2-149 Help information for the new zfsadm setauditfid command

- Command **zfsadm aggrinfo –long** displays the aggregate's auditfid. Figure 2-150 shows a sequence of commands when switching to the new format of the zFS auditids.

```
$> zfsadm aggrinfo -long -aggregate OMVS.HERING.TEST.ZFS | grep auditfid
auditfid 00000000 00000000 0000
$> swsu zfsadm setauditfid -aggregate OMVS.HERING.TEST.ZFS
IOEZ00596I Auditfid set for aggregate OMVS.HERING.TEST.ZFS
$> zfsadm aggrinfo -long -aggregate OMVS.HERING.TEST.ZFS | grep auditfid
auditfid E2C2D6E7 F3F2053B 0000
```

Figure 2-150 Listing and switching the zFS auditfid

Note: The format of the zFS auditid, new or old, depends on the auditfid stored in the aggregate.

Figure 2-151 verifies that the auditfid is created.

```
tso rexx say x2c(E2C2D6E7 F3F2)
SBOX32
tso listcat ent('OMVS.HERING.TEST.ZFS') alloc
CLUSTER ------- OMVS.HERING.TEST.ZFS
...
     VOLUME
       VOLSER------------SBOX32     PHYREC-SIZE---------4096     HI-A-RBA------
       DEVTYPE------X'3010200F'     PHYRECS/TRK-----------12     HI-U-RBA------
       VOLFLAG------------PRIME     TRACKS/CA-------------15
       EXTENTS:
       LOW-CCHH-----X'053B0000'     LOW-RBA---------------0      TRACKS--------
...
```

Figure 2-151 Volser and CCHH value of the auditfid shown in a listcat output

The new zFS auditid format is very similar to that of HFS and consists of the following:

- One byte as x"01" is fixed
- A 9-byte file system identifier, as follows:
  - 6 bytes for the volser
  - 3 bytes for the TTR
- 4 bytes for the inode of the entry
- A 2-byte uniquifier.

This similarity, together with the internal processing of the auditid tool, now allows you to use the tool for both HFS and zFS as illustrated in Figure 2-152 on page 111.

```
$> auditid test/testfile
/u/hering/test/testfile
01E2C2D6E7F0F2413C26000000000003 /
00000000000000000000000000000000 /u
01E2C2D6E7F1F4002D0C000000000003 /u/hering
E2C2D6E7F3F2053B0000000000010000 /u/hering/test
E2C2D6E7F3F2053B0000000001200000 /u/hering/test/testfile
$> auditid E2C2D6E7F3F2053B0000000001200000
/u/hering/test/testfile
```

Figure 2-152 Listing and searching the new zFS format auditids with the auditid utility

You can use the following command in authorized mode to switch the zFS auditids for all the zFS aggregates used read-write currently:

```
zfsadm lsaggr | grep R/W | awk '{system("zfsadm setauditfid " $1);}'
```

Keep the following points in mind when you compare HFS and zFS auditid values:

- If the last 8 bytes of the auditid are binary zero, the auditid is a zFS old format.
- If the first byte of the auditid is x"01", then the auditid is an HFS format.
- Otherwise, the auditid has the new zFS format.

2.26 zFS read-only mount recovery

Without the new zFS read-only mount recovery support, a zFS read-only mount may fail as a result of conditions that occurred when the aggregate was previously used, as explained with the following examples:

- The aggregate was mounted read-write and then the system failed before the zFS aggregate was unmounted.
- Or, something else occurred that prohibited the unmount from completing successfully.

2.26.1 Failing zFS read-only mount situation

Figure 2-153 shows the error message that may be seen in this situation on a failing read-only mount.

```
BPXF002I FILE SYSTEM SYS2.ISC0.USR.LOCAL WAS NOT MOUNTED.
         RETURN CODE = 0000008D, REASON CODE = EF096271
```

Figure 2-153 Failing read-only mount if recovery processing has to be done first

Reason code EF096271 indicates that the required recovery cannot be done since the aggregate is being attached read-only.

Mounting the zFS aggregate first as read-write and then switching to the desired read-only mode solves this problem, as demonstrated by the messages displayed in Figure 2-154 on page 112.

```
IOEZ00397I recovery statistics for SYS2.ISCO.USR.LOCAL:
IOEZ00391I   Elapsed time was 55 ms
IOEZ00392I   1 log pages recovered consisting of 4 records
IOEZ00393I   Modified 1 data blocks
IOEZ00394I   2 redo-data records, 0 redo-fill records
IOEZ00395I   0 undo-data records, 0 undo-fill records
IOEZ00396I   0 not written blocks
IOEZ00400I   0 blocks zeroed
```

Figure 2-154 Successful recovery messages displayed when temporary mounting the zFS R/W

The circumvention to avoid this problem was to always mount the zFS read-write first and then switch to read-only. Nevertheless, this is a condition that does not occur with HFS and therefore can be unexpected.

2.26.2 zFS new read-only recovery option

With the new support zFS provides a configuration option to enable it to handle this condition automatically. There are several ways to activate this option and to query the behavior that is used.

- There is a new zFS configuration option to control this:

  `romount_recovery={on|off}`

- The **`zfsadm config`** command can be used to set the new behavior on or off dynamically by using this new option:

  `-romount_recovery {on|off}`

- The **`zfsadm configquery`** command allows you to display the current setting by using this option:

  `-romount_recovery`

Note: The new support eliminates unexpected mount failures, as well as the need for additional procedures to avoid them. Again, this makes zFS more like HFS.

Table 2-11 lists the associated APARs.

Table 2-11 APAR number for the new zFS read-only mount recovery option

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R7 (370)	UA37118	OA22351
z/OS V1R8 (380)	UA37119	OA22351
z/OS V1R9 (390)	UA37323	OA20615

If the new option is `on` and you are mounting a zFS file system as read-only after it was previously mounted read-only but not cleanly unmounted, first mount the file system as read-write “under the covers” to allow the log to be replayed. Then unmount it and re-mount it as read-only.

Any zFS installation that normally mounts a zFS file system read-only but periodically mounts it read-write to update it and then mounts it back to read-only can avoid mount failures if `romount_recovery=on` (if the system fails before the switch back to read-only mode is done, or simply if switching back is not done).

2.27 Quiesced zFS aggregates

Quiescing a zFS aggregate is totally different compared to the way quiescing is performed for an HFS. More importantly, listing information about UNIX System Services file systems shows a quiesced HFS as an exception situation and displays it like a specific attribute belonging to the file system. (In the ISHELL mount table, for example, you see this flag when displaying the file system attributes.)

However, this is not the case for quiesced zFS aggregates; no indication is provided from the UNIX System Services file system point of view. In fact, no API function shows this status.

Note: In the zFS case, it is not the file system that is quiesced, it is the aggregate.

The following section explains how to obtain UNIX System Services file system status information.

2.27.1 UNIX System Services file system status information

To illustrate UNIX System Services file system status information, assume a situation with an HFS file system named OMVS.U.USER being quiesced and a zFS aggregate named HERING.TEST.ZFS, as well. Figure 2-155 shows the information for the aggregate.

```
$> zfsadm aggrinfo hering.test.zfs
HERING.TEST.ZFS (R/W COMP QUIESCED): 566 K free out of total 720
$> zfsadm lsfs hering.test.zfs
IOEZ00129I Total of 1 file systems found for aggregate HERING.TEST.ZFS
IOEZ00213E Error 114 (reason EF1A6246) occurred while attempting to get status for file
system HERING.TEST.ZFS in aggregate HERING.TEST.ZFS
Total file systems on-line 0; total off-line 0; total busy 0; total mounted 0
$>
```

Figure 2-155 Listing information for aggregate HERING.TEST.ZFS

The return code 114 ("72"x; EBUSY) indicates that the resource is busy. The same information is given with the zFS reason code provided.

Figure 2-156 illustrates what the **D OMVS** command displays. The HFS is clearly marked to be quiesced. Notice, however, that no information is shown regarding the zFS aggregate.

```
D OMVS,F,E
BPXO045I 13.50.28 DISPLAY OMVS 048
OMVS     000E ACTIVE             OMVS=(NF)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
HFS            82 QUIESCED                    RDWR  06/01/2007  L=43
  NAME=OMVS.U.USER                                  17.06.15    Q=51   PATH=/auto/user
  QSYSTEM=MCEVSF QJOBNAME=HERI QPID=  50332170
AUTOMNT        83 DRAIN UNMOUNT               RDWR  06/01/2007  L=42
  NAME=*AMD/auto/user/auto2                         17.06.15    Q=0
  PATH=/auto/user/auto2
```

Figure 2-156 Displaying an exception situation using D OMVS

Figure 2-157 shows the different status information for the HFS and the zFS that is displayed by using the UNIX System Services REXX SYSCALL interface from an interactive REXX routine.

```
$> rexx
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "statfs HERING.TEST.ZFS st."
OMVS Return Value (retval) = 128
...
s "getmntent mnt. (st.stfs_fsid)"
OMVS Return Value (retval) = 1
...
say bitand(D2c(mnt.mnte_status.1),D2c(mnt_quiesced))=D2c(mnt_quiesced)
0       <=== Status does not reference QUIESCED
...
s "getmntent mnt. 82"
OMVS Return Value (retval) = 1
...
say Strip(mnt.mnte_fsname.1)
OMVS.U.USER
...
say bitand(D2c(mnt.mnte_status.1),D2c(mnt_quiesced))=D2c(mnt_quiesced)
1       <=== Status does reference QUIESCED
...
```

Figure 2-157 HFS and zFS UNIX System Services file system status information

The first block retrieves information about the zFS file system. Notice that the name is specified in uppercase letters. The results show that the status QUIESCED is not set.

The HFS has device number 82. Information about the HFS is retrieved afterwards.

Restriction: The file system interface API SYSCALL command **`getmntent`** (which is similar to the API function w_getmntent() in C/C++) does *not* show a zFS QUIESCE status.

2.27.2 zFS monitoring quiesced aggregates

Beginning with z/OS V1R8, you may receive message IOEZ00581E (which is not documented in R8 manuals, but is included in R9 documentation). This message helps you to diagnose when there are quiesced aggregates mounted on the system showing the message; see Figure 2-158.

```
IOEZ00581E There are quiesced zFS aggregates
```

Figure 2-158 Message IOEZ00581E

The message is handled by a thread that wakes up every 30 seconds and checks to see if there are any quiesced aggregates.

Note: If you do not “remove” the message from the console, it will be DOMed automatically as long as no aggregate is still quiesced on a next test for quiesced aggregates.

If you receive this message, then you can use either one of the following commands to determine the list of quiesced aggregates:

```
zfsadm lsaggr | grep QUIESCE
rxlsaggr -qsd
```

See 2.19.1, “Utility RXLSAGGR” on page 98 for more information about the **rxlsaggr** command. Figure 2-133 on page 98 shows an example of using this command to display the quiesced aggregates.

An aggregate is normally quiesced during backup, and unquiesced when the backup is complete. An aggregate is also quiesced during a grow operation.

If there is no reason to quiesce an aggregate (needed temporarily when backing up an active zFS aggregate) and you decide to reset this status, you can use either of the following commands.

- You can use this command:

  `zfsadm unquiesce` *`aggregate_name`*

- Or you can use this console command on the owning system of the aggregate, which is supported for z/OS V1R7 and above:

  `F ZFS,UNQUIESCE,`*`aggregate_name`*

Important: Even if a file system is unmounted, the zFS aggregate may stay quiesced (and attached).

Therefore, you need to unquiesce the aggregate in any case to be able to work normally with the file system in the aggregate.

2.28 Monitoring aggregate full status

HFS has a monitoring interface available to set a threshold value and a further increment for showing SYSLOG messages saying that this HFS is about to fill up. If you specify FSFULL(90,2), this means that a message is shown when the HFS is 90% full and then again when 92% and 94% are reached and so on.

2.28.1 zFS monitoring for FSFULL and AGGRFULL

Looking for FSFULL in case of compatibility mode aggregates (which is the only type that should be used) is not useful and may be misleading. This section provides an example with a compatibility mode aggregate named OMVS.HERING.TEST.FSFULL. Note the following points:

- The aggregate has a size of 10 cylinders 3390; that is, 900 8 K blocks.
- It is mounted with zFS parameters `noaggrgrow fsfull(50,5) aggrfull(90,2)`.

In the beginning, the aggregate had a significant amount of data. Figure 2-159 on page 115 shows the messages displayed when the aggregate and file system contents are cleared.

```
IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 98% full (881/900)
IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 96% full (863/900)
IOEZ00069I zFS file system OMVS.HERING.TEST.FSFULL is now below 70% full (5038/7200)
IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 94% full (845/900)
IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 92% full (827/900)
IOEZ00069I zFS file system OMVS.HERING.TEST.FSFULL is now below 65% full (4672/7200)
IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 90% full (809/900)
IOEZ00069I zFS file system OMVS.HERING.TEST.FSFULL is now below 60% full (4314/7200)
IOEZ00069I zFS file system OMVS.HERING.TEST.FSFULL is now below 55% full (3956/7200)
IOEZ00069I zFS file system OMVS.HERING.TEST.FSFULL is now below 50% full (3598/7200)
```

Figure 2-159 Clearing the contents of a zFS aggregate

You can use the **zfsadm aggrinfo** command with the **-long** option, as shown in Figure 2-120 on page 87, to display the status information for a zFS aggregate. Refer to 2.19.1, “Utility RXLSAGGR” on page 98 for more information about this new tool.

After the clearing processing in the aggregate, this utility provides status as shown in Figure 2-160.

```
OMVS.HERING.TEST.FSFULL                          SC70      R/W
Monitored for full  . . : Enabled, (90,2)
HFS compatibility . . . : Enabled
Auto-extend . . . . . . : Disabled
Number of fragments . . :       7200
8K Blocks available . . :        900
Aggregate free space KB :       6943
Number free 8K blocks . :        867
Number free 1K fragments:          7
Log file size in KB . . :        112
File system table in KB :         24
Bitmap file size in KB  :          8
```

Figure 2-160 Aggregate information for OMVS.HERING.TEST.FSFULL

Next, we performed the following actions:

- Writing 2400 1 K files into the aggregate
- Deleting seven of these 1 K files again in each of the 300 8 K blocks used
- Writing files that were 8 K in size into the aggregate until it was filled up completely

The results are shown in Figure 2-161 on page 116.

```
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 90% full (811/900) (WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 92% full (829/900) (WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 94% full (847/900) (WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 96% full (865/900) (WARNING)
 IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 96% full (863/900)
 IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 94% full (845/900)
 IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 92% full (827/900)
 IOEZ00079I zFS aggregate OMVS.HERING.TEST.FSFULL is now below 90% full (809/900)
 IOEZ00068E zFS file system OMVS.HERING.TEST.FSFULL exceeds 50% full (3604/7200)
(WARNING)
 IOEZ00068E zFS file system OMVS.HERING.TEST.FSFULL exceeds 55% full (3964/7200)
(WARNING)
 IOEZ00068E zFS file system OMVS.HERING.TEST.FSFULL exceeds 60% full (4324/7200)
(WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 90% full (811/900) (WARNING)
 IOEZ00068E zFS file system OMVS.HERING.TEST.FSFULL exceeds 65% full (4684/7200)
(WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 92% full (829/900) (WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 94% full (847/900) (WARNING)
 IOEZ00068E zFS file system OMVS.HERING.TEST.FSFULL exceeds 70% full (5044/7200)
(WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 96% full (865/900) (WARNING)
 IOEZ00078E zFS aggregate OMVS.HERING.TEST.FSFULL exceeds 98% full (883/900) (WARNING)
 IOEZ00551I Aggregate OMVS.HERING.TEST.FSFULL ran out of space.
```

Figure 2-161 Using up the number of 8 K blocks in the aggregate completely

Note: The message IOEZ00551I is new with z/OS V1R9.

Figure 2-162 shows the new status information of the aggregate.

```
OMVS.HERING.TEST.FSFULL                        SC70      R/W
Number of fragments . . :       7200
8K Blocks available . . :        900
Aggregate free space KB :       1699
Number free 8K blocks . :          0
Number free 1K fragments:       1699
Log file size in KB . . :        112
File system table in KB :         24
Bitmap file size in KB  :          8
```

Figure 2-162 Aggregate information after filling it up completely

Figure 2-163 lists the aggregate monitor information using **rxzfsmon**.

```
Aggregate name                                8K blk tot 8K blk use Percent  XQR
--------------------------------------------- ---------- ---------- -------  ---
OMVS.HERING.TEST.FSFULL                              900        900 =100.0%  ---
```

Figure 2-163 Aggregate monitor information after filling it up completely

Figure 2-164 on page 117 shows the zFS file system quota information.

```
$> zfsadm lsquota -filesystem OMVS.HERING.TEST.FSFULL
Filesys Name          Quota    Used  Percent Used  Aggregate
OMVS.HERING.TEST.FSFULL   7200    5356    74    76 = 5501/7200 (zFS)
$>
```

Figure 2-164 zFS file system quota information

2.28.2 zFS monitoring for FSFULL

In our example, although the aggregate had used up all of the 8 K blocks (leaving only 1 K fragments and no chance to add data for any request that needs 8 K blocks), the FSFULL value was currently at 74%, and at 76% for all the aggregate data.

The quota value set for the compatibility mode file system is the size of the aggregate. In the case of a zFS file system in a multifile system aggregate, the quota is the number of 1 K fragments that the file system is permitted to use. Therefore, it is a logical number that is not related to the real space left in the aggregate.

Therefore, it is useful to monitor for the percentage of allowed 1 K blocks for a file system in a multifile system aggregate. For a compatibility mode aggregate, however, this does not make sense.

Although FSFULL values are based on 1 K fragments, AGGRFULL monitors the usage of the 8 K blocks that define how full an aggregate is. Even when you write small files into available 1 K fragments, a necessary enlargement of other settings may need new 8 K blocks.

The test results demonstrate that, in the case of a compatibility mode aggregate, the FSFULL information is not useful for monitoring the aggregate full status.

2.28.3 zFS monitoring for AGGRFULL

Regarding zFS, monitoring for AGGRFULL gives you an accurate idea of how full the aggregate is. Therefore, always look for the number of 8 K blocks being used up and still being available. There are several options available to do this, as explained here.

- Use AGGRFULL() to get syslog messages when certain threshold values are reached:
  - By specifying this in the IOEPRMxx zFS parameter file, for example `aggrfull(90,2)`
  - By dynamic activation of the setting with the **`zfsadm config -aggrfull 90,2`** command
  - By specifying the **`-o "AGGRFULL(90,2)"`** option on the zFS command **`/usr/sbin/mount -f ZFS`**
  - By specifying **`PARM('AGFULL(90,2)')`** on a zFS TSO **`MOUNT`** ... **`TYPE(ZFS)`** command
- Use the **`zfsadm aggrinfo -long`** command regularly.
- Use the **`rxzfsmon`** utility regularly, because it provides even clearer and more compact information.

2.29 Minor zFS updates in z/OS V1R10

The following minor changes are made to zFS in z/OS V1R10.

2.29.1 zFS man pages support

Previously, **`zfsadm`** only offered online help support via the **`-help`** option (or with **`zfsadm apropos`**), as shown in Figure 2-165.

```
$> zfsadm quiesce -help
IOEZ00243I Usage: zfsadm quiesce [{-aggregate <aggregate name> | -all}][-level]
[-help]
```

Figure 2-165 Using zfsadm -help option

Now **`zfsadm`** supports the **`man`** command when you specify **`zfsadm`** and the subcommand as one word, as shown in Figure 2-166.

```
$> man zfsadmquiesce
zfsadm quiesce

  Purpose

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


  Format

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


  Options
...
```

Figure 2-166 Using the new zfsadm man support

In addition, the zFS mount options and utilities are supported:

```
man zfsmount
man ioeagfmt
man ioeagslv
```

2.29.2 zFS support of EAV volumes

As a result of DFSMS EAV support, zFS aggregates are supported in an EAV environment:

- zFS aggregates (and file systems) can reside on an extended address volume (EAV).
- zFS aggregates (and file systems) can reside in track-managed space or cylinder-managed space (and are subject to any limitations that DFSMS may have).
- zFS still has an architected limit of 4 TB for the maximum size of a zFS aggregate.

```
Old VSAM data set size limits:
65520 cylinders/volume x 90 blocks/cylinder x 8K bytes/block x 59 volumes =
2,850,088,550,400 bytes = 2654 GB = 2.59 TB

New VSAM data set size limits:
262668 cylinders/volume x 90 blocks/cylinder x 8k bytes/block x 59 volumes =
11,425,931,919,360 bytes = 10641 GB = 10.39 TB

New limits per volume:
193,659,863,040 bytes/volume = 180.359 GB/volume
```

Figure 2-167 VSAM data set limits based on 90 8 K blocks per cylinder

2.30 DFSMSdfp indirect volume serial for zFS data sets (R12)

When z/OS system programmers have HFS file systems, they typically migrate them from one release to another using an IBM-recommended cloning process to copy the system residence volumes.

Note: The cloning of non-VSAM data sets like HFS uses the extended indirect volume serial support that uses system symbols.

With zFS, IBM introduced new inhibitors, as follows:

- zFS file systems are VSAM-based using Linear Data Sets (LDS).
- VSAM data sets need to be cataloged using a real volume serial.

Restriction: This brings up cloning and maintenance issues that prevent the migration of USS version root file systems.

2.30.1 Cloning zFS file systems

z/OS V1R12 DFSMS provides the extended indirect volume serial support that can be used to clone zFS file systems. A single catalog entry (for zFS VSAM LDS data sets) can represent different volumes on various systems.

Important: This support is the solution for several marketing requirements (such as MR0424065959) asking for extended Indirect volume serial support for cloning of zFS file systems.

Detailed specification and requirements

Here we provide the requirements that need to be met to allow a zFS aggregate to have an extended indirect volume serial.

- The zFS data set needs to be a single volume VSAM linear data set (LDS).
- A system symbol has been defined in the systems involved pointing to a system-specific volume that the zFS LDS is located on.
- On a DEFINE RECATALOG command, the VOLUMES parameter can have a special form referred to as indirect volume serial by using that system symbol. It can only be used with the RECATALOG parameter.
- This new function can be used for both SMS and non-SMS-managed zFS data sets on a DEFINE RECATALOG command of the data set.

Note: This results in the system dynamically resolving the volume serial to the system volume serial, residence (or its logical extension) serial number.

2.30.2 Enabling the support

There are some single specific actions needed to enable this support for a specific zFS data set name in your environment.

Provide a single volume zFS LDS

For example, you can define and format it as shown in Figure 2-168 on page 121.

```
$> zfsadm define omvs.zfs2.clone -storageclass openmvs -cylinders 1 0
IOEZ00248I VSAM linear dataset omvs.zfs2.clone successfully created.
$> sudo zfsadm format omvs.zfs2.clone -compat
IOEZ00077I HFS-compatibility aggregate OMVS.ZFS2.CLONE has been successfully
created
$> tsocmd "listcat ent('omvs.zfs2.clone') volume" 2>/dev/null | grep VOLSER \
> | awk -F'-' '{print $13}' | awk '{print $1}'
BH50E2
$>
```

Figure 2-168 Defining and formatting a single volume zFS aggregate

Defining a system symbol

A system symbol needs to exist or must be defined pointing to the volume serial, for example by using a definition in an IEASYSMxx parmlib member. A sample is shown in Figure 2-169.

```
SYSDEF SYMDEF(&ZFSCL.='BH50E2')
```

Figure 2-169 System symbol pointing to a volume serial

Note: You can also dynamically add a system symbol using the utility IEASYMUP.

For more information about IEASYMUP, see *Appendix B. IEASYMUP* in *z/OS Planned Outage Avoidance Checklist*, SG24-7328-00.

Deleting the catalog entry of the zFS LDS

Run an IDCAMS or TSO DELETE NOSCRATCH command against the data set, as shown in Figure 2-170.

```
$> tsocmd "delete 'omvs.zfs2.clone' noscratch"
delete 'omvs.zfs2.clone' noscratch
ENTRY (D) OMVS.ZFS2.CLONE.DATA DELETED
ENTRY (C) OMVS.ZFS2.CLONE DELETED
$>
```

Figure 2-170 DELETE NOSCRATCH

This deletes the catalog entry of the data set.

Recataloging the zFS LDS

Run an IDCAMS or TSO DEFINE RECATALOG using the system symbol (which is the indirect volume serial) against the data set. This is shown in Figure 2-171.

```
$> tsocmd "define cluster (name('omvs.zfs2.clone') volumes(&ZFSCL) \
> linear recatalog)"
define cluster (name('omvs.zfs2.clone') volumes(&ZFSCL) linear recatalog)
DATA ALLOCATION STATUS FOR VOLUME &ZFSCL IS 0
$>
```

Figure 2-171 DEFINE RECATALOG

Note: This defines the data set with the system symbol.

For more details on how to setup an indirect volume serial see manual *z/OS V1R12 DFSMS Access Method Services for Catalogs*.

2.30.3 Cloning of zFS aggregates

By setting the value of the system symbol to the desired different values on each system, it is achieved to have a single catalog entry (for zFS VSAM LDS data sets) that represents different volumes on these systems.

Note: System programmers can now use this to clone volumes, if this level of DFSMS is used.

You can clone a zFS by making copies of the existing zFS data sets using the ADRDSSU COPY command with the PHYSINDYNAM (PIDY) parameter as shown in Figure 2-172.

```
//DUMP     EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSIN    DD DATA,DLM=##
  COPY DATASET(INCLUDE(OMVS.ZFS2.CLONE)) -
  PIDY ((BH50E2)) -
  OUTDY ((BH50E1)) -
  REPLACEU -
  ALLDATA(*) -
  ALLEXCP
##
```

Figure 2-172 Clone a zFS using the PIDY parameter

Using PHYSINDYNAM is important because it does not create a catalog entry, and you can use the same name for your original and copied zFS. PHYSINDYNAM does create a VSAM volume data set (VVDS) entry for the zFS.

In the joblog for the job you should find the following information:

```
ADR396I (001)-PCVSM(01), DATA SET CLUSTER OMVS.ZFS2.CLONE COMPONENT
OMVS.ZFS2.CLONE.DATA ALLOCATED, ON VOLUME(S): BH50E1
```

Important:

- The copy processing is done with the identical name. The new data set is *not* cataloged.
- In this case of an “Indirect Volume Serial” you cannot change the name of the cluster.

2.30.4 Further sample of cloning a zFS aggregate

Using this sample zFS aggregate OMVS.ZFS2.CLONE, we demonstrate a simple sample of cloning.

Note: This is normally not the way cloning will be used. This is just for demonstration purposes.

- We removed the catalog entry again as shown in Figure 2-173.
- Then we defined a zFS with the same name on another system and provided the system symbol pointing to the new volume in that system. This is demonstrated in Figure 2-173.

```
$> echo This is system $(sysvar SYSNAME).
This is system SC75.
$> zfsadm define omvs.zfs2.clone -storageclass openmvs -cylinders 2 0
IOEZ00248I VSAM linear dataset omvs.zfs2.clone successfully created.
$> tsocmd "listcat ent('omvs.zfs2.clone') volume" 2>/dev/null | grep VOLSER \
> | awk -F'-' '{print $13}' | awk '{print $1}'
BH50E1
$> tsocmd "call 'hering.tso.load(ieasymup)' 'ZFSCL=BH50E1'"
call 'hering.tso.load(ieasymup)' 'ZFSCL=BH50E1'
$> sysvar ZFSCL
BH50E1
$>
```

Figure 2-173 Defining the target zFS with the same name on another system

Note the following information about IEASYMUP.

Note: In order to be able to use IEASYMUP successfully, you must assure the following.

- The library containing IEASYMUP must be APF-authorized.
- When using IEASYMUP from TSO it must be defined in the AUTHTSF names list of your TSO parmlib member IKJTSOxx being used.
- IEASYMUP is provided on an as-is basis, as are all samples in SYS1.SAMPLIB.

Instead of using the TSO interface, you can also use a job as shown in Figure 2-174.

```
//SYMBATCH EXEC PGM=IEASYMUP,PARM='ZFSCL=BH50E1'
//STEPLIB  DD DSN=&SYSUID..TSO.LOAD,DISP=SHR
```

Figure 2-174 Defining the system symbol dynamically using a job

- Afterwards we mounted and worked with the zFS file system in the source system. This is shown in Figure 2-175.

```
$> echo This is system $(sysvar SYSNAME).
This is system SC74.
$> sudo /usr/sbin/mount -f omvs.zfs2.clone zfs2.clone
$> touch zfs2.clone/sc74.file
$> ls -l zfs2.clone/sc74.file
zfs2.clone/sc74.file
$> sudo /usr/sbin/unmount zfs2.clone
$>
```

Figure 2-175 Working with the zFS file system in the source system

Doing the clone processing

We unloaded the source zFS aggreagte using IDCAMS REPRO and restore it to the target accordingly on the target system:

- First we unloaded the zFS on system SC74. The JCL is shown in Figure 2-176.

```
//* ------------------------------------------------------------------
//  SET ZFSAGGR=OMVS.ZFS2.CLONE                    <=== zFS aggrname
//  SET ZFSUNLOD=HERING.OMVS.ZFS2.CLONE.ULD        <=== Unloaded file
//* ------------------------------------------------------------------
//UNLOAD   EXEC PGM=IDCAMS
//ZFSAGGR  DD DSN=&ZFSAGGR.,DISP=SHR
//ZFSUNLOD DD DSN=&ZFSUNLOD.,DISP=(NEW,CATLG,DELETE),LRECL=4096,
//  BLKSIZE=20480,RECFM=FB,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//SYSIN    DD DATA,DLM=##
  REPRO INFILE(ZFSAGGR) OUTFILE(ZFSUNLOD)
##
//SYSPRINT DD SYSOUT=*
```

Figure 2-176 Unload the zFS aggregate using IDCAMS REPRO in system SC74

- Then we restored the zFS on system SC75. The JCL is shown in Figure 2-177.

```
//* ------------------------------------------------------------------
//  SET ZFSAGGR=OMVS.ZFS2.CLONE                    <=== Target zFS
//  SET ZFSUNLOD=HERING.OMVS.ZFS2.CLONE.ULD        <=== Unloaded file
//* ------------------------------------------------------------------
//RESTORE  EXEC PGM=IDCAMS
//ZFSAGGR  DD DSN=&ZFSAGGR.,DISP=OLD
//ZFSUNLOD DD DSN=&ZFSUNLOD.,DISP=(SHR,DELETE,KEEP)
//SYSIN    DD DATA,DLM=##
  REPRO INFILE(ZFSUNLOD) OUTFILE(ZFSAGGR)
##
//SYSPRINT DD SYSOUT=*
```

Figure 2-177 Restore the zFS aggregate using IDCAMS REPRO in system SC75

- Finally we demonstrated that we have two different data sets on different volumes. This is shown in Figure 2-178.

```
$> echo This is system $(sysvar SYSNAME).
This is system SC74.
$> sudo /usr/sbin/mount -f omvs.zfs2.clone -d SC75 zfs2.clone
$> mv zfs2.clone/sc74.file zfs2.clone/sc75.file
$> ls -1 zfs2.clone | grep sc7
sc75.file
$> sudo /usr/sbin/chmount -d SC74 zfs2.clone
$> ls -1 zfs2.clone | grep sc7
sc74.file
$>
```

Figure 2-178 Demonstrated that we have two different data sets on different volumes

This also shows that you should be careful using this if not done for real cloning of system residence volumes.

2.31 Changes in zFS Installation and setup in z/OS V1R13

Here we describe some changes regarding zFS installation and setup.

2.31.1 Changes in zFS Installation

This provides some information about installation changes.

- The zFS load modules have been moved from the PDS library `hlq.SIOELMOD` to a PDSE library named `hlq.SIEALNKE`.
- In previous releases the command **`zfsadm`** was a binary in the z/OS UNIX file system. Now it is a shell script that executes **`IOEZADM`**, which is an entry with the sticky bit set on. So the IOEZADM load module is executed. Figure 2-179 shows what this looks like.

```
$> command -V zfsadm
zfsadm is cached /bin/zfsadm
$> cat /bin/zfsadm | tail -3
PATH=$PATH:/usr/lpp/dfs/global/bin

IOEZADM "$@"
$> ls -l /usr/lpp/dfs/global/bin/IOEZADM | awk '{print $1}'
-rwxr-xr-t
$>
```

Figure 2-179 Command zfsadm

2.31.2 Changes in IOEPRMxx configuration options

There are several changes for the zFS parameter specifications:

- The setting for **`dir_cache_size`** is ignored in z/OS V1R13. It is no longer needed.
- The default value for the **`meta_cache_size`** has been changed from 32 M to 64 M. This is the sum of the previous defaults for the directory cache and the metadata cache.

  Note: The metadata cache is now used for caching both metadata and directory data.

- Client metadata can be stored in metadata backing cache and the size of the cache can be dynamically changed.

  Note: Metadata backing cache cannot be dynamically defined in z/OS V1R13. It must be specified in the zFS parameter file with a small size at least. However, a metadata backing cache is normally not needed.

- The setting for **`nbs`** is ignored now. It is always on and there is no performance penalty.

The default for **`client_cache_size`** has been reduced from 128 M to 32 M. It is only needed for zFS file system owners of sysplex-aware file systems that are of previous releases.

2.32 JCL and REXX procedures to run zFS-related tasks

During this project we sometimes used batch jobs for defining, formatting, and managing zFS file systems, in addition to other tasks. This made it easier to use the commands and to document our activities.

To do so, we created and used several REXX procedures that are described in Appendix B, “REXX utility procedures” on page 361. Appendix C, “JCL samples” on page 451, contains many JCL examples that you can use as skeletons. We also explain how to exploit these procedures and how to use the commands and facilities provided with zFS.

Important: The sample JCL and REXX procedures, along with information explaining how to retrieve them, are available in softcopy on the Internet from the Redbooks Web server.

Point your browser to the following URL (note that “SG” must be in uppercase):

```
ftp://www.redbooks.ibm.com/redbooks/SG246580/
```

Alternatively, you can go to:

```
http://www.redbooks.ibm.com
```

Then select **Redbooks Online**, and **Additional Materials**.

3

Migrating to zFS

This chapter describes how to migrate to zFS. It discusses the following migration considerations:

- Creating zFS file systems
- Moving the HFS root to zFS
- The automount facility and zFS
- Logical File System support for zFS in z/OS V1R7
- Migration tool BPXWH2Z
- Alternate HFS-to-zFS migration tool
- Replacing or migrating the sysplex root file system

3.1 Creating zFS file systems

You can migrate or copy existing HFS file systems into empty zFS file systems. This can be done with the OMVS **pax** command. The pax utility reads, writes, and lists archive files. An archive file is a single file containing one or more files and directories. Archive files can be HFS files or MVS data sets. A file stored inside an archive is called a component file; similarly, a directory stored inside an archive is called a component directory.

The **pax** command can be used with or without using an intermediate archive file. When the data is being copied from an HFS file system, the file system being accessed must be mounted.

Important: If you are planning to migrate HFS file systems to zFS, refer to the information about the utilities described in 3.5, “Migration tool BPXWH2Z” on page 148 and 3.6, “Alternate HFS-to-zFS migration tool” on page 158.

3.1.1 Previous restriction for zFS aggregate names

When you used special characters in HFS data set names and wanted to use the same name later for a zFS aggregate, you may have experienced a problem caused by a restriction of the characters that you were allowed to use, for example the “#” character.

When defining an aggregate using the **zfsadm define** command, you may have seen the following message:

```
IOEZ00248E VSAM linear dataset OMVS.NAME#TST.ZFS successfully created.
```

However, if doing the formatting with the **zfsadm format** command or the IOEAGFMT utility, you would have gotten the following message:

```
IOEZ00252E Aggregate name 'OMVS.NAME#TST.ZFS' contains invalid characters.
Operation terminated.
```

Characters allowed

In the old description of the IOEAGFMT utility, the characters that were allowed for the zFS file system names were the following:

- All upper case and lower case alphabetic characters (a to z, A to Z)
- All numerals (0 to 9)
- Period (.)
- Dash (-)
- Underscore (_)

The same rules hold for file system names on the **zfsadm create** command.

Important: This restriction, not to support characters such as @, #, or $ in zFS aggregate names, has been removed in z/OS V1R7.

A toleration APAR OA08611 is required to tolerate the new support on prior releases back to z/OS V1R4 (PTF UA14531 for z/OS V1R4, UA14529 for z/OS V1R5 and UA14530 for z/OS V1R6).

Note: Although introduced as toleration support, it really provides the full function, so it is a retrofit set for the z/OS releases back to z/OS V1R4.

This means that you can not only use file systems with these special characters on previous releases, but also create and format them, as shown in Figure 3-1.

```
$> zfsadm define -aggr OMVSM.AGGR#06.INVALID -cyl 1 1
IOEZ00248E VSAM linear dataset OMVSM.AGGR#06.INVALID successfully created.
$> zfsadm -level
IOEZ00020I zfsadm: z/OS    zSeries File System
Version 01.04.00 Service Level OA08633.
Created on Mon Nov  8 18:26:05 EST 2004.
$> su
#> zfsadm format -aggr OMVSM.AGGR#06.INVALID -compat
IOEZ00077I HFS-compatibility aggregate OMVSM.AGGR#06.INVALID has been
successfully created
```

Figure 3-1 Creating and formatting a zFS aggregate with special characters on z/OS V1R4

3.1.2 Using an archive file

You can use the **pax** command to copy the HFS file system into an intermediate archive file and then use it again 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 /u/hering and you want to copy this into an empty zFS file system mounted at /u/zfs/c01. Issue the following commands from OMVS:

1. Position to the source (HFS) file system mounted at /u/hering:

   ```
   cd /u/hering
   ```

2. Create a z/OS UNIX archive file called /tmp/zfs1.pax that contains the HFS file system mounted at /u/hering:

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

3. Position to the target (zFS) file system mounted at /u/zfs/c01:

   ```
   cd /u/zfs/c03
   ```

4. Read the archive file into the zFS file system mounted at /u/zfs/c01:

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

You may also use an MVS data set to be the intermediate archive file. In the sample shown in Figure 3-2, we use option **-p e** to preserve the user ID, group ID, file mode, access time, modification time, all extended attributes, and ACL entries. Option **-z** specifies that **pax** is to use the Lempel-Ziv compression. Finally, option **-X** specifies that **pax** is to skip over mount point boundaries.

```
#> cd /u/hering
#> pax -p e -wzXf "//test.u.hering.pax.z" .
#> cd /u/zfs/c02
#> pax -p e -rzXf "//test.u.hering.pax.z"
```

Figure 3-2 Migrating an HFS file system to zFS using an MVS data set as intermediate pax archive

Note: If you deal with large amounts of data, it may be advisable to preallocate the MVS sequential data set (with record format VB and record length 256).

3.1.3 Using copytree to migrate an HFS file system to zFS

The copytree utility is available from the z/OS UNIX Tools and Toys on the Internet. It is used for the logical copying of z/OS UNIX structures. Beginning with z/OS V1R3, it is distributed as a z/OS UNIX sample in the directory /samples of the root file system.

This utility has some useful functions that make it easy to use and that avoid possible loop conditions. The copytree utility has the following characteristics:

- It does not cross mount points.
- It can handle the case where the target structure is located within the source structure, even if it is the same file system.
- It ensures that the target directory is empty.
- It runs under TSO or the shell. To run under TSO, copy it to a REXX library located in SYSEXEC or SYSPROC.
- It has an option to automatically set the effective UID to 0 before starting if the user is a BPX.SUPERUSER.

To start a migration process with the copytree utility we created a small job, displayed in Figure 3-3, that uses the REXX procedure RXZFS (shown in B.3, “RXZFS” on page 366).

```
//ZFSJOB   JOB ,'COPYTREE',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Using copytree to copy or migrate UNIX structures
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET TIMEOUT=R0          <=== Timeout value in seconds, R0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
  source_dir=/u/hering/
  target_dir=/u/zfs/c03/
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
UNIXCMD="/samples/copytree -a $source_dir $target_dir";-
echo $UNIXCMD;$UNIXCMD+
# -------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
```

Figure 3-3 JCL using copytree to copy or migrate UNIX structures

The output of a successful run is shown in Figure 3-4 on page 131.

```
/samples/copytree -a /u/hering/ /u/zfs/c03/
Copying /u/hering/ to /u/zfs/c03/
Scanning for file nodes...
Skipping mountpoint: /u/hering//..
Processing 44 nodes
Creating directories
Creating other files
Setting file attributes

*******************

Copy complete.  Error count= 0
Directory errors:   0
Directories copied: 9
File errors:        0
Files copied:       14
Symlink errors:     0
Symlinks copied:    11
Char-spec errors:   0
Char-spec copied:   7
FIFO errors:        0
FIFOs copied:       0
Sparse file count:  0
ACL count:          2
```

Figure 3-4 Job output of a successful run of the JCL

3.1.4 Using pax in copy mode to migrate an HFS file system to zFS

Using **pax** in copy mode without an intermediate archive is a more efficient way of copying an HFS file system to a zFS file system. Figure 3-5 on page 132 shows a JCL example of using the **pax** utility to copy the file system, which is very similar to the steps needed when using an intermediate archive file.

- `cd /u/hering`
- `pax -pe -rwX . /u/zfs/c04`

An example JCL using this simple form is shown in Figure 3-5 on page 132, but this is not the recommended format with **pax**.

Tip: You should exploit the enhanced **pax** options available since z/OS V1R7 as illustrated in Figure 3-35 on page 153. This technique and more is used with utility COPYPAX, which is introduced next in “REXX procedure COPYPAX” on page 132.

In the JCL shown in Figure 3-5 on page 132, first, the source directory is set up as the working directory, and then **pax** is called to copy the data from the source to the target directory.

```
//ZFSJOB   JOB ,'COPYPAXS',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Simple way to use pax in copy mode for copying and migration
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET TIMEOUT=RO          <=== Timeout value in seconds, RO=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
  source_dir=/u/hering/
  target_dir=/u/zfs/c04
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
echo Copying $source_dir to $target_dir+
cd $source_dir && pax -p e -rwX . $target_dir+
# -------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
```

Figure 3-5 JCL for using pax in copy mode

3.1.5 REXX procedure COPYPAX

To make sure the copy works for all file systems, we created a REXX procedure named COPYPAX (shown in B.4, “COPYPAX” on page 374) that uses **pax** in copy mode with several additional functions. See B.4.1, “Benefits of COPYPAX” on page 375 for all the details.

Sample JCL to use this REXX procedure is shown in Figure 3-6 on page 133.

Tip: We are currently working on a further utility named “zFS REORG Tool” for reorganization of a zFS with options to use **pax**, **copytree** or IDCAMS/TSO REPRO. This can be a further option for copying a source zFS to a target zFS in a very flexible way. It works very much like the MIGRTOOL for migrating an HFS to zFS.

```
//ZFSJOB   JOB ,'COPYPAX',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=xxxx
//*MAIN SYSTEM=xxxx
//* --------------------------------------------------------------------
//* Use pax to copy source directory structure to target directory
//* Property of IBM  (C) Copyright IBM Corp. 2002-2010
//* --------------------------------------------------------------------
//  SET SOURCED='/u/hering'                      <=== Source directory
//  SET TARGETD='/tmp/hering'                    <=== Target directory
//* --------------------------------------------------------------------
//  SET TIMEOUT=RO          <=== Timeout value in seconds, RO=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* --------------------------------------------------------------------
//COPYPAX  EXEC PGM=IKJEFT01,PARM='COPYPAX &TIMEOUT &SOURCED &TARGETD'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//CPPXPARM DD DATA,DLM=##
TARGET_MUST_BE_EMPTY=Y < Target structure must be empty      (--Y--|N)
CORRECT_INV_FD_FILES=Y < Correct invalid fd files            (--Y--|N)
    COPY_PAX_VERBOSE=N < List all objects copied or created  (Y|--N--)
   NO_FILE_OVERWRITE=N < Prevent replacing of existing files (Y|--N--)
   PROCESS_INFO_MSGS=Y < Display process data in batch mode  (--Y--|N)
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* --------------------------------------------------------------------
```

Figure 3-6 JCL to use pax for copying a directory structure

The job output of a successful run is shown in Figure 3-7.

```
------------------------------------------------------------------------------
The main process ID for this job is 67305701. If you should need to stop
processing use the following UNIX command in authorized mode to do this:
kill 67305701
------------------------------------------------------------------------------
Verifying existence of source directory /u/hering...
Verifying existence of target directory /tmp/hering...
Copying the source to the target structure using pax...
Searching for and correcting invalid target file descriptor entries...
READY
END
```

Figure 3-7 Job output messages provided by the JCL

3.2 Moving the HFS root to zFS

In z/OS V1R3, you can run the z/OS UNIX environment with the root file system, or the version file system in case of sysplex sharing, as a zFS root instead of an HFS root.

Note: This test was performed to demonstrate that this is possible. It does *not* describe how to switch to zFS.

HFS-to-zFS migration tools are described in 3.5, “Migration tool BPXWH2Z” on page 148 and 3.6, “Alternate HFS-to-zFS migration tool” on page 158.

3.2.1 Creating a compatibility mode aggregate

We determined the number of blocks used and available in the HFS root file system and then decided how to create the new zFS file system to be large enough to have about the same size as the HFS. We used JCL to determine this, but will only reference the commands used to create a zFS root using the shell in SU mode; see Figure 3-8 on page 134.

```
#> zfsadm define -aggregate OMVS.ROOTCOMP.ZFS -storageclass SCCOMP -megabytes
2000 100
IOEZ00248E VSAM linear dataset OMVS.ROOTCOMP.ZFS successfully created.
#> zfsadm format -aggregate OMVS.ROOTCOMP.ZFS -compat
IOEZ00077I HFS-compatibility aggregate OMVS.ROOTCOMP.ZFS has been successfully
created
#> mkdir -m 700 /u/zfs/clone
#> /usr/sbin/mount -f OMVS.ROOTCOMP.ZFS -t ZFS /u/zfs/clone
```

Figure 3-8 Creating and formatting the compatibility mode aggregate

3.2.2 Copying the HFS root data into the zFS file system

To copy the HFS root we used the JCL shown in Figure 3-9. It took approximately 10 minutes to copy about 2,000 MB of data. Because there were many file systems mounted on the root, which included many mount points that had to be skipped during the copy, it was then necessary to create these missing directories afterwards.

```
//ZFSJOB   JOB ,'COPYPAX',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Use pax to copy source directory structure to target directory
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET SOURCED='/'                       <=== Source directoy
//  SET TARGETD='/u/zfs/clone'            <=== Target directoy
//* -------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='COPYPAX &TIMEOUT &SOURCED &TARGETD'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
```

Figure 3-9 Job to copy the HFS root to the zFS root

To recreate the missing directories, we created another REXX procedure named ADDMNTPS (shown in B.5, “ADDMNTPS” on page 385). We used it in the JCL shown in Figure 3-10 on page 135.

```
//ZFSJOB   JOB ,'ADDMNTPS',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Add missing mount point directories after clone process
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET SOURCED='/'                                 <=== Source directoy
//  SET TARGETD='/u/zfs/clone'                      <=== Target directoy
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC                <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='ADDMNTPS &TIMEOUT &SOURCED &TARGETD'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

Figure 3-10 JCL to add missing mount point mounts after the copy process

For our root, the output messages from the submitted job are shown in Figure 3-11. You can see that the REXX procedure looks for the missing directories and recreates them.

```
Looking for and creating the missing directory mount points...
Mount point /u/zfs/clone/SYSTEM/tmp exists already...
Mount point /u/zfs/clone/usr/lpp/esa110 has been created...
Mount point /u/zfs/clone/usr/lpp/customizer has been created...
Mount point /u/zfs/clone/was/hod60 has been created...
Mount point /u/zfs/clone/web/hod60 has been created...
Mount point /u/zfs/clone/usr/lpp/HOD has been created...
Mount point /u/zfs/clone/was/juha has been created...
Mount point /u/zfs/clone/web/juha has been created...
Mount point /u/zfs/clone/usr/lpp/java213b has been created...
Mount point /u/zfs/clone/usr/lpp/java18 has been created...
Mount point /u/zfs/clone/usr/lpp/imsweb has been created...
Mount point /u/zfs/clone/SYSTEM/var has been created...
Mount point /u/zfs/clone/u exists already...
Mount point /u/zfs/clone/SYSTEM/etc exists already...
```

Figure 3-11 Job output when adding missing mount point mounts

Note that some of the directories had been defined already because we did special tests between running the two jobs.

3.2.3 Using the chroot command to test the new zFS file system

If you have the appropriate privileges, you can use the **chroot** command to gain access to the HFS root as a zFS file system. This command changes the root directory to the directory specified by the directory parameter of the specified command. The new root directory also contains its children. To be able to issue this command, you should:

- Be a superuser (UID=0)

- Be a member of the BPX.SUPERUSER facility class

The directory path name is always relative to the current root. For your process to operate properly after the **chroot** command is issued, you need to have all the files that your programs depend on in your new root, as explained here:

- If your new root is /tmp and you issue an ls, you will get a `Not found` error. To use ls with /tmp as your new root, you will need a /tmp/bin with ls in it before you issue the **chroot** command.
- After **chroot** is issued, your current working directory is the new root (directory); **chroot** does not change environment variables.

You can use this command to perform multiple tests using the new root file system as the root temporarily after initializing a shell environment; see in Figure 3-12.

```
HERING:/u/hering:$> su
HERING:/u/hering:#> pwd
/u/hering
HERING:/u/hering:#> /usr/sbin/chroot /u/zfs/clone sh
HERING:/:#> pwd
/
HERING:/:#> su hering
FSUM5019 Enter the password for hering:
HERING:/:$> sh -L
You are logon on system SC43.
HERING:/u/hering:$> pwd
/u/hering
HERING:/u/hering:$> id
uid=888(HERING) gid=0(SYS1)
HERING:/u/hering:$>
```

Figure 3-12 Setting up a UNIX System Services shell environment using the new file system as root

Note: You cannot use **su -s hering** because after running **chroot**, your MVS identity is BPXROOT. Do not give BPXROOT read access to the profile BPX.SRV.HERING in the SURROGAT class.

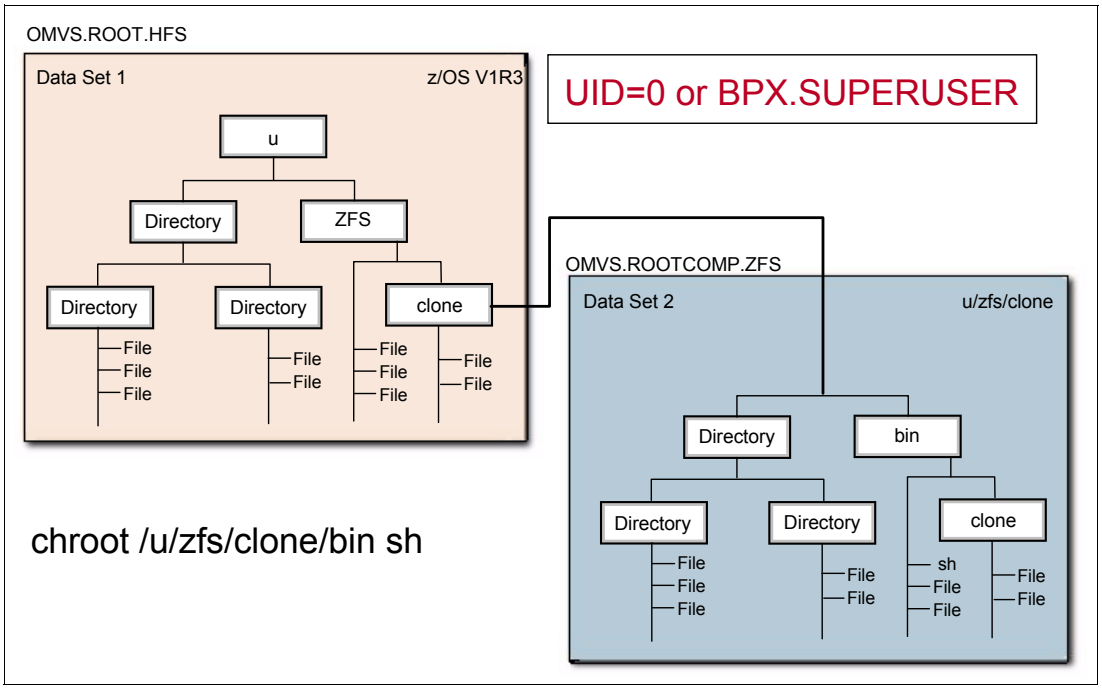


Figure 3-13 Using the chroot command to test the root as a zFS root

3.2.4 IPLing the system with the zFS file system as the root

The final step is to replace the root statement in the BPXPRMxx parmlib member, as shown in Figure 3-14, to take the new zFS file system as the root.

```
ROOT     FILESYSTEM('OMVS.ROOTCOMP.ZFS')
         TYPE(ZFS)              /* TYPE OF FILE SYSTEM              */
         MODE(RDWR)             /* (OPTIONAL) CAN BE READ OR RDWR.  */
                                /*  DEFAULT = RDWR                  */
```

Figure 3-14 BPXPRMxx ROOT statement for the zFS file system

We compared important time stamps during IPL until the following message was displayed:

```
BPXI004I OMVS INITIALIZATION COMPLETE
```

This message, which was issued for the IPL with a zFS root file system and another IPL for the one that had the HFS file system as the root, showed no essential differences in time intervals measured from the moment of the start of OMVS for the following events:

- zFS is active.
- The root file system is mounted.
- ETC file system is mounted.
- BPXOINIT has been started.
- OMVS initialization is completed.

Thus, zFS in z/OS V1R3 can be a possible alternative to HFS for all possible situations and purposes, including the important case of the root or version file system.

3.2.5 Switching from an HFS to a zFS version root without an IPL

In a UNIX System Services sysplex sharing environment, the comparable situation is switching the version root file system. Although the best solution is to do this when a new IPL is planned anyway, there is at least the possibility of avoiding an IPL.

After creating the new zFS copy, and using a process that is very similar to the process described in 3.2.2, “Copying the HFS root data into the zFS file system” on page 134, you can use the option **-q** of the **/usr/sbin/mount** command to discover all the lower mount points down the HFS version root structure, as shown in Figure 3-15.

```
$> /usr/sbin/mount -q /'$VERSION'
/Z16RD1/usr/lpp/Tivoli
/Z16RD1/usr/lpp/java
/Z16RD1/usr/lpp/ixm
/Z16RD1
$>
```

Figure 3-15 Listing mount point directories in the version structure

The following steps will replace the HFS version root with the new zFS version root while keeping all the file systems mounted down the current version as is. Do the following:

1. Create a new mount point directory in the sysplex root, for example /Z16RD1Z, and mount the new zFS version file system at this new mount point.
2. Replace all the directories in this new zFS file system corresponding to a mount point within the current HFS version root with a symbolic link pointing to this corresponding mount point in the HFS.
3. Run the system command SETOMVS VERSION='Z16RD1Z' to switch the version file system to the new zFS file system.

   Note: Although the procedure described here is not a recommended one, the impact to applications may be low because the HFS file system is retained and available in parallel.

4. Before IPLing and using the zFS version root correctly, replace the symbolic links with directories again.

3.2.6 The impact of stopping zFS for other mounted file systems

When using a zFS root file system, stopping zFS will always force an unmount for all zFS file systems and all other file systems mounted below the zFS root file system.

However, in z/OS V1R3 there are still situations that require a restart of zFS. For example, if you want to change the default or currently set cache size values specified as processing options in the IOEFSPRM parameter file, zFS must be stopped and restarted. This restriction has been removed with z/OS V1R4; see 2.8.5, “Dynamic configuration” on page 39 for more information.

In the case of a zFS root, this means that all file systems in the system will get unmounted, as shown in Figure 3-16 on page 139 and Figure 3-17 on page 140.

Attention: Avoid conditions that would stop zFS if you run with a zFS root file system.

```
P ZFS
IOEZ00050I zFS kernel: Stop command received.
IOEZ00048I Detaching aggregate OMVS.CMP04.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP04.ZFS.
IOEZ00048I Detaching aggregate OMVS.CMP03.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP03.ZFS.
IOEZ00048I Detaching aggregate OMVS.CMP02.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP02.ZFS.
IOEZ00048I Detaching aggregate OMVS.CMP01.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.CMP01.ZFS.
IOEZ00048I Detaching aggregate OMVS.ROOTCOMP.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.ROOTCOMP.ZFS.
IOEZ00048I Detaching aggregate OMVS.MUL02.ZFS
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.MUL02.M01.ZFS.
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.MUL02.M02.ZFS.
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.MUL02.m03.ZFS.
IOEZ00061I zFS kernel: forcing unmount of file system OMVS.MUL02.M04.ZFS.
IOEZ00048I Detaching aggregate OMVS.MUL01.ZFS
IOEZ00057I zFS kernel program IOEFSCM is ending
...
```

Figure 3-16 Unmount of zFS file systems when stopping zFS

The other file systems are also unmounted, as shown in Figure 3-17 on page 140.

```
...
IEF196I IGD104I OMVS.SC43.ESA110.JAVA213.HFS                RETAINED,
IEF196I DDNAME=SYS00026
IEF196I IGD104I OMVS.SC43.WSC206.HFS                        RETAINED,
IEF196I DDNAME=SYS00024
IEF196I IGD104I OMVS.SC43.WAS.HOD60.HFS                     RETAINED,
IEF196I DDNAME=SYS00023
IEF196I IGD104I OMVS.SC43.WEB.HOD60.HFS                     RETAINED,
IEF196I DDNAME=SYS00022
IEF196I IGD104I OMVS.SC43.HODV6GA.HFS                       RETAINED,
IEF196I DDNAME=SYS00021
IEF196I IGD104I OMVS.SC43.WAS.JUHA                          RETAINED,
IEF196I DDNAME=SYS00020
IEF196I IGD104I OMVS.SC43.WEB.JUHA                          RETAINED,
IEF196I DDNAME=SYS00019
IEF196I IGD104I OMVS.SC43.JAVA213.V010207.HFS               RETAINED,
IEF196I DDNAME=SYS00018
IEF196I IGD104I OMVS.SC43.JAVA118.V001122.HFS               RETAINED,
IEF196I DDNAME=SYS00017
IEF196I IGD104I OMVS.SC43.IMSWEB.HFS                        RETAINED,
IEF196I DDNAME=SYS00007
IEF196I IGD104I OMVS.SC43.VAR                               RETAINED,
IEF196I DDNAME=SYS00003
IEF196I IGD104I OMVS.SC43.USERS                             RETAINED,
IEF196I DDNAME=SYS00002
IEF196I IGD104I OMVS.SC43.ETC                               RETAINED,
IEF196I DDNAME=SYS00001
IEF196I IGD104I OMVS.SC43.ADSM.LOGS                         RETAINED,
IEF196I DDNAME=SYS00025
IEF196I IGD104I OMVS.SC43.BIN.HFS                           RETAINED,
IEF196I DDNAME=SYS00028
IEF196I IGD104I OMVS.SC43.HERING.HFS                        RETAINED,
IEF196I DDNAME=SYS00027
IEF196I IGD104I OMVS.SC43.OS390R7.PDF                       RETAINED,
IEF196I DDNAME=SYS00016
IEF196I IGD104I OMVS.SC43.MDH                               RETAINED,
IEF196I DDNAME=SYS00015
IEF196I IGD104I OMVS.SC43.HDM                               RETAINED,
IEF196I DDNAME=SYS00014
IEF196I IGD104I OMVS.SC43.GRAAFF                            RETAINED,
IEF196I DDNAME=SYS00013
IEF196I IGD104I OMVS.SC43.NICHOLS                           RETAINED,
IEF196I DDNAME=SYS00012
IEF196I IGD104I OMVS.SC43.MIKEM                             RETAINED,
IEF196I DDNAME=SYS00011
IEF196I IGD104I OMVS.SC43.VAINI                             RETAINED,
IEF196I DDNAME=SYS00010
IEF196I IGD104I OMVS.SC43.SHARKMN                           RETAINED,
IEF196I DDNAME=SYS00009
IEF196I IGD104I OMVS.SC43.ROGERS.HFS                        RETAINED,
IEF196I DDNAME=SYS00008
IEF196I IGD104I OMVS.SC43.PAUL                              RETAINED,
IEF196I DDNAME=SYS00006
IEF196I IGD104I OMVS.SC43.AOPSS                             RETAINED,
IEF196I DDNAME=SYS00005
IEF196I IGD104I OMVS.SC43.PEGGYR.HFS                        RETAINED,
IEF196I DDNAME=SYS00004
*628 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO IGNORE.
```

Figure 3-17 Unmount of other file systems when stopping zFS

Figure 3-18 on page 141 shows that finally, only the SYSROOT is left.

```
D OMVS,F
BPXO045I 13.52.51 DISPLAY OMVS 818
OMVS     000F ACTIVE          OMVS=(RH,3B)
TYPENAME   DEVICE ----------STATUS----------- MODE
BPXFTCLN        2 ACTIVE                      RDWR
  NAME=SYSROOT
  PATH=/
```

Figure 3-18 Dummy SYSROOT left after an unmount of all file systems

3.3 Automount facility for zFS

You can change your current automount facility to manage your user file systems as zFS file systems. Then, any zFS file system that was previously mounted will be mounted by the automount facility when the mount point is referenced by a user. With the automount facility, you do not need to mount user file systems at system IPL.

You also do not need to request that operators perform mounts for other file systems. In addition, the facility simplifies the addition of new users, because you do not need to update your parmlib definitions for mounts. You can establish a simple automount policy to manage user home directories, as shown in Figure 3-19 on page 142.

Using automount with a zFS multifile aggregate containing the file systems for users confers the following benefits:

- By defining a multifile aggregate for the z/OS UNIX users, all users can share the allocated space in the aggregate. This eliminates having to increase space for individual users in their file systems. If all the space is used in the aggregate, you can issue the **`zfsadm grow`** command to increase the space for all users.
- Users tend to have very different needs for their amount of space, so you avoid wasting disk space because it is not needed to provide a separate contingent of disk space for each individual user.
- You need to define a larger logical amount of space for users using the **`zfsadm setquota`** command only if those users have used up their allowed quota. No physical change is necessary if there is still enough space available in the aggregate.

Restriction: Using a multifile system aggregate is inappropriate if you are running in a sysplex sharing environment. See Chapter 5, “Sysplex considerations” on page 201 for further information about this topic.

We are describing a zFS option regarding automount when you are running in a single system without sharing file systems in R/W mode in a sysplex.

Note: When using automount in sysplex sharing environments, compatibility mode aggregates provide a zFS means equivalent to HFS.

3.3.1 Setting up the automount facility

Figure 3-19 on page 142 illustrates the steps required to define the automount environment.

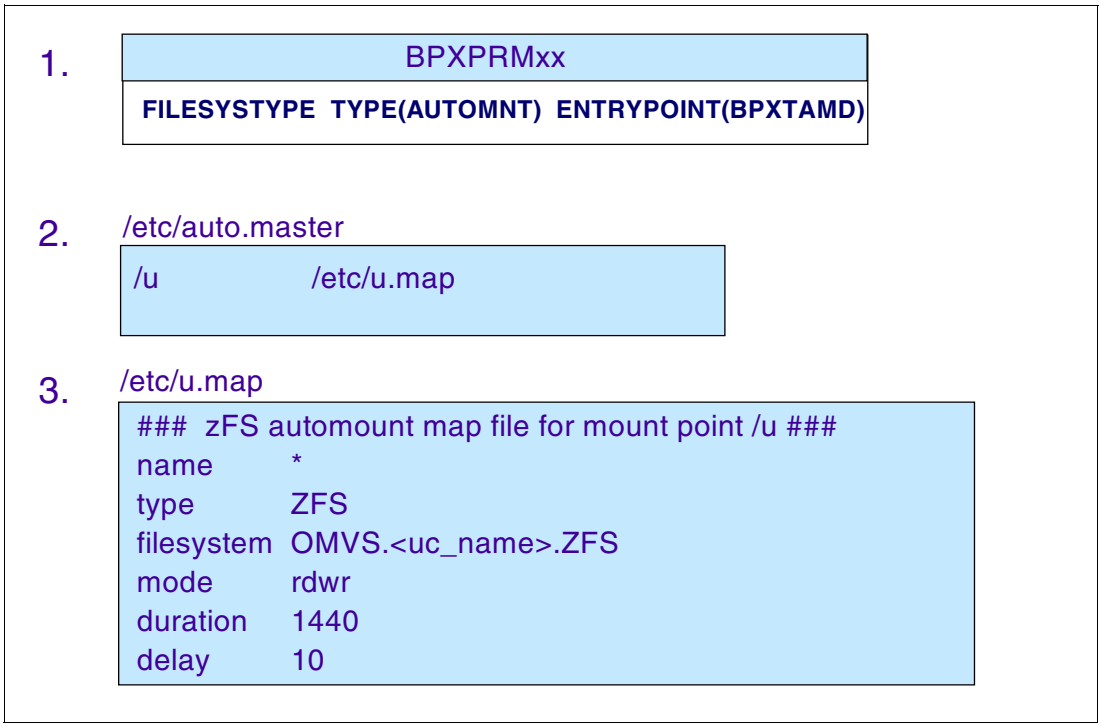


Figure 3-19 Setting up the automount facility

1. As shown in Figure 3-19, you must have the automount FILESYSTYPE active, either by having the statement in the active BPXPRMxx member during IPL as shown or by using the **SETOMVS RESET=(xx)** command to do it dynamically.
2. The auto.master file in /etc must contain all mount points that should be managed by automount.
3. A map file describes the automount policy for a specific mount point. If you choose to include a system name, you can specify &SYSNAME instead of <system>, which exploits the new system symbols support of automount.

3.3.2 Defining the automount multifile system aggregate

First you define and attach the aggregate. You can do that from the OMVS shell, as shown in Figure 3-20.

```
#> zfsadm define -aggregate OMVS.MULTUSER.ZFS -storageclass SCCOMP -megabytes
10 10
IOEZ00248E VSAM linear dataset OMVS.MULTUSER.ZFS successfully created.
#> zfsadm format -aggregate OMVS.MULTUSER.ZFS
Done.  OMVS.MULTUSER.ZFS is now a zFS aggregate.
#> zfsadm attach -aggregate OMVS.MULTUSER.ZFS
IOEZ00117I Aggregate OMVS.MULTUSER.ZFS attached successfully
```

Figure 3-20 Defining and attaching the aggregate

In parallel, we added a define_aggr statement to the IOEFSPRM parameter file, shown in Figure 3-21 on page 143.

```
define_aggr R/W attach nbs aggrfull(85,5)
            cluster(OMVS.MULTUSER.ZFS)
```

Figure 3-21 Including the aggregate in the list of aggregates in the IOEFSPRM file

Defining file systems by quota

Figure 3-22 shows how to create user file systems with a sum of allowed quotas (15 MB), which is already more than the initial size of the whole aggregate as it has been defined (10 MB). This demonstrates the flexibility of this method of defining automount user file systems.

```
#> zfsadm create -filesystem OMVS.ALAN.ZFS -aggregate OMVS.MULTUSER.ZFS -size
5000 -owner ALAN -perms o700
IOEZ00099I File system OMVS.ALAN.ZFS created successfully
#> zfsadm create -filesystem OMVS.TOLOD.ZFS -aggregate OMVS.MULTUSER.ZFS -size
5000 -owner TOLOD -perms o700
IOEZ00099I File system OMVS.TOLOD.ZFS created successfully
#> zfsadm create -filesystem OMVS.YVES.ZFS -aggregate OMVS.MULTUSER.ZFS -size
5000 -owner YVES -perms o700
IOEZ00099I File system OMVS.YVES.ZFS created successfully
```

Figure 3-22 Creating user file systems

Figure 3-23 shows the z/OS UNIX automount file structure and the zFS multifile aggregate with the user file system that it contains.

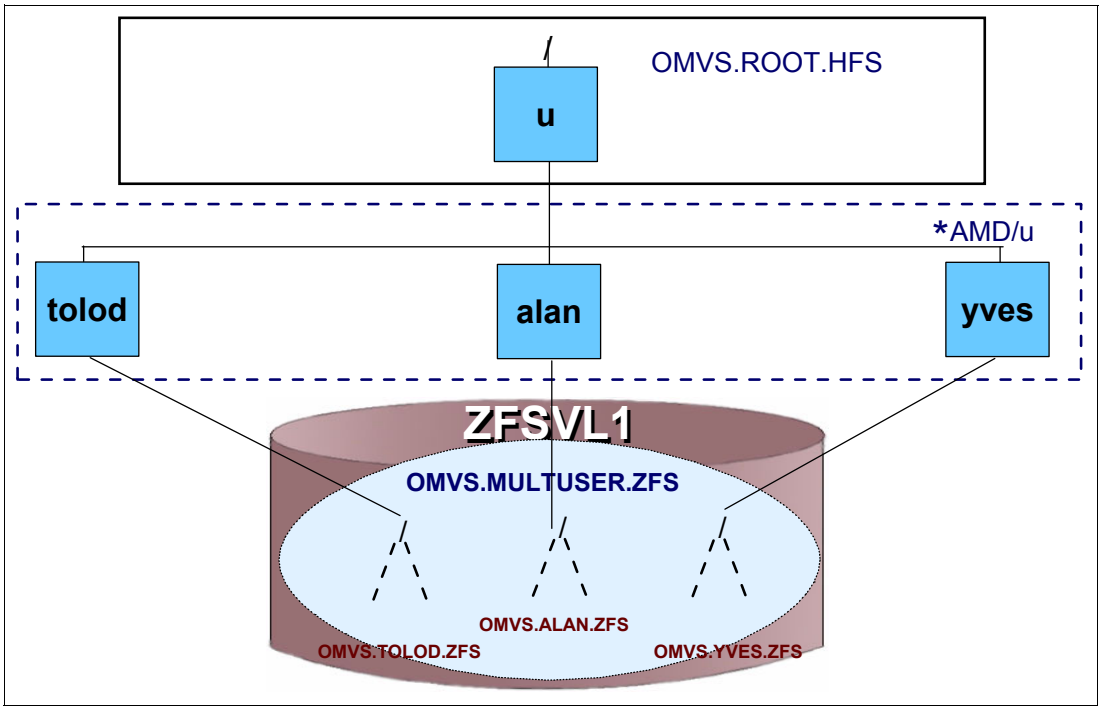


Figure 3-23 zFS multifile system aggregate used with automount

After starting (or restarting) automount as shown in Figure 3-24 on page 144, the new automount policy becomes active.

```
#> /usr/sbin/automount
FOMF0107I Processing file /etc/u.map
FOMF0108I Managing directory /u
#>
```

Figure 3-24 Starting automount with the new automount policy

Figure 3-25 shows commands to verify that automount is working as desired.

```
#> ls -ER /u

/u:
total 0
#> ls -Ed /u/alan
drwx------        2 ALAN     SYS1         256 Apr 10 19:21 /u/alan
#> echo xxxxxxxxxxxxxxxxxxx > /u/tolod/testfile
#> ls -ER /u

/u:
total 32
drwx------        2 ALAN     SYS1         256 Apr 10 19:21 alan
drwx------        2 TOLOD    SYS1         288 Apr 10 19:29 tolod

/u/alan:
total 0

/u/tolod:
total 16
-rw-r--r--  --s-  1 TOLOD    SYS1          21 Apr 13 16:36 testfile
#>
```

Figure 3-25 Verifying automount processing

3.3.3 Automount assistance for HFS-to-zFS migration

A new UNIX System Services function in z/OS V1R6 now allows you to have a single automount policy to manage both HFS and zFS file systems with a single map name file entry.

Note: This has been made available for z/OS V1R5 with APAR OA06364 and PTF UA10075.

A necessary prerequisite to allow for this is the dynamic creation of zFS file systems to have the same possibilities as HFS. See 2.11.5, “Dynamic creation of automounted zFS file systems” on page 53 for more information about this topic. In previous releases of z/OS, such a generic automount policy was not available. All file systems had to be of the same type.

In HFS-to-zFS migration scenarios, clients are more likely to migrate file systems over time rather than migrating all file systems at once. The capability of allowing automount to be able to manage both HFS and zFS file systems in one automount policy is a basic capability to help enable this migration.

The new support is used when HFS or zFS is specified as the file system type in the automount policy. If either `allocany` or `allocuser` is specified, a new file system is allocated as before, using the file system type specified in the automount policy. However, if the data set already exists, a check is done to see whether it is an HFS file system or a zFS aggregate, and then the mount is directed to the appropriate PFS.

Attention: An automount **`parm`** specification must not be used to make use of this capability.

This characteristic can be used to effectively disable the support for the dynamic PFS Type determination.

Example of an automount policy handling HFS and zFS in parallel

Figure 3-26 shows the activation of an automount policy that exploits this and then lists it. Because the intention is to go to zFS, the “type” is specified as zFS to create new file systems as zFS aggregates. This is done for new UNIX System Services users referencing their home directory and file structure the first time.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.05.00
#> /usr/sbin/automount
FOMF0107I Processing file /etc/auto.map
FOMF0108I Managing directory /u
$> /usr/sbin/automount -q

/u

name                 *
filesystem           OMVS.<uc_name>.HFS
type                 ZFS
allocuser            space(1,1) storclas(OPENMVS)
mode                 rdwr
duration             1440
delay                360

$>
```

Figure 3-26 Activating and listing an automount policy having ZFS set as “type”

Automount now assures that zFS aggregates are mounted if a corresponding UNIX file structure is referenced and the file system is not yet mounted. But this is true only if the data set is HFS, and not of type zFS. This is shown in Figure 3-27 on page 146.

Note: For type zFS, there is no need to specify the unit type for a space value in an `allocuser` or `allocany` setting because `cyl` (cylinder) is always chosen.

This is currently not documented in *UNIX System Services Command Reference*.

Important: Even if you specify unit type as `tracks` or `blocks` this is ignored, and cyl is selected instead.

```
$> su
#> /usr/sbin/unmount /u/heritst
#> exit
$> ls -En /u
total 80
drwxr-xr-x        3 0        1           8192 Aug 28 17:48 bin
drwxr-xr-x       27 888      0           8192 Aug 30 19:55 hering
drwxr-xr-x        2 0        2           8192 Jun 28  2002 ldapsrv
drwxr-xr-x        4 68216    2           8192 Aug 30 06:30 patrick
drwxr-xr-x        2 0        2           8192 Nov 15  2001 syslogd
$> ls -End /u/heritst
drwx------        3 88       2           8192 Feb  5  2004 /u/heritst
$> df -v /u/heritst
Mounted on     Filesystem                Avail/Total    Files      Status
/u/heritst     (OMVS.HERITST.HFS)        992/1440       4294967282 Available
HFS, Read/Write, Device:196, ACLS=Y
File System Owner : SC63        Automove=Y      Client=N
Filetag : T=off   codeset=0
```

Figure 3-27 Automount processing to mount an HFS while policy “type” is zFS

3.4 Logical file system support for zFS in z/OS V1R7

In z/OS V1R7, zFS is the preferred file system. Continued use of HFS is discouraged because HFS may no longer be supported in future releases. At that time, all remaining HFS file systems must be migrated to zFS. This migration to zFS file systems needs to be well planned because it will take significant effort to migrate all data from HFS file systems to zFS file systems.

3.4.1 Migration and coexistence considerations

No capability exists to convert an in-place file system to zFS, or to use an HFS data set with zFS. Therefore, file systems that are used in a read-write mode will have to be made unavailable for updates while being migrated to zFS, and two data sets will exist at least during the migration.

The unavailability of the data for read-write access must be planned for; the space for two file systems that are about the same size must be allocated as well.

Mounts using a place holder

It is now possible to specify a file system name with substitution place holders. The place holder is ///, which represents the string HFS or zFS as appropriate if the file system type is HFS. An example is shown in Figure 3-28.

```
MOUNT FILESYSTEM('OMVS.ZOSR17.MAN.///') TYPE(HFS) MODE(READ)
MOUNTPOINT('/usr/man')
```

Figure 3-28 Mount command using placeholder ///

Mount processing will first substitute zFS for the placeholder, then check if the data set exists. If non-existent, HFS will be used in the placeholder, and existence of the data set will be checked again. The mount will be directed either to zFS or HFS, depending on which data set

was found. This allows easier migration from HFS file systems to zFS file systems because the need to change mount policies and scripts is simplified or eliminated.

If a generic entry contains a file system parameter similar to `OMVS.HFS.<uc_name>`, you can change it to use a pattern that will replace the HFS string with zFS when appropriate by using `OMVS.///.<uc.name>`.

Attention: Using the HFS file system type either as a generic type or as a file system name substitution place holder is not compatible in a mixed-level sysplex. Pre-z/OS V1R7 systems will fail the mounts.

HFS as a generic file system type

HFS and zFS are now generic file system types that can be specified as either HFS or zFS.

Mount processing will first search for a data set matching the file system name. If the data set is not an HFS data set and zFS has been started, the file system type will be changed to zFS and the mount proceeds to zFS. If the data set was not found, the mount proceeds to zFS (assuming zFS is started). If zFS is not started, the type is not changed and the mount proceeds to HFS.

Note: If the zFS data set names are to be the same as the HFS data set names, it is likely that scripts or policies that are used to mount file systems would not need to be changed.

Figure 3-29 lists a sequence of commands demonstrating aspects of the new LFS support for zFS.

```
#> /usr/sbin/mount -f OMVS.HERING.TEST.ZFS /u/hering/test
#> /usr/sbin/unmount /u/hering/test
#> /usr/sbin/mount -t HFS -f OMVS.HERING.TEST.ZFS /u/hering/test
#> /usr/sbin/unmount /u/hering/test
#> /usr/sbin/mount -f OMVS.HERING.TEST./// /u/hering/test
#> df -v /u/hering/test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.TEST.ZFS)    1056/22750     4294966910 Available
ZFS, Read/Write, Device:790, ACLS=Y
File System Owner : SC65        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.TEST.ZFS
#> /usr/sbin/unmount /u/hering/test
#> /usr/sbin/mount -t HFS -f OMVS.HERING.TEST./// /u/hering/test
#> df -v /u/hering/test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.TEST.ZFS)    1056/22750     4294966910 Available
ZFS, Read/Write, Device:791, ACLS=Y
...
```

Figure 3-29 Samples using the new LFS support for zFS in z/OS V1R7

3.4.2 Considerations for migrated data sets

Currently, OMVS initialization will suspend mounts until the recalls are complete for HFS data sets. Mounts will complete async to this processing for zFS data sets. Parmlib mounts in z/OS V1R7 will change to suspend for zFS mounts and fail HFS mounts for migrated data sets.

Attention: Critical file systems should not be HSM migrated; therefore, mounts performed from BPXPRMxx should not specify data sets that are eligible for HSM migration.

Non-parmlib zFS mounts are done asynchronously. There is an accommodation in automount for HFS so that an HSM migrated data set is recalled prior to entering the mount flow.

Tip: Consider moving noncritical mounts for HSM data sets that can be migrated to a script such as /etc/rc.

zFS has higher performance characteristics than HFS, and it is the strategic file system. The LFS Filecache function is no longer needed so it was removed.

Important: The LFS Filecache function is no longer supported because zFS handles caching within the PFS.

3.4.3 zFS file system considerations

If export lists are used with NFS or SMB, those export lists must be changed if the data set name is changed. You can make these changes before migrating to zFS by exporting both names.

zFS file systems that are greater than 4 GB (about 5825 cylinders of a 3390) must be defined with a data class that includes extended addressability.

zFS file systems cannot be striped.

Multifile system aggregates must not contain file systems with a name that is the same as a cataloged HFS data set.

3.5 Migration tool BPXWH2Z

BPXWH2Z is an ISPF-based tool, available beginning with z/OS V1R7, that assists you in migrating from an HFS file system to zFS. The benefit is that the complex process of migrating from HFS to zFS has been simplified. Many details needed for the migration of HFS file systems to zFS are handled by this tool, including the actual copy operation.

Note the following points regarding BPXWH2Z:

- This HFS-to-zFS migration tool is located in a partitioned data set named SYS1.SBPXEXEC.
- It is ISPF panel-based and dynamically builds its own panels.
- It uses ISPF to set up the file system migration.
- The actual migration work can be run in UNIX background as well as TSO foreground.
- It migrates HFS file systems, both mounted and unmounted, to zFS file systems.
- It uses **pax** to perform the actual copy operation.
- It defines zFS aggregates by default to be approximately the same size as the HFS. The new allocation size can be changed.

- “Enhancements to BPXWH2Z” on page 155 describes the many new functions that have been added to BPXWH2Z since it was released with z/OS V1R7.

After getting the list of data sets to migrate, this utility will obtain data set information about each data set. Data sets that do not exist, are HSM migrated, or are not HFS data sets will be excluded from the list. The resulting data set list is presented in a table.

Source HFS file systems should not be in use for update when you are doing the migration to prevent loss of updates during this time.

Attention: Mounted file systems will be switched to read-only during the migration.

3.5.1 Using BPXWH2Z

The utility can be run with no arguments and it will prompt for an HFS file system name. You can also specify one or more file system names as an argument. The names can be simple names or patterns that you might use for the ISPF data set list panel. To use the migration tool, enter `bpxhw2x` on the ISPF Option 6 command line; the panel shown in Figure 3-30 will be displayed. To migrate a file system, specify the file system name on the panel as shown, `OMVS.ROGERS.TEST`.

```
---------------------------  DATA SET SPECIFICATION  ---- Enter required field
Command ===>

Use the HELP command for usage information on this tool.
Enter the name of the HFS file system to migrate to a zFS file system.
Note that a data set pattern can be used.
If the file system is not currently mounted it will be mounted on a
temporary directory during the migration.


File system name: OMVS.ROGERS.TEST_____________________________________________
```

Figure 3-30 First panel of migration tool

Pressing Enter displays the panel shown in Figure 3-31, where you can specify SMS classes if they are required.

```
---------------------------  CLASS AND VOLUME DEFAULTS -------------------------
Command ===>

The volume or SMS classes for zFS allocations will default to the
same names as the current HFS allocation.  You can change these
individually for each new allocation.  If you want the default for
each new allocation to be set to specific values other than that of
the current HFS allocation, enter those values here and Press Enter.

Default volume  . . . . .: _______
Default data class  . . .: _______
Default storage class . .: _______
Default management class : _______
```

Figure 3-31 Panel to specify SMS classes if needed

Pressing Enter displays the panel in Figure 3-32 on page 150. This panel shows the HFS and zFS data sets names and the space allocations for the migration.

By default, the panels will be initialized such that your HFS data set will be renamed with a .SAV suffix and the zFS data set will be renamed to the original HFS name.

This migration process makes use of /tmp.

Important: The file system containing /tmp should not be migrated with this utility.

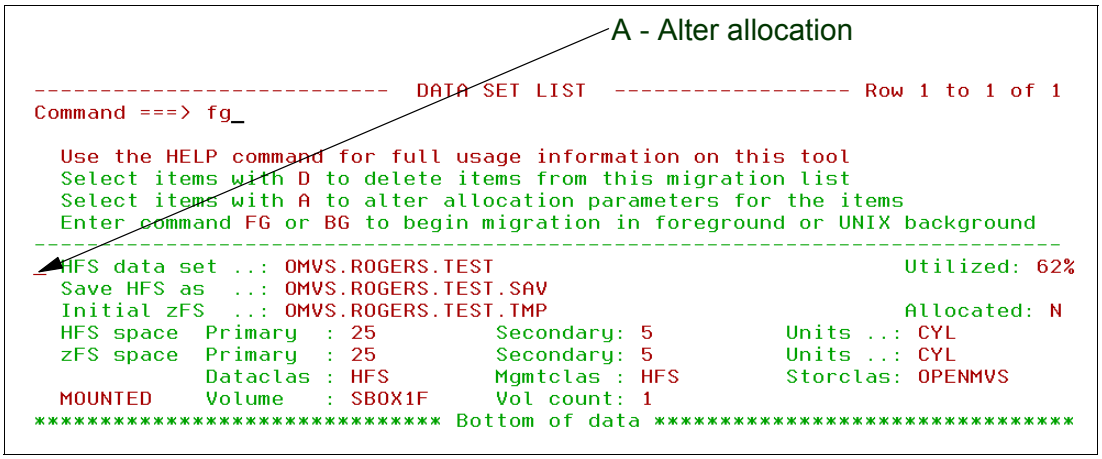


Figure 3-32 Panel describing the migration data sets

To begin the migration, enter the command **FG** or **BG** to run the migration in foreground or background. If this is run in background, the tool will keep a standard output log and also a summary log. The prefix for the pathnames will be displayed. The commands **D** or **A** may be specified for each HFS data set to delete items from the migration list or to alter allocation parameters for the items.

Each table entry shows the data set names that will be used, current HFS space utilization and space allocation, and allocation parameters that will be used to create the zFS file system. Select a row to alter the allocation attributes or data set names. Rows can also be deleted. Additional rows cannot be added.

Pressing Enter displays the messages shown in Figure 3-33.

```
Migrating OMVS.ROGERS.TEST
 creating zFS OMVS.ROGERS.TEST.TMP
 copying OMVS.ROGERS.TEST to OMVS.ROGERS.TEST.TMP Blocks to copy: 2832
 IGD01009I MC ACS GETS CONTROL &ACSENVIR=RENAME
 IGD01009I MC ACS GETS CONTROL &ACSENVIR=RENAME
 IDC0531I ENTRY OMVS.ROGERS.TEST.TMP ALTERED
 IDC0531I ENTRY OMVS.ROGERS.TEST.TMP.DATA ALTERED
 mount /u/rogers/tmp OMVS.ROGERS.TEST ZFS 1
 ***
```

Figure 3-33 Messages issued by the migration tool following the migration

After an HFS file system is migrated to zFS, the HFS data set is renamed to the name in the “Save HFS as” field shown in Figure 3-32 and the zFS data set is renamed to the old HFS data set name, as shown in the mount command in Figure 3-33. If you do not want a data set to be renamed, set it to the HFS data set name or clear the field and the tool will set it to the HFS name.

This utility will best handle file systems that are not currently mounted or file systems that are mounted but do not have other file systems mounted below them. If the file system is mounted, it will be unmounted and the new zFS put in its place.

Attention: If the file system contains active mount points, the tool will attempt to unmount that subtree to replace the HFS with the zFS and put back everything it needed to unmount.

Altering allocation options

If your zFS file system is not preallocated, it will be allocated based on the attributes for the HFS data set. You will have an opportunity to alter any of these attributes. To alter an attribute, enter `A`, as shown in Figure 3-32 on page 150. The panel shown in Figure 3-34 displays, where you can alter the space allocation.

Normally, the size is set at the same size as your HFS if current utilization is below about 75%. Otherwise, it adds about 10% if your utilization is below 90%. It adds about 20% for utilization above 90%.

Note: For general purpose file systems, it appears that they consume about the same amount of space.

```
----------------------  CHANGE ALLOCATION ATTRIBUTES  ----------------------
Command ===> _

Use the HELP command for usage information on this tool.
Change the allocation attributes for this new zFS file system.
If the file system is already allocated enter Y for preallocated and the
name of the data set as the temp name.  The attributes will not be used.


File system name . . . OMVS.ROGERS.HFS
Save HFS as  . . . . . OMVS.ROGERS.HFS.SAV
Temp name for zFS. . . OMVS.ROGERS.HFS.TMP
Preallocated zFS . . . N

Primary allocation . . 10
Secondary allocation   5
Allocation units . . . CYL      (CYL or TRK)
Data class . . . . . . HFS
Management class . . . HFS
Storage class  . . . . OPENMVS
Volume . . . . . . . . SBOX39
```

Figure 3-34 Panel to alter the allocation attributes

Migrating to a multivolume zFS

If your HFS is multivolume, you must preallocate your zFS for it to have similar attributes and be multivolume. However, do not mount it. Specify `A` for alter allocation. On the panel shown in Figure 3-34, specify `Y` on the Preallocated zFS line.

Options with BPXWH2Z

The first argument can specify option flags. The option flags are preceded by a dash (-) followed immediately by the flags. The following flags are valid:

-v Additional messages are issued at various points in processing.

-c Summary information goes to a file when background is selected. If this is not specified, then summary information goes to the console.

The tool can display many different messages. These messages are documented in the online help panels.

You can preallocate the zFS file system or specify a character substitution string. If you specify a substitution string, the panel will be primed such that the data sets will not be renamed. The substitution string is specified as the first argument on the command line as:

```
/fromstring/tostring/
```

For example, if your data set is OMVS.HFS.WJS and you want your zFS data set to be named OMVS.ZFS.WJS you can specify a command argument as follows:

```
/hfs/zfs/ omvs.hfs.wjs
```

Attention: All strings matching the from string in the data set name will be replaced. You must update all mount scripts and policies according to new names as needed when the file system name contains HFS and the new name contains zFS. The tool does not modify or search for any type of mount scripts or policies.

Migration considerations

Use the **End** or the **Cancel** command to exit from this tool. No changes or allocations have been made.

In general, if you are using HFS as your /tmp, it would be best to start with a clean zFS.

Also, you should not migrate your /dev file system. As mentioned, start with a clean zFS. If you are migrating your root file system, do this in foreground. As part of the migration all file systems will be unmounted, including /tmp and /dev, which will keep you from seeing the status of the migration if run in background.

Restrictions using BPXWH2Z

- BPXWH2Z can only be executed on systems running at least a z/OS V1R7 system because of the dependence on the version of the **pax** utility introduced in z/OS V1R7.
- zFS file systems that will span volumes must be preallocated prior to invoking the tool. This restriction has been mainly removed (see “Enhancements to BPXWH2Z” on page 155).
- The tool cannot be used in a multilevel security environment.
- Specific file systems cannot or should not be handled by the tool (for example, if the /tmp structure is included, or if it is a version root or sysplex root file system).

3.5.2 Enhancements of the pax utility

The **pax** command was selected as the utility for data movement for the migration tool because it is well established. Enhancements were made for the copy mode in z/OS V1R7 to support the HFS-to-zFS migration tool. It now has the following abilities:

- Keep sparse files sparse during a copy by default. *Sparse files* do not use real disk storage for pages of file data that contain only zeros. Copying files as sparse saves on disk space.
- Continue after encountering a read error on the source file system. In addition, **pax** will print an error message and return a non-zero value after the command ends.

Note: The ability to skip read errors may allow clients to salvage some files from corrupted file systems.

- Create empty directories within the target directory tree for each active mount point encountered within the source directory tree.
- Preserve all file attributes from the source to the target of the copy, including user-requested audit attributes and auditor-requested audit attributes (see the **chaudit** command).

Note: The enhancements added to **pax** provide a supported utility for migration tool data movement and some enhanced **pax** functionality.

All functions and options from previous levels of **pax** will still be valid at this level. All automated scripts that use **pax** at the previous level will still work without errors. Any archives created with previous levels of **pax** can be extracted by the new version without problems.

Restriction: If scripts or commands that use these new options are run on an older level of **pax**, then **pax** will fail with a usage message.

If you want to copy data from an HFS source data set mounted at /tmp/hfssource into a new zFS file system mounted at /tmp/zfstarget with the same options that the conversion utility BPXWH2Z uses, you can do it as shown in Figure 3-35.

```
#> cd /tmp/hfssource && /bin/pax -rw -peW -XCM . /tmp/zfstarget
```

Figure 3-35 Exploiting pax enhancements in z/OS V1R7 when copying file system data

For detailed information about **pax** options, see *z/OS UNIX System Services Command Reference,* SA22-7802.

3.5.3 Converting multiple file systems

This section describes the steps we used to perform a conversion of two similar named HFS data sets to zFS aggregates.

The two HFS data sets are named OMVS.BPXWH2Z.HFS1 and OMVS.BPXWH2Z.HFS2. They are mounted at /u/hering/hfs1 and /u/hering/hfs2, and are to be migrated to zFS.

The conversion processing was started from ISPF, as shown in Figure 3-36.

```
tso bpxwh2z OMVS.BPXWH2Z.H*
Not running UID=0 - attempting to set UID=0
***
```

Figure 3-36 Starting migration utility BPXWH2Z

Because the user ID running the tool was not a permanent superuser, BPXWH2Z tried to switch to superuser mode. This was successful because the user had read access to FACILITY profile BPX.SUPERUSER.

The argument **omvs.bpxwh2z.h*** is a generic HFS data set selection criteria for possible HFS data sets.

Note: Quotes are not needed around the name specifications, which are always expected to be related to complete MVS data set names.

The panel shown in Figure 3-37 was displayed next.

```
-------------------------  CLASS AND VOLUME DEFAULTS ----------------------

 The volume or SMS classes for zFS allocations will default to the
 same names as the current HFS allocation.  You can change these
 individually for each new allocation.  If you want the default for
 each new allocation to be set to specific values other than that of
 the current HFS allocation, enter those values here and Press Enter.

 Default volume  . . . . .: ________
 Default data class  . . .: ________
 Default storage class . .: ________
 Default management class : ________
```

Figure 3-37 Displaying class and volume defaults panel

We could have provided some or all of the possible defaults here, but we simply left the fields empty. After receiving `Skipping non-HFS data set ...` messages, we reached the table of selected HFS data sets shown in Figure 3-38.

```
-------------------------  DATA SET LIST  ----------------- Row 1 to 2 of 2

   Use the HELP command for full usage information on this tool
   Select items with D to delete items from this migration list
   Select items with A to alter allocation parameters for the items
   Enter command FG or BG to begin migration in foreground or UNIX background
 ------------------------------------------------------------------------------
 _ HFS data set ..: OMVS.BPXWH2Z.HFS1                            Utilized: 6%
   Save HFS as  ..: OMVS.BPXWH2Z.HFS1.SAV
   Initial zFS  ..: OMVS.BPXWH2Z.HFS1.TMP                        Allocated: N
   HFS space  Primary  : 1          Secondary: 1          Units ..: CYL
   zFS space  Primary  : 1          Secondary: 1          Units ..: CYL
              Dataclas : HFS        Mgmtclas : HFS        Storclas: OPENMVS
   MOUNTED    Volume   : SBOX1A     Vol count: 1
 ------------------------------------------------------------------------------
 _ HFS data set ..: OMVS.BPXWH2Z.HFS2                            Utilized: 6%
   Save HFS as  ..: OMVS.BPXWH2Z.HFS2.SAV
   Initial zFS  ..: OMVS.BPXWH2Z.HFS2.TMP                        Allocated: N
   HFS space  Primary  : 1          Secondary: 1          Units ..: CYL
   zFS space  Primary  : 1          Secondary: 1          Units ..: CYL
              Dataclas : HFS        Mgmtclas : HFS        Storclas: OPENMVS
   MOUNTED    Volume   : SBOX1E     Vol count: 1
 ****************************** Bottom of data ******************************


 Command ===> bg
```

Figure 3-38 Data set display table for the conversion processing

To start the conversion in the background, we entered command **bg** on the panel.

Tip: Press PF1 (and then Enter until the end of the text is reached) to view a description with usage information for BPXWH2Z. You can also use commands **Right** and **Left** to scroll up and down.

We received the messages shown in Figure 3-39. They indicated where to find actual status and the results of the conversion processing when it was completed.

```
Background status can be found in files
   /tmp/bpxwh2z.HERING.09:40:51.*
Remove these files when you no longer need them
***
```

Figure 3-39 Reference messages for the background conversion processing

The following two files were created in the /tmp directory:

```
bpxwh2z.HERING.09:40:51.err
bpxwh2z.HERING.09:40:51.log
```

During processing the old HFS file systems were remounted as read-only. In the end they were automatically unmounted, renamed to OMVS.BPXWH2Z.HFS1.SAV and OMVS.BPXWH2Z.HFS2.SAV, and replaced by the new compatibility mode zFS file systems, which also got the original HFS names.

Note: You do not receive an indication when processing is complete, so you must check whether it has ended.

When processing was finished, we found that the error file was empty. The contents of the log file are listed in Figure 3-40.

```
creating zFS OMVS.BPXWH2Z.HFS1.TMP
copying OMVS.BPXWH2Z.HFS1 to OMVS.BPXWH2Z.HFS1.TMP Blocks to copy: 11
IDC0531I ENTRY OMVS.BPXWH2Z.HFS1.TMP ALTERED
IDC0531I ENTRY OMVS.BPXWH2Z.HFS1.TMP.DATA ALTERED
mount /u/hering/hfs1 OMVS.BPXWH2Z.HFS1 ZFS 0
creating zFS OMVS.BPXWH2Z.HFS2.TMP
copying OMVS.BPXWH2Z.HFS2 to OMVS.BPXWH2Z.HFS2.TMP Blocks to copy: 11
IDC0531I ENTRY OMVS.BPXWH2Z.HFS2.TMP ALTERED
IDC0531I ENTRY OMVS.BPXWH2Z.HFS2.TMP.DATA ALTERED
mount /u/hering/hfs2 OMVS.BPXWH2Z.HFS2 ZFS 0
```

Figure 3-40 Listing of the log file after completion of the conversion

3.5.4 Enhancements to BPXWH2Z

Several important enhancements have been made to the BPXWH2Z migration tool, which was first released with z/OS V1R7.

APAR OA13154 (SPE 1)

The first set of changes arrived with UNIX System Services APAR OA13154, which is listed in Table 3-1 on page 156.

Table 3-1 APAR OA13154

z/OS release	zFS release	PTF
z/OS V1R7 (720)	370	UA23876

Note these significant new functions:

- A migration complete message with the time stamp is issued to the summary log file when running as background processing with the **-C** option; or to the operator console when running as background processing without the **-C** option; or to the terminal when running as foreground processing.
- A TSO notification is sent to the user when the HFS data set-to-zFS data set migration is complete.
- When a zFS allocation fails, zFS allocation error details are displayed on the terminal when running as foreground processing, or issued to log file when running as background processing.
- The tool allows the user to set the allocations unit, and then the tool verifies it. The allocations unit for zFS must be either CYL or TRK.
- The BPXWH2Z tool now can be run in an ISPF batch job. It only works when the HFS data set is allocated in CYL or TRK, and the zFS data set is not preallocated.

APAR OA18196 (SPE 2)

The second SPE arrived with APAR OA18196, as listed in Table 3-2.

Table 3-2 APAR OA18196

z/OS release	zFS release	PTF
z/OS V1R7 (720)	370	UA32289
z/OS V1R8 (730)	380	UA32290

Following are the most significant changes and new functions of APAR OA18196 (SPE 2):

- If your HFS file system is an SMS-managed multi-volume data set, your zFS will be allocated based upon the allocation attributes of the HFS. You will have an opportunity to modify these attributes via the Change Allocation Attributes panel.
- The tool will ensure that the zFS file system allocation is not less than the minimum allocation supported by the zFS file system.
- A new option **-e** is added to the BPXWH2Z tool to exactly match the HFS file system name. When the new option is used during the invocation of the BPXWH2Z tool, it will match the input HFS file system name as an exact name and will not consider it as a pattern string, which is the default behavior.

3.5.5 Using BPXWH2C in batch mode

This section provides a sample that demonstrates how to use the migration tool BPXWH2Z in batch mode, converting a single HFS data set to zFS. Figure 3-41 on page 157 lists the job used for this task.

```
//ZFSJOB   JOB ,'BPXWH2Z',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=SC70
//* --------------------------------------------------------------------
//* Running BPXWH2Z as an ISPF Batch Job
//* --------------------------------------------------------------------
//TSOBATCH EXEC PGM=IKJEFT01,DYNAMNBR=90
//SYSTSIN  DD DATA,DLM=##
ISPSTART CMD(BPXWH2Z -eVC OMVS.HERING.TESTBAT)
##
//SYSEXEC  DD DSNAME=&SYSUID..REXX.EXEC,DISP=SHR
//         DD DSNAME=SYS1.SBPXEXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=VB
//ISPPROF  DD DSN=&&TMPPROF,DISP=(NEW,PASS,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,1))
//ISPLOG   DD DUMMY
//ISPMLIB  DD DSN=ISP.SISPMENU,DISP=SHR
//         DD DSN=SYS1.SBPXMENU,DISP=SHR
//ISPPLIB  DD DSN=&&TMPPLIB,DISP=(NEW,DELETE),UNIT=SYSALLDA,
//            LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,1))
//ISPSLIB  DD DSN=ISP.SISPSLIB,DISP=SHR
//ISPTLIB  DD DSN=ISP.SISPTENU,DISP=SHR
//* --------------------------------------------------------------------
```

Figure 3-41 JCL to run BPXWH2Z in batch mode

Figure 3-42 shows the messages that are received in the joblog when this JCL is running successfully.

```
READY
ISPSTART CMD(BPXWH2Z -eVC OMVS.HERING.TESTBAT)
Not running UID=0 - attempting to set UID=0
-getting ds list for OMVS.HERING.TESTBAT
Skipping matching patterns of input data set name OMVS.HERING.TESTBAT.OTHER1
Skipping matching patterns of input data set name OMVS.HERING.TESTBAT.OTHER2
-getting ds info for OMVS.HERING.TESTBAT
-getting ds info for OMVS.HERING.TESTBAT.SAV
-getting ds info for OMVS.HERING.TESTBAT.TMP
Migrating OMVS.HERING.TESTBAT   20:22:00
creating zFS OMVS.HERING.TESTBAT.TMP
copying OMVS.HERING.TESTBAT to OMVS.HERING.TESTBAT.TMP Blocks to copy: 79
- cd /u/hering/testbat; /bin/pax -rw -peW -XCM . /tmp/bpxwh2z.HERING.20:21:59.zfs.
IDC0531I ENTRY OMVS.HERING.TESTBAT.TMP ALTERED
IDC0531I ENTRY OMVS.HERING.TESTBAT.TMP.DATA ALTERED
mount /u/hering/testbat OMVS.HERING.TESTBAT ZFS 0
Migration complete for OMVS.HERING.TESTBAT    20:22:13
 NULLFILE was preallocated (no free was done).
READY
END
```

Figure 3-42 Messages in the joblog after running BPXWH2Z in batch mode

Important: There is also a JCL sample named ISPBTCH in SYS1.SAMPLIB to invoke BPXWH2Z as an ISPF batch job. Be sure to read the Notes section before you run the job.

3.6 Alternate HFS-to-zFS migration tool

Some clients who considered migrating HFS data sets to zFS aggregates shared their ideas regarding how to perform such migrations. Even after IBM provided BPXWH2Z, many additional ideas were added to it as new function.

Eventually a separate tool known as MIGRTOOL was created and released at ITSO Poughkeepsie to implement many of the requests from clients. Information about this tool is available in the IBM Redpaper™ publication *HFS to zFS Migration Tool*, REDP-4328. The paper is available at the following URL:

http://www.redbooks.ibm.com/redpieces/abstracts/redp4328.html

3.6.1 Implementation and references

The main focus of MIGRTOOL is to divide an HFS-to-zFS migration into two parts:

- The migration definition, exploiting all the ISPF means in a TSO/ISPF foreground
- Running the migration processing in a TSO batch job

The description section of MIGRTOOL provides installation instructions that explain how to get it to your z/OS. Scenario 1, shown in Figure 3-43, illustrates how to get the data directly to your z/OS system.

```
ftp www.redbooks.ibm.com
User: anonymous
Password: my.email@xx.com
cd redbooks/REDP4328/
lcd 'myuser'
locsite blk=3120 lrecl=80 recfm=fb
binary
get zfs.migrtool.unload.bin zfs.migrtool.unload
quit
```

Figure 3-43 Scenario 1 - getting the data directly to your z/OS system

A second scenario describes how to get the file to your workstation first, before transferring it to z/OS. Refer to REDP-4328 for more information about this topic. This paper also details how to perform the customization task and how to use the tool, and provides several examples.

3.6.2 MIGRTOOL functions and choices

The migration job provides a clear and simple way of starting migration processing with many functions and options. It can be run independent of the definition step at any time later.

The tool has the following supported functions:

- Support for using the standard **pax** utility of the system running the migration
- Allows migration processing in z/OS releases previous to z/OS V1R7
- Automatically detects whether the version of **pax** is at a z/OS V1R7+ level
- Automatically adds all missing mount points if the version of **pax** is prior to R7
- Support for using copytree for copy processing

- Provides an extra migration utility called copymigr that performs the same tasks as **pax** R7 for releases prior to R7

MIGRTOOL allows several intermediate interruption possibilities for stopping migration and continuing processing in a subsequent run of the batch job. You can:

- Force stopping after a formal syntax check of conversion statements
- Force stopping when HFS, and if the existing zFS is mounted
- Force stopping after formatting the zFS aggregate
- Specify filesystypes blocking replacement of a mounted HFS by the zFS file system
- If desired, run copy processing only if the target zFS structure is empty

3.7 Replacing or migrating the sysplex root file system

All the migration tools previously described require that all UNIX work relying on file system structure availability in the sysplex must be stopped before replacing the current sysplex root by a new one. This is a significant restriction that inhibits installations from performing a full migration to the zFS file system. In the following sections we address that restriction.

3.7.1 The sysplex root

The sysplex root is used by the system to redirect addressing to other directories. It is very small and should be mounted read-only. The sysplex root contains directories and symbolic links that allow redirection of directories, as shown in Figure 3-44. Only one sysplex root file system is used for all systems participating in a shared file system.

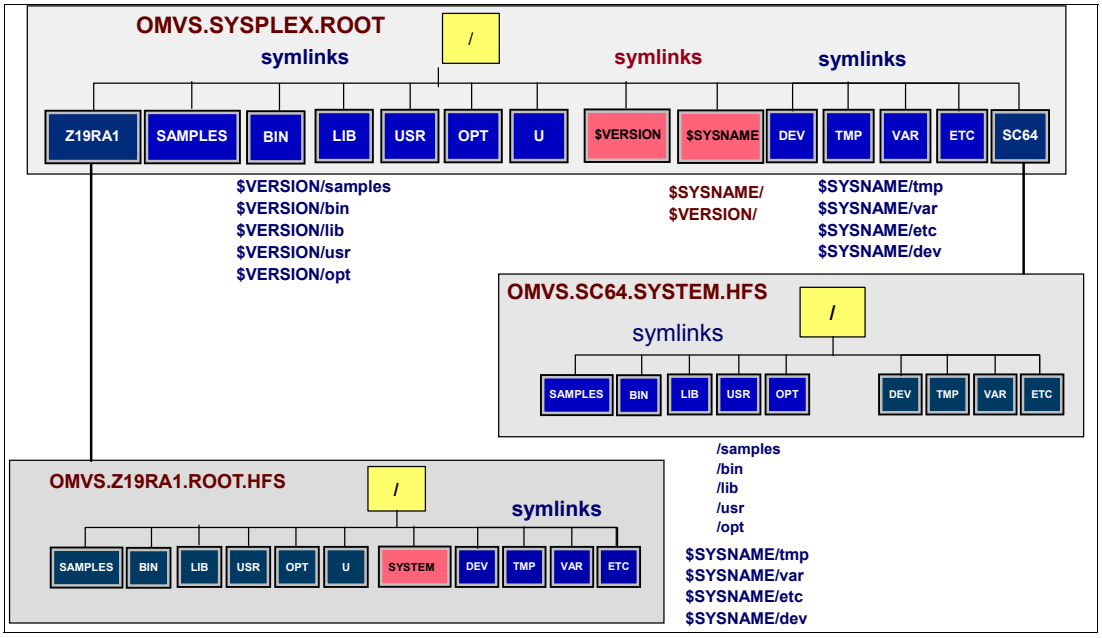


Figure 3-44 z/OS UNIX files systems in a sysplex

The sysplex root is a file system that is used as the sysplex-wide root. This file system can be initially mounted read-write and designated AUTOMOVE. Then you should switch the mode mode to read-only for normal production mode.

No files or code should reside in the sysplex root file system. It consists of directories and symbolic links only. Therefore, the size of the data set representing the sysplex root is very small.

In general, the contents of the sysplex root should only change when you need a new version root or system-specific root file system directory for your shared file system configuration. When a system is IPLed (initialized), the mount processing for the sysplex root file system will include defining the appropriate $SYSNAME or $VERSION directory in the sysplex root file system if the sysplex root is mounted as read/write at that time. These new directories can also be defined longer before IPLing a system that needs them.

3.7.2 Sysplex root replacement

Previously, migrating a sysplex root file system in a UNIX System Services sysplex file system sharing environment from HFS to zFS required that all file systems be unmounted before the replacement could be done.

In z/OS V1R10, however, the replacement of the sysplex root file system can be initiated with an operator command on any system in the UNIX System Services sharing environment. The benefit is that the sysplex root file system can be dynamically replaced while it is in use, without disrupting active workloads.

Using this dynamic root replacement, it is possible to do the following:

- You can change the sysplex root file system to the same or a different PFS, either HFS or zFS.
- Installations can perform a migration of the sysplex root file system to the zFS file system.
- The sysplex root file system can be migrated without unmounting all file systems.

The current sysplex root file system and the new sysplex root file system must be either HFS or zFS in any combination. In addition to being useful for HFS-to-zFS migration, this dynamic root replacement is also useful for moving the sysplex root generally.

For example, when an installation obtains a new storage subsystem and moves the data from the old storage subsystem to the new one and discontinues use of the old storage subsystem, one of the data sets to be moved might be the sysplex root. Being able to perform this move nondisruptively is a benefit of this new function.

New command to migrate the sysplex root

The replacement of the sysplex root file system is initiated via the following operator command on any system in the shared file system configuration:

```
F OMVS,NEWROOT=new.root.file.system.name,COND=<Yes|No>
```

The following values are accepted:

COND=Yes Proceed conditionally if any active usage was found in the current sysplex root file system (this is the default)

COND=No Proceed unconditionally if any active usage was found in the current sysplex root file system

The command can be issued on any system in the UNIX System Services file system sharing environment. It is processed on the system that is the owner of the current sysplex root file system.

Mount parameters are preserved for the same PFS file system type, but will be dropped for different PFS types on the current and new sysplex root file systems.

Using COND=Yes

When the **COND=Yes** parameter is used and if activity in the current sysplex root file system is found, the following messages are displayed:

```
BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILE SYSTEM:
Path Name: pathname INODE: InodeNumber
BPXF244E F OMVS,NEWROOT COMMAND FAILED. RETURN CODE=00000072, REASON
CODE=124F0626
```

If the active path entry is located in a subdirectory of the sysplex root file system and not at the root level, then only the plain pathname without the directory structure is displayed. In addition, the inode number is provided to identify the entry. Otherwise, the inode number is not given because it is not needed.

If sysplex root file system resources are currently being used by active workloads, wait until the current active workloads are completed, or cancel the active workloads and then reissue the command.

Using COND=No

Command **F OMVS,NEWROOT=new.root.file.system.name,COND=No** can be used to proceed unconditionally even if activities are found in the current sysplex root file system. When the **COND=No** parameter is used, the active connections that will be broken on replacement of the sysplex root file system include the following:

- Open files in the root
- Open directories in the root
- Current working directories (CWDs) in the root, except the root CWD

Old connections in the old sysplex root file system may get EIO errors.

After replacing the sysplex root

Remember to update the ROOT statement in the BPXPRMxx parmlib member that is currently in use.

- The root CWD (/) is updated on all systems in the sysplex to point to the new sysplex root file system.
- New open requests go to the new sysplex root file system.
- The current sysplex root for the root directory is replaced for all processes in all systems.
- The current working directory for the sysplex root is replaced for any active processes using it.

New console messages

Following are sysplex root replacement messages indicating failure or success of the operator command:

```
BPXF243E F OMVS,NEWROOT COMMAND HAS BEEN TERMINATED DUE TO THE FOLLOWING
REASON(S):     text

BPXF244E F OMVS,NEWROOT COMMAND FAILED. RETURN CODE=retcode REASON CODE=rsncode

BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILE SYSTEM:
```

```
Path Name: pathname INODE: InodeNumber

BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED
SUCCESSFULLY.

BPXF247I SYSPLEX ROOT MOUNT PARMS ARE DROPPED ON REPLACEMENT.
```

3.7.3 Restrictions for replacement

Although a shared file system configuration is required, the sysplex can be a single system. When the sysplex root is mounted read-only, there are no function-shipping clients as long as physical paths to the DASD are available to all systems in the sysplex. Directories, data, files, and links are not copied from one file system to another.

The following constraints must be met for dynamic replacement of the sysplex root file system:

- The system must be in a shared file system configuration.
- The sysplex root file system must be mounted read only.
- There must be no function shipping clients on the sysplex root file system.
- The current and the replacement sysplex root file system physical file system (PFS) must be active on all systems in the configuration.
- The user allocates and sets up the new sysplex root file system containing all active mountpoints and symlinks similar to the current system.
- The UID, GID, and permission bits for the root directory of the new sysplex root file system must be the same as the current.
- The current sysplex root file system, and any directories on it, cannot be exported by the DFS or SMB server.
- The new sysplex root file system cannot be DFHSM-migrated or mounted or in use.
- The current and new sysplex root must be either HFS or zFS in any combination.
- All systems in the shared file system environment must be at the minimum system level of z/OS V1R10.
- If the SECLABEL class is active and the MLFSOBJ option is active, the security label of the new and current sysplex root file system must match.

This support does not allow a dummy sysplex root for the current sysplex root. A file system must be mounted as the current root.

Migration and coexistence considerations

The minimum system level for all systems in the shared file system configuration is z/OS V1R10 to use this function. Systems at a lower release level may join the sysplex after the sysplex root replacement is completed.

3.7.4 Sample processing

This section describes the steps we performed to dynamically migrate the sysplex root file system from HFS to zFS, or simply to replace the current version.

Preparing for dynamically migrating the sysplex root

The following rules need to be taken into account or followed:

- All systems in the sysplex must be, at minimum, at level z/OSV1R10.
  - This is not the case in the example shown in Figure 3-48 on page 165.
- The new zFS sysplex root file system needs to be allocated and set up.
  - You can simply create the new version root and use the copytree utility, because the structure of the sysplex root should be very simple and small.
  - We provide another example using the alternate HFS-to-zFS migration tool named MIGRTOOL (which is described in the IBM Redpaper REDP-4328).
- The new zFS sysplex root file system cannot be HSM-migrated, mounted, or in use.
- The new zFS sysplex root file system must contain all active mountpoints and symlinks similar to the current HFS sysplex root.
- The UID, GID, and the permission bits of the root directory in the new sysplex root must be the same as in the current root.
- All of the other restrictions mentioned in “Restrictions for replacement” on page 162 must be met.

Allocating and setting up the new sysplex root file system

If you are not changing the type of the new sysplex root file system, then using the ADRDSSU utility is probably the best way to create the new sysplex root. This assures that the internal structure of the new sysplex root is exactly the same as the old one.

You can use:

```
ADDRSSU COPY
```

Or you can use:

`ADDRDSSU DUMP` followed by `ADRDSSU RESTORE`

Figure 3-45 on page 164 illustrates how to migrate from HFS to zFS by using MIGRTOOL. Use the **`DEFMIGR`** command entered from TSO, and then change the initial data presented as shown in the figure.

```
EDIT       SYS08129.T141324.RA000.HERING.R0300614          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 ZFS_MIGRATE_DEFINE_DSN=HERING.ZFS.MIGRATE.DEFINE
000003 ZFS_MIGRATE_DEFINE_MBR=sysroot
000004 ZFS_MIGRATE_DEFINE_DSP=replace
000005 # ------------------------------------------------------------------- #
000006
000007 # ------------------------------------------------------------------- #
000008 ZFS_DEF_DATACLASS=
000009 ZFS_DEF_MANAGEMENTCLASS=
000010 ZFS_DEF_STORAGECLASS=
000011 # ------------------------------------------------------------------- #
000012
000013 # ------------------------------------------------------------------- #
000014 HFS_DATA_SETS_TO_MIGRATE=WTSCPLX2.SYSPLEX.ROOT
000015 ZFS_AGGRNAME_CHANGE_CMD=
000016 # ------------------------------------------------------------------- #
```

Figure 3-45 Defining MIGRTOOL migration statements with DEFMIGR, part 1

Press PF3 to save the changes. The suggested migration statements are displayed for review and changes. Figure 3-46 shows the changes involved in specifying the new sysplex root.

```
EDIT       HERING.ZFS.MIGRATE.DEFINE(SYSROOT) - 01.00      Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 # MIGRATE CONTROL DEFINITIONS, CREATED 2008-05-08 14:27:35             #
000003 # ------------------------------------------------------------------- #
000004
000005 # ------------------------------------------------------------------- #
000006 HFS_NAME=                      WTSCPLX2.SYSPLEX.ROOT
000007 # ------------------------------------------------------------------- #
000008 #HFS_#_VOLUMES=                1
000009 #HFS_DEVICE_TYPE=              3390
000010 #HFS_ALLOC_UNIT=               CYLINDER
000011 #HFS_ALLOC_SPACE=              1 0
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  7 Line(s) not Displayed
000019 ZFS_NAME=                      WTSCPLX2.SYSPLEX.ROOT.NEW
000020 #ZFS_#_VOLUMES=                1
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 Line(s) not Displayed
000024 ZFS_ALLOC_NUM_CAND_VOLUMES=    0
000025 ZFS_ALLOC_NUM_SEC_ALLOCS=      0
000026 ZFS_DATACLASS=
000027 ZFS_MGMNTCLASS=
000028 ZFS_STORCLASS=
000029 ZFS_REPLACES_HFS=              N
000030
```

Figure 3-46 Defining MIGRTOOL migration statements with DEFMIGR, part 2

Next, use the MIGRDATA JCL to create the zFS aggregate and copy the data into the new sysplex root using **copytree**, as shown in Figure 3-47 on page 165.

```
EDIT       HERING.ZFS.JOB.CNTL(MIGRDATA) - 01.41          Columns 00001 00072
000004 //* ---------------------------------------------------------------
000005 //* Migrate HFS data sets to zFS compat mode aggregates
000006 //* Property of IBM  (C) Copyright IBM Corp. 2002-2007
000007 //* ---------------------------------------------------------------
000008 //  SET MGRTOOL=COPYTREE                 <=== Copy utility to be used
000009 //*             COPYMIGR: Use copy tool provided with migration tool
000010 //*             COPYPAX : Use accessible (std) pax version for copying
000011 //*             COPYTREE: Use accessible (std) copytree for copying
000012 //  SET VERBOSE=N                <=== Y or N, list all objects copied
000013 //*               VERBOSE is used only if COPYMIGR or COPYPAX are set.
000014 //  SET DEFMIGR=HERING.ZFS.MIGRATE.DEFINE(SYSROOT)   <=== MIGRATE DEFs
000015 //* ---------------------------------------------------------------
000016 //  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
000017 //  SET STEPLIB=HERING.ZFS.MIGRTOOL.LOADLIB      <=== STEPLIB library
000018 //* ---------------------------------------------------------------
000019 //COPYMIGR EXEC PGM=IKJEFT01,
000020 //  PARM='COPYMIGR &MGRTOOL. &VERBOSE. &OVERWRT.'
000021 //STEPLIB  DD DSNAME=&STEPLIB.,DISP=SHR
000022 //*        DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
000023 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000024 //STDENV   DD DATA,DLM=##
000025 # ---------------------------------------------------------------
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 18 Line(s) not Displayed
```

Figure 3-47 Using the MIGRTOOL JCL to copy the data using copytree

Replacing the sysplex root file system

If all requirements are met, you can issue the command shown in Figure 3-47 from an operator's console.

To force a situation that blocks switching the sysplex root, we created a file named testfile in the root directory before copying the contents of the file system and opened the file for reading.

We also started with two systems not running at level z/OS V1R10, as shown in Figure 3-48.

```
F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEWX,COND=YES
BPXF243E F OMVS,NEWROOT COMMAND HAS BEEN TERMINATED DUE TO THE FOLLOWING
REASON(S):
ONE OR MORE SYSTEM IS NOT AT THE REQUIRED LFS VERSION
```

Figure 3-48 A failing F OMVS,NEWROOT command

Then we carefully took down OMVS on the systems that were not running at level z/OS V1R10 by using the **F OMVS,SHUTDOWN** command, as shown in Figure 3-49 on page 166.

```
F OMVS,SHUTDOWN
BPXI055I OMVS SHUTDOWN REQUEST ACCEPTED
D OMVS
BPXO042I 08.07.42 DISPLAY OMVS 038
OMVS     0011 SHUTTING DOWN    0   OMVS=(9B)
...
*BPXI056E OMVS SHUTDOWN REQUEST HAS COMPLETED SUCCESSFULLY
 BPXN002I UNIX SYSTEM SERVICES PARTITION CLEANUP COMPLETE FOR SYSTEM SC64
```

Figure 3-49 Shutting down OMVS on systems running not at level z/OS V1R10

Figure 3-50 shows the **F OMVS,NEWROOT** command issued again. It fails because the file is open. This is not a normal situation because you should not have any files located directly in the sysplex root file system. It could also have been the case where a process had its current activity directly set to a directory in the sysplex root, but the directory was not used as an active mountpoint.

```
F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEW,COND=YES
BPXF245I LIST OF ACTIVITIES IN THE CURRENT SYSPLEX ROOT FILESYSTEM:
PATH NAME: /testfile
BPXF244E F OMVS,NEWROOT COMMAND FAILED.
RETURN CODE = 00000072, REASON CODE = 124F0626
IOEZ00048I Detaching aggregate WTSCPLX2.SYSPLEX.ROOT.NEW
```

Figure 3-50 F OMVS,NEWROOT fails because of activities in the sysplex root

The reason code 124F0626 shown (=JrActivityFound) means that an activity is found on the sysplex root file system. Look for the BPXF245I message explanation and system programmer response for further details.

Finally, we closed the file and issued the **F OMVS,NEWROOT** command again. This time it successfully replaced the sysplex root HFS with the new zFS file system, as shown in Figure 3-51.

```
F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.NEW,COND=YES
BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED
SUCCESSFULLY.
D OMVS,F,N=WTSCPLX2.SYSPLEX.ROOT.NEW
BPXO045I 08.41.04 DISPLAY OMVS 070
OMVS     0011 ACTIVE             OMVS=(1A)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
ZFS             1 ACTIVE                      READ  05/08/2008  L=210
  NAME=WTSCPLX2.SYSPLEX.ROOT.NEW                    08.29.50    Q=0
  PATH=/
  AGGREGATE NAME=WTSCPLX2.SYSPLEX.ROOT.NEW
  OWNER=SC70     AUTOMOVE=Y CLIENT=N
```

Figure 3-51 F OMVS,NEWROOT showing the message COMPLETED SUCCESSFULLY

Steps after migrating the sysplex root file system

Next, the BPXPRMxx parmlib member needed to be updated with the new sysplex root file system information.

We moved the sysplex root again twice to end up with the original name of the sysplex root name, as shown in Figure 3-52.

```
tso alter 'wtscplx2.sysplex.root' newname('wtscplx2.sysplex.root.old')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT ALTERED

 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT.OLD,COND=YES
...
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED SUCCESSFULLY.
 IOEZ00048I Detaching aggregate WTSCPLX2.SYSPLEX.ROOT.NEW

tso alter 'wtscplx2.sysplex.root.new' newname('wtscplx2.sysplex.root')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT.NEW ALTERED
tso alter 'wtscplx2.sysplex.root.new.data' newname('wtscplx2.sysplex.root.data')
IDC0531I ENTRY WTSCPLX2.SYSPLEX.ROOT.NEW.DATA ALTERED

 F OMVS,NEWROOT=WTSCPLX2.SYSPLEX.ROOT,COND=YES
 IEF196I IEF285I   WTSCPLX2.SYSPLEX.ROOT.OLD                    KEPT
...
 BPXF246I THE SYSPLEX ROOT FILE SYSTEM MIGRATION PROCESSING COMPLETED SUCCESSFULLY.
```

Figure 3-52 Moving the sysplex root two times more to restore the original name

As a result, the update in the BPXPRMxx member was not needed.

Note: Specifying type HFS in the BPXPRMxx parmlib member on the ROOT statement mounts a zFS file system. A correction to specify zFS instead can be done later.

4

Backup and recovery

This chapter describes the considerations for the backup and recovery of zFS file systems.

It discusses the following topics:

- Backup file system or clone
- Creating clones
- Aggregate recovery
- Backing up zFS file systems
- Problem determination

4.1 Backup file system

You can make a clone of a zFS file system. The zFS file system that is the source file system for the clone is referred to as the read-write file system. The zFS file system that is the result of the clone operation is called the backup file system. The backup file system is a read-only file system and can only be mounted as read-only. You can create a backup file system for every read-write file system that exists.

Attention: In 1.11, “zFS statement of direction” on page 9 it is mentioned that IBM plans to withdraw support for zFS clones with the next release after z/OS V1R13. When this support is withdrawn, you cannot use this function any longer.

4.1.1 Create a file system clone

The aggregate containing the read-write file system to be cloned must be attached. You create a file system clone by using the **zfsadm clone** command, as follows:

```
zfsadm clone -filesystem OMVS.CMP01.ZFS
```

Look for the following “successful clone” message:

```
IOEZ00225I File system OMVS.CMP01.ZFS successfully cloned.
```

When a file system is cloned, a copy of the file system is created in the same aggregate; space must be available in the aggregate for the clone to be successful. 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, as shown in Figure 4-1. This means that you need to limit the length of a file system name to 40 characters if you want to clone it.

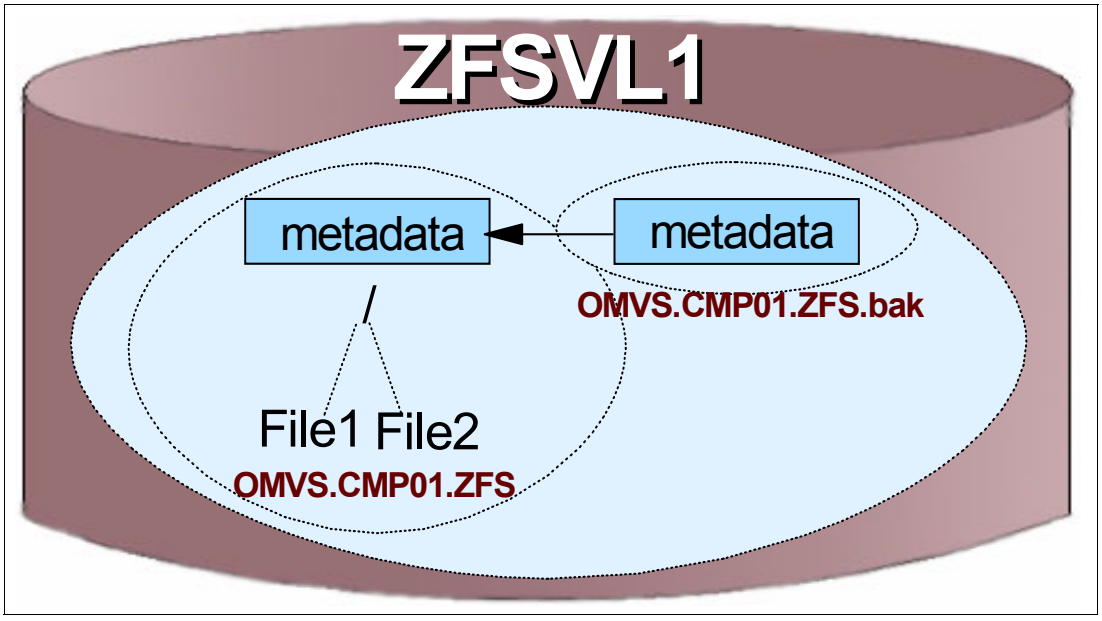


Figure 4-1 zFS file system clone

Mount the clone

After you create the clone, you must mount it before anyone can use it. You can use the TSO/E MOUNT command for the backup file system (clone), as follows:

```
MOUNT FILESYSTEM('''OMVS.CMP01.ZFS.bak''') MOUNTPOINT('/u/zfs/c01cl')
TYPE(READ)                NOAUTOMOVE
```

To use an OMVS **mount** command instead, enter:

```
/usr/sbin/mount -t ZFS -r -a no -f OMVS.CMP01.ZFS.bak /u/zfs/c01cl
```

Note: The backup file system or clone 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.

The backup file or clone is mounted as shown in Figure 4-2.

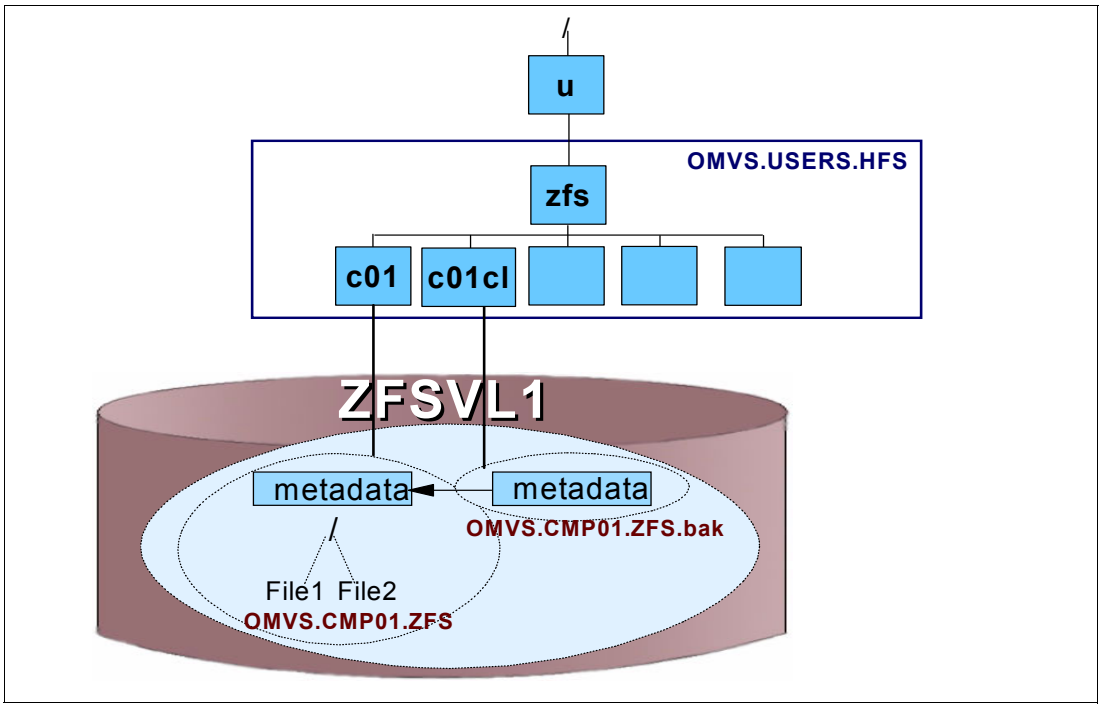


Figure 4-2 Mount the backup file or clone

4.1.2 Using the clone

The clone uses a 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 creates the backup file system, when the read-write file system user data is updated, zFS does the following:

- Makes sure that new physical blocks are allocated to hold the updates
- Maintains the backup file system's data pointers to the original data
- Keeps the backup file system as an exact copy (at the point in time when the clone was made) when updates to the read-write file system are made

If users accidentally erase a file from the read-write file system, they can simply copy the file from the clone into the read-write file system to restore the file to the time the backup was created.

4.1.3 Updating the clone

The read-write file system can be cloned again (reclone). If the clone file system already exists, it is replaced during this clone operation. 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, as follows:

```
zfsadm clonesys -aggregate OMVS.MUL02.ZFS
```

This command makes a clone of the four file systems created in the aggregate and issues the messages shown in Figure 4-3.

```
ROGERS @ SC43:/>zfsadm clonesys -aggregate OMVS.MUL02.ZFS
IOEZ00219I Clonesys starting for aggregate OMVS.MUL02.ZFS, prefix * all
filesystems *
IOEZ00225I File system OMVS.M01.ZFS successfully cloned.
IOEZ00225I File system OMVS.M02.ZFS successfully cloned.
IOEZ00225I File system OMVS.m03.ZFS successfully cloned.
IOEZ00225I File system OMVS.M04.ZFS successfully cloned.
IOEZ00216I Clone ending for aggregate OMVS.MUL02.ZFS (Total: 4, Failed: 0,
Time.340)
```

Figure 4-3 A clone of all the file systems in a single aggregate

Using the **zfsadm lsfs** command, Figure 4-4 shows the aggregate and the file systems with their clones. Notice that the clones are not mounted.

```
ROGERS @ SC43:/>zfsadm lsfs
.........
IOEZ00129I Total of 8 file systems found for aggregate OMVS.MUL02.ZFS
OMVS.m03.ZFS RW (Mounted R/W)       8 K alloc      9 K quota On-line
OMVS.m03.ZFS.bak BK (Not Mounted)       9 K alloc      9 K quota On-line
OMVS.M01.ZFS RW (Mounted R/W)      24 K alloc     77 K quota On-line
OMVS.M01.ZFS.bak BK (Not Mounted)      77 K alloc     77 K quota Online
OMVS.M02.ZFS RW (Mounted R/W)       8 K alloc      9 K quota On-line
OMVS.M02.ZFS.bak BK (Not Mounted)       9 K alloc      9 K quota On-line
OMVS.M04.ZFS RW (Mounted R/W)       8 K alloc      9 K quota On-line
OMVS.M04.ZFS.bak BK (Not Mounted)       9 K alloc      9 K quota On-line
.........
```

Figure 4-4 Display of file systems and their clones

4.2 Aggregate recovery

zFS provides an aggregate recovery command, **ioeagslv**, that attempts to recover a zFS aggregate and repairs problems it detects in the structure of the aggregate. This command is not normally needed when the following failure occurs.

If the system fails, the aggregate log is replayed automatically as follows:

- The next time the aggregate is attached
- 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.

The **ioeagslv** command is similar to the z/OS UNIX **fsck** command. However, it does not verify or repair the format of user data contained in the files contained in the aggregate.

The command should be used to verify or repair the structure of the aggregate if any of the following occurs:

- If problems are detected in the basic structure of the aggregate
- If the log mechanism is damaged
- If the storage medium of the aggregate has errors

4.2.1 The ioeagslv command

The **ioeagslv** command invokes a utility called the Salvager. If changes are made, the Salvager displays the path names of the files affected by the modifications, when the path names can be determined. The owners of the files can then verify the contents of the files, and the files can be restored from backups if necessary.

The type of recovery that is attempted by the Salvager utility depends on the following options, which indicate the type of operations to be performed when specified on the command:

-recoveryonly Directs the Salvager to recover the specified aggregate. The Salvager replays the log of metadata changes that resides on the aggregate.

-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.

-salvageonly Directs the Salvager to salvage the specified aggregate. The Salvager attempts to repair any inconsistencies it finds on the aggregate.

Security access required

The issuing user may require RACF authority, depending on which option is specified with the command:

- With the **-verifyonly** option included, the issuer needs UPDATE authority for the specified VSAM LDS (aggregate), although nothing is changed in this situation.
- If either the **-recoveronly** or **-salvageonly** option is included, or if all three options are omitted, the issuer must have ALTER authority for the specified VSAM LDS.

In addition, the user must be in superuser mode by either:

- Being a permanent superuser with a UID of 0
- Having READ authority to the SUPERUSER.FILESYS.PFSCTL profile in the UNIXPRIV class. See “UNIXPRIV profiles” on page 18 for additional details.

Using the utility

The utility can be used by issuing the command from the OMVS shell or by submitting a JCL job as shown in Figure 4-5 on page 174.

```
//ZFSJOB   JOB ,'ZFS Salvager',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//* Recover an Aggregate using the Salvager
//FORMAT   EXEC   PGM=IOEAGSLV,REGION=0M,
//         PARM=('-aggregate OMVS.MUL01.ZFS' -recoveronly)
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD    <--LOADLIB FOR ZFS
//SYSPRINT DD   SYSOUT=*
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CEEDUMP  DD   SYSOUT=*
//*
```

Figure 4-5 Job to execute the Salvager utility

Note: The job in Figure 4-5 will not execute if you only have security access with BPX.SUPERUSER. There is no way to check for this authority in this situation.

The format of the shell command is:

```
ioeagslv -aggregate OMVS.MUL01.ZFS' -recoveronly
```

4.2.2 Using the Salvager utility

The Salvager utility scans an aggregate and reports any inconsistencies found. The aggregate can be scanned in the following ways:

- Verified
- Recovered by scanning the log
- Salvaged or repaired

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.

Using the Salvager utility from the UNIX shell

To use the Salvager utility as a BPX.SUPERUSER, we defined an external link to IOEAGSLV in the UNIX file structure:

```
#> ln -e IOEAGSLV /u/hering/bin/zfssalvage
```

A Salvager utility job

The JCL member shown in Example C-19 on page 461 can be used to run the utility. This utility job executes the following UNIX shell commands:

```
#> zfssalvage -aggregate OMVS.CMP01.ZFS -verbose
#> zfssalvage -aggregate OMVS.CMP01.ZFS -salvageonly
#> zfssalvage -aggregate OMVS.CMP01.ZFS -verifyonly
```

The JCL output for this job is shown in Example C-20 on page 462.

4.3 Backing up zFS aggregates

You can back up zFS file systems by performing the following actions:

- A zFS aggregate can be backed up and restored using IDCAMS REPRO.

  The aggregate must be quiesced before the backup.
- A zFS aggregate can be backed up and restored using DFSMSdss.

  The aggregate must be quiesced before the backup.

When the aggregate is backed up, all the file systems in the aggregate are backed up. To allow use of the file systems, after the backup is complete, use the **unquiesce** command or use a batch job with PGM=IOEZADM.

When you create backups with DFSMS utilities HSM or ADRDSSU while the VSAM LDS is marked as a zFS aggregate in the catalog entry, then quiescing the aggregate before and unquiescing it again afterwards is done automatically and you must not do it yourself. Refer to 4.3.4, “Catalog indication for zFS aggregates” on page 179 for more detailed information about this topic.

Note: You may also use a program that logically saves UNIX file system data, which you may already be doing with HFS file systems.

Example C-42 on page 473 shows a sample job that backs up a zFS aggregate using ADRDSSU.

4.3.1 Restoring the backup

zFS aggregates can be restored by using the DFSMSdss logical restore. If the original aggregate is to be kept, simply restore into a new aggregate and rename it.

Compatibility mode aggregate restores

Following the restore, perform these steps:

1. Unmount the original aggregate.
2. Mount the new restored aggregate.

Multifile system aggregates

After restoring the aggregate, perform the following steps to restore the file systems:

1. Unmount the file systems in the aggregate.
2. Detach the aggregate.
3. Attach the restored aggregate.
4. Mount each file system in the restored aggregate.

See Example C-43 on page 474 for a sample job that restores a zFS aggregate that has been backed up.

4.3.2 Using a clone for taking a backup

If you have huge zFS file systems with fairly simple directory and file structures, and thus rather small metadata, there is another choice available for taking backups with zFS. You can create a read-only clone, mount it read-only, and then take a backup of the clone while the original file system may be used and changed. A sequence of creating a clone and mounting it is shown in Figure 4-6.

Note: In 1.11, “zFS statement of direction” on page 9 it is mentioned that IBM plans to withdraw support for zFS clones with the next release after z/OS V1R13. When this support is withdrawn, you cannot use this function any longer.

```
#> zfsadm clone -filesystem OMVS.M02A.ZFS
IOEZ00225I File system OMVS.M02A.ZFS successfully cloned.
#> /usr/sbin/mount -f OMVS.M02A.ZFS.bak -t ZFS /u/zfs/m02acl
FOMF0504I mount error: 8D EF09605B
EROFS: The specified file system is read only
#> /usr/sbin/mount -rf OMVS.M02A.ZFS.bak -t ZFS /u/zfs/m02acl
#> zfsadm clone -filesystem OMVS.M02A.ZFS
IOEZ00224E The clone for file system OMVS.M02.ZFS is mounted. Clone
terminating.
#> /usr/sbin/unmount /u/zfs/m02acl
#> zfsadm clone -filesystem OMVS.M02A.ZFS
IOEZ00225I File system OMVS.M02.ZFS successfully cloned.
#> /usr/sbin/mount -rf OMVS.M02A.ZFS.bak -t ZFS /u/zfs/m02acl
```

Figure 4-6 Creating a clone and mounting it for backup

Figure 4-7 shows how to create a backup **pax** archive in a preallocated MVS data set.

```
#> cd /u/zfs/m02acl && pax -o saveext -pe -wzXf "//'omvs.m02a.zfs.bkup'" .
#>
```

Figure 4-7 Creating a pax archive backup file

To list the contents of the archive file just created, you can use the following command:

```
pax -Ef "//'omvs.m02a.zfs.bkup'"
```

To restore a single file from the archive, use the following command:

```
pax -pe -rf "//'omvs.m02a.zfs.bkup'" single_file_name
```

To restore the complete backup data starting at the current working directory, use the following command:

```
pax -pe -rf "//'omvs.m02a.zfs.bkup'" .
```

4.3.3 Using IDCAMS REPRO

You can use IDCAMS REPRO to get easily transportable backup unload files of zFS aggregates. These are much more flexible than ADDRDSU unloads (which still have included SMS class settings, for example).

Important: The recommended and supported utility for copying zFS data sets is DFdss DUMP and RESTORE. This is ADRDSSU.

- REPRO is (seen as) not officially supported for manipulating zFS aggregates because it has never been tested and verified by development to work okay in all possible situations.
- REPRO should *not* be used to take backups of zFS aggregates to be used later for necessary restore processing. In this case ADRDSSU should definitely be used and is recommended.
- Using REPRO should be restricted for those situations where it makes sense and the result can be validated at once and before using the new aggregate in production mode.

Figure 4-8 on page 177 shows a JCL sample that demonstrates how to back up a zFS aggregate by using this technique.

```
//* ------------------------------------------------------------------
//  SET  ZFSAGGR=HERING.TEST.ZFS                 <=== zFS aggrname
//  SET ZFSUNLOD=HERING.TEST.ZFS.UNLOAD          <=== Unloaded file
//* ------------------------------------------------------------------
//UNLOAD   EXEC PGM=IDCAMS
//ZFSAGGR  DD DSN=&ZFSAGGR.,DISP=SHR
//ZFSUNLOD DD DSN=&ZFSUNLOD.,DISP=(NEW,CATLG,DELETE),LRECL=4096,
//            BLKSIZE=20480,RECFM=FB,SPACE=(CYL,(10,10))
//SYSIN    DD DATA,DLM=##
  REPRO INFILE(ZFSAGGR) OUTFILE(ZFSUNLOD)
##
//SYSPRINT DD SYSOUT=*
```

Figure 4-8 Creating a backup using IDCAMS REPRO

Important: If the aggregate that you are unloading or copying (shown in Figure 4-11 on page 179) is active in read-write mode, quiesce it before copying the data and unquiesce it after copying the data. This is necessary because REPRO does not automatically do that based on the catalog indication in the catalog. See 4.3.4, “Catalog indication for zFS aggregates” on page 179 for more information about this topic.

Figure 4-9 shows sample JCL for restoring the zFS aggregate from the backup file.

```
//* ------------------------------------------------------------------
//  SET  ZFSAGGR=HERING.TEST3.ZFS                 <=== Target zFS
//  SET ZFSUNLOD=HERING.TEST.ZFS.UNLOAD           <=== Unloaded file
//* ------------------------------------------------------------------
//RESTORE  EXEC PGM=IDCAMS
//ZFSAGGR  DD DSN=&ZFSAGGR.,DISP=OLD
//ZFSUNLOD DD DSN=&ZFSUNLOD.,DISP=SHR
//SYSIN    DD DATA,DLM=##
  REPRO INFILE(ZFSUNLOD) OUTFILE(ZFSAGGR)
##
//SYSPRINT DD SYSOUT=*
```

Figure 4-9 Restoring the zFS data from the backup file

When you are restoring such an unloaded zFS aggregate at a new location to a new aggregate, it is very important to only *define* the new zFS aggregate.This is because IDCAMS REPRO would simply append the new data at the end of the just-formatted aggregate. The result, effectively, would be still a newly formatted aggregate because the other data is not visible for zFS.

Restriction: Keep in mind the following two important rules.

1. You must *not* format the new aggregate before restoring the data from the IDCAMS REPRO backup file, because this would destroy the aggregate format.
2. *Never ever* use the salvager tool for testing whether a zFS is empty because this will remove this status of being empty and REPRO will have a similar problem as after formatting the aggregate. See 4.2, “Aggregate recovery” on page 172 for more information about the salvalger tool IOEAGSLV.

Copying a zFS aggregate using IDCAMS REPRO

Copying a zFS aggregate is even simpler to do. First define the new aggregate as shown in Figure 4-10 on page 178 and do not format it.

```
#> zfsadm define SYS1.TEST.NONSMS.ZFS -volumes VSM7C1 -cylinders 10 10
IOEZ00248E VSAM linear dataset SYS1.TEST.NONSMS.ZFS successfully created.
#>
```

Figure 4-10 Defining a new zFS aggregate

Figure 4-11 on page 179 shows sample JCL to copy an existing zFS to this new defined VSAM linear data set.

```
//* ------------------------------------------------------------------
//  SET SOURCE=OMVSM.TEST.EXTADDR.ZFS                <=== SRC aggrname
//  SET TARGET=SYS1.TEST.NONSMS.ZFS                  <=== TRG aggrname
//* ------------------------------------------------------------------
//COPY     EXEC PGM=IDCAMS
//ZSOURCE  DD DSN=&SOURCE.,DISP=SHR
//ZTARGET  DD DSN=&TARGET.,DISP=OLD
//SYSIN    DD DATA,DLM=##
  REPRO INFILE(ZSOURCE) OUTFILE(ZTARGET)
##
//SYSPRINT DD SYSOUT=*
```

Figure 4-11 Copy a zFS aggregate to a new defined VSAM LDS

Because the zFS aggregate names that are used associate in this case a zFS aggregate defined with a data class having set extended addressability (and being smaller than 4 GB in size), it is copied to a non-SMS-managed VSAM LDS. This could not be done with **ADRDSSU COPY**.

Tip: Using this technique of IDCAMS REPRO is a useful way to migrate a zFS aggregate that is defined without extended addressability, and that is growing close to 4 GB, to a new one that has the right data class set with a data set name type of EXTENDED and extended addressability YES.

New zFS reorganization tool

We created a further utility named zFS reorganization tool (or simply ZFSREORG) for reorganization of a zFS aggregate with options to use **pax**, **copytree** or IDCAMS/TSO REPRO for copying a source zFS to a new zFS.

Tip: This REORG tool can be a further option for using REPRO, especially as it carefully tests a target zFS for being empty in case the zFS exists already when the tool is called to copy a source zFS to the target zFS using REPRO.

Information about this tool is available in the IBM Redpaper publication *zFS Reorganization Tool*, REDP-4769. The paper is available at:

http://www.redbooks.ibm.com/redpieces/abstracts/redp4769.html

The paper describes in detail how to retrieve, install and use the tool.

4.3.4 Catalog indication for zFS aggregates

At the time zFS was introduced, there was no indication available for application- and system-related programs that identified the underlying VSAM linear data sets to be zFS aggregates. This fact needs to be taken into account in many situations.

Most important were the impacts to DSS (ADRDSSU) and HSM because these programs simply acted as though the data sets were of type “normal” VSAM. This caused the problem that in both situations, inconsistent backup copies could most likely be taken when the zFS aggregate was active and used in parallel.

For DSS, the circumvention was to quiesce the aggregate before taking the dump and to unquiesce it again afterwards. In the case of HSM, there was no useful means available that assured that a quiesce and unquiesce procedure was performed.

Example C-42 on page 473 contains a sample job that has a QUIESCE step before and an UNQUIESCE step after doing the backup using ADRDSSU.

Several APARs (and PTFs) and follow-on APARs to correct remaining problems were made available to provide a catalog flag entry that identifies the zFS VSAM linear data set to be a zFS aggregate. Figure 4-12 lists the most important APARs.

```
OW57015  ICF: Catalog support for indication data set is a zFS aggregate
OW57046  zFS: Set bit in catalog to indicate VSAM LDS is a zFS aggregate
OW57017  IDCAMS: New function support for IDCAMS (LISTCAT)
OA02713  HSM: New support for backup of zFS aggregate data sets
OW57141  DSS: New support for copy and dump functions for zFS aggregates
```

Figure 4-12 Catalog support APARs for zFS aggregates

Now this flag allows the programs to recognize zFS aggregates and, for example, perform a quiesce operation before taking a backup and perform an unquiesce afterward. There is no longer a need to quiesce an aggregate before and unquiesce it again after the call to ADDRDSSU to dump a zFS aggregate. Therefore, remove the QUIESCE and the UNQUIESCE steps in your job. See 4.3.5, “Restrictions on quiescing zFS aggregates” on page 181 for further information about this topic.

Having zFS set the bit is supported in z/OS V1R4 (zFS PTF UW95768). All the other DFSMS-related PTFs are for Release 1G0 (z/OS V1R3 and V1R4).

Attention: To be able to set the catalog flag, zFS must have ALTER access for the zFS aggregate. Otherwise, you will receive a RACF ICH408I message and a zFS message. An example is shown in Figure 4-13.

```
ICH408I USER(DFS     ) GROUP(DFSGRP  ) NAME(DFS                 )
  OMVS.SC43.COMPAT01.ZFS CL(DATASET ) VOL(TOTZF1)
  INSUFFICIENT ACCESS AUTHORITY
  FROM OMVS.SC43.** (G)
  ACCESS INTENT(ALTER  )  ACCESS ALLOWED(READ   )
IOEZ00336I OMVS.SC43.COMPAT01.ZFS could not be marked as a zFS
aggregate in the catalog, rc=56 rsn=6
```

Figure 4-13 Error messages if zFS is not authorized to set the catalog flag

As with OMVS, it is useful to run the zFS started task with the RACF TRUSTED attribute. This eases processing because access to any resources needed is provided, and does not need to be given to the user ID in the Started Class profile. This is a further small benefit by itself. Figure 4-14 illustrates how to activate this setting. Note that the setting is not used until the STC is started the next time.

```
RALTER STARTED (ZFS.**) STDATA(TRUSTED(YES))
RACLISTED PROFILES FOR STARTED WILL NOT REFLECT THE UPDATE(S) UNTIL A SETROPTS
REFRESH IS ISSUED.
SETROPTS RACLIST(STARTED) REFRESH
SETROPTS command complete.
```

Figure 4-14 Setting the RACF TRUSTED attribute for the zFS started task

4.3.5 Restrictions on quiescing zFS aggregates

With z/OS V1R6, when quiescing zFS aggregates, there is still the restriction that quiesce must be run on the system that owns the aggregate, that is, where the aggregate is attached. Figure 4-15 shows a situation where an authorized user is running a successful quiesce because the file system is owned by that system.

```
#> zfsadm quiesce -aggregate OMVS.HERING.TEST.ZFS
IOEZ00163I Aggregate OMVS.HERING.TEST.ZFS successfully quiesced
```

Figure 4-15 Successfully quiescing an aggregate

Figure 4-16 shows what happens if the file system is owned by another system.

```
#> sysvar SYSNAME
SC65
#> /usr/sbin/chmount -d SC70 testfs
#> zfsadm quiesce -aggregate OMVS.HERING.TEST.ZFS
IOEZ00164I Could not quiesce aggregate OMVS.HERING.TEST.ZFS because it was not
attached
```

Figure 4-16 Unsuccessfully quiescing an aggregate

Important: This restriction that the **zfsadm** interface and the APIs are not sysplex-aware has been removed in z/OS V1R7. See 5.8, “Sysplex awareness of the zfsadm command interface” on page 234 for details.

In environments where some systems are still not at the z/OS V1R7 level, there is a problem when any of the programs that are now aware of whether a VSAM data set is of type zFS tries to back up a zFS aggregate that is attached on another system. Nevertheless, this is not a real exposure (ending up with inconsistent backup data) because it will be recognized if the quiesce is not successful and then the backup will not be done.

The job shown in Figure 4-17 on page 182 will try to quiesce a zFS aggregate, dump it, and then unquiesce it again.

```
...
//DUMP     EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//OUT      DD DSN=OMVS.HERING.TEST.ZFS.BKUP,
//            DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(10,5),RLSE)
//SYSIN    DD DATA,DLM=##
  DUMP DATASET(INCLUDE(OMVS.HERING.TEST.ZFS)) -
  OUTDD(OUT)
##
```

Figure 4-17 Sample JCL to back up a zFS aggregate

If successful, you receive the following message in the job output:

```
ADR454I ... THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
             CLUSTER NAME   OMVS.HERING.TEST.ZFS
             CATALOG NAME   MCAT.SANDBOX.VSBOX01
             COMPONENT NAME OMVS.HERING.TEST.ZFS.DATA
```

If the aggregate is owned by another system, you receive the following message:

```
ADR980E ... THE BPX1PCT PROGRAM FAILED DURING QUIESCE PROCESSING FOR DATA SET
OMVS.HERING.TEST.ZFS WITH RETURN CODE 00000081 AND REASON CODE EF04623E
```

BPX1PCT is the pfsctl API that is used to send physical file system requests to a PFS. The return code x"81"(ENOENT) means: `No such file, directory, or IPC member exists`, and the zFS reason code x"623E" means that the specified aggregate cannot be found.

Furthermore, if the aggregate is already quiesced at the time ADRDSSU gets control, the backup fails with the following message:

```
ADR980E ... THE BPX1PCT PROGRAM FAILED DURING QUIESCE PROCESSING FOR DATA SET
OMVS.HERING.TEST.ZFS WITH RETURN CODE 00000074 AND REASON CODE EF17623F
```

Return code x"74" (EDEADLK) states: `A resource deadlock is avoided`, and the zFS reason code x"623F" means that the aggregate is already quiesced.

Important: If the catalog support for zFS aggregates is active in your system, you must remove the QUIESE and UNQUIESE steps from the backup jobs that you may have used before.

4.3.6 UNQUIESCE modify command

Previously, if a zFS aggregate was quiesced and the job failed to unquiesce it, you had to use the UNIX System Services **`zfsadm unquiesce`** command to do it. In z/OS V1R7, the operator can unquiesce a zFS aggregate from the operator's console by using the MODIFY ZFS,UNQUIESCE command. This must be done from the system that owns the zFS aggregate.

Let us assume a quiesce situation with systems SC65 and SC70 running z/OS V1R7 and a file system mounted on SC70 and quiesced as shown in Figure 4-18.

```
$> zfsadm aggrinfo -aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS (R/W COMP QUIESCED): 528 K free out of total 11520
$> df -v /u/hering/test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.TEST.ZFS)    1056/22750     4294966910 Available
ZFS, Read/Write, Device:791, ACLS=Y
File System Owner : SC70        Automove=Y      Client=Y
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.TEST.ZFS
```

Figure 4-18 Quiesced zFS aggregate mounted on system SC70

Figure 4-19 on page 183 shows system commands with corresponding output run from a console on system SC65.

```
 F ZFS,UNQUIESCE,OMVS.HERING.TEST.ZFS
 IOEZ00425E UNQUIESCE FAILURE: rc = 129 rsn = 10
 IOEZ00024E zFS kernel: MODIFY command - UNQUIESCE,OMVS.HERING.TEST.ZFS failed.
 SC70 F ZFS,UNQUIESCE,OMVS.HERING.TEST.ZFS
 IOEZ00025I zFS kernel: MODIFY command - UNQUIESCE,OMVS.HERING.TEST.ZFS
completed successfully.
```

Figure 4-19 Unquiesce modify commands run on system SC65

As mentioned previously, the commands demonstrate that you must know on which system the zFS file system is mounted to run the **MODIFY** command on that system.

4.3.7 Using a started task to unquiesce zFS aggregates

An alternative method is to create a small STC as shown in Figure 4-20.

```
//* ------------------------------------------------------------------
//* STC using BPXBATCH to run "zfsadm unquiesce -aggregate aggrname"
//* Property of IBM  (C) Copyright IBM Corp. 2005
//* ------------------------------------------------------------------
//UNQUSCE  PROC AGGRNAME=''
//* ------------------------------------------------------------------
//OMVS     EXEC PGM=BPXBATCH,REGION=0M,
//  PARM='PGM /bin/zfsadm unquiesce -aggregate &AGGRNAME.'
//STDOUT   DD PATH='/dev/console',PATHOPTS=(OWRONLY)
//* ------------------------------------------------------------------
```

Figure 4-20 Started task ZFSUNQSC

Run the RACF commands shown in Figure 4-21 to make it known to the system.

```
RDEFINE STARTED ZFSUNQSC.* STDATA(USER(DFS) GROUP(DFSGRP))
SETROPTS RACLIST(STARTED) REFRESH
```

Figure 4-21 RACF commands to define STC ZFSUNQSC

Assume we have the same situation as shown in Figure 4-18 on page 183. We can run the started task on system SC65, as shown in Figure 4-22.

```
S ZFSUNQSC,AGGRNAME=OMVS.HERING.TEST.ZFS
BPXF024I (DFS) IOEZ00166I Aggregate OMVS.HERING.TEST.ZFS successfully
unquiesced.
```

Figure 4-22 Running STC ZFSUNQSC on SC65 to unquiesce a zFS aggregate

The advantages of the started task over the **UNQUIESCE modify** command are as follows.

- If only systems running at level z/OS V1R7 are involved, you can run the started task on any system without the need to know which system owns the zFS aggregate. (This exploits the fact that the **zfsadm** command is sysplex-aware in z/OS V1R7. The **UNQUIESCE modify** command is not sysplex-aware.)
- If the zFS aggregate is owned by a system at a lower level than z/OS V1R7, the started task can by used successfully instead of the **MODIFY** command. (The **UNQUIESCE modify** command is not supported in a system with a level lower than z/OS V1R7.)

4.4 Abend handling and hang conditions

zFS has specific instructions that check for conditions that must be true at that point in the code. For example, if a routine is called and a pointer to some information is passed to the routine, the routine might check that the pointer is non-zero. These are called *asserts*.

If the asserted condition is true, the code simply continues. However, if the assertion is false, zFS abends with an abend code of 2C3. zFS takes this action to avoid writing bad data into the file system, that is, corrupting the file system. zFS is trying to detect, as early as possible, whether there is an internal logic problem. When this abend occurs, you should provide IBM Service with the dump and any other relevant material so that the original problem can be fixed and so does not occur again.

Also, if an application goes to end of memory (EOM) while executing in the zFS address space, this causes zFS to abend.

In these two cases, zFS causes the zFS file systems to be moved or unmounted and applications to fail.

4.4.1 Conditional asserts

In z/OS V1R6, zFS has modified all the asserts to be conditional asserts. This means that zFS will still abend but it will not go down. Rather, zFS will attempt to isolate the problem to a single aggregate (by disabling it) and allow access to other aggregates to continue. You should still provide IBM Service with the abend material so that the original problem can be fixed.

As mentioned, a conditional assert abend (X'2C3') may result in an aggregate being marked disabled, as shown in Figure 4-23.

```
IOEZ00337E zFS kernel: non-terminating exception 2C3 occurred, reason EA8001B7
abend psw 70C1000 8B4CCD46
IOEZ00422E Aggregate PLEX.JMS.AGGR001.LDS0001 disabled for writing
```

Figure 4-23 Aggregate being marked disabled after conditional assert abend

This will disallow writes to that aggregate. Again, this is an attempt to avoid corrupting the aggregate. To use this aggregate, it must be detached (for compatibility mode aggregates, this means unmounted) and then attached (for compatibility mode aggregates, this means mounted).

For compatibility mode aggregates, you can use the remount and chmount capability that UNIX System Services provides to get the aggregate detached and reattached without affecting any lower-mounted file systems. If you are in a UNIX System Services sysplex sharing environment, you must be running at z/OS V1R5 or on z/OS V1R4 with APAR OA02584.

Figure 4-24 on page 185 shows an example illustrating how to do this using the **chmount** command. To use the necessary **chmount** options, the system level must be at z/OS V1R5 or above.

```
#> zfsadm setquota PLEX.JMS.AGGR004.LDS0004 -size 8000
IOEZ00194E Error setting quota for file system PLEX.JMS.AGGR004.LDS0004, error
code=157 reason code=EF1A6233
#> /usr/sbin/chmount -r /zfsmnt2
#> df -v /zfsmnt2
Mounted on     Filesystem                Avail/Total    Files      Status
/zfsmnt2       (PLEX.JMS.AGGR004.LDS0004) 8774/8798      4294967287 Available
ZFS, Read Only, Device:28, ACLS=Y
File System Owner : DCEIMGVM    Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : PLEX.JMS.AGGR004.LDS0004
#> /usr/sbin/chmount -w /zfsmnt2
#> zfsadm setquota PLEX.JMS.AGGR004.LDS0004 -size 4400
IOEZ00193I Set quota to 4400K bytes for file system PLEX.JMS.AGGR004.LDS0004
```

Figure 4-24 Using chmount for re-enabling a compatibility mode aggregate

Otherwise, if the disabled aggregate is a compatibility mode aggregate, you must unmount the compatibility mode aggregate (and all lower-mounted file systems) and then remount it (and all the lower file systems).

If the disabled aggregate is a multifile system aggregate (regardless of the release of z/OS), you must unmount the file systems in the aggregate (and all the lower file systems of those file systems), detach the aggregate, reattach it, and mount all the file systems in the aggregate that were previously mounted (and all the lower file systems that were previously mounted).

It is possible that the attempt to detach (unmount) the disabled aggregate may fail. If this occurs, you will still be able to use other (non-disabled) aggregates. You will not be able to use the disabled aggregate that could not be detached until zFS is stopped and restarted. Figure 4-25 shows an example of the message displayed in that situation.

```
IOEZ00433E Internal error, aggregate PLEX.JMS.AGGR001.LDS0001 cannot be
detached from zFS
```

Figure 4-25 zFS message IOEZ00433E

4.4.2 Recovery code for End of Memory failures

zFS End of Memory (EOM) support is invoked when a calling application goes to end of memory while executing in zFS. For example, this can happen when a job is forced that did not respond to a cancel.

Previously, an EOM condition while executing in zFS could cause zFS to go down. In a UNIX single system without sharing, in this situation all zFS file systems were unmounted. In a sharing environment at least the auto-movable zFS file systems are not lost, but instead moved to another system.

In z/OS V1R6, for applications that go to end of memory while executing in zFS, zFS will sometimes recover and sometimes will still go down, depending on the particular area in zFS that the task is executing when the EOM occurs. If it is executing in an area where zFS has implemented EOM recovery, zFS will recover.

Note: This was a coding bandwidth issue for z/OS V1R6; nevertheless, zFS still did not handle EOM recovery in all zFS paths.

In z/OS V1R7, zFS will recover in all EOM situations. zFS does not go down and no zFS file systems are unmounted or moved. No file system needs to be remounted.

Note: This support, handling EOM recovery in all situations, is only provided in z/OS V1R7.

An EOM condition in a calling application is recovered automatically. There is nothing that needs to be invoked explicitly.

Important: With this EOM support in z/OS V1R7, IBM has withdrawn the recommendation to not use zFS for a root, sysplex root, or version root structure. In fact, zFS is now the *preferred* file system.

4.4.3 zFS hang detection

Determining the cause of a hang in zFS can be difficult. Usually, the hang is not noticed right away. This means that one or more applications are not making progress and that the information needed to diagnose the problem may be lost.

Hang detection support in z/OS V1R8

zFS has added support to try to detect when a hang has occurred in the zFS address space. The zFS hang detector, intended for use by IBM Service, monitors the current location of the various tasks processing in zFS. At a set interval, the hang detector thread wakes up and scans the current user requests that have been called into zFS.

The hang detector processes this list of tasks and notes various pieces of information that allow it to determine the location of the task. When the hang detector determines that a task

has remained in the same location for a predefined period of time, it flags the task as a potential hang and generates message IOEZ00524I or IOEZ00547I to the console. If, on a subsequent iteration, the hang detector recognizes that this task has finally progressed, it will DOM the message (remove it from the console). If the message is removed by zFS, it means that there was no hang.

If zFS finds that a user thread is at the same (wait instruction) for three consecutive minutes, then zFS displays a message to the operator console indicating that there may be a hang in zFS, as shown in Figure 4-26.

Messages IOEZ00524I and IOEZ00547I are also issued and cleared when a slowdown occurs. This is not an indication of a real hang. Instead, it indicates that things are progressing slowly because of a stressful workload or some other issue. In this case, you can discard the dump.

Continually monitor for the messages shown in the following two figures. In Figure 4-26, *UserList* is a list of address space IDs and TCB addresses causing the hang.

```
IOEZ00524I zFS has a potentially hanging thread caused by: UserList
```

Figure 4-26 Hang detection message IOEZ00524I

Figure 4-27, *Systemnames* is the list of system names.

```
IOEZ00547I zFS has a potentially hanging XCF request on systems: Systemnames
```

Figure 4-27 Hang detection message IOEZ00547I

Finding a hanging thread

To find a hanging thread, the operator can use the **F ZFS,QUERY,THREADS** command to determine if a thread is hanging, as shown in Figure 4-28.

```
F ZFS,QUERY,THREADS
IOEZ00438I Starting Query Command THREADS.
            zFS and USS Tasks
            --------------------
Recov     TCB       ASID  Stack     Routine               State
--------  --------  ----  --------  --------              --------
79D4A218  008EE650  0020  7993E6A0  comm_daemon           RUNNING
       since  Sep 8 17:03:01 2006


xcf_stg_pool: 0
--------------
>>size 4084 num subpools 1
    subpool 0 total 2147483647 available 2147483647
>>size 36852 num subpools 1
    subpool 0 total 2147483647 available 2147483647
>>size 92148 num subpools 1
    subpool 0 total 2147483647 available 2147483647

xcf_stg_pool: 1
--------------
. . . . (text omitted)
--------------
xcf thread pool: 0  size 10
---------------------------
  all threads in this pool are idle

xcf thread pool: 1  size 1
---------------------------
  all threads in this pool are idle

xcf thread pool: 2  size 4
---------------------------
  all threads in this pool are idle
. . . . (text omitted)
IOEZ00025I zFS kernel: MODIFY command - QUERY,THREADS completed successfully.
```

Figure 4-28 Command to find a hanging thread

4.4.4 Command to break hangs

If hangs are detected or you suspect that users are hung in zFS, the operator can use the following command to attempt to break the hang condition:

```
F ZFS,HANGBREAK
```

This command posts any threads that zFS suspects to be in a hang with an error and can cause abends and dumps to occur, which you can ignore. After issuing the command, the hang message can remain on the console for up to one minute. If you question the hang condition, or if the command does not seem to have resolved the situation, contact IBM Support and provide the dump and SYSLOG information.

Attention: The command MODIFY ZFS,HANGBREAK has been changed to cause a zFS internal restart beginning with z/OS V1R13. This produces the same result as issuing MODIFY ZFS,ABORT. See 4.5.3, “zFS internal restart” on page 190 for details.

4.5 zFS Internal Restart

This is an availability improvement in case of zFS internal failures.

In previous releases, when zFS had an internal failure that required zFS to abort and have z/OS UNIX restart zFS, this cleared the failure but could cause some file systems to be unmounted.

zFS in z/OS V1R13 has been restructured so that internally, a controller task attaches the zFS kernel task. When an internal failure occurs in the zFS kernel, this zFS controller task can stop and restart the zFS kernel and internally remount the zFS file systems.

- zFS can recover from internal failures without losing mounts.
- zFS time to recovery is improved.

4.5.1 zFS internal restart processing

As in previous releases, zFS restart can be explicitly invoked to resolve hangs involving zFS via the operator command MODIFY ZFS,ABORT but now in z/OS V1R13, it will attempt an internal restart.

- zFS internal restart occurs automatically if zFS detects an internal failure.
- zFS ownership of zFS sysplex-aware file systems may change during internal restart.
- As before, applications may see failures as a result of zFS restart.

Note: You can query the number of zFS internal restarts that have occurred since zFS was started by using the MODIFY ZFS,QUERY,STATUS command.

4.5.2 Previous behavior on zFS internal errors or abort

Figure 4-29 on page 190 shows the previous behavior and sequence of processing in the zFS address space.

- The zFS kernel is the main task in the zFS address space.
- When zFS has to go down based on a failure or an abort request there is nothing left to initiate a restart.
- z/OS UNIX has to restart zFS.

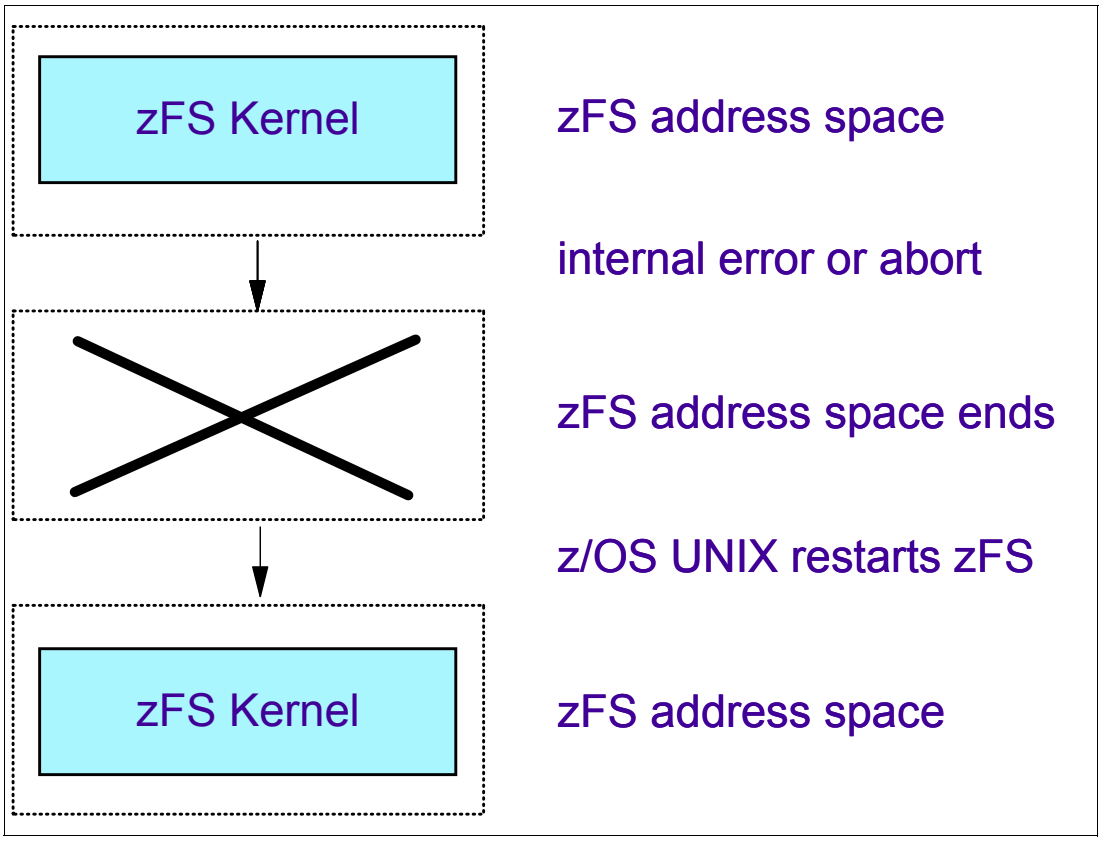


Figure 4-29 Previous behavior on zFS internal errors or abort

4.5.3 zFS internal restart

Figure 4-30 on page 191 shows the new behavior and sequence of processing in the zFS address space.

- The main task started in the zFS address space now is the controller task. This zFS controller then attaches the zFS kernel.
- On an internal error or abort the zFS kernel is ended but the controller is still active.
- The controller then initiates the zFS kernel to be restarted. This recover processing avoids losing mounted zFS file systems.

Notes:

1. zFS does not read (again) the zFS parameter settings in the IOEPRMxx parmlib member on an internal restart.
2. If a zFS internal restart does occur, all zFS mounted file systems should remain mounted.
3. The command MODIFY ZFS,HANGBREAK now causes a zFS internal restart. This produces the same result as issuing MODIFY ZFS,ABORT.

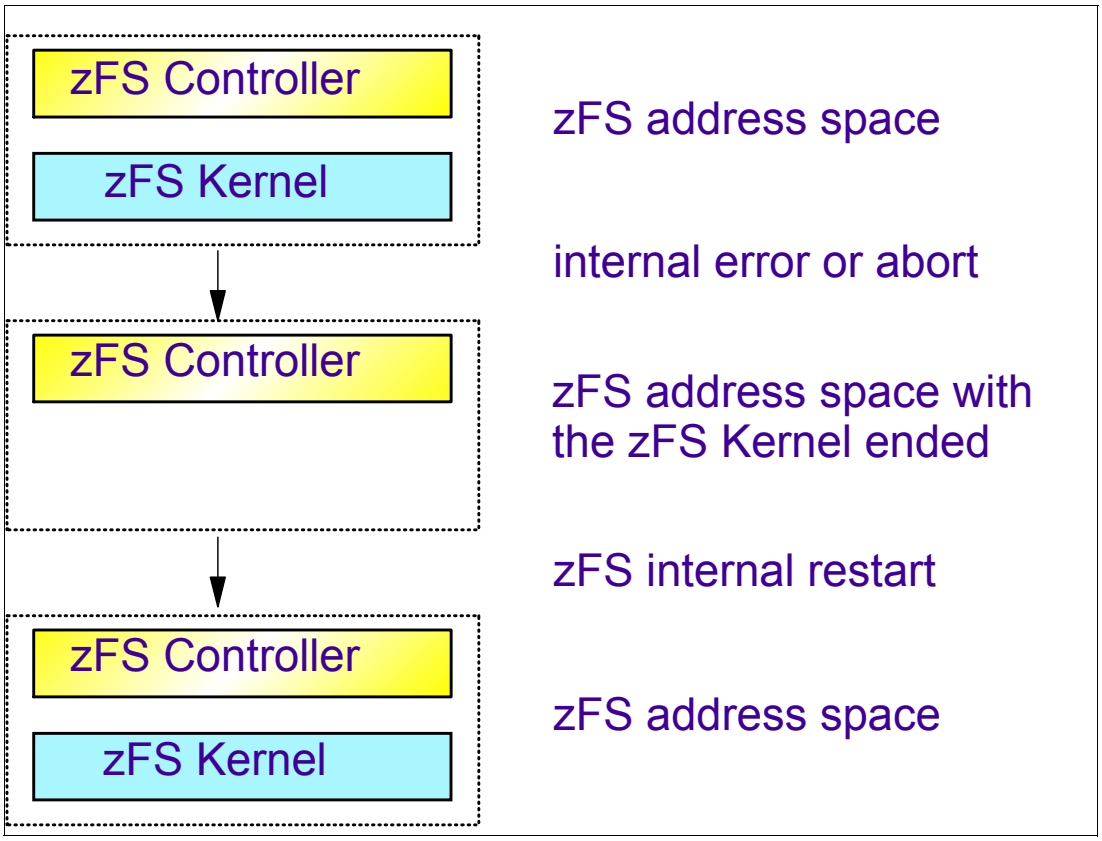


Figure 4-30 zFS internal restart

4.5.4 Forcing a zFS internal restart

Figure 4-31 shows information about the zFS environment on system SC75 from system SC74. Most zFS file systems are mounted at SC74.

```
$> echo This is system $(sysvar SYSNAME).
This is system SC74.
$> sudo /usr/sbin/mount -qv ~
----A- HERING.ZFS /u/hering
$> rxlsaggr | grep " SC75"
PLEX75.SC75.SYSTEM.ZFS                          SC75      R/W Sysplex-aware
HERING.ZFS                                      SC75      R/W Sysplex-aware
JES3.HFS                                        SC75      R/W Sysplex-aware
LUTZ.ZFS                                        SC75      R/W
$> zfsadm configquery -trace_table_size -system SC75
IOEZ00317I The value for configuration option -trace_table_size is 64M.
$> zfsadm configquery -trace_dsn -system SC75
IOEZ00317I The value for configuration option -trace_dsn is PLEX75.SC75.ZFS.TRACE.
$> cn "route sc75,f zfs,query,status"
 IOEZ00736I zFS kernel initiated at Apr 29 11:16:05 2011
   Current status: active
   Internal restart count 0
 IOEZ00025I zFS kernel: MODIFY command - QUERY,STATUS completed successfully.
$>
```

Figure 4-31 zFS information for system SC75

In Figure 4-32 an interactive REXX environment is started, a file in the home file system is opened and a zFS restart is forced by running command **f zfs,hangbreak**.

```
$> rexx
SH> "pwd"
/u/hering
SH> s "open test.zfs.internal.restart" o_creat+o_wronly 640
OMVS Return Value (retval) = 3
SH> data = "XxXxX"
SH> s "write 3 data"
OMVS Return Value (retval) = 5
SH> "cn route sc75,f zfs,hangbreak"
 IOEZ00338A zFS kernel: restarting exception 2C3 occurred, reason EA150384 abend
 psw 77C1000 9258961E
...
IOEZ00041I No start record, trace wrapped, total records 1677720, 67108800 bytesto
 format.
SH> s "write 3 data"
OMVS Return Value (retval) = 5
SH> s "write 3 esc_n"
OMVS Return Value (retval) = 1
SH> s "close 3"
OMVS Return Value (retval) = 0
SH> exit
$>
```

Figure 4-32 Forcing a zFS restart in system SC75

In Figure 4-33 we show a first excerpt of the operlog with SC74 trace and SC75 dump information.

```
 ROUTE SC75,F ZFS,HANGBREAK
 F ZFS,HANGBREAK
 IOEZ00338A zFS kernel: restarting exception 2C3 occurred, reason
 EA150384 abend psw 77C1000 9258961E
 IOEZ00337E zFS kernel: non-terminating exception 2C3 occurred, reason
 EA2F0385 abend psw 77C1000 9258961E
 IOEZ00064I General Registers R0: 4000000 42C3000 71D6E000 124967EA
...
 IOEZ00065I zFS kernel recovery: estae psw 70C0000 924FC7A8
...
 IOEZ00034I zFS kern: printing contents of trace to dataset
 <PLEX75.SC74.ZFS.TRACE(ZFSKNT01)>
...
 IOEZ00036I Printing contents of table at address 7EC5FF50 name: Main Trace Table
 IOEZ00041I No start record, trace wrapped, total records 1677720,
 67108800 bytes to format.
 IEA794I SVC DUMP HAS CAPTURED: 318
 DUMPID=001 REQUESTED BY JOB (ZFS     )
 DUMP TITLE=zFS abend 02C3 reason EA150384 May 13 18:47:57 in module IOEFSCM at
 offset 0010A61E
 IOEZ00334I Return code and reason code for dump is 40000000
```

Figure 4-33 Excerpt 1 of operlog during zFS restart

Figure 4-34 on page 193 shows a second excerpt of the operlog with SC75 trace information. A dump for zFS in SC74 is forced by SC75 in addition.

```
 IOEZ00034I zFS kern: printing contents of trace to dataset
 <PLEX75.SC75.ZFS.TRACE(ZFSKNT01)>
 IOEZ00036I Printing contents of table at address 7EC58F50 name: Main Trace Table
 IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00004)
 IEF196I         DSN (DUMP.D110513.H18.SC75.ZFS.S00001            )
 IEF196I         STORCLAS (DUMPS) MGMTCLAS (        ) DATACLAS (        )
 IEF196I         VOL SER NOS= STPDP2
 IOEZ00041I No start record, trace wrapped, total records 1677720,
 67108800 bytes to format.
...
 IOEZ00043I zFS kern: print of in-memory trace table has completed.
...
 IOEZ00051I An error occurred in program IOEFSKN
 IOEZ00357I Successfully left group IOEZFS.
 IOEZ00057I zFS kernel program IOEFSKN is ending.
 IOEZ00387E System SC75 has left group IOEZFS, aggregate recovery in progress.
...
IOEZ00579I Restarting zFS kernel, Restart count 1
```

Figure 4-34 Excerpt 2 of operlog during zFS restart

Figure 4-35 shows the completion of the zFS restart and the takeover of aggregate HERING.ZFS by SC74 with recovery and resume processing.

```
 IOEZ00388I Aggregate takeover being attempted for aggregate HERING.ZFS
...
 IOEZ00397I Recovery statistics for HERING.ZFS:
   elapsed time 6 ms 1 log pages
   2 log records, 1 data blocks modified
   1 redo-data, 0 redo-fill records
   0 undo records, 0 unwritten blocks
...
 IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System 340
 Version 01.13.00 Service Level z130224 - HZFS3D0.
 Created on Thu Mar 3 09:24:20 EST 2011.
...
*IOEZ00727I Pre-processing 32768 vnodes for file system restart 352
 IOEZ00388I Aggregate takeover being attempted for aggregate JES3.HFS
 IOEZ00044I Aggregate HERING.ZFS attached successfully.
 IOEZ00044I Aggregate JES3.HFS attached successfully.
 IOEZ00617I zFS is running sysplex filesys,norwshare with interface level 4
...
*IOEZ00646I zFS Kernel is restarting 34 file systems 361
 IOEZ00055I zFS kernel: initialization complete.
...
IOEZ00728I Resuming user operations for file system HERING.ZFS
   Waking 0 waiting tasks
   Re-cached 4 vnodes for this file system
 IOEZ00728I Resuming user operations for file system PFA.HFS
   Waking 1 waiting tasks
   Re-cached 2011 vnodes for this file system
...
 IOEZ00731I zFS internal restart complete.
```

Figure 4-35 Excerpt 3 of operlog during zFS restart

In addition note the following information.

Notes:

- Aggregate takeover took place for PLEX75.SC75.SYSTEM.ZFS and JES3.HFS the same way as for HERING.ZFS.
- Recovery took place for PLEX75.SC75.SYSTEM.ZFS and for LUTZ.ZFS as well.
- Resuming user operations was done for most of the zFS file systems in the USS sysplex file system sharing environment.

4.5.5 Results of the zFS restart

In Figure 4-36 essential results of the zFS restart are listed.

```
$> cat test.zfs.internal.restart | od -tcx1 | head -l2
0000000000    \0  \0  \0  \0  \0   X   x   X   x   X  \n
              00  00  00  00  00  E7  A7  E7  A7  E7  15
$> cn "route sc75,f zfs,query,status"
 IOEZ00736I zFS kernel initiated at Apr 29 11:16:05 2011
   Current status: active
   Internal restart count 1 (Last restart: May 13 18:48:08 2011)
 IOEZ00025I zFS kernel: MODIFY command - QUERY,STATUS completed successfully.
$> cn "f zfs,query,status"
 IOEZ00736I zFS kernel initiated at Apr 29 11:08:56 2011
   Current status: active
   Internal restart count 0
 IOEZ00025I zFS kernel: MODIFY command - QUERY,STATUS completed successfully.
$> rxlsaggr | grep " SC75"
LUTZ.ZFS                                        SC75      R/W
$> rxlsaggr | grep PLEX75.SC75.SYSTEM.ZFS
PLEX75.SC75.SYSTEM.ZFS                          SC74      R/W Sysplex-aware
$> rxlsaggr | grep HERING.ZFS
HERING.ZFS                                      SC74      R/W Sysplex-aware
$> rxlsaggr | grep JES3.HFS
JES3.HFS                                        SC74      R/W Sysplex-aware
$>
```

Figure 4-36 Results of the zFS restart

Note the following information for file systems and I/O processing:

- The zFS ownership of file systems previously owned by the system that did a zFS restart may go to another system.
- If so, this happens independent of the automove setting and the USS ownership of the file system.
- Active I/O operations may fail and data, not on DASD already, may be lost.

4.6 zFS automatic re-enablement of disabled aggregates

This is an availability improvement for zFS aggregate failures.

4.6.1 Behavior in previous releases

In previous releases, when zFS had an internal failure that required zFS to disable an aggregate, you had two possibilities to solve the problem.

- The file system needed to get explicitly unmounted and then mounted to recover it.
- Or you needed to perform a remount samemode processing for the file system.

4.6.2 The new R13 automatic re-enablement of disabled aggregates

The implementation works as follows:

- zFS automatic re-enablement of disabled aggregates occurs automatically if zFS disables an aggregate.
- So, zFS can recover a disabled aggregate without administrator intervention.
- zFS ownership of a disabled zFS sysplex-aware file system may change during automatic re-enablement of the disabled aggregate.
- As before, applications may see failures as a result of zFS disabling an aggregate.
- Administrators should run the salvager utility (IOEAGSLV) against the aggregate at the earliest convenience.

Note:

- Automatic re-enablement of a disabled aggregate that becomes disabled again is tried up to three times.
- If this occurs, the file system must be manually unmounted and mounted again.

4.6.3 Sample messages on automatic re-enablement

Figure 4-37 shows automatic re-enablement messages when a zFS aggregate gets disabled.

```
IOEZ00422E Aggregate HERING.TEST.ZFS disabled
IOEZ00548I Requesting that SC75 takeover aggregate HERING.TEST.ZFS
IOEZ00388I Aggregate takeover being attempted for aggregate HERING.TEST.ZFS
IOEZ00044I Aggregate HERING.TEST.ZFS attached successfully.
```

Figure 4-37 Messages on automatic re-enablement

In Figure 4-38 you see the message shown when automatic re-enablement of a zFS aggregate is halted after repeated failures.

```
IOEZ00422E Aggregate HERING.TEST.ZFS disabled
IOEZ00746E Automatic re-enablement of file system HERING.TEST.ZFS
halted after repeated failures
```

Figure 4-38 Halting automatic re-enablement after repeated failures

In this situation you can use one of the following commands as you could do before introducing this new function:

```
tsocmd “unmount filesystem('hering.test.zfs') remount(samemode)”
sudo rexx 's "unmount HERING.TEST.ZFS" mtm_samemode'
```

```
sudo /usr/sbin/unmount -f HERING.TEST.ZFS
```

Notes:

- The utility rexx allows you to run commands interactively. It is available from the ITSO tools disk at:

  `ftp://www.redbooks.ibm.com/redbooks/SG247035/`

- In case of the unmount a new mount is needed to have the file system available again.

4.7 Problem determination

zFS provides a trace facility that should be used in problem determination. The trace is an internal (wrap-around) trace table that traces certain events. The size of this trace table is controlled by the IOEFSPRM trace_table_size option, shown in Figure A-1 on page 360.

4.7.1 Trace data set

To use the trace facility for printing purposes, you need to specify a trace output data set, as shown in Figure 4-39. This data set must be allocated before you can begin to process trace output. We created the data set OMVS.SC65.ZFS.TRACEOUT and specified this name in the IOEFSPRM member.

Trace data set size

The trace data set should be allocated as a PDSE, with RECFM=VB and LRECL=133, and the size should be with a primary allocation of at least 50 cylinders and a secondary allocation of 30 cylinders.

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 32 MB trace_table_size, which is the default, can generate a trace output member of 100 cylinders of 3390s. Therefore, the trace output data set should be large enough to hold the trace output because the complete trace will not be captured if it runs out of room sending the trace to the trace output data set.

```
trace_dsn=OMVS.SC65.ZFS.TRACEOUT
*debug_settings_dsn=data.set.name(membername)
trace_table_size=64M
*storage_details=ON
*storage_details_dsn=data.set.name
```

Figure 4-39 IOEFSPRM data set trace entries

A trace_table_size of 64 MB is considered to be a good value to keep enough information for problem determination. Note that this value cannot be changed dynamically without restarting zFS.

Note: A separate trace output data set is required for each member of a sysplex. This requires separate IOEFSPRM files. Refer to “IOEFSPRM file considerations” on page 203 for more information about this topic.

4.7.2 zFS abends and trace output

Each trace output is created as a new member with the name ZFSKNTnn. When the first tracing begins, nn starts at 01 and is incremented for each trace output until zFS is restarted.

When a problem occurs, the current trace data set is processed for print, as shown in Figure 4-40. ZFSKNT01, being the first trace member, becomes the first member of the PDS trace output.

```
IOEZ00051I An error occurred in program IOEFSCM:, terminating zFS kernel
IOEZ00034I zFS kernel: printing contents of trace to dataset
<OMVS.SC65.ZFS.TRACEOUT(ZFSKNT01)
IOEZ00036I Printing contents of table at address 7D66D000 name: Main Trace
Table
IOEZ00042I Start record found, total records 1812, 103620 bytes to format.
IOEZ00043I zFS kernel: print of in-memory trace table has completed.
```

Figure 4-40 Log messages showing creation of first PDS member

Restart of zFS

After a zFS restart, when the next trace output is sent to the trace output data set, ZFSKNT01 is overlaid. You should not be accessing the trace output data set while a trace is being sent to it.

4.7.3 Debugging using the trace

When it is necessary to trace something for debugging purposes, the trace table can be reset by issuing the following command:

```
f zfs,trace,reset
```

This command clears the trace table, which allows recreating a problem with the data related to the problem and also to minimize the amount of information generated.

Printing the trace

The trace table can be formatted and sent to the trace output data set by using the following operator command:

```
f zfs,trace,print
```

This command creates the processing shown in Figure 4-41.

```
IOEZ00034I zFS kernel: printing contents of trace to dataset
<OMVS.SC65.ZFS.TRACEOUT(ZFSKNT02)>
IOEZ00036I Printing contents of table at address 7D66D000 name: Main Trace
Table
IOEZ00042I Start record found, total records 56, 3180 bytes to format.
IOEZ00043I zFS kernel: print of in-memory trace table has completed.
IOEZ00025I zFS kernel: MODIFY command - TRACE,PRINT completed successfully.
```

Figure 4-41 Output from the print trace command

4.7.4 Taking a zFS-related dump dynamically

If a IEADMCZS parmlib member with the contents shown in Figure 4-42 on page 198 exists, you may use the following system command to take a dump dynamically:

```
dump parmlib=zs
```

Parmlib member IEADMCZS, shown in Figure 4-42, does not dump the zFS data spaces. They are not referenced assuming that this will be too much data and that they do not have really useful information that would help in problem determination. It may also be useful to add the address spaces or jobnames that are involved in the actual problem.

```
COMM=(zFS Dump)
JOBNAME=(ZFS,OMVS),DSPNAME=('OMVS'.*),
SDATA=(NUC,CSA,LPA,LSQA,RGN,SQA,SUM,SWA,TRT,PSA)
```

Figure 4-42 IEADMCZS parmlib member

4.7.5 Analyzing hang conditions

To start investigating, enter the **D OMVS,W** command to check the state of sysplex messages or waiters. Message IOEZ00547I (hanging XCF request) can indicate an XCF issue. Check any outstanding message that might need a response to determine if a system is leaving the sysplex or not (for example, IXC402D). This might look like a zFS hang until that message gets a response. Then do the following:

- Enter the **F ZFS,QUERY,THREADS** command as shown in Figure 4-28 on page 188 to determine if, and why, any zFS threads are hanging.

  Note: The type and amount of information displayed as a result of this command is for internal use and can vary between releases or service levels.

- Enter the **D A,ZFS** command to determine the zFS ASID.
- Enter the **F ZFS,QUERY,THREADS** command at one- to two-minute intervals for six minutes.
- Interrogate the output for any user tasks (tasks that do not show the zFS ASID) that are repeatedly in the same state during the time you requested **F ZFS,QUERY,THREADS**. If there is a hang, this user task will persist unchanged over the course of this time span. If the information is different each time, it means that there is no hang.
- Verify that no zFS aggregates are in the QUIESCED state by checking their status using the **zfsadm lsaggr** or **zfsadm aggrinfo** command. For example, quiesced aggregates display as follows:

```
DCESVPI:/home/susvpi/> zfsadm lsaggr

IOEZ00106I A total of 1 aggregates are attached
SUSVPI.HIGHRISK.TEST                          DCESVPI   R/W QUIESCE
DCESVPI:/home/susvpi/> zfsadm aggrinfo
IOEZ00370I A total of 1 aggregates are attached.
SUSVPI.HIGHRISK.TEST (R/W COMP QUIESCED): 35582 K free out of total 36000
DCESVPI:/home/susvpi/>
```

Resolve the QUIESCED state, while continuing to determine if there is a real hang condition. The hang condition message can remain on the console for up to a minute after the aggregate is unquiesced.

Note: Message IOEZ00581E appears on the system that contains at least one zFS aggregate that is quiesced. There is a time delay between when the aggregate is quiesced and when the message appears. When there are no quiesced zFS aggregates on the system, this message is DOMed.

There is also a delay between when the last aggregate is unquiesced and when the message is DOMed. This message is handled by a thread that wakes up every 30 seconds and checks for any quiesced aggregates owned by this system.

It is possible for an aggregate to be quiesced and unquiesced in the 30-second sleep window of the thread and no quiesce message to appear. This message remains if one aggregate is unquiesced and another is quiesced within the 30-second sleep window.

Final considerations

Finally, if the previous steps do not clear the hang, do one of the following:

- Enter the F ZFS,HANGBREAK command to attempt to break the hang condition. The F ZFS,HANGBREAK command posts any threads that zFS suspects are in a hang condition with an error and can cause abends and dumps to occur, which you can ignore.

  After entering the F ZFS,HANGBREAK command, the hang message can remain on the console for up to one minute. When the F ZFS,HANGBREAK command completes, it issues message IOEZ00025I. However, IOEZ00025I does not mean the system cleared the hang. Enter the F ZFS,QUERY,THREADS command to check the output for indication the hang is clear.

  It is possible that the F ZFS,HANGBREAK command can clear the current hang condition only to encounter yet another hang. You might have to enter the F ZFS,HANGBREAK command several times.

- If users are hung in the file system, forcefully unmount the file system by entering the **F** ZFS,ABORT command.

Important: Keep in mind that the behavior of command MODIFY ZFS,HANGBREAK has been changed completely and being different than just described. It now causes a zFS internal restart (beginning with z/OS V1R13). This produces the same result as issuing MODIFY ZFS,ABORT. See 4.5.3, “zFS internal restart” on page 190 for details.

z/OS V1R9 enhancements

With z/OS V1R9, there is a new zFS parameter file specification that can be used to change the hang detection interval.

```
hang_detection_interval=45 - default is 45 seconds
```

This improves potential hangs to avoid reporting false hangs. However, normally it is only used when suggested by IBM for problem analysis. The hang detection can be turned on and off with an operator MODIFY F command, as follows:

```
F ZFS,HANGDETECT,ON  -     to turn on
F ZFS,HANGDETECT,OFF -     to turn off
```

Hang detection is set ON by default.

Important: This specification and the F ZFS,HANGDETECT command are not documented at the moment. They are normally only used if advised by IBM, especially in case of analysis for a reported problem.

4.7.6 zFS bpxmtext enhancements

zFS reason codes are returned to APIs and are sometimes displayed in messages. Users do not always know where to find the meaning of the reason code. z/OS UNIX provides a **bpxmtext** command that can be issued from a shell session that displays the meaning of z/OS UNIX reason codes.

Previously, the **bpxmtext** command could not display the meaning of zFS reason codes. With z/OS V1R8, support is added to allow the **bpxmtext** command to do this. See *z/OS Distributed File Service Messages and Codes,* SC24-5917, to obtain detailed information about these codes.

Following is a message for a zFS error code in which you can see the new functionality to display the meaning of zFS reason codes:

```
ROGERS @ SC75:/u/rogers>bpxmtext EF096800
zFS Fri Jul 28 10:36:15 EDT 2006
Description: Mount for file system contain in multi-file system aggregate is
not allowed

Action: Using a release of z/OS prior to z/OS V1R8, attach the aggregate,
mount the file system and copy the file system data to a compatibility mode
aggregate.
```

5

Sysplex considerations

This chapter describes using zFS file systems in sysplex sharing mode. zFS supports a shared file system capability in a multisystem sysplex environment. The term *shared file system environment* refers to a sysplex that has a BPXPRMxx specification of SYSPLEX(YES). That is, users in a sysplex can access zFS data that is owned by another system in the sysplex.

For full sysplex support, zFS must be running on all systems in the sysplex in a shared file system environment, and all zFS file systems must be compatibility mode file systems (that is, they cannot be file systems in multifile system aggregates). zFS multifile system aggregates are not supported by automount.

zFS sysplex-aware support has no effect on z/OS UNIX file system movement definitions. z/OS UNIX still performs movements in the sysplex-aware group of systems independent of zFS ownership movement.

This chapter contains the following functional changes:

- zFS shared sysplex support
- Sharing compatibility mode file systems
- Using automount in a sysplex environment
- zFS sysplex-aware mounts for read-write file systems

5.1 zFS shared sysplex support

zFS file systems can be shared in a sysplex environment. This means that users in a sysplex can access a zFS file in a file system that is owned and mounted on another system in the sysplex. zFS uses the same support for sharing as shared HFS, which became available with OS/390 V1R9.

With shared HFS support, all file systems that are mounted by a participating system are available to all participating systems. When a zFS file system is mounted by a participating system, that file system is accessible by any other participating system. It is not possible to mount a file system so that it is restricted to just one of those systems.

5.1.1 Compatibility mode file systems

As in shared HFS support, zFS file systems can be mounted as AUTOMOVE. In Figure 5-1, file system OMVS.CMP01.ZFS is mounted as AUTOMOVE on system SC63. If system SC63 fails the zFS file system is automatically moved to another system in the sysplex, in this case system SC64, because zFS is not started on system SC65. The shared sysplex support works as follows:

- zFS must be running on the other system in the sysplex.
- zFS file systems must be compatibility mode file systems.
- It requires both zFS and UNIX System Services support for UNIX System Services aggregate awareness and UNIX System Services automove support for zFS.
- zFS file systems owned by system SC63 are accessible from systems SC64 and SC65 because those systems in the sysplex are running zFS.

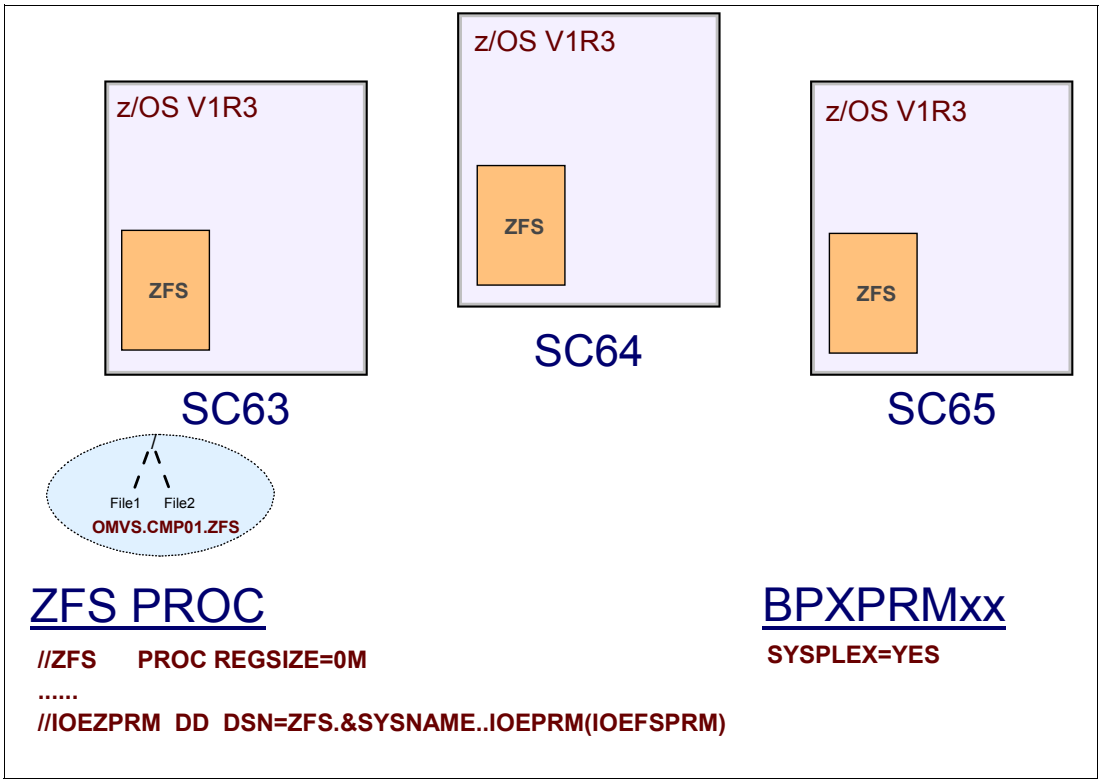


Figure 5-1 Shared sysplex support for zFS file systems

IOEFSPRM file considerations

The same IOEFSPRM file cannot be shared with z/OS V1R3 across systems in a sysplex if the file contains the following types of specifications:

- A multifile system aggregate specification such as:

  define_aggr

- Data sets that cannot be shared such as:
  - A msg_output_dsn specification
  - A trace_dsn specification
  - debug_settings_dsn specification

Recommendation: Figure 5-1 on page 202 shows the zFS PROC, used for all systems, in which it specifies the use of the &SYSNAME system variable in the data set name of the IOEZPRM DD. This allows for the specification of different IOEFSPRM parameters for different systems in the sysplex.

System symbols in the IOEFSPRM file

In z/OS V1R4 system symbols can now be specified for data set names in the IOEFSPRM configuration file, to make it easier to share a single IOEFSPRM file in a sysplex.

Figure 5-2 illustrates uses of system symbols in IOEFSPRM configuration options.

```
trace_dsn=OMVS.&SYSNAME..ZFS.TRACEOUT
define_aggr R/W attach aggrgrow cluster(OMVS.&SYSNAME..AGGR1)
```

Figure 5-2 IOEFSPRM configuration file using system symbols

Compatibility mode aggregate with a clone

The read-write file system and the read backup file system, called a *clone*, must be mounted on the same system. If you want to use AUTOMOVE, you cannot mount the clone together with the read-write file system when the owning system is about to leave the sysplex. This is because in z/OS V1R3, there are no controls that would assure that both file systems get mounted on the same system again if the old owning system fails.

This restriction has been removed in z/OS V1R4. UNIX logical file system code now assures that the read-write file system and the clone are mounted at the same new system. However, if you want to perform a move manually by, for example, using the **chmount** command, you must first unmount the clone, then move the ownership of the read-write file system, and mount the clone again.

5.1.2 Multiple file system aggregates

All zFS file systems in a multifile system aggregate must be owned and mounted on the same system in a sysplex with the NOAUTOMOVE option specified.

z/OS UNIX is not aware of the relationship between zFS file systems in the same aggregate. Therefore, in a shared sysplex mode, if you want to move the ownership of file systems in a multifile system aggregate, you must do the following:

1. Manually unmount each file system in the multifile system aggregate.
2. Detach the aggregate.
3. Attach the aggregate on the other system.
4. Mount all the file systems on the other system.

Note: You cannot move one zFS file system that resides in a multifile system aggregate to another system without moving the whole multifile system aggregate and all the file systems in the aggregate. The current AUTOMOVE code does not support this.

5.6, “Multifile system aggregates behavior when zFS stops” on page 232 describes an example in which file systems in a multifile system aggregate are unmounted when zFS fails in the system where the aggregate is attached.

Automount considerations

A zFS file system in a multifile system aggregate should not be automounted, because an automounted file system is always mounted AUTOMOVE. Also, automounted file systems might be mounted and owned by any system.

Removal of support for multifile system aggregates

As mentioned previously, IBM recommended that multifile system aggregates should not be shared in a sysplex environment because there is no support added to the UNIX Logical File System (LFS) code to handle this in case the system owning the aggregate is leaving the sysplex.

In February 2005, IBM announced that z/OS V1R7 is planned to be the last release to allow mounting zFS file systems contained in multifile system aggregates in a UNIX System Services sysplex sharing environment.

After this support is removed, attempts to mount zFS file systems contained in multifile system aggregates fail in a UNIX System Services shared file system environment. Mounting zFS compatibility mode aggregates, which have a single R/W file system per data set, will continue to be supported in all environments.

Attention: In the release after z/OS V1R7, mounting a file system contained in a multifile system aggregate will be denied when running in a UNIX System Services shared file system environment.

In addition, as a statement of direction (SOD), z/OS V1R10 is planned to be the last release to allow attaching a zFS multifile system aggregate in a UNIX System Services sysplex file system sharing environment.

See also 1.6.2, “Multifile system aggregates” on page 6 for reference to a note saying that support for multifile system aggregates will be generally removed in a future release.

5.2 Automount for compatibility mode file systems

We ran a small automount test within the sysplex environment shown in Figure 5-1 on page 202 and had zFS active on all three systems. The automount policy shown in Figure 5-3 on page 205 was used on all systems in the sysplex.

```
name               *
type               ZFS
filesystem         OMVS.<uc_name>.ZFS
mode               rdwr
duration           1440
delay              360
```

Figure 5-3 Automount policy in /etc/auto.map

We had three user file systems mounted in different systems at the beginning of our test, as shown in Figure 5-4. All the commands were issued from system SC63.

```
 D OMVS,F
 BPXO045I 16.07.19 DISPLAY OMVS 097
 OMVS     000F ACTIVE          OMVS=(3A)
 TYPENAME   DEVICE ----------STATUS----------- MODE
 ZFS             95 ACTIVE                     RDWR
   NAME=OMVS.USER01.ZFS
   PATH=/u/user01
   OWNER=SC63     AUTOMOVE=Y CLIENT=N
 ZFS             96 ACTIVE                     RDWR
   NAME=OMVS.USER02.ZFS
   PATH=/u/user02
   OWNER=SC63     AUTOMOVE=Y CLIENT=N
ZFS             97 ACTIVE                     RDWR
   NAME=OMVS.USER03.ZFS
   PATH=/u/user03
   OWNER=SC64     AUTOMOVE=Y CLIENT=Y
AUTOMNT         28 ACTIVE                     RDWR
   NAME=*AMD/u
   PATH=/u
   OWNER=SC64     AUTOMOVE=Y CLIENT=N
...
```

Figure 5-4 Automounted zFS user file systems

We used the **chmount** command to change the ownership of OMVS.USER03.ZFS to SC63. This is shown in Figure 5-5.

```
#> /usr/sbin/chmount -d SC63 /user/compat03
```

Figure 5-5 Changing the ownership of file system OMVS.USER03.ZFS

Now all the user file systems were owned by SC63, as shown in Figure 5-6 on page 206.

```
D OMVS,F
 BPXO045I 16.10.12 DISPLAY OMVS 435
 OMVS     000F ACTIVE          OMVS=(3A)
 TYPENAME   DEVICE ----------STATUS----------- MODE
 ZFS            95 ACTIVE                      RDWR
   NAME=OMVS.USER01.ZFS
   PATH=/u/user01
   OWNER=SC63     AUTOMOVE=Y CLIENT=N
 ZFS            96 ACTIVE                      RDWR
   NAME=OMVS.USER02.ZFS
   PATH=/u/user02
   OWNER=SC63     AUTOMOVE=Y CLIENT=N
ZFS            97 ACTIVE                      RDWR
   NAME=OMVS.USER03.ZFS
   PATH=/u/user03
   OWNER=SC63     AUTOMOVE=Y CLIENT=N
AUTOMNT        28 ACTIVE                      RDWR
   NAME=*AMD/u
   PATH=/u
   OWNER=SC64     AUTOMOVE=Y CLIENT=N
...
```

Figure 5-6 New ownership of the automounted zFS file system OMVS.USER03.ZFS

Next, we stopped zFS on system SC63. The zFS file systems were no longer available in system SC63, as shown in Figure 5-7. This occurred because the z/OS level on SC63 was lower than V1R6 when doing these tests; refer to 5.4, “Logical file system support of zFS in z/OS V1R6” on page 213 for more information about changes in LFS in z/OS V1R6.

```
#> ls -ER /u/
/u/:
total 0
```

Figure 5-7 zFS file systems no longer seen by system SC63

After restarting zFS, the file systems were back in the system, as shown in Figure 5-8.

```
#> ls -E /u/
total 48
drwxr-xr-x        2 DFS      SYS1         256 Apr 25 02:53 compat01
drwxr-xr-x        2 DFS      SYS1         256 Apr 25 02:54 compat02
drwxr-xr-x        2 DFS      SYS1         256 Apr 25 02:54 compat03
```

Figure 5-8 zFS file systems seen again after zFS restart

The file systems were then owned by SC64 and SC65, as shown in Figure 5-9 on page 207.

```
D OMVS,F
 BPXO045I 16.14.03 DISPLAY OMVS 355
 OMVS     000F ACTIVE          OMVS=(3A)
 TYPENAME   DEVICE ----------STATUS----------- MODE
 ZFS            95 ACTIVE                      RDWR
   NAME=OMVS.USER01.ZFS
   PATH=/u/user01
   OWNER=SC64     AUTOMOVE=Y CLIENT=Y
 ZFS            96 ACTIVE                      RDWR
   NAME=OMVS.USER02.ZFS
   PATH=/u/user02
   OWNER=SC65     AUTOMOVE=Y CLIENT=Y
ZFS            97 ACTIVE                      RDWR
   NAME=OMVS.USER03.ZFS
   PATH=/u/user03
   OWNER=SC64     AUTOMOVE=Y CLIENT=Y
AUTOMNT        28 ACTIVE                      RDWR
   NAME=*AMD/u
   PATH=/u
   OWNER=SC64     AUTOMOVE=Y CLIENT=N
...
```

Figure 5-9 New ownership of the automounted zFS user file systems

5.3 zFS remount considerations

The remount function makes it possible to change the mount mode of a file system from RDWR to READ or READ to RDWR. For sysplex sharing environments, this function has been made available in z/OS V1R4 with APAR OA02584 and PTF UA04906.

All sysplex members must have this support. If one or more members are back level, errno EINVAL (79x) and errnojr JrNotSupInSysplex (58804A5x) will still be returned when remount is requested. This reason code simply says that remount is not supported in sysplex.

5.3.1 Remount processing in sysplex sharing

The syntax for remount processing in a sysplex is the same as it is for non-sysplex. There are several commands and interfaces that allow you to use it. Following is a list of possibilities:

- The TSO UNMOUNT command with the REMOUNT parameter supports it.
- Use MtmRemount set on in the MtmFlags with an Assembler BPX1UMT API call.
- Use the mtm_remount flag with REXX Syscall command unmount.
- In z/OS V1R5, this is also supported with the **chmount** command (options **-r** or **-w**).

In the following section we discuss considerations regarding the use of remount with zFS.

Important: In a sysplex-sharing environment, a remount always needs to be able to switch the aggregate attach mode at the same time. Otherwise, the remount is not performed.

Refer to 5.4.5, “Mounting zFS file systems R/O” on page 225 for additional information about this topic.

5.3.2 zFS remount considerations

There are two considerations where remount processing for zFS does not work:

- If both the primary file system and its clone are mounted, remount will fail because you cannot detach the aggregate.
- If the aggregate is attached R/O, then file systems in it can only be mounted R/O and remount to R/W will result in a zFS error.

Compatibility mode aggregates with a clone

For zFS HFS-compatible aggregates, if both the clone and the primary file system are mounted, then sysplex remount will be rejected with errno EINVAL and errnojr JrAggregateErr. The suggested action is to unmount the clone and retry the remount. This is because the unmount phase of remount will only unmount the primary, leaving the clone mounted and leaving the aggregate in an attached state.

This is shown in Figure 5-10, which was run in a z/OS V1R5 system participating in UNIX System Services sysplex sharing.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.05.00
#> /usr/sbin/mount -f OMVS.HERING.TEST.ZFS -t zFS /u/hering/testfs
#> zfsadm clone -filesystem OMVS.HERING.TEST.ZFS
IOEZ00225I File system OMVS.HERING.TEST.ZFS successfully cloned.
#> /usr/sbin/mount -f OMVS.HERING.TEST.ZFS.bak -t zFS -r /u/hering/testfs.clone
#> zfsadm lsfs -aggregate OMVS.HERING.TEST.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS     RW (Mounted R/W)       8 K alloc      9 K quota On-line
OMVS.HERING.TEST.ZFS.bak BK (Mounted R/O)       9 K alloc      9 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
#> zfsadm aggrinfo -aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS (R/W COMP): 558 K free out of total 720
#> /usr/sbin/chmount -r /u/hering/testfs
FOMF0504I remount error: 79 5090576
EINVAL: The parameter is incorrect
JrAggregateErr: Remount is not allowed for a filesystem in an HFS-compatible aggregate
if the clone is also mounted.
```

Figure 5-10 zFS remount restriction if both clone and primary are mounted in sysplex sharing

This is different in a non-sysplex sharing situation (UNIX System Services single system). We demonstrate that by commands run in a z/OS V1R4 system, as shown in Figure 5-11 on page 209.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.04.00
#> zfsadm aggrinfo -aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS (R/W COMP): 2357 K free out of total 7200
#> zfsadm lsfs -aggregate OMVSM.COMP002.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS        RW (Mounted R/W)      16 K alloc   4698 K quota On-line
OMVSM.COMP002.ZFS.bak    BK (Mounted R/O)    4698 K alloc   4698 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
#> tso -t "unmount filesystem(OMVSM.COMP002.ZFS) remount"
unmount filesystem(OMVSM.COMP002.ZFS) remount
#> zfsadm aggrinfo -aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS (R/W COMP): 2341 K free out of total 7200
#> zfsadm lsfs -aggregate OMVSM.COMP002.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS        RW (Mounted R/O)      16 K alloc   4698 K quota On-line
OMVSM.COMP002.ZFS.bak    BK (Mounted R/O)    4698 K alloc   4698 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
```

Figure 5-11 zFS remount processing for a compatibility mode aggregate in non-sysplex sharing

Nevertheless, a problem is seen if a remount is performed before mounting the clone and trying to run a further remount, as shown in Figure 5-12 on page 210. In message BPXF137E the return code 8Dx (EROFS) means: `The specified file system is read only.` and the zFS reason code 605Ax means: `Attempt to mount a file system R/W on a R/O aggregate.`

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.04.00
#> zfsadm aggrinfo -aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS (R/W COMP): 2341 K free out of total 7200
#> zfsadm lsfs -aggregate OMVSM.COMP002.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS        RW (Mounted R/W)      16 K alloc   4698 K quota On-line
OMVSM.COMP002.ZFS.bak    BK (Not Mounted)    4698 K alloc   4698 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 1
#> tso -t "unmount filesystem(OMVSM.COMP002.ZFS) remount"
unmount filesystem(OMVSM.COMP002.ZFS) remount
#> zfsadm aggrinfo -aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS (R/O COMP): 2341 K free out of total 7200
#> zfsadm lsfs -aggregate OMVSM.COMP002.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS        RW (Mounted R/O)      16 K alloc   4698 K quota On-line
OMVSM.COMP002.ZFS.bak    BK (Not Mounted)    4698 K alloc   4698 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 1
#> /usr/sbin/mount -f OMVSM.COMP002.ZFS.bak -t zFS -r /u/hering/test.clone
#> zfsadm lsfs -aggregate OMVSM.COMP002.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS        RW (Mounted R/O)      16 K alloc   4698 K quota On-line
OMVSM.COMP002.ZFS.bak    BK (Mounted R/O)    4698 K alloc   4698 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
#> zfsadm aggrinfo -aggregate OMVSM.COMP002.ZFS
OMVSM.COMP002.ZFS (R/O COMP): 2341 K free out of total 7200
#> tso -t "unmount filesystem(OMVSM.COMP002.ZFS) remount"
unmount filesystem(OMVSM.COMP002.ZFS) remount
RC(12)
BPXF137E RETURN CODE 0000008D, REASON CODE EF09605A.  THE UNMOUNT FAILED FOR FILE SYSTEM
OMVSM.COMP002.ZFS.
```

Figure 5-12 zFS remount error for a compatibility mode aggregate in non-sysplex sharing

zFS multifile system aggregates and remount

If zFS multifile system aggregates are involved and the aggregate is attached R/O, then file systems in it can only be mounted R/O and remount to R/W will result in a zFS error. This problem is seen the same way in a UNIX System Services non-sysplex sharing environment. This is shown in Figure 5-13 (and it is the same situation as shown in Figure 5-12).

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.04.00
#> zfsadm aggrinfo -aggregate OMVSM.MULTIF02.ZFS
OMVSM.MULTIF02.ZFS (R/O MULT): 20606 K free out of total 20880
#> /usr/sbin/mount -f OMVSM.MULTIF02.M01.ZFS -t zFS -r /u/hering/test
#> tso -t "unmount filesystem(OMVSM.MULTIF02.M01.ZFS) remount"
unmount filesystem(OMVSM.MULTIF02.M01.ZFS) remount
RC(12)
BPXF137E RETURN CODE 0000008D, REASON CODE EF09605A.  THE UNMOUNT FAILED FOR FILE SYSTEM
OMVSM.MULTIF02.M01.ZFS.
```

Figure 5-13 zFS remount error for a multifile system aggregate attached read only

5.3.3 Switching from R/O to R/W and back again to R/O

Finally, we want to demonstrate what happens in different systems when performing a remount. To do so, the effects are listed for a UNIX System Services sysplex sharing environment with four systems: SC63, SC64, SC65, and SC70. As shown in Figure 5-14, in the beginning all systems show the same information when listing information for aggregate OMVS.HERING.ZFS.

```
$> echo System $(sysvar SYSNAME)
System SC63
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/O COMP): 565 K free out of total 720

$> echo System $(sysvar SYSNAME)
System SC64
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/O COMP): 565 K free out of total 720

#> echo System $(sysvar SYSNAME)
System SC65
#> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/O COMP): 565 K free out of total 720

$> echo System $(sysvar SYSNAME)
System SC70
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/O COMP): 565 K free out of total 720
```

Figure 5-14 Four systems sharing a compatibility mode aggregate R/O

The results of listing more information about the file system and then doing a remount from system SC65 are shown in Figure 5-15.

```
#> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/O COMP): 565 K free out of total 720
#> df -v test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.ZFS)         1130/1150      4294967291 Available
ZFS, Read Only, Device:234, ACLS=Y
File System Owner : SC70        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.ZFS
#> /usr/sbin/chmount -w test
#> df -v test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.ZFS)         1130/1150      4294967291 Available
ZFS, Read/Write, Device:234, ACLS=Y
File System Owner : SC70        Automove=Y      Client=Y
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.ZFS
```

Figure 5-15 Remounting the compatibility mode file system in R/W mode

In parallel for each of the four systems, an “aggregate detaching” message is displayed as shown in Figure 5-16 on page 212.

```
IOEZ00048I Detaching aggregate OMVS.HERING.ZFS
```

Figure 5-16 Aggregate detaching message in the syslog

Figure 5-17 shows the results when now listing aggregate information again in the systems.

```
$> echo System $(sysvar SYSNAME)
System SC63
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
IOEZ00210E Error 121 (reason EF04624E) occurred while attempting to get status for
aggregate OMVS.HERING.ZFS

$> echo System $(sysvar SYSNAME)
System SC64
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
IOEZ00210E Error 121 (reason EF04624E) occurred while attempting to get status for
aggregate OMVS.HERING.ZFS

#> echo System $(sysvar SYSNAME)
System SC65
#> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
IOEZ00210E Error 121 (reason EF04624E) occurred while attempting to get status for
aggregate OMVS.HERING.ZFS

$> echo System $(sysvar SYSNAME)
System SC70
$> zfsadm aggrinfo -aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS (R/W COMP): 565 K free out of total 720
```

Figure 5-17 Only one system (the owner) having the aggregate attached in R/W mode

The reason is that now SC70 has the file system mounted R/W and the aggregate attached R/W. The other systems cannot have the aggregate attached and need to access the file system by using XCF function shipping.

Finally the remount operation back to R/O is shown in Figure 5-18.

```
#> /usr/sbin/chmount -r test
#> df -v test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.ZFS)         1130/1150      4294967291 Available
ZFS, Read Only, Device:234, ACLS=Y
File System Owner : SC70        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.ZFS
```

Figure 5-18 Remounting the compatibility mode file system in R/O mode

In parallel to the remount, the aggregate detaching message shown in Figure 5-16 is displayed in the syslog for system SC70, because it is the only system this time performing a detach. The final result that we see when listing the aggregate information again is that of the beginning, shown in Figure 5-14 on page 211.

5.4 Logical file system support of zFS in z/OS V1R6

In z/OS V1R6 a change is being made to LFS termination of a PFS, such as zFS, to improve the availability of file systems on the system where a PFS is terminating.

5.4.1 New LFS support for zFS

The old design of PFS termination is that file systems for the terminating PFS, and subtrees of those file systems, get moved to another system (if locally owned), and then get locally unmounted and become unavailable on the system where the PFS is terminating. If they could not be moved, then they become globally unmounted.

The new design is that if the ownership of these file systems can be moved to another system in the sysplex, and then allow for function-shipping requests on the system where a PFS is terminating and avoid the local unmounts. This provides improved availability of file systems.

If the file system is sysplex-aware (locally mounted), but not owned by the system where the PFS is terminating, then the file system will be converted to function-shipping to the owner (no move occurs). Note the following definitions:

Sysplex-aware Capable of mounting locally in the systems; for example, R/O zFS file systems.

Sysplex-unaware Not capable of mounting locally in the systems. Function ships the request to the owner. For example, R/W zFS file systems are sysplex-unaware currently.

With the new LFS support of sysplex zFS, you can continue to access the file systems owned by the system where PFS was terminated.

Note: Each file system is moved or converted to function-shipping one at a time. As a result there is a window during which the PFS is dead and all file systems are not yet function-shipping, in which ops (requests) will fail. This design accepts that window.

The benefits of this support are:

- Improved availability of file systems.
- Applications can continue to access the file systems using function-shipping on the system where a PFS is terminating.

5.4.2 Example of the sysplex sharing support

Figure 5-19 on page 214 shows a UNIX System Services sysplex sharing environment with three systems that share two zFS file systems. In this example, the following activity is taking place:

- System SY1 owns the zFS file systems OMVS.TEST1.ZFS (R/W) and OMVS.TEST2.ZFS (R/O).
- The other two systems, SY2 and SY3, have the R/O zFS file system locally mounted as read (R/O).
- Any R/W requests from SY2 and SY3 to the R/W file system owned by SY1 must be passed through the XCF messaging function, and this is referred to as function-shipping requests.

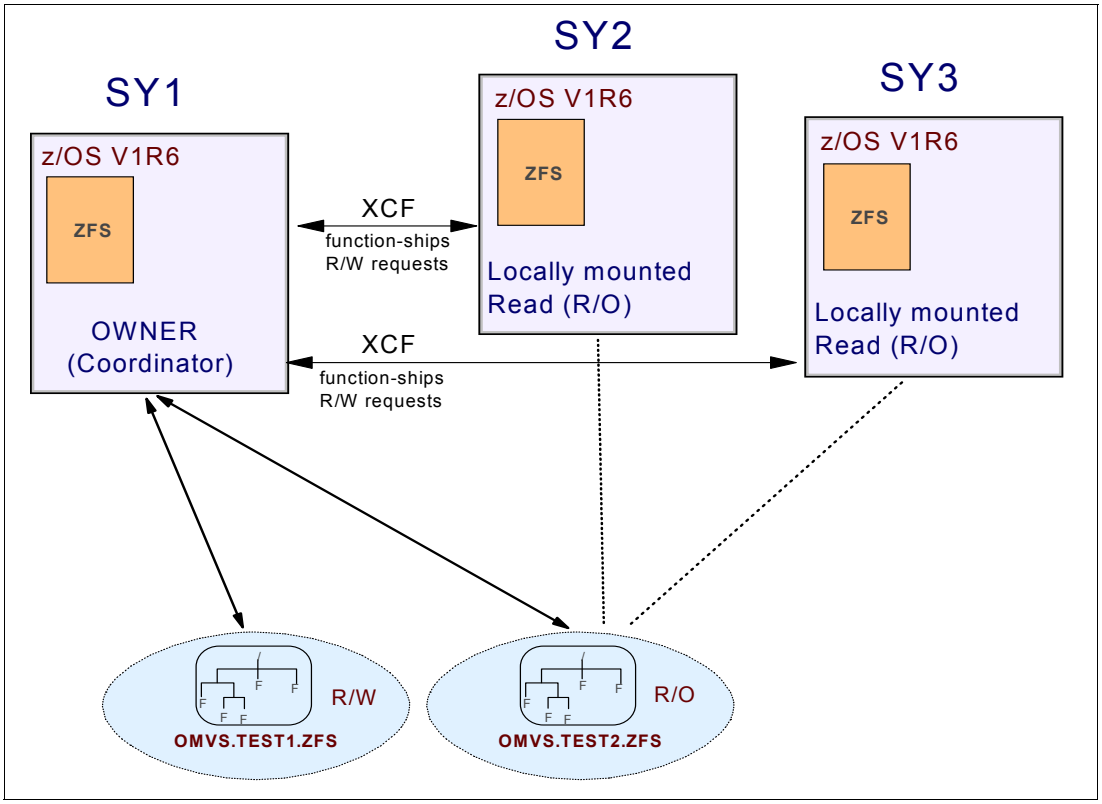


Figure 5-19 Three systems in a UNIX System Services sharing environment accessing two file systems

zFS PFS terminates

The zFS PFS can be terminated by an operator console command or by a zFS PFS abend. When the zFS PFS terminates, the terminating PFS will attempt to move all the file systems it owns to another system in the sysplex, because they are mounted as AUTOMOVE.

When the PFS terminates, there is a system prompt message waiting for a reply to be answered in system SY1; for example:

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

- If the reply to restart the PFS is `I` (do not restart the PFS), then the file systems will be locally unmounted as before.
- If the reply to restart the PFS is `R`, then any sysplex-aware file systems will convert back from function-shipping to local mount. Sysplex-unaware file systems remain function-shipping to the current owner.

Example 1: Before z/OS V1R6

This example describes the processing flow before z/OS V1R6, when a zFS PFS terminates. File systems from the terminating PFS are moved to another system in the sysplex as illustrated in Figure 5-20 on page 215.

1. The file systems owned by SY1, OMVS.TEST1.ZFS (R/W) and OMVS.TEST2.ZFS (R/O) are automoved to SY2. SY2 becomes the new owner.
2. The other two systems, SY2 and SY3, still maintain the R/O zFS file system locally mounted as read (R/O). SY2 is the new owner of the R/O file system.

3. Any R/W request from SY3 to the R/W file system now owned by SY2 is passed through the XCF messaging function, which is referred to as function-shipping requests.
4. All users on the SY1 system that were using the (R/O) and (R/W) file systems on SY1 no longer have any access to them.

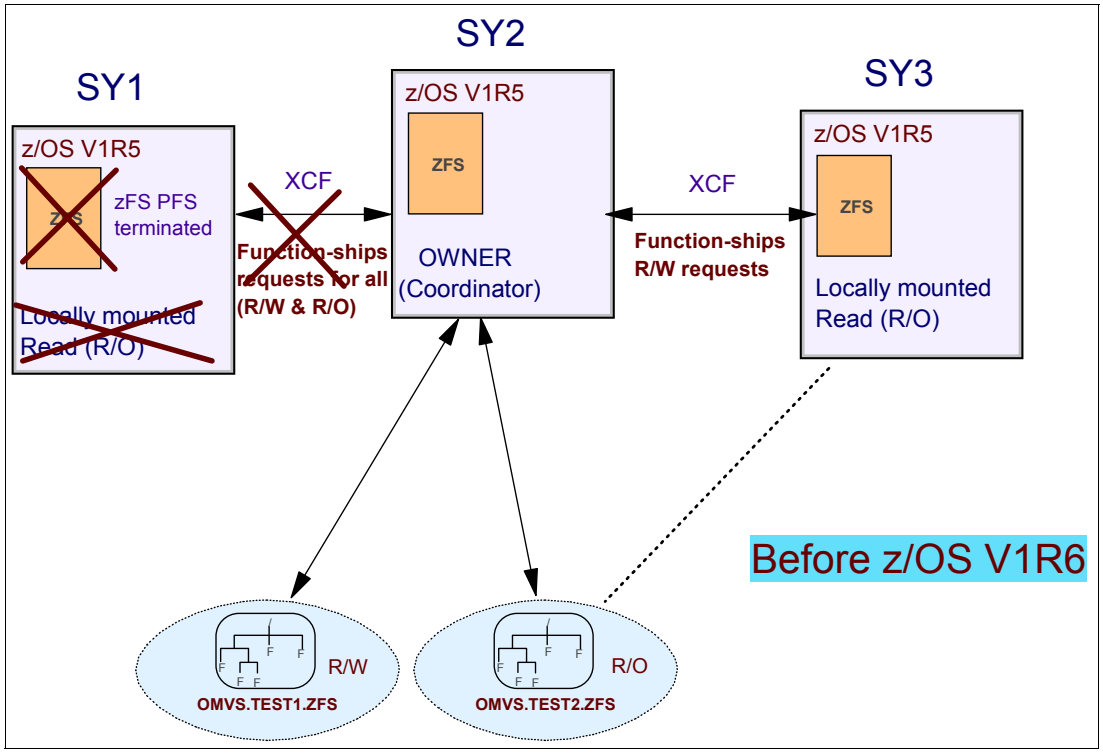


Figure 5-20 For all releases prior to z/OS V1R6 for move of file systems

Example 2: Changes with z/OS V1R6

This example describes the processing flow with z/OS V1R6 when a zFS PFS terminates. File systems from the terminating PFS are moved to another system in the sysplex illustrated in Figure 5-20, as follows:

Figure 5-21 on page 216 shows that SY2 is now the new owner (coordinator) of the two file systems.

1. The file systems owned by SY1, OMVS.TEST1.ZFS (R/W) and OMVS.TEST2.ZFS (R/O) are automoved to SY2. SY2 becomes the new owner.
2. The other two systems, SY2 and SY3, still maintain the R/O zFS file system locally mounted as read (R/O). SY2 is the new owner of the R/O file system.
3. Any R/W request from SY3 to the R/W file system now owned by SY2 is passed through the XCF messaging function, which is referred to as function-shipping requests.
4. All users on the SY1 system that were using the (R/O) and (R/W) file systems on SY1 now have access to them through the XCF messaging function, which is referred to as function-shipping requests.

Note: The XCF function-shipping is also for a R/O file system.

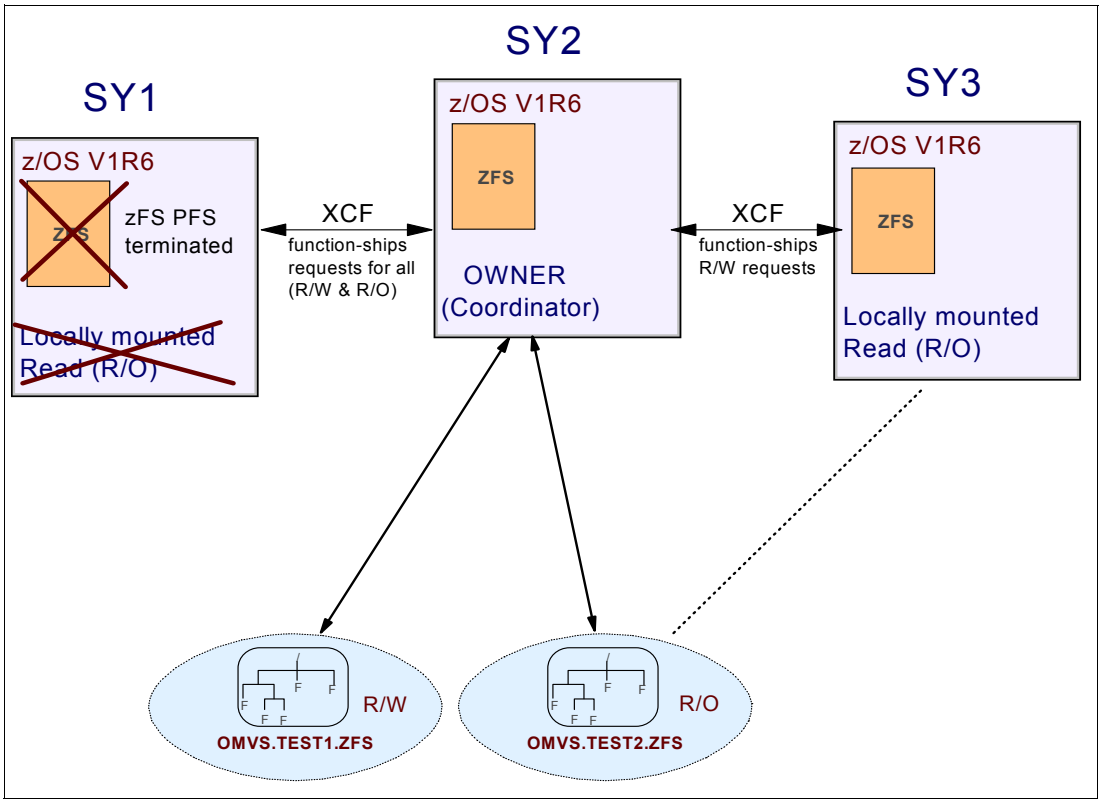


Figure 5-21 The same UNIX System Services sharing environment after changing the ownership to another system

5.4.3 Automove behavior with z/OS releases between V1R6 and V1R8

For sysplex-aware file systems (R/O file systems), the behavior in automove situations is changed with z/OS V1R6. It now has the following characteristics:

- MOUNT allows AUTOMOVE(YES) or AUTOMOVE(UNMOUNT).
- If AUTOMOVE(NO) or if an automove syslist is specified, it changes to AUTOMOVE(YES) and a new message, BPXF234I, is issued.
- Remount does not change the AUTOMOVE setting. Therefore, a remount from R/W to R/O when the AUTOMOVE is NO does not change it to AUTOMOVE(YES), even though it is now sysplex-aware.
- PFS termination ignores AUTOMOVE(NO) or AUTOMOVE(UNMOUNT) if the file system is sysplex-aware, and goes ahead and tries to move ownership and then perform a local to function-ship conversion. This is shown in Figure 5-21.
- A move to any systems in the sysplex (SYSNAME=*) ignores an automove syslist if the file system is sysplex-aware and considers all systems as move candidates. It has always ignored AUTOMOVE(NO) and AUTOMOVE(UNMOUNT) if sysplex-aware.
- Dead system recovery and takeover has always ignored AUTOMOVE(NO) and AUTOMOVE(UNMOUNT) for sysplex-aware, and has still attempted to have all systems try takeover. But it was honoring the automove syslist regardless of sysplex-awareness. It now ignores an automove syslist as well if sysplex-aware, and allows all systems to try to takeover.

- For sysplex-aware file systems, if no system could take it over, then AUTOMOVE(UNMOUNT) unmounts the file system and its subtree. However, for AUTOMOVE(NO), or with a syslist, the file system becomes unowned.

Note: A move of a file system that is either AUTOMOVE(NO) or has an AUTOMOVE syslist to a new z/OS V1R6 owner changes to AUTOMOVE(YES), and issues message BPXF234I. This will be true for manual move, file system Dead System Recovery, and unowned file system takeover processing.

Although MOUNTs will now only allow AUTOMOVE(YES) or AUTOMOVE(UNMOUNT) for sysplex-aware, you can still wind up with a sysplex-aware file system with other AUTOMOVE settings by either having it mounted on a down-level system, or mounted R/W and re-mounted R/O.

The combination of sysplex-aware and syslist will be treated as follows:

- AUTOMOVE(YES) - this will try to move it anywhere. If it cannot, it will turn it into unowned, rather than unmount it.
- Prior to z/OS V1R6, an AUTOMOVE syslist for sysplex-aware behaved like sysplex-unaware (honoring the syslist and causing it to be unmounted if it could not be taken over).

Several examples shown in the following section are based on the changed automove behavior in z/OS V1R6.

Important: The automove changes introduced with z/OS V1R6 caused several problems. Therefore, this was changed again in z/OS V1R9 to have a consistent behavior for all situations.

For more information about automove consistency, refer to IBM Redbooks publications *z/OS Version 1 Release 9 Implementation*, SG24-7427, or to the most current version of *UNIX System Services z/OS Version 1 Release 7 Implementation*, SG24-7035.

5.4.4 More detailed examples for the new LFS support

In the following examples we demonstrate how this new support is exploited when the zFS PFS is terminated in one of the systems participating in a sysplex sharing environment. The sysplex in which these tests were done had four systems: two running with z/OS V1R5, and the other two at level z/OS V1R6.

Mounting zFS file systems with several different mount settings

In the first examples we mount several zFS file systems, using the OMVS shell session with the **`/usr/sbin/mount`** command with different settings, as shown in Figure 5-22 on page 218.

Where:

`-r` This is used to mount the file system R/O.

`-a` This is the AUTOMOVE option followed by the AUTOMOVE specifications, as follows:

unmount

When UNMOUNT is specified for a file system, it indicates that the file system will not be moved and will be unmounted if the file system's owner should fail. This file system and any file systems mounted within its subtree will be unmounted. When

AUTOMOVE is specified for a file system and the file system's owner goes down, AUTOMOVE indicates that ownership of the file system can be automatically moved to another system participating in shared HFS. AUTOMOVE is the default.

yes

When AUTOMOVE is specified for a file system and the file system's owner goes down, AUTOMOVE indicates that ownership of the file system can be automatically moved to another system participating in shared HFS. AUTOMOVE is the default.

no

When NOAUTOMOVE is specified for a file system, this indicates that ownership should not be moved to another system participating in shared HFS if the file system's owner should fail.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.06.00
#> /usr/sbin/mount -a unmount -f ZFSFR.ZFSA.ZFS -t ZFS /z/zfsa
#> /usr/sbin/mount -a no -f ZFSFR.ZFSB.ZFS -t ZFS /z/zfsb
#> /usr/sbin/mount -a yes -f ZFSFR.ZFSC.ZFS -t ZFS /z/zfsc
#> zfsadm clone -filesystem ZFSFR.ZFSC.ZFS
IOEZ00225I File system ZFSFR.ZFSC.ZFS successfully cloned.
#> /usr/sbin/mount -r -a yes -f ZFSFR.ZFSC.ZFS.bak -t ZFS /z/zfsc.clone
#> /usr/sbin/mount -a yes -f ZFSFR.ZFSD.ZFS -t ZFS /z/zfsd
#> zfsadm clone -filesystem ZFSFR.ZFSD.ZFS
IOEZ00225I File system ZFSFR.ZFSD.ZFS successfully cloned.
#> /usr/sbin/chmount -r /z/zfsd
#> /usr/sbin/mount -r -a yes -f ZFSFR.ZFSD.ZFS.bak -t ZFS /z/zfsd.clone
#> /usr/sbin/mount -r -a unmount -f ZFSFR.ZFSE.ZFS -t ZFS /z/zfse
```

Figure 5-22 Mounting several zFS compatibility mode aggregates in a z/OS V1R6 system

Specific information about the aggregates and the file systems contained in them after performing these mounts is shown in Figure 5-23 on page 219.

```
$> zfsadm lsaggr
IOEZ00106I A total of 7 aggregates are attached
ZFSFR.ZFSH.ZFS                                  SC65        R/O
ZFSFR.ZFSE.ZFS                                  SC65        R/O
ZFSFR.ZFSD.ZFS                                  SC65        R/O
ZFSFR.ZFSC.ZFS                                  SC65        R/W
ZFSFR.ZFSB.ZFS                                  SC65        R/W
ZFSFR.ZFSA.ZFS                                  SC65        R/W
OMVS.HERING.TEST.ZFS                            SC65        R/O
$> zfsadm lsfs
IOEZ00370I A total of 7 aggregates are attached.
IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSH.ZFS
ZFSFR.ZFSH.ZFS           RW (Mounted R/O)  1132419 K alloc 1132419 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSE.ZFS
ZFSFR.ZFSE.ZFS           RW (Mounted R/O)     673 K alloc    673 K quota On-line

IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSD.ZFS
ZFSFR.ZFSD.ZFS           RW (Mounted R/O)      24 K alloc    673 K quota On-line
ZFSFR.ZFSD.ZFS.bak       BK (Mounted R/O)     673 K alloc    673 K quota On-line

IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSC.ZFS
ZFSFR.ZFSC.ZFS           RW (Mounted R/W)       8 K alloc 512335 K quota On-line
ZFSFR.ZFSC.ZFS.bak       BK (Mounted R/O)  512335 K alloc 512335 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSB.ZFS
ZFSFR.ZFSB.ZFS           RW (Mounted R/W)  512335 K alloc 512335 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSA.ZFS
ZFSFR.ZFSA.ZFS           RW (Mounted R/W)     673 K alloc    673 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS     RW (Mounted R/O)   10847 K alloc  10847 K quota On-line
Total file systems on-line 9; total off-line 0; total busy 0; total mounted 9
```

Figure 5-23 zFS aggregate and file system information after performing the mounts

Stopping and restarting a zFS PFS

When stopping the zFS PFS in system SC65, as shown in Figure 5-24 on page 220, the following actions occurred:

- The two R/W file systems that are not mounted as automovable are unmounted on every system in the sysplex.
- The two systems, SC63 and SC64, which are running z/OS V1R5, receive messages IKJ56225I and IOEZ00003E when trying to access the R/W zFS aggregate having the R/O clone active.
- No further problem is seen on SC70, as finally it becomes the owner for both the R/W file system and its R/O clone.

```
 SC65 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSH.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSE.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSD.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSC.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSB.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSA.ZFS
 IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
 IOEZ00057I zFS kernel program IOEFSCM is ending
...
 BPXF063I FILE SYSTEM ZFSFR.ZFSA.ZFS WAS SUCCESSFULLY UNMOUNTED.
...
 BPXF063I FILE SYSTEM ZFSFR.ZFSB.ZFS WAS SUCCESSFULLY UNMOUNTED.
...
 IKJ56225I DATA SET ZFSFR.ZFSC.ZFS ALREADY IN USE, TRY LATER+
 IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
 IOEZ00003E While opening minor device 10019, could not open dataset ZFSFR.ZFSC.ZFS.
 IKJ56225I DATA SET ZFSFR.ZFSC.ZFS ALREADY IN USE, TRY LATER+
 IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
 IOEZ00003E While opening minor device 10023, could not open dataset ZFSFR.ZFSC.ZFS.
*015 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO
IGNORE.
```

Figure 5-24 Stopping zFS in system SC65

Because the zFS kernel is not active at this time, the **zfsadm** command interface is not usable, as shown in Figure 5-25.

```
$> zfsadm lsaggr
IOEZ00109E Could not retrieve parameter dataset name from zFS kernel
```

Figure 5-25 zfsadm command interface not usable while zFS kernel not active

Stopping zFS in a z/OS V1R5 system

Note the following:

- When zFS goes down in z/OS V1R6, IOEZ00061I messages are not shown.
- When zFS goes down in a z/OS V1R5 system, these messages *are* shown, as seen in Figure 5-26.

```
SC63 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSI.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system ZFSFR.ZFSI.ZFS.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSH.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system ZFSFR.ZFSH.ZFS.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSG.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system ZFSFR.ZFSG.ZFS.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSF.ZFS
 IOEZ00061I zFS kernel: forcing unmount of file system ZFSFR.ZFSF.ZFS.
...
 IOEZ00057I zFS kernel program IOEFSCM is ending
```

Figure 5-26 Stopping zFS in a z/OS V1R5 system

Note: IOEZ00061I messages are no longer displayed in this situation with z/OS V1R6 because access to zFS file systems in general is still available via XCF function-shipping.

Mounted file systems after zFS stopped on SC65

Figure 5-27 lists the results of all the file system movements.

```
$> df -v /z/zfsc
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsc   (ZFSFR.ZFSC.ZFS)          115520/1140190 4294967275 Available
ZFS, Read/Write, Device:214, ACLS=Y
File System Owner : SC70    .Automove=Y.Client=Y
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSC.ZFS
$> df -v /z/zfsc.clone
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsc.clone (ZFSFR.ZFSC.ZFS.bak)      115520/1140190 4294967275 Available
ZFS, Read Only, Device:215, ACLS=Y
File System Owner : SC70    .Automove=Y.Client=Y
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSC.ZFS
$> df -v /z/zfsd
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsd   (ZFSFR.ZFSD.ZFS)          12748/14094    4294967209 Available
ZFS, Read Only, Device:216, ACLS=Y
File System Owner : SC64    .Automove=Y.Client=Y
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSD.ZFS
$> df -v /z/zfsd.clone
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsd.clone (ZFSFR.ZFSD.ZFS.bak)      12748/14094    4294967209 Available
ZFS, Read Only, Device:217, ACLS=Y
File System Owner : SC70    .Automove=Y.Client=Y
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSD.ZFS
$> df -v /z/zfse
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfse   (ZFSFR.ZFSE.ZFS)          12748/14094    4294967209 Available
ZFS, Read Only, Device:218, ACLS=Y
File System Owner : SC63    .Automove=U.Client=Y
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSE.ZFS
```

Figure 5-27 Listing the results of all the file system movements after stopping zFS

Restarting of zFS on SC65

After restarting zFS system SC65 also sees messages IKJ56225I and IOEZ00003E, as shown in Figure 5-28 on page 222.

```
R 15,R
IEE600I REPLY TO 015 IS;R
IAT6100 ( DEMSEL ) JOB ZFS      (JOB08293), PRTY=15, ID=DFS
ICH70001I DFS      LAST ACCESS AT 17:13:43 ON WEDNESDAY, SEPTEMBER 1, 2004
IEF403I ZFS - STARTED - TIME=17.13.44 - ASID=009A - SC65
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.06.00 Service Level OA07700 - HZFS360.
Created on Fri Jun 11 13:31:26 EDT 2004.
IOEZ00178I SYS1.TEST.IOEFSZFS(IOEFSPRM) is the .configuration dataset currently
IOEZ00055I zFS kernel: initialization complete.
IKJ56225I DATA SET ZFSFR.ZFSC.ZFS ALREADY IN USE, TRY LATER+ -now on SC65!
IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IOEZ00003E While opening minor device 10002, could not open dataset ZFSFR.ZFSC.ZFS.
```

Figure 5-28 Restarting zFS by answering the reply message with R

After restarting zFS in SC65 the R/O file systems are mounted again locally, as shown in Figure 5-29.

```
$> zfsadm lsaggr
IOEZ00106I A total of 4 aggregates are attached
ZFSFR.ZFSH.ZFS                                SC65        R/O
ZFSFR.ZFSE.ZFS                                SC65        R/O
ZFSFR.ZFSD.ZFS                                SC65        R/O
OMVS.HERING.TEST.ZFS                          SC65        R/O
$> zfsadm lsfs
IOEZ00370I A total of 4 aggregates are attached.
IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSH.ZFS
ZFSFR.ZFSH.ZFS           RW (Mounted R/O)  1132419 K alloc 1132419 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSE.ZFS
ZFSFR.ZFSE.ZFS           RW (Mounted R/O)     673 K alloc    673 K quota On-line

IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSD.ZFS
ZFSFR.ZFSD.ZFS           RW (Mounted R/O)      24 K alloc    673 K quota On-line
ZFSFR.ZFSD.ZFS.bak       BK (Mounted R/O)     673 K alloc    673 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS     RW (Mounted R/O)   10847 K alloc  10847 K quota On-line
Total file systems on-line 5; total off-line 0; total busy 0; total mounted 5
```

Figure 5-29 Displaying information for all R/O file systems mounted locally again

Stopping of zFS on SC65

Figure 5-30 on page 223 shows a further stop of zFS in SC65.

```
 SC65 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSH.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSE.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSD.ZFS
 IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
 IOEZ00057I zFS kernel program IOEFSCM is ending
...
*016 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO
IGNORE.
```

Figure 5-30 Stopping zFS in system SC65 again

Replying I to stop of zFS on SC65

Figure 5-31 shows the consequences when I is used to answer the reply message. This causes all zFS file systems in the UNIX System Services sysplex sharing environment to be unmounted on SC65. Message BPXF218I is displayed in parallel, because zFS is no longer active and available on SC65.

```
 R 16,I
 IEE600I REPLY TO 016 IS;I
 BPXF063I FILE SYSTEM ZFSFR.ZFSI.ZFS 729
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF218I ONE OR MORE FILE SYSTEMS DID NOT MOUNT DUE TO INCONSISTENT 730
 FILESYSTYPE STATEMENTS.
 BPXF063I FILE SYSTEM ZFSFR.ZFSH.ZFS 731
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSG.ZFS 732
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSF.ZFS 733
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM OMVS.HERING.TEST.ZFS 734
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSC.ZFS 735
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSC.ZFS.bak 736
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSD.ZFS 737
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSD.ZFS.bak 738
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM ZFSFR.ZFSE.ZFS 739
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM TWS.TWSABIN.RVA.ZFS 740
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM TWS.TWSABIN.ZFS 741
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM TWS.TWSAWRK.ZFS 742
 WAS SUCCESSFULLY UNMOUNTED.
 BPXF063I FILE SYSTEM TWS.TWSAWRK.RVA.ZFS 743
 WAS SUCCESSFULLY UNMOUNTED.
```

Figure 5-31 zFS no longer active when answering the reply message with I

Restarting zFS using the SETOMVS command

When executing a **SETOMVS RESET=** command as shown in Figure 5-33, we also received message BPXF221I. The zFS reason code 6058x says:

"HFS-compatibility mount, - error attaching aggregate or was found not to be HFS-compatibility.**"**.

Note: The **SETOMVS RESET=** command can be used to start the zFS PFS if it has not been started at IPL. It can also be used to restart the zFS PFS if it has been terminated, by replying `i` to the BPXF032D operator message (after stopping the zFS PFS).

BPXPRMZF parmlib member for restart

The parmlib member used to restart zFS is shown in Figure 5-32.

```
FILESYSTYPE TYPE(ZFS)           /* Type of file system to start  */
ENTRYPOINT(IOEFSCM)             /* Entry Point of load module    */
ASNAME(ZFS)                     /* Null PARM for physical file   */
```

Figure 5-32 BPXPRMZF parmlib member with the zFS filesystype definition

The zFS restart processing using the BPXPRMZF parmlib member is shown in Figure 5-33.

```
SETOMVS RESET=(ZF)
IEE252I MEMBER BPXPRMZF FOUND IN SYS1.PARMLIB
IEF196I IEF285I   SYS1.SYSPROG.PARMLIB                          KEPT
IEF196I IEF285I   VOL SER NOS= SBOX01.
IEF196I IEF285I   SYS1.PARMLIB                                  KEPT
IEF196I IEF285I   VOL SER NOS= 037CAT.
IEF196I IEF285I   CPAC.PARMLIB                                  KEPT
IEF196I IEF285I   VOL SER NOS= Z16CAT.
IEF196I IEF285I   SYS1.IBM.PARMLIB                              KEPT
IEF196I IEF285I   VOL SER NOS= Z16CAT.
IAT6100 ( DEMSEL ) JOB ZFS      (JOB08363), PRTY=15, ID=DFS
ICH70001I DFS      LAST ACCESS AT 15:02:26 ON THURSDAY, SEPTEMBER 2, 2004
IEF403I ZFS - STARTED - TIME=15.02.26 - ASID=0099 - SC65
IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
Version 01.06.00 Service Level OA07700 - HZFS360.
Created on Fri Jun 11 13:31:26 EDT 2004.
IOEZ00178I SYS1.TEST.IOEFSZFS(IOEFSPRM) is the .configuration dataset currently in use.
IOEZ00055I zFS kernel: initialization complete.
BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
IKJ56225I DATA SET ZFSFR.ZFSC.ZFS ALREADY IN USE, TRY LATER+
IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IOEZ00003E While opening minor device 10001, could not open dataset ZFSFR.ZFSC.ZFS.
BPXF221I FILE SYSTEM ZFSFR.ZFSC.ZFS.bak FAILED TO MOUNT LOCALLY.
RETURN CODE = 00000004, REASON CODE = EF096058
THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH A MOUNT ON A REMOTE SYSTEM.
```

Figure 5-33 Activating zFS again by using system command SETOMVS RESET=

Note: Beginning with z/OS V1R7, you can use **SET OMVS=(ZS)** instead of **SETOMVS RESET=(ZS)**.

5.4.5 Mounting zFS file systems R/O

When you mount a zFS compatibility mode aggregate R/O and specify the automove setting as NO in a z/OS V1R6 system, as shown in Figure 5-34, the operation is successful and no further message is displayed in the shell environment.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.06.00
#> /usr/sbin/mount -r -a no -f ZFSFR.ZFSA.ZFS -t ZFS /z/zfsa
```

Figure 5-34 Mounting a zFS aggregate R/O with AUTOMOVE=NO in z/OS V1R6

Syslog message

However, in the syslog you see a new message BPXF234I, as shown in Figure 5-35, indicating that the automove setting is changed to YES.

```
BPXF234I FILE SYSTEM ZFSFR.ZFSA.ZFS WAS MOUNTED WITH AUTOMOVE(YES).
```

Figure 5-35 New message BPXF234I

Changing to automove=no

Furthermore, it is not possible to change the automove setting to NO if the owning system of the file system is running at the z/OS V1R6 level; see Figure 5-36.

```
#> /usr/sbin/chmount -a no /z/zfsa
FOMF0504I mount error: 79 119E05E1
EINVAL: The parameter is incorrect
JrAutoMoveable: A filesystem mounted in a mode for which it is capable of being
directly mounted to the PFS on all systems is considered automoveable.
```

Figure 5-36 No longer allowed to set AUTOMOVE=NO for a R/O zFS mounted at z/OS V1R6

Moving the file system to z/OS V1R5 and back to z/OS V1R6

When moving the zFS file system to a z/OS V1R5 system, then switching to AUTOMOVE=NO, and finally moving it back to the z/OS V1R6 system, we found that the setting of AUTOMOVE was not modified again.

-d destsys To designate a specific reassignment, use **-d destsys**, where destsys becomes the logical owner of a file system in a shared HFS environment.

This is shown in Figure 5-37 on page 226.

```
#> /usr/sbin/chmount -d SC63 /z/zfsa
#> /usr/sbin/chmount -a no /z/zfsa
#> df -v /z/zfsa
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsa   (ZFSFR.ZFSA.ZFS)          12748/14094    4294967209 Available
ZFS, Read Only, Device:261, ACLS=Y
File System Owner : SC63        Automove=N      Client=N
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSA.ZFS
#> /usr/sbin/chmount -d SC65 /z/zfsa
#> df -v /z/zfsa
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsa   (ZFSFR.ZFSA.ZFS)          12748/14094    4294967209 Available
ZFS, Read Only, Device:261, ACLS=Y
File System Owner : SC65        Automove=N      Client=N
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSA.ZFS
```

Figure 5-37 Moving a R/O zFS file system to a z/OS V1R5 system and back to z/OS V1R6

In 5.3.1, “Remount processing in sysplex sharing” on page 207, we mention that a remount of a compatibility mode file system forces the aggregate mode to be switched. In general the rule is that mounting a compatibility mode file system R/O forces the aggregate itself to be implicitly attached R/O also.

This is the only reasonable choice to be able to mount the file system locally R/O in the other systems of the sysplex, because the aggregate needs to be attached there also.

Note: It is not possible to move a zFS file system that is mounted R/O to another system, or to mount this file system locally on another system, if the aggregate itself is attached R/W.

A similar effect had been recognized in a situation where a compatibility mode zFS file system was mounted together with its R/O clone at the same time; see message BPXF221I, which is displayed in Figure 5-33 on page 224.

R/O file system in a R/W attached aggregate

In a compatibility mode aggregate, having the only file system mounted R/O while the aggregate is attached R/W will normally not happen or should be avoided. Nevertheless, it is easy to force it, as shown in Figure 5-38 on page 227, if you have a R/W compatibility file system mounted together with its R/O clone. The reason code is the same as described for message BPXF221I in Figure 5-33 on page 224, as follows:

1. The aggregate ZFSFR.ZFSC.ZFS has two file systems, with one of them being the clone.
2. The file system is unmounted, leaving only the R/O clone mounted in the aggregate.
3. A list of the aggregate information is displayed.
4. A list of the file systems in the aggregate is displayed, showing their mount status.
5. A display of the amount of free space in the file system and its status is issued for the clone.
6. Finally, an attempt is made to designate the clone for a specific reassignment by using the **-d destsys** option, where destsys becomes the logical owner of a file system in a shared HFS environment. This fails with a mount error and the messages shown in Figure 5-39 on page 227 appear in the syslog.

```
#> zfsadm lsfs -aggregate ZFSFR.ZFSC.ZFS
IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSC.ZFS
ZFSFR.ZFSC.ZFS           RW (Mounted R/W)       8 K alloc 512335 K quota On-line
ZFSFR.ZFSC.ZFS.bak       BK (Mounted R/O)  512335 K alloc 512335 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
#> /usr/sbin/unmount /z/zfsc
#> zfsadm aggrinfo -aggregate ZFSFR.ZFSC.ZFS
ZFSFR.ZFSC.ZFS (R/W COMP): 57752 K free out of total 576000
#> zfsadm lsfs -aggregate ZFSFR.ZFSC.ZFS
IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSC.ZFS
ZFSFR.ZFSC.ZFS           RW (Not Mounted)       8 K alloc 512335 K quota On-line
ZFSFR.ZFSC.ZFS.bak       BK (Mounted R/O)  512335 K alloc 512335 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 1
#> df -v /z/zfsc.clone
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/zfsc.clone (ZFSFR.ZFSC.ZFS.bak)      115520/1140190 4294967275 Available
ZFS, Read Only, Device:284, ACLS=Y
File System Owner : SC65        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : ZFSFR.ZFSC.ZFS
#> /usr/sbin/chmount -d SC70 /z/zfsc.clone
FOMF0504I mount error: 4 EF096058
```

Figure 5-38 Not being able to move a R/O file system if aggregate is attached R/W

The messages IKJ56225I and IOEZ00003E are displayed in the syslog again for system SC70, as shown in Figure 5-39.

```
IKJ56225I DATA SET ZFSFR.ZFSC.ZFS ALREADY IN USE, TRY LATER+
IKJ56225I DATA SET IS ALLOCATED TO ANOTHER JOB OR USER
IOEZ00003E While opening minor device 10022, could not open dataset ZFSFR.ZFSC.ZFS.
```

Figure 5-39 Messages IKJ56225I and IOEZ00003E displayed for system SC70

Note: If a compatibility mode file system is mounted R/W together with its R/O clone, data from the cloned file system is only read via XCF function-shipping from a remote system.

5.5 Effects on applications having zFS files open

Tests were performed to see the effects on applications that had files opened in a zFS file system when the zFS PFS is stopped on the system that current owns the zFS file system.

Running on a z/OS V1R6 system, there were no essential differences whether zFS was going down on that system, a system in the sysplex sharing was stopped, or a system in the sysplex sharing went down completely. Because the impact is likely to be greater if the file system is owned locally and zFS is stopped there, we only discuss that situation here.

In the examples and excerpts of a series of commands, we used a small UNIX REXX script named “rexx” that allowed REXX, UNIX System Services shell, and SYSCALL commands to run interactively. This provided an easy way to demonstrate the interaction of UNIX commands on an application.

We also used another UNIX REXX script named “swsu” to switch to SU (superuser) mode for running just one command. This allowed the application to run with a UID that was not zero and to run commands needing superuser authorization when needed.

These two REXX procedures were taken from the UNIX System Services Tools package, which is available at the IBM z/OS UNIX Tools disk. For more information, refer to B.8, “SU procedure” on page 397.

Note: Moving a file system manually to a new owning system has no impact on an application having files opened within that file system.

5.5.1 Small application opening files in a zFS file system

The first series of commands, shown in Figure 5-40, is used to demonstrate the previous statement, as follows:

1. A display of the amount of free space in the file system and its status is issued for the file system OMVS.HERING.TEST.
2. The REXX script is invoked and is followed by a series of Syscall and SH commands to the file system.

```
$> echo System $(sysvar SYSNAME) running $(uname -I) Version $(uname -Iv).$(uname -Ir)
System SC65 running z/OS Version 01.06.00
$> id
uid=888(HERING) gid=2(SYS1) groups=1047(USSTEST)
$> df -v /u/hering/test
Mounted on     Filesystem                Avail/Total    Files      Status
/u/hering/test (OMVS.HERING.ZFS)         1132/1150      4294967292 Available
ZFS, Read/Write, Device:234, ACLS=Y
File System Owner : SC65        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.HERING.ZFS
$> rexx
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "open /u/hering/test/testfile" o_creat+o_trunc+o_wronly 644
OMVS Return Value (retval) = 3
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
linedata = "This is a sample line."||esc_n
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 3 linedata"
OMVS Return Value (retval) = 23
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "swsu /usr/sbin/chmount -d SC70 /u/hering/test"
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 3 linedata"
OMVS Return Value (retval) = 23
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "swsu /usr/sbin/chmount -d SC65 /u/hering/test"
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "zfsadm lsfs -aggregate OMVS.HERING.ZFS"
IOEZ00129I Total of 1 file systems found for aggregate OMVS.HERING.ZFS
OMVS.HERING.ZFS          RW (Mounted R/W)       9 K alloc      9 K quota On-line
Total file systems on-line 1; total off-line 0; total busy 0; total mounted 1
...
```

Figure 5-40 Moving file systems with open files to new owning systems

Stopping zFS

When stopping zFS in system SC65, no file system is unmounted and all are still accessible while the zFS PFS is waiting to restart. All the file systems previously mounted on system SC65 now have a new owner; see Figure 5-41.

```
 SC65 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate OMVS.HERING.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSD.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSE.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSH.ZFS
 IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
 IOEZ00057I zFS kernel program IOEFSCM is ending
...
*022 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO
IGNORE.
```

Figure 5-41 Stopping zFS in system SC65

Status of application after stopping zFS

In this situation an application trying to run an I/O operation to an open file located in a zFS file system previously owned by SC65 is likely to suffer a specific I/O error; see Figure 5-42.

```
...
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 3 linedata"
OMVS Return Value (retval) = -1
OMVS Return Code (errno )  = 7A
OMVS Reason Code (errnojr) = 52C00B6
OMVS Return Code Explanation - EIO: An I/O error occurred
OMVS Reason Code Explanation - JRPfsSuspend: The PFS is waiting to restart.
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "close 3"
OMVS Return Value (retval) = 0
...
```

Figure 5-42 Application with an I/O error on an open file when zFS is stopped

Restarting zFS

When attempting commands to the application, the same is true when zFS is restarted but with a different error condition; see Figure 5-43 on page 230.

```
...
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 3 linedata"
OMVS Return Value (retval) = -1
OMVS Return Code (errno )  = 7A
OMVS Reason Code (errnojr) = 52C04B8
OMVS Return Code Explanation - EIO: An I/O error occurred
OMVS Reason Code Explanation - JrSysplexDataSyncLost: The I/O request is rejected
because the file integrity was lost due to the failure of the file system server.
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "close 3"
OMVS Return Value (retval) = -1
OMVS Return Code (errno )  = 7A
OMVS Reason Code (errnojr) = 52C04B8
OMVS Return Code Explanation - EIO: An I/O error occurred
OMVS Reason Code Explanation - JrSysplexDataSyncLost: The I/O request is rejected
because the file integritywas lost due to the failure of the file system server.
...
```

Figure 5-43 Application with an I/O error on an open file when zFS is restarted again

Suggestion: For an application suffering a specific I/O error caused by a stop of the owning zFS PFS, the application should close that file to get rid of the file descriptor (whether the close is successful or not) and open it again to continue processing.

This suggestion applies to both situations, as referenced in Figure 5-42 on page 229 and Figure 5-43.

5.5.2 Stopping zFS being transparent for applications in z/OS V1R8

Figure 5-44 demonstrates that the enhanced zFS stop behavior, which is controlled by OMVS, avoids the problems that were encountered with the old stop command. Figure 5-44 shows again an application opening files in a zFS file system.

```
$> echo System $(sysvar SYSNAME) running $(uname -I) Version $(uname -Iv).$(uname -Ir)
System SC70 running z/OS Version 01.10.00
$> df -v /u/hering/test | grep "File System Owner"
File System Owner : SC70        Automove=Y      Client=N
$> df -v /u/hering/test | grep "Aggregate Name"
Aggregate Name : OMVS.HERING.ZFS
$> zfsadm lsfs OMVS.HERING.ZFS -fast
OMVS.HERING.ZFS
$> rexx
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "open /u/hering/test/testfile" o_creat+o_trunc+o_wronly 644
OMVS Return Value (retval) = 4
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
linedata = "This is a sample line."||esc_n
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 4 linedata"
OMVS Return Value (retval) = 23
...
```

Figure 5-44 Opening and writing to a zFS file in system SC70

Stopping zFS in z/OS V1R8 or above

Figure 5-45 shows the changed zFS stop command introduced in z/OS V1R8. The old interface does not work anymore.

```
 P ZFS
 IOEZ00523I zFS no longer supports the stop command. Please issue f omvs,stoppfs=zfs
 F OMVS,STOPPFS=ZFS
*753 BPXI078D STOP OF ZFS REQUESTED. REPLY 'Y' TO PROCEED. ANY OTHER
 REPLY WILL CANCEL THIS STOP.
...
 R 753,Y
 IEE600I REPLY TO 753 IS;Y
 IOEZ00048I Detaching aggregate OMVS.HERING.ZFS
...
 IOEZ00048I Detaching aggregate OMVS.PELEG.HFS
 IOEZ00048I Detaching aggregate HFS.ZOSR19.VARWBEM
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00388I Aggregate takeover being attempted for aggregate HFS.ZOSR1A.Z1ARD1.ROOT
 IOEZ00044I Aggregate HFS.ZOSR1A.Z1ARD1.ROOT attached successfully.
 IOEZ00416I Aggregate HFS.ZOSR1A.Z1ARD1.ROOT moved to system SC64 at shutdown.
 IOEZ00048I Detaching aggregate HFS.ZOSR19.Z19RC1.SHPJROOT
...
 IOEZ00048I Detaching aggregate OMVS.HERING.ZFS
...
 IOEZ00057I zFS kernel program IOEFSCM is ending
...
*754 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO
IGNORE.
```

Figure 5-45 Stopping zFS in system SC70

Status of application after stopping zFS

The new stop processing is coordinated by OMVS. Thus, OMVS first tries to move the zFS file systems mounted on SC70 to a new owning system, depending on the automove setting. When this is complete, the zFS PFS is stopped.

Figure 5-46 shows that the new stop processing is transparent for applications, just as normal moves of file systems in older releases were already.

```
...
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "df -v /u/hering/test | grep 'File System Owner'"
File System Owner : SC65        Automove=Y      Client=Y
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 4 linedata"
OMVS Return Value (retval) = 23
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
...
```

Figure 5-46 zFS stop being transparent for the application

Restarting zFS

After restarting zFS and moving back the file system to SC70, there is no new problem; see Figure 5-47 on page 232.

```
...
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "swsu chmount -d SC70 /u/hering/test"
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
sh "df -v /u/hering/test | grep 'File System Owner'"
File System Owner : SC70        Automove=Y      Client=N
REXX - Enter SH, Syscall, TSO, rexx commands or EXIT:
s "write 4 linedata"
OMVS Return Value (retval) = 23
...
```

Figure 5-47 zFS restart and file system move back to SC70

Beginning with z/OS V1R8, stopping and restarting zFS in a system owning a zFS file system is transparent for applications using this file system.

5.6 Multifile system aggregates behavior when zFS stops

In this section, we show that there is no way to attach multifile system aggregates in R/W mode and mount file systems in them successfully with AUTOMOVE=YES.

Assume we have a multifile system aggregate attached to zFS in a z/OS V1R6 system. We have two file systems mounted on that system, as shown in Figure 5-48, where one file system is mounted R/W and the other one is mounted R/O.

```
#> echo System $(sysvar SYSNAME) running $(uname -I) Version $(uname -Iv).$(uname -Ir)
System SC65 running z/OS Version 01.06.00
#> zfsadm aggrinfo -aggregate OMVS.TEST.MULTIFS.ZFS
OMVS.TEST.MULTIFS.ZFS (R/W MULT): 20597 K free out of total 20880
#> zfsadm lsfs -aggregate OMVS.TEST.MULTIFS.ZFS
IOEZ00129I Total of 2 file systems found for aggregate OMVS.TEST.MULTIFS.ZFS
OMVS.TEST.MULTIFS.FS01.ZFS RW (Mounted R/W)       9 K alloc      9 K quota On-line
OMVS.TEST.MULTIFS.FS02.ZFS RW (Mounted R/O)       9 K alloc      9 K quota On-line
Total file systems on-line 2; total off-line 0; total busy 0; total mounted 2
#> df -v /z/test01
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/test01 (OMVS.TEST.MULTIFS.FS01.ZFS) 11982/12000    4294967292 Available
ZFS, Read/Write, Device:274, ACLS=Y
File System Owner : SC65        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.TEST.MULTIFS.ZFS
#> df -v /z/test02
Mounted on     Filesystem                Avail/Total    Files      Status
/SC65/z/test02 (OMVS.TEST.MULTIFS.FS02.ZFS) 11982/12000    4294967292 Available
ZFS, Read Only, Device:275, ACLS=Y
File System Owner : SC65        Automove=Y      Client=N
Filetag : T=off   codeset=0
Aggregate Name : OMVS.TEST.MULTIFS.ZFS
```

Figure 5-48 Two multifile system aggregate file systems mounted on system SC65

5.6.1 Stopping zFS

If zFS is stopped in that system, then both file systems are unmounted in all systems of the sysplex, as shown in Figure 5-49 on page 233. This is normal processing and it emphasizes

that you cannot use these types of file systems and mount them as automovable. It does not matter whether you mount them with AUTOMOVE=YES or not; they will not be automoved.

```
 SC65 P ZFS
 IOEZ00050I zFS kernel: Stop command received.
 IOEZ00048I Detaching aggregate ZFSFR.ZFSI.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSH.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSG.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSF.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSE.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSD.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSC.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSB.ZFS
 IOEZ00048I Detaching aggregate ZFSFR.ZFSA.ZFS
 IOEZ00048I Detaching aggregate OMVS.TEST.MULTIFS.ZFS
 IOEZ00048I Detaching aggregate OMVS.HERING.TEST.ZFS
 IOEZ00057I zFS kernel program IOEFSCM is ending
...
 BPXF063I FILE SYSTEM OMVS.TEST.MULTIFS.FS02.ZFS WAS SUCCESSFULLY UNMOUNTED.
...
 BPXF063I FILE SYSTEM OMVS.TEST.MULTIFS.FS01.ZFS WAS SUCCESSFULLY UNMOUNTED.
...
*031 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO RESTART. REPLY 'I' TO
IGNORE.
```

Figure 5-49 Unmounting of R/W multifile system aggregate file systems on STOP ZFS

5.7 Deny mount of multifile system aggregate

zFS file systems contained in multifile system aggregates cannot be mounted in a shared file system environment on a z/OS V1R8 system as of the z/OS V1R8 release. Any data in those file systems must be copied into a compatibility mode aggregate.

Important: This must be done using a system prior to z/OS V1R8 or a non-shared file system environment.

In a shared file system environment, with z/OS V1R8 and z/OS V1R7 (or earlier) systems, a file system contained in a multifile system aggregate can be mounted on a z/OS V1R7 (or earlier) system and the z/OS V1R8 system can access that file system via z/OS UNIX function shipping, as shown in Figure 5-50 on page 234.

However, the file system *cannot* be mounted on the z/OS V1R8 system by any of the following means:

- Via AUTOMOVE on system failure
- Explicit move (**chmount** command)
- Automount
- System shutdown

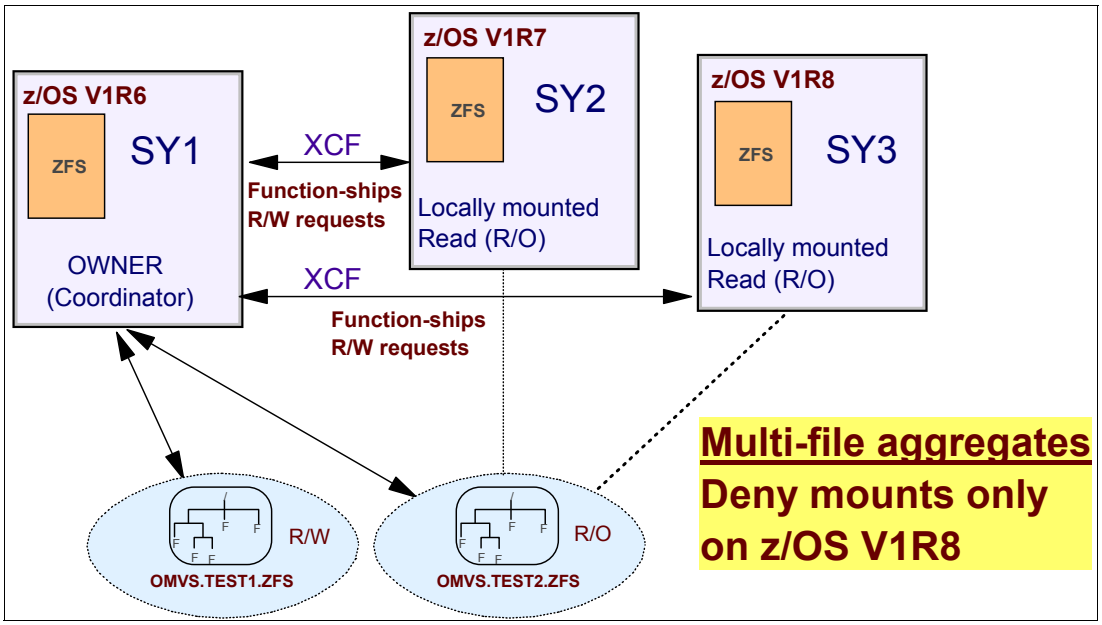


Figure 5-50 Three-system sysplex in a shared file system environment

Deny mount message

When you try to mount a multifile system on a z/OS V1R8 system in a shared environment, you will receive the following error message:

```
IOEZ00522I Filesystem LUTZ.MULTI.ZFS is not in a compat aggregate.  It cannot
be mounted
```

Important: Before you migrate to z/OS V1R8, we strongly recommend that you migrate all multifile aggregates to a compatibility mode aggregate.

Aggregates in a non-shared file system environment

You can still create and format a multifile system, but you will receive a warning message like the one shown in Figure 5-51.

```
SYSPROG @ SC75:/u/lutz>zfsadm format -aggregate LUTZ.MULTI.ZFS
IOEZ00552I Multi-file system aggregates are restricted and support will be
removed; plan to migrate.
Done.  LUTZ.MULTI.ZFS is now a zFS aggregate.
```

Figure 5-51 Warning message for a multifile system aggregate with z/OS V1R8

If you are in a non-shared environment, you can still mount a multifile aggregate.

5.8 Sysplex awareness of the zfsadm command interface

Previously, **zfsadm** commands (and the pfsctl APIs) could only display or change information that is located or owned on the current member of the sysplex. Therefore, they must be issued from the owning system. This required you to know which system was the owner.

The new function provides the ability to issue zFS file system commands from anywhere within a sysplex and to allow quiescing of zFS aggregates from any LPAR in a sysplex.

In previous z/OS levels, a particular problem is quiescing a zFS aggregate. When DFSMS is used to back up a zFS aggregate, the aggregate is quiesced by DFSMS before the backup. For this quiesce to be successful, it must be issued on the system that owns the aggregate. If it is issued on any other system in the sysplex, the quiesce fails and the backup fails, too.

It is a problem to issue the backup on the owning system because ownership can change at any time as a result of an operator command or system failure. See 4.3.5, “Restrictions on quiescing zFS aggregates” on page 181 for more information about this topic.

In z/OS V1R7, if you are in a shared file system environment, **zfsadm** commands and zFS pfsctl APIs generally work across the sysplex.

There are two main changes:

- The **zfsadm** commands act globally and report or modify all zFS objects across the sysplex.
- They support a new **–system** option that limits output to zFS objects on a single system or sends a request to a single system.

5.8.1 New zfsadm command functions

In reference to single **zfsadm** commands, the new functions can be summarized as follows:

- The **zfsadm** commands are going to, and display information about, *all* members of the sysplex.
  - zfsadm aggrinfo, zfsadm clonesys, zfsadm lsaggr, zfsadm lsfs
- The **zfsadm** commands are going to the correct system automatically.
  - zfsadm aggrinfo, zfsadm clone, zfsadm create, zfsadm delete
  - zfsadm grow, zfsadm lsfs, zfsadm lsquota, zfsadm quiesce
  - zfsadm rename, zfsadm setquota, zfsadm unquiesce
- The **zfsadm** commands can be limited or directed to a specific system.
  - zfsadm aggrinfo, zfsadm attach, zfsadm clonesys, zfsadm config
  - zfsadm configquery, zfsadm define, zfsadm detach, zfsadm format
  - zfsadm lsaggr, zfsadm lsfs, zfsadm query

Exploiting the zfsadm sysplex interface

With this sysplex awareness, information can be displayed for all aggregates in the sysplex on systems running at least z/OS V1R7. This is shown in Figure 5-52 on page 236 using **zfsadm lsaggr** and exploiting the pfsctl interface.

For more information about rxlsaggr, which uses an API function, see 2.19, “zFS application programming interfaces” on page 97. The procedure itself is listed in Example B-12 on page 410.

```
$> zfsadm lsaggr
IOEZ00106I A total of 10 aggregates are attached
ZFSFR.ZFSB.ZFS                                SC65      R/O
ZFSFR.ZFSA.ZFS                                SC70      R/O
OMVS.HERING.ZFS                               SC65      R/W
ZFSFR.ZFSC.ZFS                                SC70      R/W
ZFSFR.ZFSH.ZFS                                SC65      R/W
ZFSFR.ZFSG.ZFS                                SC65      R/W
ZFSFR.ZFSF.ZFS                                SC65      R/W
ZFSFR.ZFSE.ZFS                                SC65      R/W
ZFSFR.ZFSD.ZFS                                SC65      R/W
OMVS.HERING.TEST.ZFS                          SC70      R/W
$> rxlsaggr
A total of 10 aggregates are attached.
ZFSFR.ZFSB.ZFS                                SC65
ZFSFR.ZFSA.ZFS                                SC70
OMVS.HERING.ZFS                               SC65
ZFSFR.ZFSC.ZFS                                SC70
ZFSFR.ZFSH.ZFS                                SC65
ZFSFR.ZFSG.ZFS                                SC65
ZFSFR.ZFSF.ZFS                                SC65
ZFSFR.ZFSE.ZFS                                SC65
ZFSFR.ZFSD.ZFS                                SC65
OMVS.HERING.TEST.ZFS                          SC70
```

Figure 5-52 Displaying zFS aggregate information

You can route commands to a specific system, as shown in Figure 5-53 on page 237.

```
$> zfsadm lsaggr -system sc65
IOEZ00106I A total of 7 aggregates are attached
ZFSFR.ZFSD.ZFS                                SC65      R/W
ZFSFR.ZFSE.ZFS                                SC65      R/W
ZFSFR.ZFSF.ZFS                                SC65      R/W
ZFSFR.ZFSG.ZFS                                SC65      R/W
ZFSFR.ZFSH.ZFS                                SC65      R/W
OMVS.HERING.ZFS                               SC65      R/W
ZFSFR.ZFSB.ZFS                                SC65      R/O
$> zfsadm lsaggr -system sc64
IOEZ00363E The system name SC64 is not known.
$> zfsadm aggrinfo -system sc70
IOEZ00368I A total of 3 aggregates are attached to system SC70.
OMVS.HERING.TEST.ZFS (R/W COMP): 528 K free out of total 11520
ZFSFR.ZFSC.ZFS (R/W COMP): 57752 K free out of total 576000
ZFSFR.ZFSA.ZFS (R/O COMP): 6374 K free out of total 7200
$> zfsadm lsfs -system sc70
IOEZ00368I A total of 3 aggregates are attached to system SC70.
IOEZ00129I Total of 1 file systems found for aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS     RW (Mounted R/W)   10847 K alloc  10847 K quota On-line

IOEZ00129I Total of 2 file systems found for aggregate ZFSFR.ZFSC.ZFS
ZFSFR.ZFSC.ZFS           RW (Mounted R/W)       8 K alloc 512335 K quota On-line
ZFSFR.ZFSC.ZFS.bak       BK (Not Mounted)  512335 K alloc 512335 K quota On-line

IOEZ00129I Total of 1 file systems found for aggregate ZFSFR.ZFSA.ZFS
ZFSFR.ZFSA.ZFS           RW (Mounted R/O)     673 K alloc    673 K quota On-line
Total file systems on-line 4; total off-line 0; total busy 0; total mounted 3
```

Figure 5-53 Listing aggregate information for a specific system

Alternatively, you can route commands automatically to the owning system, as shown in Figure 5-54.

Note: The **zfsadm** commands cannot be routed to systems not running at z/OS V1R7.

```
#> echo This is system $(sysvar SYSNAME).
This is system SC65.
#> zfsadm lsaggr -system sc70
IOEZ00106I A total of 3 aggregates are attached
OMVS.HERING.TEST.ZFS                          SC70      R/W
ZFSFR.ZFSC.ZFS                                SC70      R/W
ZFSFR.ZFSA.ZFS                                SC70      R/O
#> zfsadm aggrinfo -aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS (R/W COMP): 528 K free out of total 11520
#> zfsadm quiesce -aggregate OMVS.HERING.TEST.ZFS
IOEZ00163I Aggregate OMVS.HERING.TEST.ZFS successfully quiesced
#> zfsadm aggrinfo -aggregate OMVS.HERING.TEST.ZFS
OMVS.HERING.TEST.ZFS (R/W COMP QUIESCED): 528 K free out of total 11520
```

Figure 5-54 Listing aggregate information and running a zfsadm command on another system

Note: Because **zfsadm** command forwarding allows the quiesce to be issued from any member of the sysplex, DFSMS backup (ADRDSSU) will automatically exploit this function. Therefore, backups can now be issued from any member of the sysplex.

5.8.2 The zFS sysplex group

There are no installation requirements for this support. However, there are configuration dependencies. Whether **zfsadm** commands act globally across the sysplex depends on the BPXPRMxx SYSPLEX option. If SYSPLEX(YES) is specified, then you are in a shared file system environment and **zfsadm** commands will act globally. The zFS tasks are using XCF connections to talk to each other in this situation. If the BPXPRMxx SYSPLEX option specifies SYSPLEX(NO), then **zfsadm** commands will not work globally.

There is a new **zfsadm** command available in z/OS V1R7 that lists all the systems in the same zFS sysplex group; see Figure 5-55.

```
$> zfsadm lssys
IOEZ00361I A total of 2 systems are in the XCF group for zFS
SC65
SC70
```

Figure 5-55 Listing all systems in the zFS sysplex group

A new **zfsadm configquery** option **-group** displays the XCF group used by zFS for communication between sysplex members; see Figure 5-56.

```
$> zfsadm configquery -group
IOEZ00317I The value for configuration option -group is IOEZFS.
```

Figure 5-56 Displaying the name of the zFS XCF group

The value of the zFS XCF group can be set in the zFS parameter file as shown here:

```
group=zFS_groupname
```

IOEZFS is the default value if you do not code the option.

Furthermore, the option **-sysplex_state** displays the sysplex state of zFS. A value of **0** indicates that zFS is not in a shared file system environment. A value of **1** indicates that it is in a shared file system environment; see Figure 5-57.

```
$> zfsadm configquery -sysplex_state
IOEZ00317I The value for configuration option -sysplex_state is 1.
```

Figure 5-57 Displaying the sysplex state of zFS

A new XCF trace table was introduced with z/OS V1R7. The size of this table can be set in the zFS parameter file as shown here:

```
xcf_trace_table_size=nnM
```

The default value is 4 M. If you add the statement, then a number from 1 M up to 2048 M can be specified.

If you have two systems, SC65 and SC70, running at z/OS V1R7, you normally use the same zFS XCF group name to have all the zFS interfaces working globally within the group. If zFS in SC65 is up and running already, you will see new messages when zFS in SC70 is starting and trying to contact the partner zFS in SC65; see Figure 5-58.

```
 IEF403I ZFS - STARTED - TIME=18.59.13 - ASID=008F - SC70
...
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
 Version 01.07.00 Service Level 0000000 - HZFS370.
 Created on Mon Apr  4 16:06:17 EDT 2005.
...
 IOEZ00350I Successfully joined group IOEZFS
*IOEZ00525I Starting initialization with SC65
 IOEZ00529I Preparing for initialization with SC70. <===== SC65
 IOEZ00530I Ready to initialize with SC70. <============== SC65
*IOEZ00526I Requesting aggregate information from SC65
 IOEZ00532I Sending aggregate information to SC70. <====== SC65
...
 IOEZ00528I Initialization with SC65 complete.
 IOEZ00533I Done initializing with SC70. <================ SC65
...
 IOEZ00055I zFS kernel: initialization complete.
```

Figure 5-58 Syslog messages on zFS start up in a second z/OS V1R7 system

Note: The messages marked with SC65 are coming from system SC65. It is an interactive conversation between the systems in the same zFS XCF group.

However, if you restart zFS in one of the systems with a different group name, you lose these connections. Thus, the zFS tasks do not know about each other, although the sysplex state is still showing that you are running in a shared file system environment; see Figure 5-59 and Figure 5-60 on page 240.

```
 IEF403I ZFS - STARTED - TIME=18.04.02 - ASID=00BA - SC65
...
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
 Version 01.07.00 Service Level 0000000 - HZFS370.
 Created on Mon Apr  4 16:06:17 EDT 2005.
...
 IOEZ00350I Successfully joined group IOEZFS1
...
 IOEZ00055I zFS kernel: initialization complete.
...
```

Figure 5-59 Starting zFS in system SC65 with zFS group name set to IOEZFS1

Figure 5-60 on page 240 shows that SC70 is no longer known to zFS in SC65. Both systems act independently, as was the case in the previous z/OS release levels.

```
$> zfsadm configquery -sysplex_state
IOEZ00317I The value for configuration option -sysplex_state is 1.
$> zfsadm configquery -group
IOEZ00317I The value for configuration option -group is IOEZFS1.
$> zfsadm lsaggr -system SC70
IOEZ00363E The system name SC70 is not known.
$> zfsadm lssys
IOEZ00361I A total of 1 systems are in the XCF group for zFS
SC65
```

Figure 5-60 zFS in SC65 and SC70 being in different zFS XCF groups

5.9 zFS file system sharing enhancements in z/OS V1R11 base

With the zFS file system sharing enhancements in z/OS V1R11, the following terms are introduced:

Catch-up mount When a file system mount is successful on a system in a shared file system environment, z/OS UNIX automatically issues a corresponding local mount, the catch-up mount, to every other system's PFS that is running sysplex-aware for that mode (read/write or read/only).

If the corresponding local mount is successful, then z/OS UNIX does not function ship from that system to the z/OS UNIX owning system when that file system is accessed. Instead, the file request is sent directly to the local PFS. This is sometimes referred to as Client=N.

If the corresponding local mount is unsuccessful (for instance, DASD is not accessible from that system), then z/OS UNIX function ships requests to the z/OS UNIX owning system when that file system is accessed (message BPXF221I might be issued). This is sometimes referred to as Client=Y.

Local mount A local mount means that z/OS UNIX issues a successful mount to the local PFS. This is performed when either the PFS is running sysplex-aware for that mode (read/write or read/only) or the system is the z/OS UNIX owner.

Sysplex-unaware A file system is sysplex-unaware (or non-sysplex aware) if the PFS supporting that file system requires it to be accessed through the remote owning system from all other systems in a sysplex (allowing only one connection for update at a time) for a particular mode (read/only or read/write).

Access from a system that is not the owner (sometimes referred to as being a client) is provided through XCF communication to the owning system (sometimes called function shipping). This is controlled by z/OS UNIX and the interface is also known as a PFS named XPFS (Cross-System PFS).

Sysplex-aware A PFS that is sysplex-aware for a particular mode (read/only or read/write) allows file requests for file systems that are mounted in that mode to be handled by the local PFS. Furthermore, the file systems are locally mounted on that system. z/OS UNIX does not function ship those file requests to the z/OS UNIX owning system.

zFS namespace Starting with z/OS V1R7, zFS has supported administration commands and APIs that are sysplex-wide in a shared file system

environment. zFS communicates between sysplex members using XCF protocols.

The zFS XCF protocol exchanges information among members about zFS ownership and other attributes of zFS mounted file systems. This information, which is kept in the memory of each zFS member, is called the zFS namespace.

For more details about terminology see the chapter “zSeries File System (zFS) Overview” in *z/OS Distributed File Service zSeries File System Administration*, SC24-5989.

5.9.1 Main file system sharing concepts before z/OS V1R11

Prior to z/OS V1R11, in a UNIX System Services file system sharing environment a concept of *file system ownership* exists, as illustrated in Figure 5-61 on page 242. In the figure, SC65 is the owning system of a read/write (R/W) and read/only (R/O) file system; that is, the sysplex member system that owns the file system is the first system that processes the mount. This system always accesses the read-write (R/W) file system locally. The other systems do not access the R/W file system directly, although each system can access the R/O file system directly.

However, having to access a file system mounted as R/W from a system that is not the file system owner means that the request must be addressed forward to the owning system. Other non-owning systems in the sysplex, SC64 and SC63, access the R/W file system by issuing a function-ship request to the owning system using XCF. Prior to z/OS V1R11, this support is sysplex-unaware.

Note: For file systems that are not sysplex-aware, the file system owner is the system to which z/OS UNIX forwards file requests. 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 multifile system aggregates beginning with z/OS V1R8).

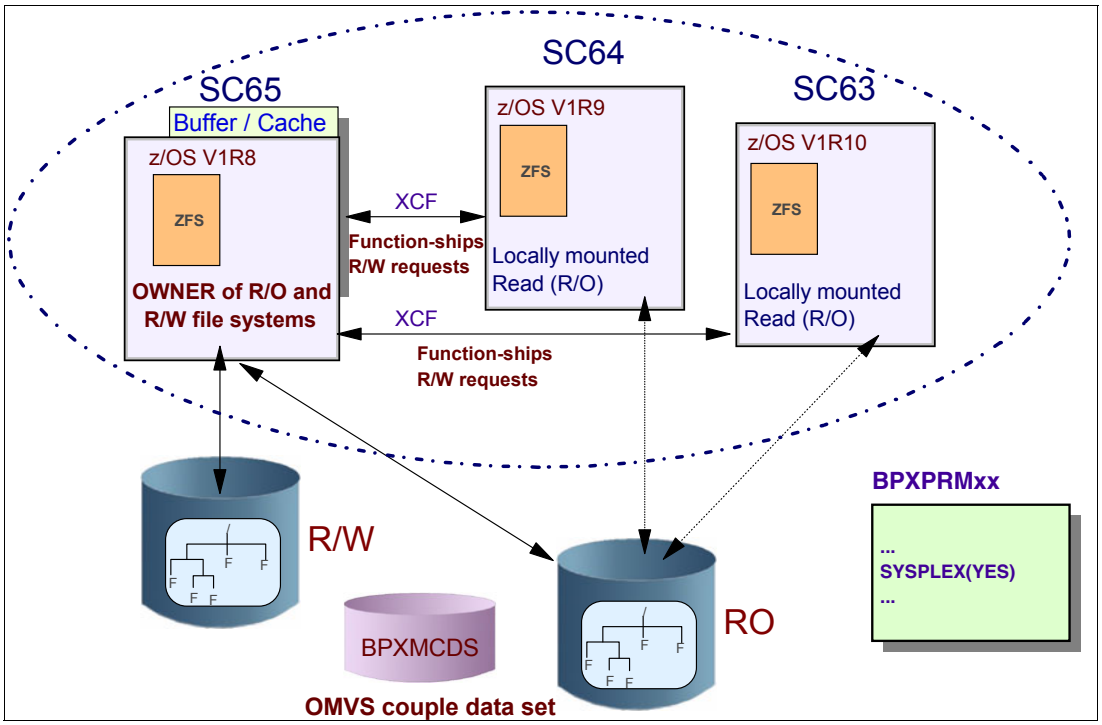


Figure 5-61 UNIX System Services file system sharing prior to z/OS V1R11

5.9.2 UNIX System Services file system sharing disadvantages for read-write file systems

UNIX System Services sysplex file system sharing offers many benefits as compared to single system mode:

- It allows you to write to file systems from all systems in the sysplex.
- It provides high availability of file systems for applications in the UNIX System Services sysplex environment.
- It allows you to make small changes to file systems used in R/O mode by following these steps:
  - Switch to R/W mode temporarily.
  - Perform the change.
  - Switch back to R/O mode.

Note that all data to be written or read must use the function-shipping path (not simply the request to be routed). This causes overhead compared to being on the owning system with all the buffer and cache management available locally and performing all I/O requests directly.

File systems mounted as R/O have been sysplex-aware from the beginning, and HFS and zFS file systems mounted in R/W mode were sysplex-unaware in all releases previous to z/OS V1R11.

Note: The performance disadvantage on a client system was the same, whether the request was a write or a read.

5.9.3 zFS support in UNIX System Services sysplex sharing prior to z/OS V1R11

In z/OS V1R6 and older systems, zFS knows only about zFS file systems that are mounted locally on each system, as illustrated in Figure 5-62.

- zFS is not sysplex-aware, except for R/O mounted file systems.
- UNIX System Services forwards file requests from a non-owning system (remote) to the owning system.
- zFS administration requests are handled locally.
- zFS is unaware of other remote zFS file systems and aggregates.

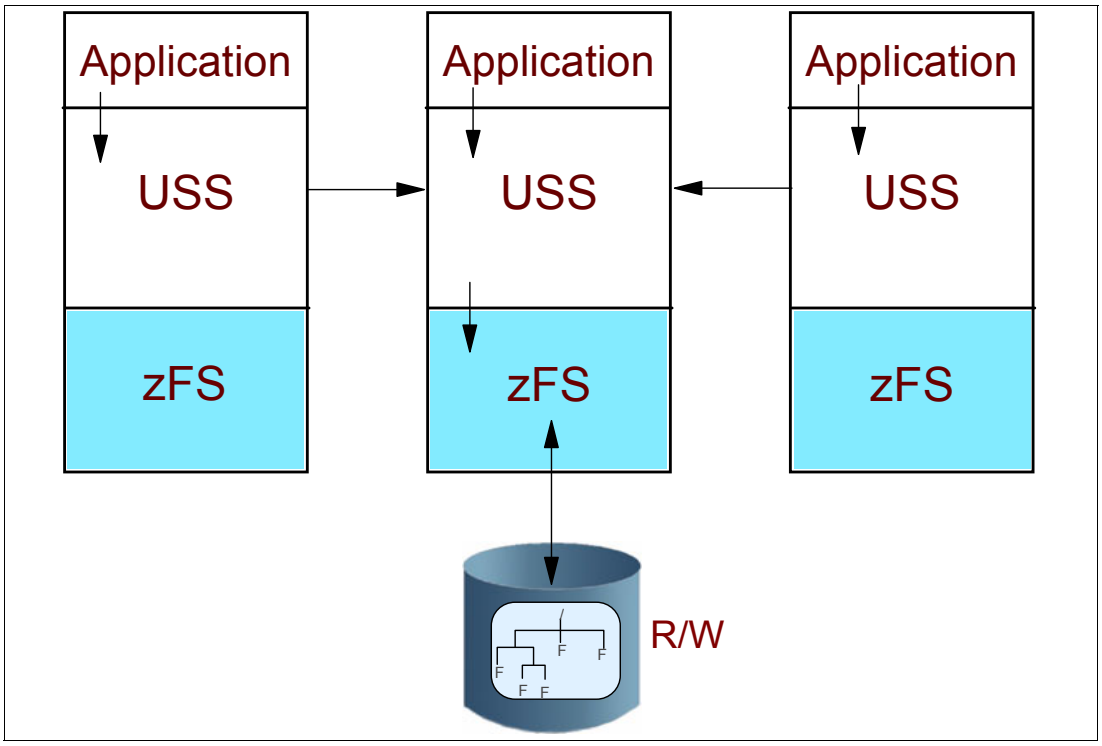


Figure 5-62 zFS support in z/OS V1R6 and earlier

z/OS V1R7 zFS support

In z/OS V1R7, the zFS administration interface works across the sysplex as illustrated in Figure 5-63 on page 244.

- zFS is sysplex-unaware (for file systems mounted R/W).
- UNIX System Services still forwards file requests to the z/OS UNIX owning system.
- zFS administration requests are handled across the sysplex.
- zFS knows about other remote zFS file systems and aggregates.

Requests are forwarded by UNIX System Services to the z/OS UNIX owning system, and then given to zFS. Responses are returned from zFS to UNIX System Services and then returned to the issuing system. As mentioned previously, this can cause a significant increase in the instruction pathlength as opposed to local access to a file system.

IOEZFS group specification

In addition, zFS support in z/OS V1R7 provided support for **zfsadm** commands that apply to zFS aggregates or file systems to work against all aggregates and file systems across the sysplex. The following **zfsadm** subcommands can optionally direct their operation to a particular member of the sysplex: **aggrinfo**, **attach**, **clonesys**, **config**, **configquery**, **define**, **detach**, **format**, **lsaggr**, **lsfs**, and **query**.

This support is represented by Admin as shown in Figure 5-63.

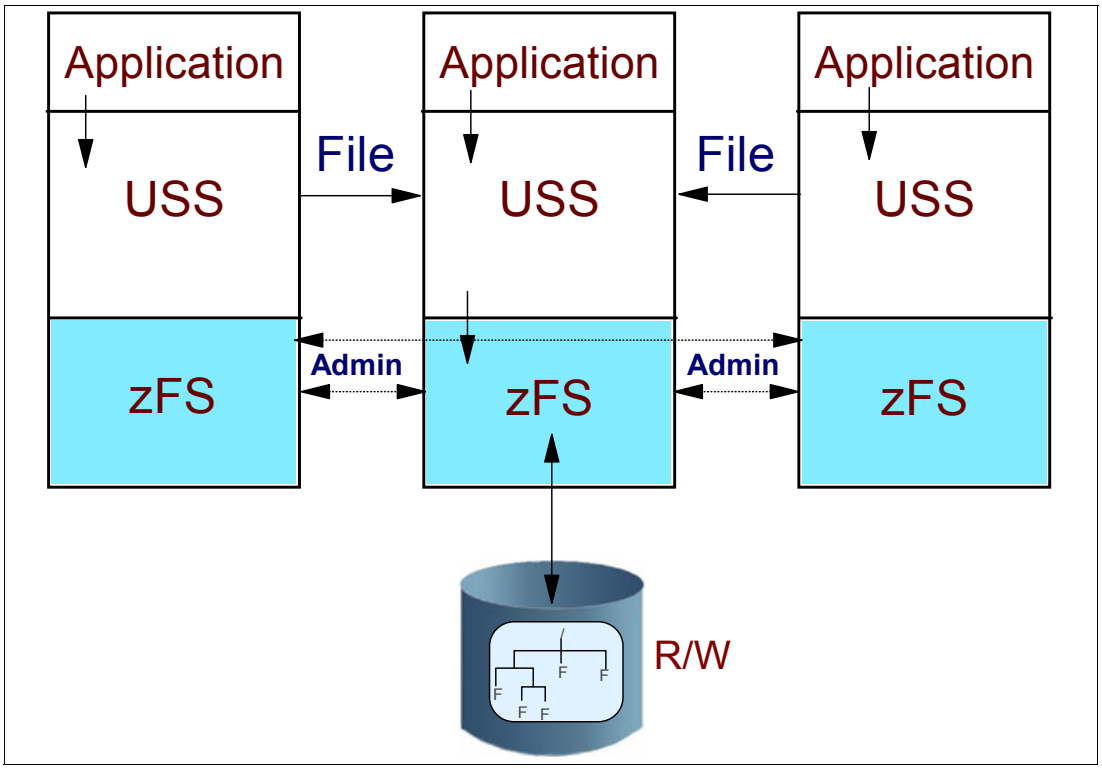


Figure 5-63 zFS support in z/OS V1R7 to z/OS V1R10

Note: This is exactly the same technique that is used for HFS file system sharing in a sysplex.

5.9.4 zFS configuration options in z/OS V1R11 (base code)

The following new configuration options can be defined either in the IOEPRMxx parmlib member or in the IOEFSPRM DD member in the ZFS procedure.

sysplex={ on | off }

The sysplex configuration option controls whether zFS runs sysplex-aware or not. The BPXPRMxx parmlib member option, SYSPLEX(YES), must be set to allow zFS to run sysplex-aware. See “zFS sysplex-aware in z/OS V1R11 for R/W mounts” on page 245 for more information.

group = ioezfs

The group configuration option specifies the group name that zFS uses for XCF communications. (This option was implemented in z/OS V1R7.) See “IOEZFS group specification” on page 244 for more information.

token_cache_size	This option specifies the maximum number of tokens in the server token manager cache to use for cache consistency between zFS members. The default value is double the number of vnode_cache_size.
file_threads = 40	The file_threads configuration option determines how many threads are in the pool to handle requests from remote clients.
client_cache_size=128M	The client_cache_size configuration option specifies the size of the user data cache for client access. The user_cache_size is used for the user data cache for server access.
client_reply_storage = 40M	The client_reply_storage configuration option specifies how much storage is used for sysplex server replies.
recovery_max_storage=256M	This option indicates the maximum amount of zFS address space storage to use for concurrent log recovery during multiple concurrent aggregate mounts.

5.9.5 zFS sysplex-aware in z/OS V1R11 for R/W mounts

In z/OS V1R11, zFS is sysplex-aware for file systems mounted R/W. This means that UNIX System Services sends all file requests directly to the local zFS physical file system (PFS). It does not function ship the requests. The local zFS either sends the request to the owning zFS system or it satisfies the request from the local zFS cache. In many cases, this improves the pathlength over the function shipping model. The request and data flow are illustrated in Figure 5-64 on page 246.

With z/OS V1R11, the zFS PFS allows a file system to be locally accessed on all systems in a sysplex for a particular mode, and then the PFS is sysplex-aware for that mode. Therefore, performance can be improved for zFS in the UNIX System Services sysplex shared file system mode by zFS becoming sysplex-aware for R/W mounted file systems. This is a new specification that is specified in the IOEPRMxx parmlib member or in the IOEFSPRM DD member in the ZFS procedure as follows:

```
sysplex=on
```

This zFS sysplex file system support improves the response time for zFS file system requests.

- zFS becomes sysplex-aware for zFS R/W file systems.
- zFS uses less XCF communication because a new client caching is introduced.
- The client requirement MR1211023911 asking for improved performance in a UNIX System Services sysplex file system sharing environment is answered.

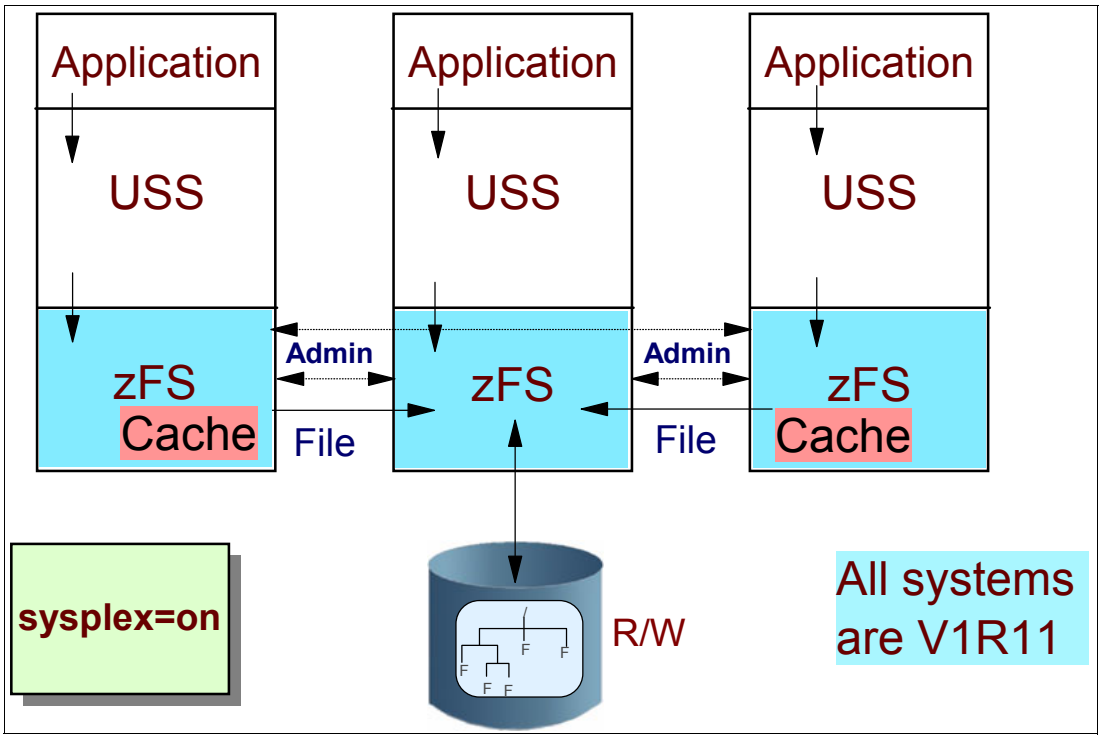


Figure 5-64 zFS sysplex support in z/OS V1R11

Defining zFS to run sysplex-aware

When the BPXPRMxx parmlib member specifies **SYSPLEX(YES)**, use one of the following methods to specify zFS options to define sysplex-aware for z/OS V1R11 systems:

- Use the IOEPRMxx parmlib member to contain the configuration options that control the zFS environment and the zFS aggregates to be attached at zFS startup. The IOEPRMxx files are contained in the logical parmlib concatenation. Use the **sysplex=on** parameter.

  Using this option means that zFS runs as sysplex-aware and that z/OS UNIX does not forward file requests to the z/OS UNIX owning system.

- Or, if you still use the IOEZPRM DD statement in the ZFS PROC, specify the zFS configuration option in the IOEFSPRM file, which is pointed to by the DD name. However, using this method to set zFS parameters is no longer recommended.

zFS cache management

When the zFS PFS is sysplex-aware, z/OS UNIX sends requests to the local zFS and zFS becomes responsible for forwarding the request, if necessary. For many requests, it is not necessary because zFS caches data locally and can satisfy the request from its cache. Cache consistency is maintained through a token management mechanism. This may provide improved performance in a shared file system environment.

Note: With z/OS V1R11, zFS mounted file systems can be sysplex-aware become zFS has it own concept of file system ownership.

A zFS client caches data as long as it holds a token for the data. Clients request tokens from the token manager. Token management for an aggregate runs on the zFS owning system. This administration task is handled by the token manager for an aggregate and handles token requests and manages token conflicts. It revokes conflicting tokens from clients.

Important: The new support in zFS has implemented a control mechanism for all the files and directories in a file system (with a possible granularity of 64 KB blocks of data, if needed). This “token management” is controlled by zFS on the system that is the zFS owner. It allows zFS on a specific system to have full control according to the type of token that it owns (reads, writes, or both), as long as the system owns the token (or both tokens).

In addition to its use of caching to improve performance, the zFS owning system also keeps track of the number of requests it receives from each client (and local) system. Based on a specific calculation and at internally set intervals, zFS can move zFS ownership to attempt to optimize zFS aggregate ownership to the system making the most requests.

On a system failure, zFS will move zFS ownership to another system.

zFS cache processing

When a request is received by the local zFS PFS, zFS determines whether the request can be satisfied from its cache. If it is a read, lookup, or readdir and the data is contained in the cache, then the request can be satisfied without communicating with the zFS owning system. Write requests are written forwarded to the owner, unless you know that the space is already allocated on disk.

With this new support in z/OS V1R11, the following conditions exist:

- zFS is sysplex-aware.
- UNIX System Services sends requests directly to local zFS PFS, regardless of the UNIX System Services owner.
- zFS decides whether it needs to forward the request to the zFS owning system.
- zFS caching and tokens allow it to sometimes avoid XCF communications.
- zFS moves zFS aggregates on system failure and to minimize XCF processing.
- Only compatibility mode zFS aggregates are supported.

New support details

When all systems are running at the z/OS V1R11 level, only compatibility mode aggregates are supported in a sysplex. Starting with z/OS V1R8, a zFS file system that resides in a multifile system aggregate cannot be mounted in a sysplex sharing environment.

The new options can be set or queried using the **zfsadm** command interface:

- The **zfsadm config** command allows setting all the new configuration options except for **sysplex=on|off**.
- The **zfsadm configquery** command can show all the new configuration options and also the new additional value for sysplex_state (option 2):

  0 This indicates that zFS is not in a shared file system environment; this is normal for V1R6 and prior releases and for single system configurations including monoplex and xcflocal.

  1 This indicates that zFS is in a shared file system environment; this is normal for z/OS V1R7 to z/OS V1R10.

  2 This indicates that zFS is running in a sysplex-aware environment; this is normal for V1R11 in a shared file system environment where zFS is running sysplex-aware for read-write file systems, for example:

```
ROGERS @ SC74:/u/rogers>zfsadm configquery -sysplex_state
IOEZ00317I The value for configuration option -sysplex_state is 2.
```

Defining zFS to run sysplex-unaware in z/OS V1R11

When all systems are running at z/OS V1R11, you can define the zFS sysplex setting as **`sysplex=off`** and perform sysplex file sharing as in all previous releases, as illustrated in Figure 5-65.

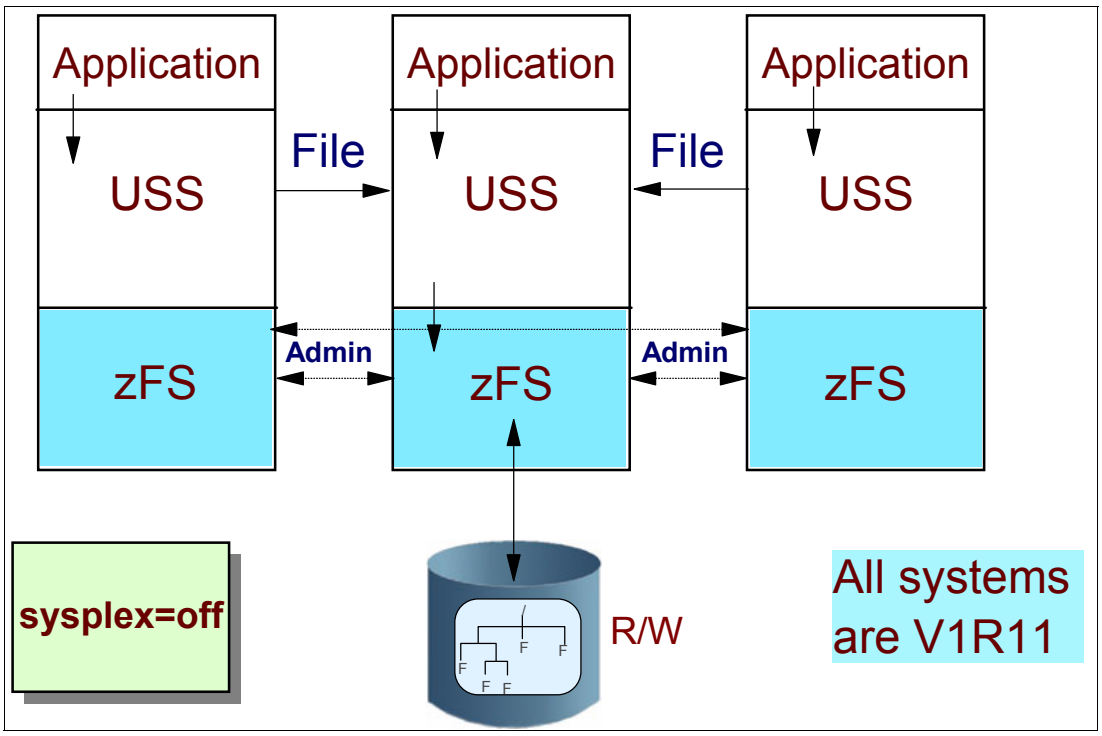


Figure 5-65 z/OS V1R11 sysplex running zFS with sysplex=off

Normal migration path to zFS being sysplex-aware

The normal migration path to zFS being sysplex-aware consists of the following steps:

1. zFS in z/OS V1R11 is running with option sysplex=off as sysplex-unaware for R/W mounted file systems. This is the default.
2. Then zFS in R11 is started with sysplex=on as sysplex-aware via rolling IPL or carefully stopping and restarting zFS.

zFS sysplex-aware restrictions

The following restrictions exist for zFS running sysplex-aware for R/W mounted file systems.

- The z/OS SMB server cannot export zFS sysplex-aware R/W file systems.
- Fast Response Cache Accelerator support of the IBM HTTP Server V5R3 for z/OS cannot cache files contained in zFS sysplex-aware file systems.
- z/OS UNIX ownership of a sysplex-aware zFS aggregate mounted R/W cannot be moved to a prior release or z/OS V1R11 zFS which is sysplex-unaware, as illustrated in Figure 5-88 on page 264.

5.9.6 Read-only mounted file systems (sysplex-aware)

When a file system is mounted read-only, as illustrated in Figure 5-66 on page 249, the mount request is sent to the local physical file system (in this case, zFS) and zFS opens the file system data set (for read). If the mount is successful on that system, z/OS UNIX records the mount and sends a signal to the other sysplex member systems to issue a “catch-up” mount

on each system. Each z/OS UNIX on each other system then reads the couple data set (CDS) and determines that it needs to send a mount request to the local zFS for that file system. Each “local mount” causes zFS to open the data set (for read). In this way, the mount on SC65 causes the file system to be mounted on every member of the sysplex.

For R/O mounted file systems, file requests are sent directly to the local physical file system, which directly reads the file system data on DASD (as shown in Figure 5-66). That means each zFS on each system has the zFS file system opened (for read) and directly accesses the data.

Note: Read-only mounted file systems are referred to as being sysplex-aware.

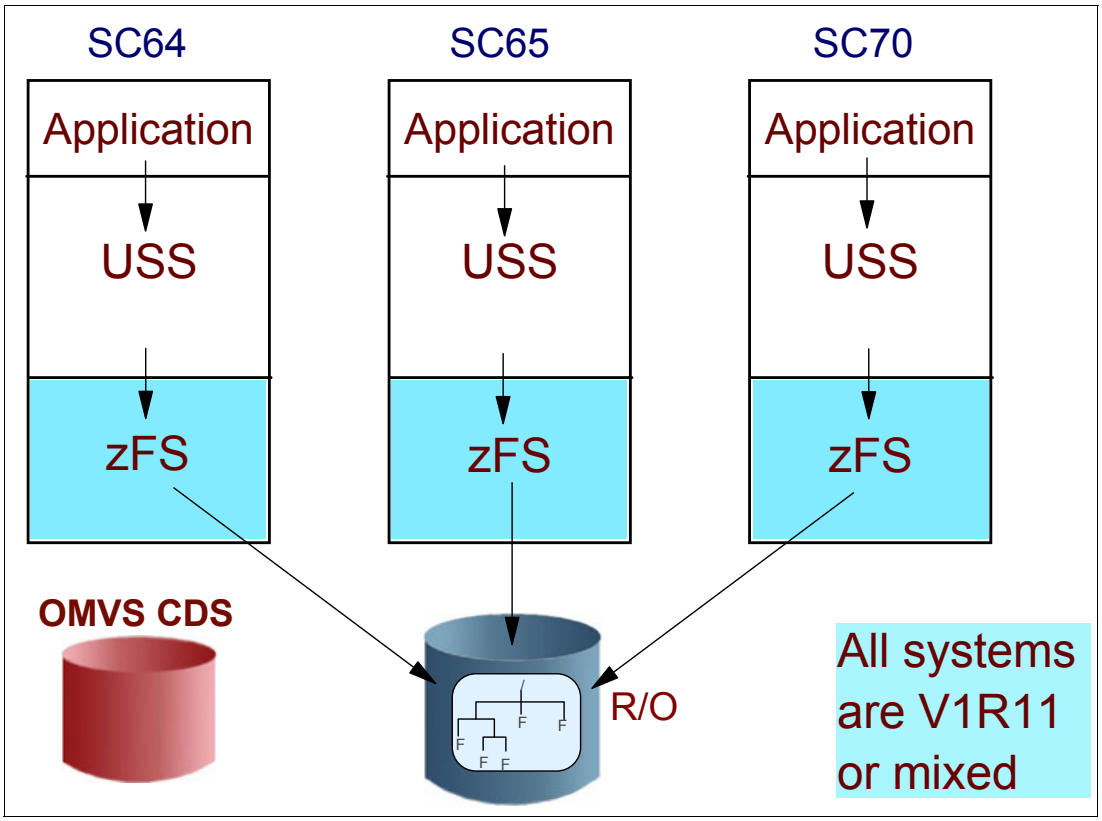


Figure 5-66 Sysplex-aware file system (read-only)

5.9.7 z/OS UNIX file system ownership versus zFS aggregate ownership

With z/OS V1R11 there are two different types of file system owners for zFS aggregates:

- z/OS UNIX file system ownership
- zFS aggregate ownership

z/OS UNIX file system ownership

z/OS UNIX ownership of file systems determines where z/OS UNIX will forward requests for sysplex-unaware file systems.

However, when zFS runs sysplex-aware, z/OS UNIX ownership does not have the same significance for zFS file systems as it does for HFS file systems because z/OS UNIX does not

normally function ship requests for zFS file systems in this situation. UNIX System Services still chooses a z/OS UNIX owning system for each zFS file system but that does not control where local file requests are forwarded to by z/OS UNIX. File requests to sysplex-aware file systems are sent directly to the local PFS.

z/OS UNIX ownership does, however, control which distributed Byte Range Lock Manager byte range lock requests are forwarded to it.

The z/OS UNIX file system owner can be different from the zFS file system owner. z/OS UNIX file system ownership can be determined by using the **df -v** command, as shown in Figure 5-67.

```
/pp/db2v9/batch17/db2910_base: >df -v .
Mounted on    Filesystem          Avail/Total   Files    Status
/pp/db2v9/batch17/db2910_base (OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS) 6884/21600
4294967272 Available
ZFS, Read/Write, Device:60, ACLS=Y
File System Owner : SC63      Automove=Y    Client=N
Filetag : T=off  codeset=0
Aggregate Name : OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS
```

Figure 5-67 The df -v command for a single file system

zFS aggregate ownership

zFS aggregate ownership determines where zFS forwards requests when zFS is running sysplex-aware.

zFS may move the zFS ownership of an aggregate in the following situations:

- Based on usage of the zFS aggregate in the systems of the sharing environment
- When the owning system fails or is being shut down
- When a stop or failure of zFS occurs

As a result, the zFS owner of the aggregate may be different from the z/OS UNIX owner of the file system. Both ownerships may change independently. The zFS aggregate owner coordinates I/O requests, quiesce, unquiesce, and unmount, among other things.

Note: zFS aggregate ownership is not as critical as z/OS UNIX file system ownership was in previous releases, due to zFS client caching and to the possibility of dynamic aggregate movement.

You can determine zFS aggregate ownership by using the **zfsadm lsaggr** command, as illustrated in Figure 5-68.

```
zfsadm lsaggr
OMVS.DB2V9.SDSNAHFS.BATCH17.ZFS                SC63      R/W
OMVS.DB2V9.SDSNWORF.BATCH17.ZFS                SC63      R/W
OMVS.DARIO.HFS                                 SC70      R/W
OMVS.HERING.TESTBAT                            SC63      R/W
OMVS.HERING.TEST.HFS.TAV                       SC70      R/W
OMVS.TEST.MULTIFS.ZFS                          SC63      R/O
ZFSFR.ZFSE.ZFS                                 SC64      R/W
```

Figure 5-68 The zfsadm lsaggr command that displays the zFS file system owner

Note: In the examples shown in Figure 5-67 on page 250 and Figure 5-68 on page 250, the z/OS UNIX and zFS ownership for the file system is the same.

Displaying file owner information

Both z/OS UNIX file system ownership and zFS aggregate ownership are initially determined during mount processing. When a file system is mounted, a z/OS UNIX file system owner is assigned by z/OS UNIX. Initially, for a compatibility mode aggregate, the zFS owner of the aggregate and the z/OS UNIX owner of the file system in the aggregate are the same.

Thereafter, they can move independently. Figure 5-69 shows an example listing both ownerships for a specific zFS aggregate.

awk information: An instruction of the form: pattern {actions} indicates that **awk** is to perform the given set of actions on every record that meets a certain set of conditions. The conditions are given by the pattern part of the instruction.

In Figure 5-69 on page 251, the pattern of an instruction often looks for records that have a particular value in some field. The notation $5 stands for the fifth field of a record, $2 stands for the second field, and so on.

```
$> /usr/sbin/mount -qv test
----A- OMVS.HERING.TEST.ZFS /u/hering/test
$> df -v test | grep Owner | awk '{print "USS owner: "$5}'
USS owner: SC70
$> zfsadm lsaggr | grep OMVS.HERING.TEST.ZFS | awk '{print "zFS owner: "$2}'
zFS owner: SC65
$>
```

Figure 5-69 Listing the z/OS UNIX file system and the zFS aggregate owner

Note: You can be less concerned about z/OS UNIX ownership when zFS runs sysplex-aware because it does not control request forwarding. As mentioned, you can also be less concerned about zFS ownership because of caching and because zFS will try to automatically move zFS ownership intelligently.

Utilities for displaying ownership information

In addition to the standard commands and functions, for our testing we created two REXX routines to display owner and owner-related information about UNIX System Services file systems and zFS aggregates:

- RXDOWNER - Display OWNER information
- ZFSOWNER - zFS aggregate OWNER information

B.18, “RXDOWNER and ZFSOWNER” on page 433 provides a detailed description of the procedures and the contents of the common procedure.

Figure 5-70 on page 252 lists corresponding information for the aggregate referenced in Figure 5-69.

```
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

$> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
$>
```

Figure 5-70 Listing owner information with rxdowner and zfsowner

In addition to zFS aggregates, routine rxdowner also lists information for other types of file systems, as shown in Figure 5-71.

```
$> rxdowner -d /etc

MP Directory : /SC70/etc
File System  : HFS.SC70.ETC
PFS Type     : HFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
$> rxdowner -f *AMD/u

MP Directory : /u
File System  : *AMD/u
PFS Type     : AUTOMNT
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC64     - File System read-only=N
$>
```

Figure 5-71 Listing owner information with rxdowner for other PFS types

5.9.8 zFS Admin support in z/OS V1R11

To avoid a possible multi-system outage due to a zFS namespace inconsistency that had been seen with an earlier implementation of the enhanced zFS sysplex file system support, the following enhancements have been added:

- The zFS XCF Admin protocol (for mount and unmount processing and zfsadm commands) has been simplified. Only information about locally mounted file systems is exchanged between sysplex members.
- The zFS file system ownership is kept externally in GRS enqueues.
- Also provided is automatic validation and, if needed, a correction of the zFS namespace.

Toleration support for zFS in z/OS V1R11

Table 5-1 on page 253 lists the PTF information for the necessary toleration support of zFS in z/OS V1R11 for systems z/OS V1R9 and z/OS V1R11.

Table 5-1 Toleration support for zFS Admin protocol in z/OS V1R9 and V1R10

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA45616	OA25026
z/OS V1R10 (3A0)	UA45614	OA25026

z/OS V1R9 and z/OS V1R10 with this toleration support added are the only z/OS releases that are supported together with zFS of z/OS V1R11.

zFS sysplex Admin levels

The following zFS Admin levels are available.

0 This is the zFS Admin interface protocol used for zFS running on systems z/OS V1R9 or V1R10 without APAR OA25026. This is the same as for z/OS levels z/OS V1R7 and V1R8.

1 This is the default zFS Admin level used in z/OS V1R9 and V1R10 when APAR OA25026 is applied, also called the *conditioning level*.

2 This the zFS Admin level for zFS of z/OS V1R9 and z/OS V1R10 that allows them to run together with zFS of z/OS V1R11. Therefore, it is called the *toleration mode*. It also exploits the benefits of the new interface.

3 This is the zFS Admin level of z/OS V1R11. Any specification of parameter sysplex_admin_level in z/OS V1R11 is ignored because zFS in z/OS V1R11 always runs at level 3.

With the three z/OS levels z/OS V1R9 plus PTF UA45616, z/OS V1R10 plus PTF UA45614, and V1R11, you need to understand the following situations:

- zFS in z/OS V1R9 or z/OS V1R10 with **sysplex_admin_level=1** set uses and follows the XCF Admin protocol among systems with level 0 and 1 and does not initialize if a system with XCF Admin level 3 is active in the sysplex.
- zFS in z/OS V1R9 or z/OS V1R10 with **sysplex_admin_level=2** set uses and follows the new XCF Admin protocol among systems with levels 2 or 3 and does not initialize if a system with XCF Admin level 0 is active in the sysplex.
- zFS in z/OS V1R11 running with **sysplex_admin_level=3** by default uses and follows the new XCF Admin protocol among systems with levels 2 or 3 and does not initialize if a system with XCF Admin level 0 or 1 is active in the sysplex.

zFS namespace validation

In a shared file system environment, zFS in z/OS V1R11 verifies the zFS namespace on the following occasions:

- When an administration command experiences an XCF message timeout
- At zFS initialization
- When an inconsistency is detected
- When the operator **F ZFS,NSVALIDATE** command is issued, for example:

```
F ZFS,NSVALIDATE

IOEZ00025I zFS kernel: MODIFY command - NSVALIDATE completed successfully.
```

If a zFS namespace problem is found, then zFS runs zFS namespace correction to resolve the problem. In the worst case this may cause a zFS restart, which is disruptive. However, this does not normally occur.

The command initiates zFS namespace validation on the system where the command is entered. The **F ZFS,NSVALIDATE** command is typically only used as a part of a recovery procedure when a problem with zFS is suspected. If the command finds an inconsistency, it might cause zFS to abort and restart the zFS address space on one or more systems to correct the zFS namespace inconsistency. The **F ZFS,NSVALIDATE** command consists of the following option:

print The optional print parameter displays additional name space information obtained after validation.

Because the zFS file system ownership is kept externally in GRS enqueues, you can use MVS system command:

```
D GRS,RES=(SYSZIOEZ,*)
```

This command displays all zFS owners for all the zFS aggregates in your sysplex, if the owning system uses zFS Admin level 3. Figure 5-89 on page 265 shows an example for a specific aggregate.

Activating toleration of R11 zFS in z/OS V1R9 or z/OS V1R10

This activation is a two-step process:

1. Install APAR OA25026 on all z/OS V1R9 and V1R10 systems.
2. After OA25026 is active on all systems, specify **sysplex_admin_level=2** in the zFS parameter file for all systems and activate sysplex Admin level 2 on all systems. zFS of z/OS V1R11 can enter the sysplex at this point.

Using zFS admin level 2 in z/OS V1R9 and z/OS V1R10 in general

As mentioned, you need to run with **sysplex_admin_level=2** specified in the zFS parameter file if a zFS R11 is used in parallel. However, it is also recommended that you always use this level even if no zFS R11 systems are in the sysplex. This avoids possible outages and problems in situations leading to an inconsistency of state information between zFS members in a sysplex shared file system environment. For more detailed information, refer to DOC APAR OA30198, *zFS R9/R10 namespace implementation change*.

Install APAR OA25026 on all systems

This Documentation APAR describes the resolution of an existing problem in zFS that can occur due to an inconsistency of state information between zFS members in a sysplex (that is, a shared file system environment). The inconsistency can occur as a result of the timing of systems entering the sysplex, abnormally leaving the sysplex and file system administration activity (for example, mount and unmount of zFS file systems).

If this inconsistency occurs, you may see mount failures or change owner failures and it may require a sysplex-wide IPL to resolve the problem. This applies to zFS on z/OS V1R9 and z/OS V1R10. This documentation describes the mechanism to provide a modified zFS sysplex XCF protocol, which avoids this problem. Because it is a modification to the existing zFS XCF protocol, it requires a conditioning step and then a separate fix step.

The conditioning step occurs upon installation of APAR OA25026 across the sysplex via a rolling IPL. APAR OA25026 was available in March 2009 (R9 PTF is UA45616, R10 PTF is UA45614). The fix step occurs when a new option is specified on all members after running with the APAR on all members of the sysplex and is activated via another rolling IPL.

The fix step cannot be run until the conditioning step is completed on all members of the sysplex. If you attempt to run the fix step on a system without having the APAR running on all members of the sysplex first, you will receive the following message and the zFS on the fix step system will not initialize:

```
IOEZ00614A zFS has detected an incompatible interface level 0 for member
<membername>.
```

5.9.9 zfsadm commands in z/OS V1R11

This section explains the changes and new functions of the **zfsadm** command interface.

- The **zfsadm attach** command can no longer attach multifile system aggregates in a UNIX System Services file system sharing environment.

  Important: In a z/OS V1R11 UNIX System Services file system sharing environment, multifile system aggregates cannot be attached any more. This is a further step in the removal of multifile system aggregates.

- The **zfsadm config** command supports all new configuration options except for setting the parameters **sysplex** and **sysplex_admin_level**. These options cannot be set or changed dynamically.
- The **zfsadm configquery** command displays the sysplex state with the (already existing) option **-sysplex_state**.
- Using the **zfsadm configquery -syslevel** command displays information about the PFS level in a similar way to the MVS system command **F ZFS,QUERY,LEVEL**; see Figure 5-72 on page 256.

UNIX zfsadm command:

```
$> zfsadm configquery -syslevel
IOEZ00644I The value for configuration option -syslevel is:
zFS kernel: z/OS    zSeries File System
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
sysplex(file) interface(3)
$>
```

MVS system command:

```
F ZFS,QUERY,LEVEL
IOEZ00639I zFS kernel: z/OS    zSeries File System 836
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
sysplex(file) interface(3)
IOEZ00025I zFS kernel: MODIFY command - QUERY,LEVEL completed successfully.
```

Figure 5-72 Showing information about PFS and sysplex interface level in z/OS V1R11

Figure 5-73 shows corresponding information for zFS in z/OS V1R9 or z/OS V1R10 with toleration support APAR OA25026 applied.

```
F ZFS,QUERY,LEVEL
IOEZ00639I zFS kernel: z/OS    zSeries File System 831
Version 01.10.00 Service Level OA27728 - HZFS3A0.
Created on Mon Feb  2 12:09:18 EST 2009.
sysplex(admin-only) interface(2)
IOEZ00025I zFS kernel: MODIFY command - QUERY,LEVEL completed successfully.
```

Figure 5-73 Showing information about PFS and sysplex interface level in z/OS V1R10

zFS kernel messages during initialization

Similar information is provided when zFS is starting, as shown in Figure 5-74 on page 257.

zFS in z/OS V1R10 with sysplex_admin_level=2

```
IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
Version 01.10.00 Service Level OA27728 - HZFS3A0.
Created on Mon Feb  2 12:09:18 EST 2009.
Address space asid x7B
IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
IOEZ00617I zFS is running sysplex admin-only with interface level 2
IOEZ00350I Successfully joined group IOEZFS
IOEZ00055I zFS kernel: initialization complete.
```

zFS in z/OS V1R11 with sysplex=on

```
IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
Version 01.11.00 Service Level z110117 - HZFS3B0.
Created on Sun Mar 22 17:07:39 EDT 2009.
Address space asid x6E
IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
IOEZ00617I zFS is running sysplex file-support with interface level 3
IOEZ00350I Successfully joined group IOEZFS
IOEZ00055I zFS kernel: initialization complete.
```

Figure 5-74 zFS initialization messages

5.10 zFS sysplex migration scenarios (R11 base code)

When migrating to z/OS V1R11, there are special considerations for the new support when previous levels of z/OS are in the same sysplex as the V1R11 system. This is an important change with zFS when accessing, opening, and working with aggregates in z/OS V1R11.

Important: When zFS handles a mount request prior to V1R11 (or when V1R11 runs sysplex=off), zFS allocates the aggregate exclusive and then opens the aggregate for R/W.

When zFS handles a mount request for R11 running sysplex-aware (sysplex=on), zFS allocates the aggregate shared and then opens the aggregate for R/W on each V1R11 system in the sysplex.

This is why administrators cannot force a reopen of an aggregate by moving it to another system. However, the UNIX unmount (remount) samemode function can used to perform this and other tasks. Refer to 5.10.6, “Exploiting the UNIX Unmount (Remount) samemode” on page 269 for more information about this topic.

5.10.1 zFS migration support for z/OS V1R11

To verify the readiness of your environment to migrate to zFS of z/OS V1R11 and highlight another step in the removal of zFS multifile system aggregates, there are two zFS migration health check routines available with APAR OA27198, as listed in Table 5-2.

Table 5-2 Health checker for migration to zFS in z/OS V1R11

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA49074	OA27198
z/OS V1R10 (3A0)	UA49072	OA27198

Health check ZOSMIGV1R11_ZFS_INTERFACELEVEL

This health check queries the zFS interface level and verifies that **sysplex_admin_level=2** is specified. Otherwise, no z/OS V1R11 zFS can initialize as shown in Figure 5-75.

```
 IOEZ00559I zFS kernel: Initializing z/OS    zSeries File System
...
 Version 01.11.00 Service Level z110117 - HZFS3B0.
 Created on Sun Mar 22 17:07:39 EDT 2009.
 Address space asid x3A
 IOEZ00374I No IOEZPRM DD specified in ZFS proc. Parmlib search being used.
 IOEZ00617I zFS is running sysplex admin-only with interface level 3
 IOEZ00614A zFS has detected an incompatible interface level 1 for member SC63.
 IOEZ00057I zFS kernel program IOEFSCM is ending
*108 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN READY TO
 RESTART. REPLY 'I' TO IGNORE.
```

Figure 5-75 zFS of z/OS V1R11 not starting because of incompatible Admin level

Health check ZOSMIGREC_ZFS_RM_MULTIFS

In a z/OS V1R11 UNIX System Services file system sharing environment, multifile system aggregates cannot be attached. This health check queries aggregates that are attached (not mounted) and reports them, as shown in Figure 5-76 on page 258.

```
CHECK(IBMZFS,ZOSMIGREC_ZFS_RM_MULTIFS)
START TIME: 05/07/2009 15:06:50.136149
CHECK DATE: 20071231  CHECK SEVERITY: LOW

IOEZH0022I Attached multi-file system aggregates:
AGGREGATE                                      # OF FILE SYSTEMS
-----------------------------------------      -----------------
OMVS.TEST.MULTIFS.ZFS                               2

* Low Severity Exception *

IOEZH0021E Multi-file system aggregates are attached.

  Explanation:  Multi-file system aggregates are attached to this system.
    In zFS V1R11, multi-file systems cannot be attached in a sysplex environment.
...
    All multi-file system aggregates should be converted to compatibility mode
    aggregates.
...
  System Programmer Response:  Convert your multi-file system aggregates
    to compatibility mode aggregates.

    Remove all define_aggr configuration options in your IOEFSPRM file
    and run zfsadm detach -all.
...
  Check Reason:  Verifies no multi-file system aggregates attached.

END TIME: 05/07/2009 15:06:50.646808  STATUS: EXCEPTION-LOW
```

Figure 5-76 ZOSMIGREC_ZFS_RM_MULTIFS showing a multifile system aggregate

If no multifile system aggregate is remaining the check runs successfully, as shown in Figure 5-77.

```
CHECK(IBMZFS,ZOSMIGREC_ZFS_RM_MULTIFS)
START TIME: 05/07/2009 12:11:15.935653
CHECK DATE: 20071231  CHECK SEVERITY: LOW

IOEZH0020I No multi-file system aggregates are attached.
This check ran successfully and found no exceptions.

END TIME: 05/07/2009 15:11:16.197562  STATUS: SUCCESSFUL
```

Figure 5-77 ZOSMIGREC_ZFS_RM_MULTIFS running successfully

Note: In any environment, you should migrate off multifile system aggregates.

5.10.2 zFS aggregate mounted on a system prior to z/OS V1R11

z/OS V1R9 and z/OS V1R10 are the only systems that can be in the same sysplex as z/OS V1R11. The first example, which is shown in Figure 5-79 on page 260, represents a situation in which an aggregate is mounted on a system prior to z/OS V1R11.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC64: z/OS Version 01.10.00
#> /usr/sbin/mount -t zfs -f OMVS.HERING.TEST.ZFS /u/hering/test
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#>
```

Figure 5-78 Mounting zFS aggregate on system SC64 running z/OS V1R10

When a primary zFS file system mount occurs on a system and it becomes the z/OS UNIX owner, UNIX System Services issues “catch-up” mounts on the other sysplex members (because every system needs to know about all file systems in the sharing environment).

- These catch-up mounts go to the local zFS when that system runs sysplex-aware.
- For a PFS that is not sysplex-aware, the catch-up mount goes to the z/OS UNIX internal “Cross System PFS” (XPFS) for function shipping.

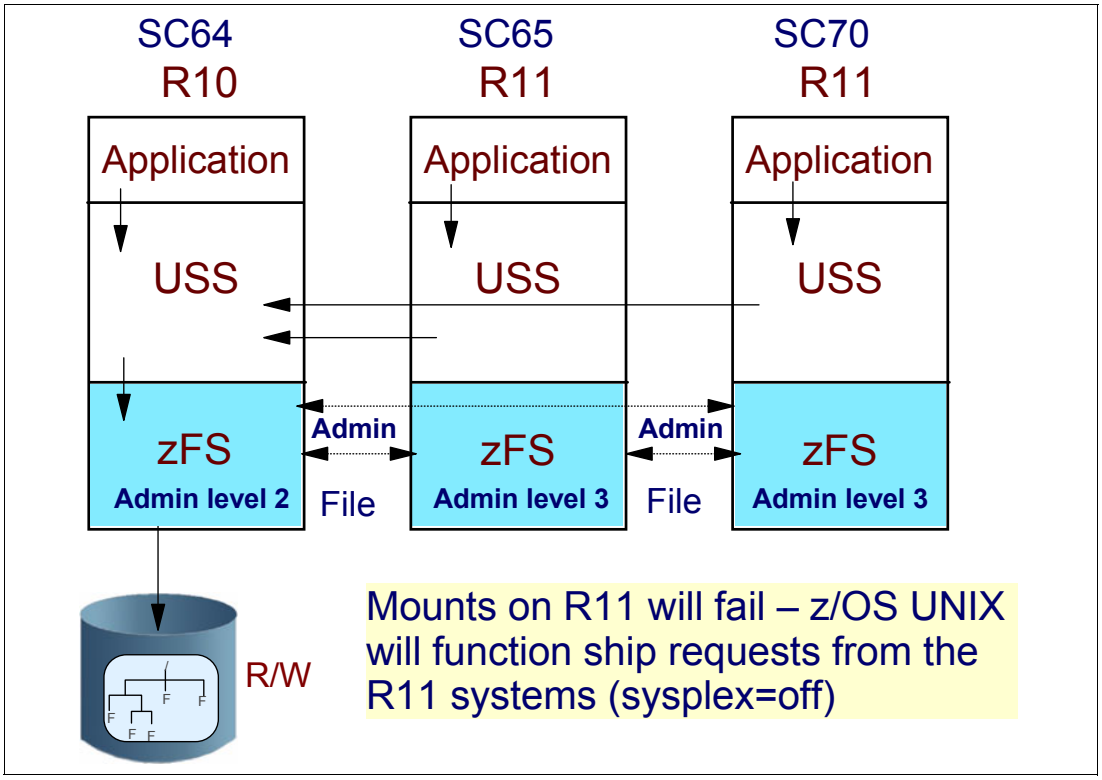


Figure 5-79 zFS aggregate mounted on a system prior to z/OS V1R11

The other systems in this sysplex, SC65 and SC70, are running z/OS V1R11 and get the messages shown in Figure 5-80 on page 260 on these systems.

```
SC65:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 076
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
SC70:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 185
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
```

Figure 5-80 "Catch-up" zFS mount error messages on z/OS V1R11 systems

zFS cannot allocate the file system shared on the z/OS V1R11 systems because the zFS on the older z/OS release has allocated the file system exclusively. This is shown in Figure 5-81.

```
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 17.33.00 GRS STATUS 597
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                0030       007FD0C0 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                0030       007EAE88 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                0030       007EAE88 EXCLUSIVE    OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC64      ZFS                0030       007EAE88   SHARE      OWN
```

Figure 5-81 zFS aggregate allocated exclusively on SC64 running z/OS V1R8

Therefore, z/OS UNIX will function ship zFS requests from the z/OS V1R11 systems to the owning z/OS V1R10 system in this situation.

5.10.3 zFS aggregate mounted on z/OS V1R11 system (sysplex=off)

Figure 5-84 on page 262 shows a mixed release sysplex. When a mount is issued on a R11 system and there are other systems running pre-R11 zFS (or a R11 zFS with sysplex=off), then the catch-up mounts on the pre-R11 zFS systems go to the z/OS UNIX internal XPFS PFS for function shipping. No error messages occur.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC65: z/OS Version 01.11.00
#> /usr/sbin/mount -t zfs -f OMVS.HERING.TEST.ZFS /u/hering/test
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#>
```

Figure 5-82 Mounting zFS aggregate on system SC65 running z/OS V1R11

A message is issued on each z/OS V1R11 system that is running zFS sysplex-aware, as shown in Figure 5-83.

```
SC70:  BPXF221I FILE SYSTEM OMVS.HERING.TEST.ZFS 185
       FAILED TO MOUNT LOCALLY.
       RETURN CODE = 00000079, REASON CODE = EF0969A8
       THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH
       A MOUNT ON A REMOTE SYSTEM.
```

Figure 5-83 "Catch-up" zFS mount error messages on z/OS V1R11 systems

Systems SC64 and SC70 function ship requests to the owning system SC65.

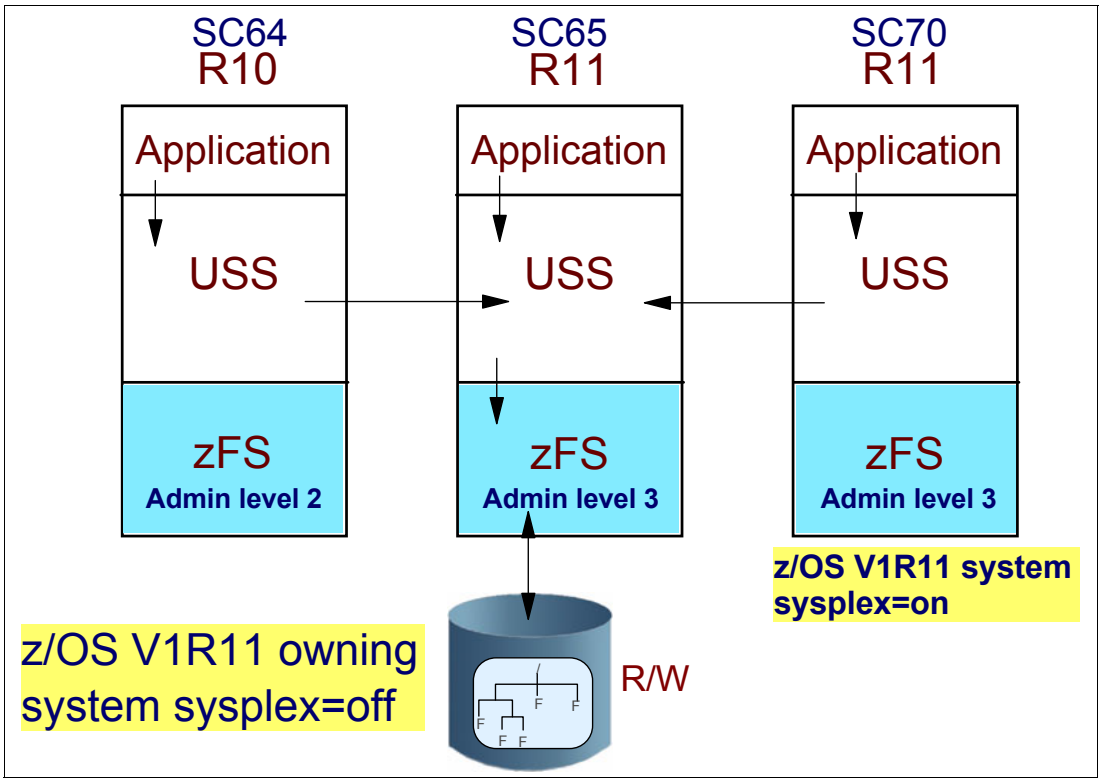


Figure 5-84 z/OS V1R11 systems sysplex=off and sysplex=on

Moving a mounted file system

Be aware, however, that if you try to move the R11 owned zFS file system on system SC65 (Figure 5-84) to the prior release system SC64 (for example, via a **`chmount`** command), and there are other R11 zFS sysplex-aware systems (system SC70), the move will fail (see Figure 5-88 on page 264).

5.10.4 zFS aggregate mounted on z/OS V1R11 system (sysplex=on)

Figure 5-85 on page 263 shows an aggregate that is mounted on a system running at level z/OS V1R11 with zFS being sysplex-aware. This causes z/OS UNIX to send file requests directly to the local zFS PFS and to not perform function shipping from the sysplex=on z/OS V1R11 systems.

When the request is received by the local zFS PFS, zFS determines if the request can be satisfied from its cache. If it is a read, lookup, or readdir request from the V1R11 sysplex=on system and the data is contained in the cache, then the request can be satisfied without communicating with the zFS owning system. Write requests are written through, unless you know the space is already allocated on disk.

In this mixed zFS sysplex sharing environment, when a read-write zFS mount is issued on a non-z/OS V1R11 system, a function ship request is made to the owning system.

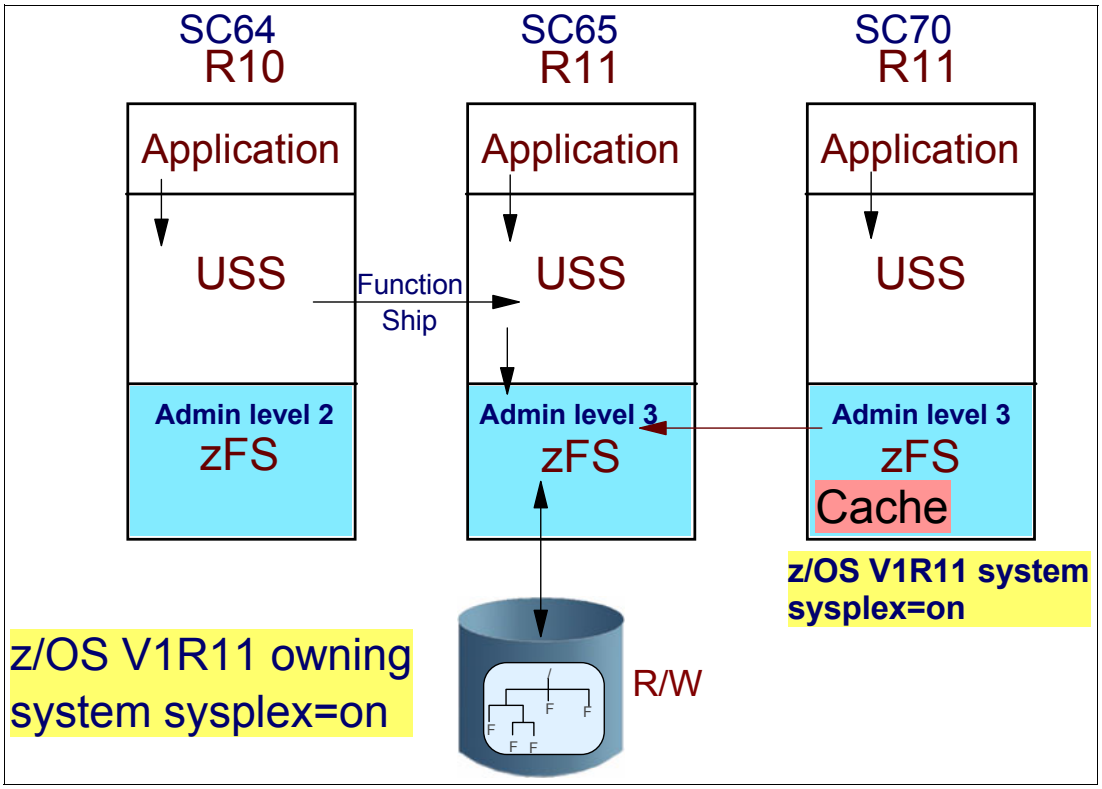


Figure 5-85 zFS aggregate mounted on a system running z/OS V1R11 (sysplex=on)

Moving a R/W aggregate between systems in a mixed environment

Figure 5-86 on page 263 shows the file system ownership for UNIX System Services and zFS, with starting of the ownership being on system SC64 which is running z/OS V1R10.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC64: z/OS Version 01.10.00
#> rxdowner -a OMVS.HERING.TEST.ZFS
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N
#> /usr/sbin/chmount -d SC65 /u/hering/test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC64     - File System local-client=Y
USS Owner    : SC65     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 5-86 Moving the zFS aggregate to system SC65 running z/OS V1R11

If you try to move the zFS file system owned on the z/OS V1R11 system to a prior release system, and there are other R11 zFS sysplex-aware systems, the move fails because z/OS UNIX does not unmount the zFS file system from the R11 systems.

Notice that the R11 systems have the zFS file system allocated shared, and that the zFS of the system that is running at a lower level of z/OS attempts to allocate the zFS file system exclusively. Figure 5-87 shows the new situation.

```
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 18.50.55 GRS STATUS 832
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FFB00   SHARE      OWN
SC65      ZFS                 001E       007FFB00   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC70      ZFS                 002D       007FF028   SHARE      OWN
SC65      ZFS                 001E       007FF5E8   SHARE      OWN
```

Figure 5-87 zFS aggregate allocated shared on systems running z/OS V1R11

However, the zFS file system can be moved to other R11 zFS sysplex-aware systems. This changes the z/OS UNIX ownership only. Therefore, after a zFS file system is owned by an R11 sysplex-aware system, it *cannot* be moved to a system with zFS being sysplex-unaware. Figure 5-88 on page 264 shows example commands.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC70: z/OS Version 01.11.00
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
#> /usr/sbin/chmount -d SC64 /u/hering/test
FOMF0504I mount error: 9D EF096058
EMVSERR: A MVS environmental or internal error has occurred
Description: HFS-compat mount, - error attaching aggregate or was found not to
be HFS-compat.
#> /usr/sbin/chmount -d SC70 /u/hering/test
#> rxdowner -d /u/hering/test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 5-88 Moving the zFS aggregate UNIX System Services ownership between systems running z/OS V1R11

Reason code EF096058 says that this can also mean that you attempted to move ownership of a zFS sysplex-aware R/W file system to a system that does not support zFS sysplex-aware. Thus, either the target system is prior to z/OS V1R11 or it is z/OS V1R11 but is running sysplex=off. This is not allowed.

There is one exception to this rule: if the R11 system is the last (or only) sysplex-aware system in the sysplex, then z/OS UNIX will unmount the sysplex-aware file system before it moves it to the other system. In this case, the move is successful.

Figure 5-89 on page 265 shows that system SC65 is the only sysplex-aware system and owns the aggregate.

```
D GRS,RES=(SYSZIOEZ,IOEZLT.OMVS.HERING.TEST.ZFS*)
ISG343I 16.18.01 GRS STATUS 321
S=SYSTEMS SYSZIOEZ IOEZLT.OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                 003A       007EBE88 EXCLUSIVE    OWN
D GRS,RES=(*,OMVS.HERING.TEST.ZFS*)
ISG343I 18.24.56 GRS STATUS 259
S=SYSTEMS SYSDSN   OMVS.HERING.TEST.ZFS
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                 003A       007FF808   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   I
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                 003A       007FF2E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFS.DATAMCAT.SANDBOX.VSBOX01   O
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                 003A       007FF2E8   SHARE      OWN
S=SYSTEMS SYSVSAM  OMVS.HERING.TEST.ZFSMCAT.SANDBOX.VSBOX01   N
SYSNAME        JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC65      ZFS                 003A       007FF2E8   SHARE      OWN
```

Figure 5-89 Enqueue resource information for zFS aggregate

Figure 5-90 shows that the move to a sysplex-unaware system is successful in this situation.

```
#> zfsowner OMVS.HERING.TEST.ZFS
zFS Owner    : SC65     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
#> /usr/sbin/chmount -d SC64 /u/hering/test
#> rxdowner -d /u/hering/test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=Y
USS Owner    : SC64     - File System read-only=N
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N

#>
```

Figure 5-90 Moving the aggregate ownership from a last or only sysplex-aware system

When all systems are at level z/OS V1R11 and zFS is running sysplex-aware, all catch-up mounts will be successful if the aggregate can be opened at each member. If the aggregate

cannot be opened at a member (for example, because of no DASD connectivity), then the catch-up mount will fail and z/OS UNIX again will function ship file requests to the z/OS UNIX owner system.

Moving from a sysplex-aware to a sysplex-unaware environment

If you really need to move from a sysplex-aware environment to a sysplex-unaware environment to avoid an explicit unmount, and you can temporarily switch to read-only mode, then use this method. First, switch to read-only mode as shown in Figure 5-91 on page 266.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC65     - File System read-only=N
zFS Owner    : SC70     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> /usr/sbin/chmount -r test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC65     - File System read-only=Y
zFS Owner    : SC65     - Aggregate read-only=Y, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 5-91 Switching a sysplex-aware read-write file system to read-only

Next, move the UNIX System Services owner to a sysplex-unaware environment (for read-write mounted file systems), as shown in Figure 5-92.

```
#> /usr/sbin/chmount -d SC64 test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC64     - File System read-only=Y
zFS Owner    : SC65     - Aggregate read-only=Y, compat-mode=Y, sysplex-aware=Y

#>
```

Figure 5-92 Moving the read-only file system to a sysplex-unaware system

Finally, switch back to read-write mode; you will end up with the zFS owner also being on the sysplex-unaware system, as shown in Figure 5-93 on page 267.

```
#> /usr/sbin/chmount -w test
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST.ZFS
Aggregate    : OMVS.HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=Y
USS Owner    : SC64     - File System read-only=N
zFS Owner    : SC64     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=N

#>
```

Figure 5-93 Switching back the file system to read-write mode

5.10.5 Further migration considerations for zFS in z/OS V1R11

Several other changes are introduced in z/OS V1R11.

- zFS ignores a vnode cache limit that has been specified.
- Unmount processing for a zFS aggregate fails if the aggregate is quiesced, cloning, or growing, unless the FORCE option is issued.

Figure 5-94 illustrates an example of trying to unmount a quiesced aggregate in z/OS V1R11.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.HERING.TEST
Aggregate    : OMVS.HERING.TEST
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> zfsadm quiesce OMVS.HERING.TEST
IOEZ00163I Aggregate OMVS.HERING.TEST successfully quiesced
#> /usr/sbin/unmount test
FOMF0504I unmount error: 72 EF0969B5
EBUSY: The resource is busy
Description: An unmount was requested and the file system was busy with an
administration command.

#>
```

Figure 5-94 Unmounting a quiesced aggregate in z/OS V1R11 fails

In case of cloning or growing an aggregate, you may wait for the command to complete or issue an unmount command with the FORCE option to interrupt the administration command. If the aggregate is quiesced, first unquiesce it.

This unmount behavior was quite different in the past, as illustrated in Figure 5-95 on page 268.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
MCEVSF: z/OS Version 01.10.00
#> rxdowner -d test
MP Directory : /u/hering/test
File System  : OMVSU.HERING.TEST
Aggregate    : OMVSU.HERING.TEST
PFS Type     : ZFS
Local Sysname: MCEVSF   - File System local-client=N
USS Owner    : MCEVSF   - File System read-only=N
zFS Owner    : MCEVSF   - Aggregate read-only=N, compat-mode=Y, sysplex-aware=-
#> zfsadm quiesce omvsu.hering.test
IOEZ00163I Aggregate OMVSU.HERING.TEST successfully quiesced
#> /usr/sbin/unmount test
#> rxdowner -a OMVSU.HERING.TEST
zFS Owner    : MCEVSF   - Aggregate read-only=N, compat-mode=Y, sysplex-aware=-
#> rxlsaggr -qsd
OMVSU.HERING.TEST
#> zfsadm unquiesce omvsu.hering.test
IOEZ00166I Aggregate OMVSU.HERING.TEST successfully unquiesced
#> rxdowner -a OMVSU.HERING.TEST

Aggregate OMVSU.HERING.TEST cannot be found.
#>
```

Figure 5-95 Unmounting a quiesced aggregate in z/OS V1R10

In Figure 5-95, system MCEVSF is a z/OS V1R10 single system. Aggregate OMVSU.HERING.TEST is mounted and then quiesced and successfully unmounted. Nevertheless, the aggregate is still kept attached and quiesced, as **rxlsaggr -qsd** lists all quiesced aggregates. After unquiescing the aggregate, it is automatically detached.

Attaching a compat mode aggregate for growing

In z/OS V1R11 it is no longer possible to attach a multifile system aggregate. Nevertheless, you still can attach a compat mode aggregate. This is a valid method of growing a zFS aggregate without making it available for use, as illustrated in Figure 5-96.

```
#> echo $(sysvar SYSNAME): $(uname -I) Version $(uname -Iv).$(uname -Ir)
SC74: z/OS Version 01.11.00
#> zfsadm attach HERING.TEST.GROW
IOEZ00117I Aggregate HERING.TEST.GROW attached successfully
#> zfsowner hering.test.grow
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=N, sysplex-aware=-
#> zfsadm aggrinfo HERING.TEST.GROW
HERING.TEST.GROW (R/W MULT): 566 K free out of total 720
#> zfsadm grow HERING.TEST.GROW -size 0
IOEZ00173I Aggregate HERING.TEST.GROW successfully grown
HERING.TEST.GROW (R/W MULT): 1286 K free out of total 1440
#> zfsadm detach HERING.TEST.GROW
IOEZ00122I Aggregate HERING.TEST.GROW detached successfully
#>
```

Figure 5-96 Growing an attached zFS compat mode aggregate in z/OS V1R11

Using a zFS sysplex-aware /dev structure

If the /dev structure (which is the file system containing the /dev directory) is accidentally moved to a remote system, then UNIX System Services cannot use the dev devices correctly, if the PFS is of type HFS or zFS, being sysplex-unaware. Function shipping to the owning system is not used in this situation.

Figure 5-97 shows the error information displayed in this situation when trying to open an OMVS shell session.

```
No session was started.  No pseudo-TTYs are available.+
Function = stat(), ending name = '/dev/ptyp0000', return value = -1, errno = 129
(X'00000081'), reason code = 053B006C, description = 'EDC5129I No such file or
directory.'
```

Figure 5-97 Error information when opening a shell session if /dev is owned remotely

However, if the file system containing the /dev directory is a sysplex-aware zFS aggregate, then no problem will occur, regardless of which system is the UNIX System Services and which is the zFS owner.

5.10.6 Exploiting the UNIX Unmount (Remount) samemode

zFS can use the UNIX Unmount (Remount) samemode function, introduced in z/OS V1R11, in the following situations:

- To dynamically change the zFS attributes for an aggregate that is active
- To make new candidate volumes available while the aggregate is active
- For force a change of the zFS owner of an aggregate to a specific system

The following sections explain the use of this function in more detail.

Restriction: Use of the UNIX Unmount (Remount) samemode function is only possible if the minimum required LFS protocol level of z/OS V1R11 is used by *all* systems in the UNIX System Services sysplex file system sharing environment.

Dynamically change the zFS attributes for an aggregate

Figure 5-98 on page 270 shows a situation with an aggregate that is active with auto-extension disabled; this means aggrgrow is off. However, at the system level, aggrgrow is on.

After using the Unmount (Remount) samemode function for this zFS file system, auto-extension is enabled.

```
#> rxlsaggr OMVS.TESTAGGR.ZFS | grep Auto-extend
Auto-extend . . . . . : Disabled
#> zfsadm configquery -aggrgrow
IOEZ00317I The value for configuration option -aggrgrow is ON.
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> /usr/sbin/chmount -s test
#> rxlsaggr OMVS.TESTAGGR.ZFS | grep Auto-extend
Auto-extend . . . . . : Enabled
#>
```

Figure 5-98 Dynamically enable auto-extension for an active zFS aggregate

Adding new candidate volumes

If a sysplex-aware zFS aggregate runs out of space, you cannot move the aggregate to another system to reopen it and make new candidate volumes known to zFS (see 2.13.1, “Adding additional candidate volumes” on page 61, for an explanation of that case).

However, now you can use the samemode option of unmount to force this action, as shown in Figure 5-99 and Figure 5-100 on page 271.

```
#> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TEST.ZFS
Aggregate    : OMVS.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#> tsocmd "ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)"
ALTER 'OMVS.TEST.ZFS.DATA' ADDVOLUMES(*)
ALTERED ALLOCATION STATUS FOR VOLUME *      IS 0
ENTRY OMVS.TEST.ZFS.DATA ALTERED
READY
END
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00326E Error 133 extending OMVS.TEST.ZFS
#>
```

Figure 5-99 Adding new candidate volumes to a zFS aggregate as no space is left

Using the Unmount (Remount) samemode function reopens and thereby makes the candidate volume known to zFS, as shown in Figure 5-100 on page 271.

```
#> /usr/sbin/chmount -s test
#> zfsadm grow -aggregate omvs.test.zfs -size 0
IOEZ00173I Aggregate OMVS.TEST.ZFS successfully grown
OMVS.TEST.ZFS (R/W COMP): 220950 K free out of total 223200
#>
```

Figure 5-100 Make the new candidate known to the sysplex-aware zFS and grow it

Changing the zFS ownership of an aggregate

As shown in Figure 5-101, you can change the zFS ownership of a read-write sysplex-aware file system:

- Change the z/OS UNIX ownership to the system you want to have the zFS ownership.
- Then do a remount samemode.

```
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC74     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

$> sudo /usr/sbin/chmount -d SC75 test
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC74     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

$> sudo /usr/sbin/chmount -s test
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : OMVS.TESTAGGR.ZFS
Aggregate    : OMVS.TESTAGGR.ZFS
PFS Type     : ZFS
Local Sysname: SC74     - File System local-client=N
USS Owner    : SC75     - File System read-only=N
zFS Owner    : SC75     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y

$>
```

Figure 5-101 Changing the zFS owner of a sysplex-aware aggregate

When adding candidate volumes as shown in Figure 5-99 on page 270, keep this fact in mind. You may want to change the UNIX System Services owner first because zFS has probably chosen its owning system correctly.

Restriction: If a read-only clone is mounted in parallel, then this method for moving the zFS owner will not be successful.

5.11 zFS sysplex-aware on a file system basis (R11 OA29619)

Beginning with z/OS V1R11, zFS introduced the ability to be able to enable zFS read-write file systems to be sysplex-aware. This means that USS sends all file requests directly down to the local zFS physical file system (PFS). It does not “function ship” the requests. The local zFS either sends the request to the owning zFS system or it satisfies the request from the local zFS cache. In many cases, this improves the pathlength over the function shipping model. The request and data flow is shown in Figure 5-103 on page 277.

5.11.1 zFS APAR OA29619

APAR OA29619 provides a new function that allows users to specify which zFS read-write file systems are to be made sysplex-aware. To enable zFS sysplex-aware on a file system basis, specify sysplex=filesys in the IOEFSPRM configuration file. Leave the sysplex_filesys_sharemode with its default of norwshare. You can specify it in a shared IOEFSPRM configuration file and each system picks up the specification in a rolling IPL.

Note: The sysplex option is ignored by previous releases.

This now .provides two possible sysplex settings:

- You can choose individual file systems to be sysplex-aware (requires zFS APAR OA29619).

  Attention: APAR OA29619 is for both z/OS V1R11 and z/OS V1R12.

- You can choose to have all zFS file systems be sysplex-aware (as with the R11 base code already).

APAR OA29619

Beginning with z/OS V1R11 and with z/OS V1R12, zFS can be sysplex-aware for read-write mounted file systems. With APAR OA29619, zFS provides new support for being sysplex-aware on a file system basis.

Table 5-3 zFS sysplex-aware on a file system basis

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R11 (3B0)	UA52930	OA29619

APAR OA29619 was introduced to enable zFS sysplex-aware on a file system basis, by specifying sysplex=filesys in the IOEFSPRM configuration file. In the current system at z/OS V1R11, leave the sysplex_filesys_sharemode with its default of norwshare. You can specify it in a shared IOEFSPRM configuration file and each system picks up the specification in a rolling IPL. The sysplex option is ignored by previous releases. Any z/OS V1R9 or V1R10 systems in the shared file system environment must have zFS APAR OA29786 active before you can have zFS sysplex=filesys active on any z/OS V1R11 system.

With the new support provided in APAR OA29619, several goals have been achieved by changing the support provided by z/OS V1R11, as follows:

- An environment has been provided where performance problems caused by mixed environments do not occur.
- The new possibilities allow the usage of zFS sysplex-aware file systems with servers that cannot tolerate sysplex-aware file systems such as z/OS SMB server and Fast Response Cache Accelerator support of the IBM HTTP Server in the same shared file system environment.
- It gives more granular control and flexibility over the decision to make zFS file systems sysplex-aware.

Important: Before you can run zFS sysplex-aware on a file system basis, you must have z/OS V1R11 zFS APAR OA29619 and z/OS V1R11 UNIX APAR OA29712 installed and active on all of your z/OS V1R11 systems. In addition, conditioning APAR OA29786 must be installed and active on your V1R10 systems. Finally, if you use the SMB server, APAR OA31112 must also be installed.

z/OS UNIX APAR OA29712

The zFS APAR OA29619 provides the capability to set read-write sysplex-awareness on a file system (mount) basis, which requires the UNIX support provided with APAR OA29712.

With APAR OA29712, USS is providing support for allowing a PFS to set the read-write sysplex-awareness of a file system during a mount. Prior to this support, the sysplex-awareness of a PFS that was set during PFS initialization would apply to all file systems mounted on that system for that PFS. This affects all flows where USS sends a vfs_mount to the PFS, which includes the following types of actions:

- Mount

  A local mount means that z/OS UNIX issues a successful mount to the local PFS, which in this case is zFS. z/OS UNIX does this when either the file system is mounted sysplex-aware for that mode (read-write or read-only) or the system is the z/OS UNIX owner. When a file system is locally mounted on the system, z/OS UNIX does not function ship requests to the z/OS UNIX owning system.

- Catchup mount

  When a file system mount is successful on a system in a shared file system environment, z/OS UNIX automatically issues a corresponding local mount, the catchup mount, to every other system's PFS that is running sysplex-aware for that mode (read-write or read-only) when zFS is running sysplex=on or for each zFS file system that is mounted RWSHARE when zFS is running sysplex=filesys. If the corresponding local mount is successful, z/OS UNIX does not function ship from that system to the z/OS UNIX owning system when that file system is accessed. Rather, the file request is sent directly to the local PFS. This is sometimes referred to as Client=N. If the corresponding local mount is unsuccessful (for instance, DASD is not accessible from that system), z/OS UNIX function ships requests to the z/OS UNIX owning system when that file system is accessed (message BPXF221I might be issued). This is sometimes referred to as Client=Y.

- Remount

  Remount allows you to change a mounted file system from read-only to read-write or from read-write to read-only without affecting lower mounted file systems.

- Dead system recovery

  When a system leaves the sysplex, use the recovery options for the mounted file systems.

- Unowned file system recovery or catchup

  This can occur, for example, when a physical I/O path from another system to the volume where the file system resides is not available. As a result, the file system becomes unowned; if this happens, you will see message BPXF213E. This is true if the file system is mounted either read/write or read-only. The file system still exists in the file system hierarchy so that any dependent file systems that are owned by another system are still usable. However, all file operations for the unowned file system will fail until a new owner is established. The sharedfile system support will continue to attempt recovery of AUTOMOVE file systems on all systems in the sysplex that are enabled for shared file system. If a subsequent recovery attempt succeeds, the file system moves from the unowned to the active state.

- Newroot/altroot

  Beginning with z/OS V1R11, in a sysplex configuration, the alternate sysplex root file system is a hot standby for the sysplex root file system that is used to replace the current sysplex root file system when that system becomes unowned. The alternate sysplex root file system is established by using the ALTROOT statement in the BPXPRMxx parmlib member during OMVS initialization or by using the SET OMVS command.

- PFS termination/restart

  Move file systems to any system using the AUTOMOVE settings.

5.11.2 Some information about zFS sysplex sharing behavior

We now provide some information about how zFS processed file systems in the past before z/OS V1R11 and with the base support in z/OS V1R11.

FIle system ownership

IBM defines a file system owner as the system that coordinates sysplex activity for a particular file system. In a shared file system environment, there is also the concept of file system ownership. The owner of a file system is the first system that processes the mount. This system always accesses the file system locally; that is, the system does not access the file system through a remote system. Other non-owning systems in the sysplex access the file system either locally or through the remote owning system, depending on the PFS and the mount mode.

z/OS UNIX file system owner

The file system owner is the system to which file requests are forwarded. Having the appropriate owner is important for performance. We use the term z/OS UNIX file system owner to mean the owner of the zFS file system as z/OS UNIX recognizes it. This is typically the system where the file system is first mounted, but it can differ from the zFS file system owner.

zFS file system owner

zFS has its own concept of file system ownership. We call this owner the zFS file system owner. This is also typically the system where the file system is first mounted in a sysplex-aware environment. File requests to sysplex-aware file systems are sent directly to the local zFS PFS, rather than being forwarded to the z/OS UNIX file system owner. The local zFS PFS forwards the request to the zFS file system owner, if necessary. The z/OS UNIX file system owner can be different from the zFS file system owner.

Note: In reality, zFS owns aggregates. Generally, we simplify this to say “zFS file system owner” because zFS compatibility mode aggregates only have a single read-write file system.

Function shipping

Function shipping means that a request is forwarded to the owning system and the response is returned back to the requestor through XCF communications.

Sysplex-aware file system

A file system can be mounted sysplex-aware or non-sysplex-aware. When a file system is mounted sysplex-aware, it means that the file system is locally mounted on every system (when the PFS is capable of handling a local mount on every system, that is, the PFS is running sysplex-aware) and therefore, file requests are handled by the local PFS. All read-only mounted file systems are always mounted sysplex-aware. HFS read-write mounted file systems are always mounted non-sysplex-aware. This means that file requests from non-z/OS UNIX owning systems are always function shipped by z/OS UNIX to the z/OS UNIX owning system where the file system is locally mounted and the I/O is actually done. Beginning with z/OS V1R11, zFS read-write mounted file systems can be mounted sysplex-aware when zFS is configured as sysplex-aware (zFS IOEFSPRM option sysplex=on or zFS IOEFSPRM option sysplex=filesys).

5.11.3 zFS R/W mounted file system being sysplex-unaware

Figure 5-102 on page 276 shows an example for a R/W mounted zFS file system in z/OS V1R10 or older or in z/OS V1R11 when zFS has been started with `sysplex=off`. One system is the owning system, SY2, and the other systems are “clients”. The file system is only locally mounted on the owning system, but also externally mounted and available on all systems.

Applications running on the owning system (SY2) access the file system locally and without any XCF communications.

But applications that run on the other systems (SY1 and SY3), access the file system through the use of z/OS UNIX function shipping with the XPFS and using XCF communications. That is, the request is forwarded by the z/OS UNIX on that system, SY1 or SY3, to the z/OS UNIX running on the owning system SY2, which then uses the local zFS.

The response goes back along the same path. So, access to the file system from systems other than the owner—they are called client systems—involves XCF communications.

Note: This makes it important to have the z/OS UNIX owning system be the system that is doing the most accesses. This file system is non-sysplex-aware.

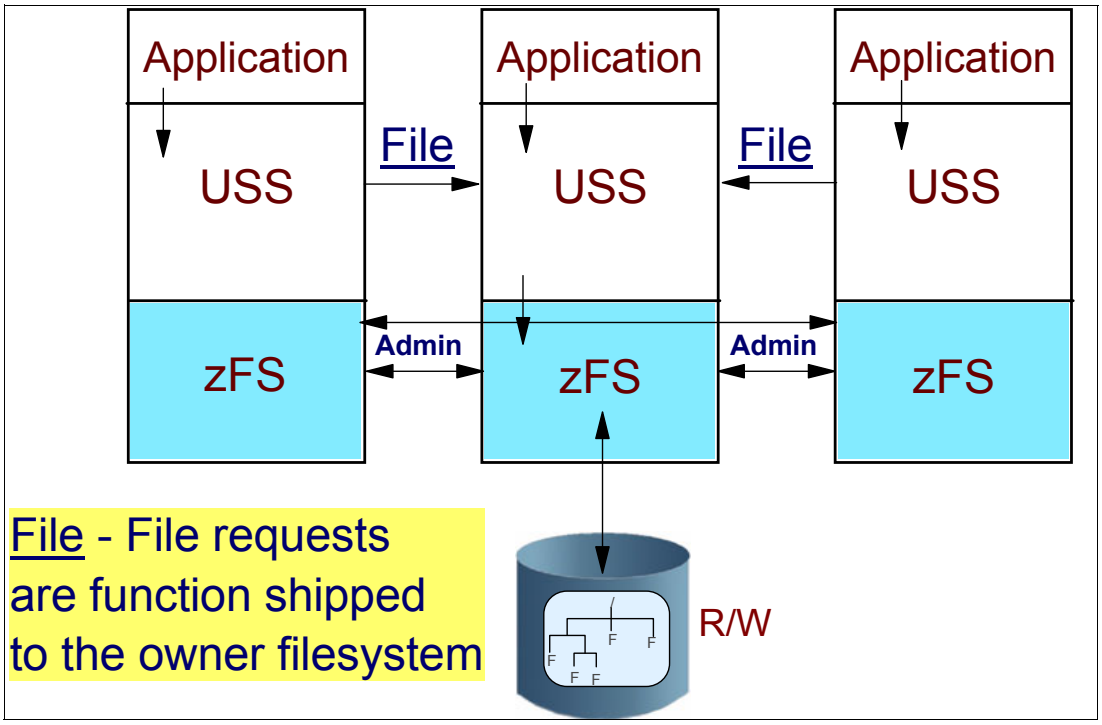


Figure 5-102 zFS R/W mounted file system being sysplex-unaware

Running all systems with zFS sysplex-unaware

If you decide to run zFS non-sysplex-aware, shared file system support works as in prior releases. Ensure that you do not specify sysplex=on or sysplex=filesys in your zFS IOEFSPRM configuration options file, as shown in “Setting up the zFS parameters” on page 282.

5.11.4 zFS being sysplex-aware in z/OS V1R11 for R/W mounts

Beginning with z/OS V1R11, zFS is sysplex-aware for file systems mounted read-write. This means that USS sends all file requests directly down to the local zFS physical file system (PFS). It does not “function ship” the requests. The local zFS either sends the request to the owning zFS system or it satisfies the request from the local zFS cache. In many cases, this improves the pathlength over the function shipping model. The request and data flow is shown in Figure 5-103 on page 277.

z/OS V1R11 supports sysplex=on

With z/OS V1R11, the zFS PFS allows a file system to be locally accessed on all systems in a sysplex for a particular mode, and then the PFS is sysplex-aware for that mode. Therefore, performance can be improved for zFS in the USS sysplex shared file system mode by zFS becoming sysplex-aware for R/W mounted file systems. This is a new specification that is specified with this new option:

```
sysplex=on
```

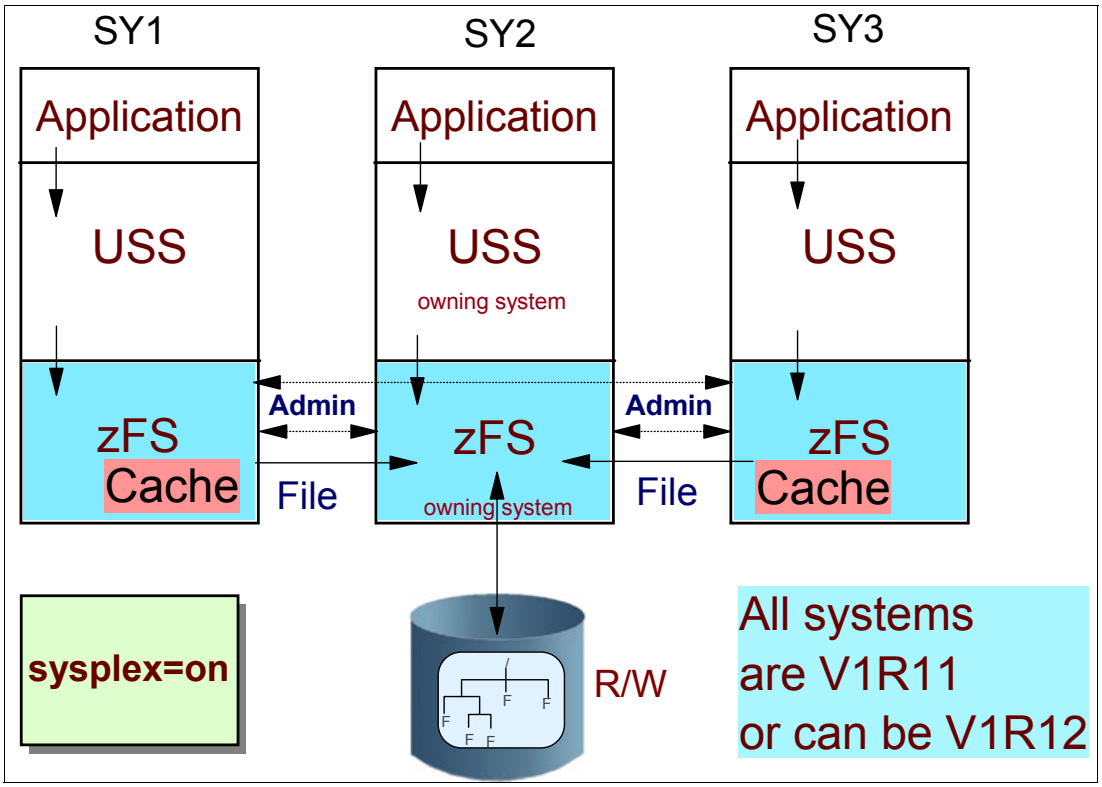


Figure 5-103 R/W mounted zFS sysplex-aware with zFS R11 started sysplex=on

Note: There is a new option available with APAR OA29619 for z/OS V1R11 and z/OS V1R12 systems, sysplex=filesys; see “Setting up the zFS parameters” on page 282.

This zFS sysplex file system support improves the response time for zFS file system requests:

- zFS becomes sysplex-aware for zFS read-write file systems.
- zFS uses less XCF communications as a new client caching is introduced.

z/OS V1R11 systems with sysplex=on

Figure 5-103 shows the z/OS V1R11 zFS sysplex-aware for read-write on a system basis support. When zFS runs sysplex-aware for read-write on all systems, a read-write mounted file system is locally mounted on all systems.

There is still a z/OS UNIX owning system but there is no z/OS UNIX function shipping to the owner. The requests from applications on any system are sent directly to the local zFS on each system. This means it is now the responsibility of the local zFS to determine how to access the file system.

One of the systems is known as the zFS owning system. This is the system where all I/O to the file system is done. zFS uses function shipping to the zFS owning system to access the file system.

Note: If this were all that zFS does, it would be essentially the same as the z/OS UNIX function shipping as shown in Figure 5-102 on page 276.

However, each zFS client system has a local cache where it keeps the most recently read file system information. So, in many cases (when the data is still in the cache), zFS can avoid the zFS function shipping (and the XCF communications) and satisfy the request locally.

Note: Any zFS read-write file system owned on SY2 is sysplex-aware.

zFS command output in z/OS V1R11 with zFS base support

Figure 5-104 shows zFS and MVS system commands with information about zFS aggregates.

```
$> zfsadm aggrinfo hering.test.zfs -long
HERING.TEST.ZFS (R/W COMP): 451 K free out of total 2880
version 1.4
auditfid E2C2D6E7 F1C3009C 0000
sysplex-aware
            52 free 8k blocks;         35 free 1K fragments
           112 K log file;             16 K filesystem table
             8 K bitmap file

$> cn f zfs,query,file
 F ZFS,QUERY<FILE
IOEZ00438I Starting Query Command FILESETS.
 File System Name                                Aggr #  Flg  Operations
 ----------------------------------------------  ------  ---  ----------
...
 HERING.TEST.ZFS                                     83  AMS           3
...
 IOEZ00025I zFS kernel: MODIFY command - QUERY,FILE completed successfully.
$> zfsowner hering.test.zfs
zFS Owner    : SC70     - Aggregate read-only=N, compat-mode=Y, sysplex-aware=Y
$>
```

Figure 5-104 zFS command output in z/OS V1R11 with zFS base support

All these commands provide the information that zFS aggregate `HERING.TEST.ZFS` is mounted sysplex-aware.

Note: For REXX commands CN and **`zfsowner`**, shown in Figure 5-104, see *z/OS Distributed File Service zSeries File System Implementation z/OS V1R11,* SG24-6580 (or later) for a detailed description of the zfsowner utility and the CN procedure.

zFS API output in z/OS V1R11 with zFS base support

The zFS Application Programming Interface (API) command `List Aggregate Status (Version 2)` indicates when an aggregate is sysplex-aware:

```
AGGR_STATUS, as_flags2, 0x40 (AS_SYSPLEXAWARE)
```

Note: The List Aggregate Status subcommand call is an aggregate operation that returns information about a specified attached aggregate on this system. Version 2 returns additional flags and fields.

See *z/OS Distributed File Service zSeries File System Administration*, SC24-5989 for more information.

Note: This `as_flags2` bit shows the zFS aggregate sysplex-aware state independent of the zFS release and service level used.

The utility zfsowner uses this bit to decide whether a R/W mounted zFS aggregate is sysplex-aware.

zFS mixed environment

Figure 5-105 on page 280 shows a zFS mixed environment with systems that are sysplex=off and sysplex=on.

WIth SY1, running sysplex=off, there are three problems with this mixed environment, as follows:

- Since zFS is running `sysplex=off`, SY1 cannot present requests for that file system to the local zFS. This is unused for that file system. This means that z/OS UNIX must forward requests to the z/OS UNIX owner, which is SY2 in this case.
- Furthermore, the z/OS UNIX directory lookup cache is not active on SY1 because the z/OS UNIX owner system SY2 is a sysplex=on system. The loss of this z/OS UNIX directory cache can increase the rate of transmissions to the z/OS UNIX owner SY2 for pathname resolution within the file system.
- Since the z/OS UNIX owner SY2 and the zFS owner SY3 are different, z/OS UNIX on SY1 forwards requests to the z/OS UNIX owner SY2. Then, z/OS UNIX calls zFS on system SY2. Since the zFS owner is on SY3, if the request is a write request or a read that has a cache-miss, a message is sent to SY3 by zFS to resolve the request. This means that two messages are sent in the sysplex for a user application request from SY1 instead of 1 or 0.

Thus a mixed environment (mixed environment with sysplex=off and sysplex=on systems) can lead to increased transmissions from sysplex=off members and a double transmit, both of which cannot occur in a non-mixed environment.

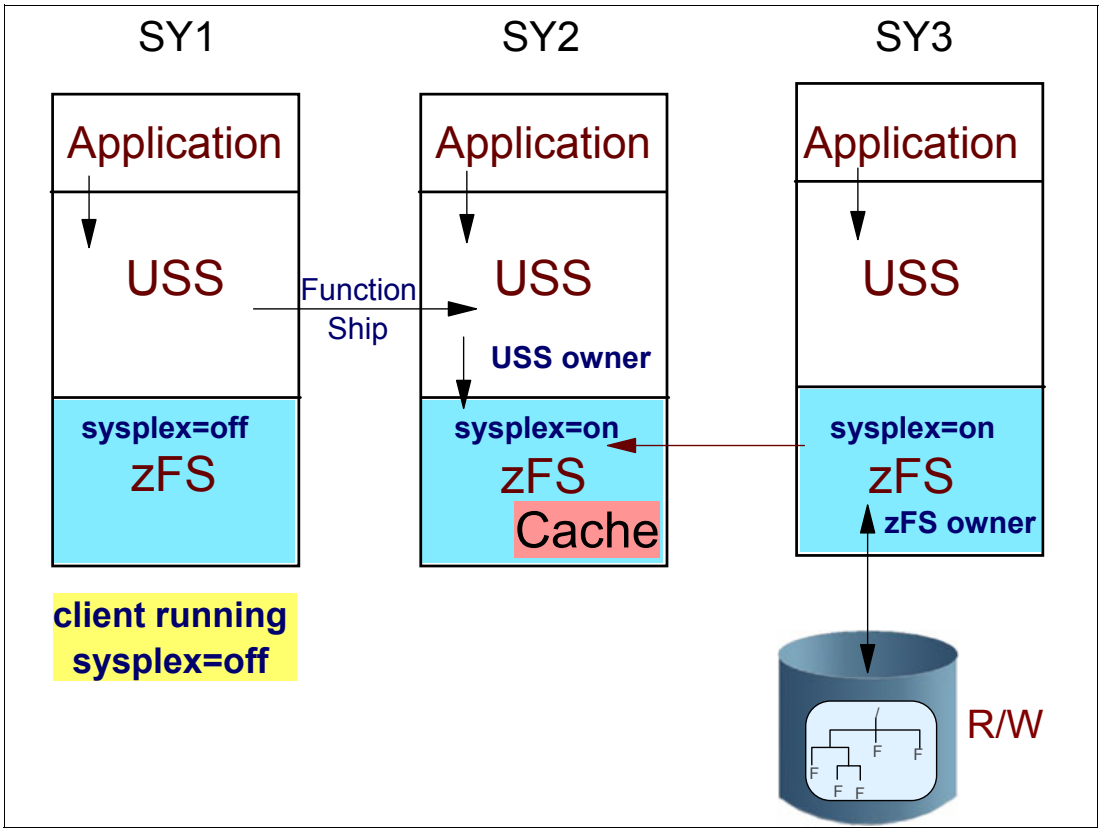


Figure 5-105 zFS mixed environment

5.11.5 New and older important zFS configuration options

Following are new configuration options that can be defined either in the IOEPRMxx parmlib member or in the IOEFSPRM DD member in the ZFS procedure. These parameters are changed as indicated by z/OS V1R11 and with APAR OA29619.

sysplex={ on | off | filesys}

Specifies whether zFS should run sysplex-aware and if so, whether zFS is sysplex-aware on a file system basis (sysplex=filesys) or sysplex-aware on a system basis (sysplex=on). When sysplex=off, zFS does not automatically move zFS ownership of the aggregate. (Changed with APAR OA29619, `filesys` is new; see “Setting up the zFS parameters” on page 282).

Default Value: OFF

Expected Value: Off, filesys, or on if BPXPRMxx specifies SYSPLEX(YES). Off if BPXPRMxx does not specify SYSPLEX(YES).

Example: sysplex=on

sysplex_filesys_sharemode

Specifies the default for the mount PARM for a zFS read-write file system mounted on a sysplex=filesys system (new with APAR OA29619; see “Setting up the zFS parameters” on page 282).

Default Value: norwshare when running zFS as sysplex=filesys or <no value> otherwise

Expected Value: rwshare or norwshare

Example: sysplex_filesys_sharemode=rwshare

token_cache_size

Specifies the maximum number of tokens in the server token manager cache to use for cache consistency between zFS members. The default value is double the number of vnode_cache_size (new with z/OS V1R11).

file_threads = 40

The file_threads configuration option determines how many threads are in the pool to handle requests from remote clients (new with z/OS V1R11).

client_cache_size=128M

The client_cache_size configuration option specifies the size of the user data cache for client access. The user_cache_size is used for the user data cache for server access. (new with z/OS V1R11)

client_reply_storage = 10M

The client_reply_storage configuration option specifies how much storage is used for sysplex server replies, (new with z/OS V1R11).

Note: With APAR OA29619, the default value is changed from 40M to 10M.

recovery_max_storage=256M

Indicates the maximum amount of zFS address space storage to use for concurrent log recovery during multiple concurrent aggregate mounts (new with z/OS V1R11).

sysplex_admin_level

Specifies the zFS XCF communication interface level that zFS is running as sysplex-aware. This is only valid for systems running zFS V1R9 or V1R10 with the proper APARs applied. The sysplex_admin_level option is ignored in z/OS V1R11 because zFS runs at sysplex_admin_level 3 on z/OS V1R11. One (1) indicates that zFS running on z/OS V1R9 or V1R10 is in preconditioning state and is using interface level 1. zFS uses the existing XCF protoc ol but also supports enough of the new XCF protocol for zFS interface level 2 to run properly. Two (2) indicates that zFS running on z/OS V1R9 or V1R10 is in toleration mode and is using interface level 2. zFS uses the new XCF protocol and is able to communicate with other zFS instances running at interface level 1 to display aggregate information for aggregates owned by zFS interface level 1 systems. At level 2, zFS is able to communicate with other zFS instances running at z/OS V1R11 (level 3). The value must be 2 on all members of the sysplex in order to bring z/OS V1R11 zFS into the shared file system environment.

Default Value: 3 in z/OS V1R11 and V1R12. 1 in z/OS V1R9 and V1R10

Expected Value: In z/OS V1R11, this option is ignored. In z/OS V1R9 and V1R10, 1 or 2.

Example: sysplex_admin_level=2

5.11.6 Setting up the zFS parameters

There is a new value for the IOEPRMxx or IOEFSPRM sysplex option, filesys. This option is for z/OS V1R11 and z/OS V1R12 systems, as follows:

```
sysplex=off | on | filesys
```

Where:

sysplex=on zFS can be configured sysplex=on to treat all zFS read-write mounted file systems owned on that system as sysplex-aware file systems.

sysplex=filesys zFS can be configured sysplex=filesys to allow some zFS read-write mounted file systems owned on that system to be sysplex-aware file systems and some to be non-sysplex aware file systems.

zFS must be running sysplex-aware for read-write in order to allow a zFS read-write file system to be mounted as sysplex-aware. Specifying `sysplex=filesys` allows individual control over which zFS read-write file systems are sysplex-aware or not (in a shared file system environment).

Note: A z/OS V1R10 system requires APAR OA29786.

5.11.7 Using sysplex=filesys

Whether the PFS is running sysplex-aware on a file system basis (referred to as filesys), or sysplex-aware on a system basis (referred to as file), or not sysplex-aware (referred to as admin-only), and the zFS XCF protocol level (normally 3 for zFS on z/OS V1R11) when running in a shared file system environment, when filesys is indicated, the default mount PARM (NORWSHARE or RWSHARE) also displays. See “Expected Value: rwshare or norwshare” on page 281.

When running with `sysplex=filesys`, a new mount parameter can be used to specify whether a file system is mounted sysplex-aware:

```
RWSHARE | NORWSHARE
```

When sysplex=filesys, the new IOEPRMxx or IOEFSPRM options control the default mount parameter:

```
sysplex_filesys_sharemode=[ norwshare | rwshare ]
```

When you run sysplex=filesys, the zFS PFS runs sysplex-aware, but each zFS file system is mounted non-sysplex aware (by default). zFS is enabled to allow zFS read-write file systems to be sysplex-aware but it must be explicitly requested on a file system basis. Note that although it is possible to change the sysplex_filesys_sharemode from its default of norwshare to rwshare at IPL using the IOEFSPRM option or dynamically using the **`zfsadm config`** command, this is not recommended while migrating from sysplex=off to sysplex=filesys because it can cause performance problems.

Note: The new `sysplex=filesys` is documented in a refresh of manual *z/OS Distributed File Service zFS Administration*, SC24-5989.

zFS command output

You can determine whether zFS on a particular system is running sysplex-aware or not by using the F ZFS,QUERY,LEVEL operator command. If the command indicates sysplex(file) or

sysplex(filesys), zFS is sysplex-aware. If the command indicates sysplex(admin-only), zFS is not sysplex-aware.

Note: In z/OS V1R11 and later in a shared file system environment, when zFS is running with sysplex=on in the IOEFSPRM configuration file, the sysplex level is (file), and the interface level is (3).

When zFS is running sysplex=filesys in the IOEFSPRM configuration file, the sysplex level is (filesys,norwshare) or (filesys,rwshare) depending on the sysplex_filesys_sharemode and the interface is (3). Otherwise, the zFS sysplex level is (admin-only) and the interface is (3).

Figure 5-106 shows zFS sysplex status information data.

```
/* From the OMVS shell ..............
$> zfsadm configquery -sysplex_state
IOEZ00317I The value for configuration option -sysplex_state is 3.
$> zfsadm configquery -syslevel
IOEZ00644I The value for configuration option -syslevel is:
zFS kernel: z/OS    zSeries File System
Version 01.11.00 Service Level OA29619 - HZFS3B0.
Created on Wed Jan 13 09:39:20 EST 2010.
sysplex(filesys,norwshare) interface(3)

/* MVS command.....................
f zfs,query,level
IOEZ00639I zFS kernel: z/OS    zSeries File System
 Version 01.11.00 Service Level OA29619 - HZFS3B0.
 Created on Wed Jan 13 09:39:20 EST 2010.
 sysplex(filesys,norwshare) interface(3)
 IOEZ00025I zFS kernel: MODIFY command - QUERY,LEVEL completed successfully.
```

Figure 5-106 zFS sysplex status queries

zFS API output

Figure 5-107 shows the output of the API example for syslevel, which is the same as what command **zfsadm configquery -syslevel** provides.

```
$> zfs_query_syslevel
zFS kernel: z/OS    zSeries File System
Version 01.11.00 Service Level OA29619 - HZFS3B0.
Created on Wed Jan 13 09:39:20 EST 2010.
sysplex(filesys,norwshare) interface(3)
$>
```

Figure 5-107 Displaying the zFS syslevel information using APIs

5.11.8 zFS sysplex-aware on a file system basis

With z/OS V1R12 and V1R11, to enable zFS sysplex-aware on a file system basis, specify sysplex=filesys in the IOEFSPRM configuration file. Leave the sysplex_filesys_sharemode with its default of norwshare. You can specify it in a shared IOEFSPRM configuration file and then each system picks up the specification in a rolling IPL. The sysplex option is ignored by

previous releases. Any V1R10 systems in the shared file system environment must have zFS APAR OA29786 active before you can have zFS sysplex=filesys active on any z/OS V1R11 and z/OS V1R12 systems.

Note: If you decide that you want some zFS read-write file systems to be sysplex-aware, you can run zFS sysplex-aware on a file system basis. Roll this support through all your sysplex members (this assumes all your members are at z/OS V1R11 with the V1R11 APAR OA29619 applied). You cannot change from zFS non-sysplex aware to zFS sysplex-aware on a file system basis dynamically. After changing IOEFSPRM, you must perform an IPL or a restart of zFS. IPL is the recommended method because in many ways it is less disruptive than a restart of zFS, which can cause zFS file systems to become unmounted.

Attention: When you run sysplex=filesys, the zFS PFS runs sysplex-aware, but each zFS file system is mounted non-sysplex-aware (by default). zFS is enabled to allow zFS read-write file systems to be sysplex-aware but it must be explicitly requested on a file system basis. Note that although it is possible to change the sysplex_filesys_sharemode from its default of norwshare to rwshare at IPL using the IOEFSPRM option or dynamically using the **`zfsadm config`** command, this is not recommended while migrating from sysplex=off to sysplex=filesys because it can cause performance problems.

Mixed sysplex example - sysplex-aware on file system basis

Figure 5-108 on page 285 shows three systems with zFS sysplex-aware on a file system basis (`sysplex=filesys`).

To enable zFS sysplex-aware on a file system basis, specify sysplex=filesys in the IOEFSPRM configuration file. Leave the sysplex_filesys_sharemode with its default of norwshare. You can specify it in a shared IOEFSPRM configuration file and each system picks up the specification in a rolling IPL. The sysplex option is ignored by previous releases. Any z/OS V1R9 or V1R10 systems in the shared file system environment must have zFS APAR OA29786 active before you can have zFS sysplex=filesys active on any z/OS V1R11 system.

After you have sysplex=filesys active on all your systems, you can consider which zFS read-write file systems you want to be sysplex-aware. Good candidates are zFS read-write file systems that are accessed from multiple systems or are mounted with AUTOMOVE and might be moved by z/OS UNIX (as a result shutdown or IPL) to systems that do not necessarily do the most accesses.

Figure 5-108 on page 285 shows two zFS read-write file systems on a sysplex running zFS sysplex-aware on file system basis on all members. One file system is mounted `NORWSHARE` and the other is mounted `RWSHARE`. They are both z/OS UNIX-owned by system SY2.

The norwshare file system is a non-sysplex aware file system. It is only locally mounted on the z/OS UNIX owner and requests from z/OS UNIX clients are function shipped to the z/OS UNIX owner by z/OS UNIX. A **`df –v`** command for the norwshare file system FS1 from SY1 would display `Client=Y`, as shown in Figure 5-109 on page 287.

The other file system shown in Figure 5-108 on page 285 is mounted rwshare. It is a sysplex-aware file system and locally mounted on all systems and z/OS UNIX never function ships requests to the z/OS UNIX owner. A **`df –v`** command for the rwshare file system FS2 from SY1 would display `Client=N`. Figure 5-110 on page 288 demonstrates this situation.

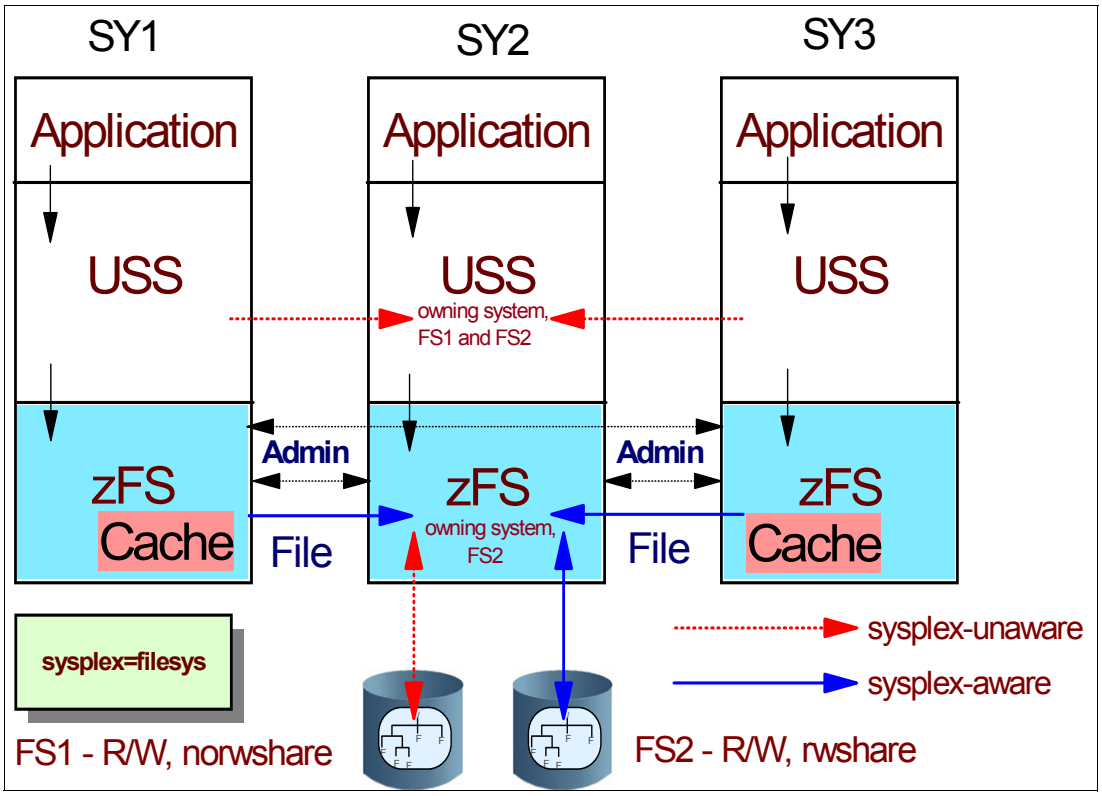


Figure 5-108 zFS sysplex-aware on a file system basis

When you run zFS sysplex-aware on a file system basis on all your members, the zFS Physical File System initializes as sysplex-aware but it can individually determine which file system is sysplex-aware and which is not based on the mount parameters rwshare and norwshare.

Note: MOUNT commands for file systems in Figure 5-108 on page 285.

- For file system FS2

```
MOUNT FILESYSTEM('OMVS.PRV2.COMPAT.AGGR001') TYPE(ZFS)
MODE(RDWR) MOUNTPOINT('/usr/mountpt2') PARM('RWSHARE')
```

- For file system FS1

```
MOUNT FILESYSTEM('OMVS.PRV1.COMPAT.AGGR001') TYPE(ZFS)
MODE(RDWR) MOUNTPOINT('/usr/mountpt1') PARM('NORWSHARE')
```

5.11.9 Running zFS sysplex-aware considerations

If you have (at least) APAR OA29619 and you intend to run zFS sysplex-aware on a file system basis (sysplex=filesys), you must ensure that you have (at least) APAR OA29786 on your prior releases (z/OS V1R10 if z/OS V1R12 is active) and z/OS V1R9 if z/OS V1R11 is your highest level of z/OS.

Note: In a shared file system environment, a zFS read-write sysplex-aware file system, in general, cannot be explicitly moved by z/OS UNIX to a prior release system or to a z/OS V1R11 system that is running zFS non-sysplex aware. In the special case that the zFS file system is sysplex-aware and every other system is doing z/OS UNIX function shipping of requests to the owning system, the zFS read-write file system can be explicitly moved, in most cases.

In a shared file system environment, when a zFS file system is mounted read-write and is z/OS UNIX-owned on a prior release (or is z/OS UNIX owned on a z/OS V1R11 system that is running zFS non-sysplex aware), a BPXF221I message similar to the following example may be issued on each system that receives a local catch-up mount, as follows:

```
BPXF221I FILE SYSTEM PLEX.JMS.AGGR006.LDS0006 FAILED TO MOUNT LOCALLY.
RETURN CODE = 00000079, REASON CODE = EF0969A8
THE FILE SYSTEM IS ACCESSIBLE ON THIS SYSTEM THROUGH A MOUNT ON A REMOTE
SYSTEM.
```

In this case, z/OS UNIX does forward file requests from the z/OS V1R11 system to the z/OS UNIX owning system.

Read-only file systems

zFS read-only mounted file systems are not affected by this support. However, if you remount a read-only file system to read-write (using the **chmount** command or the TSO/E UNMOUNT REMOUNT command), this is treated like a primary mount on the current z/OS UNIX owning system and MOUNT parameters (such as RWSHARE or NORWSHARE) or MOUNT defaults (such as the current sysplex_filesys_sharemode setting on that system) take effect when it is mounted read-write. When you remount back to read-only, those mount options are irrelevant again.

Note: These MOUNT parameters and MOUNT defaults do not take effect when a remount to the same mode is run.

Do not make any zFS read-write file systems sysplex-aware until you have all systems in the shared file system environment at z/OS V1R11 with sysplex=filesys active. To make a zFS read-write file system sysplex-aware when running sysplex=filesys on all systems, you must unmount the file system, specify RWSHARE as a mount PARM and then mount the file system. The following TSO/E example shows a zFS mount PARM of RWSHARE:

```
MOUNT FILESYSTEM('OMVS.PRV.COMPAT.AGGR001') TYPE(ZFS)
MOUNTPOINT('/etc/mountpt') PARM('RWSHARE') MODE(RO)
```

Consideration: Normally, if you are going to run any zFS read-write file systems sysplex-aware, you would run zFS sysplex-aware on a file system basis. This gives you more flexibility and control in determining which zFS file systems should be sysplex-aware for read-write. It also allows you to change whether a zFS read-write file system is sysplex-aware or not simply by unmounting and mounting it with a different mount PARM. However, if you want to give zFS complete control over which system actually does the I/O to the DASD for any zFS read-write file system and you don't want to concern yourself with which system is the z/OS UNIX owning system and you are not using any servers that are restricted from supporting zFS sysplex-aware, then you can run all your zFS read-write file systems sysplex-aware.

Interface level

In z/OS V1R11 and later in a shared file system environment, when zFS is running with sysplex=on in the IOEFSPRM configuration file, the sysplex level is (file), and the interface level is (3). When zFS is running sysplex=filesys in the IOEFSPRM configuration file, the sysplex level is (filesys,norwshare) or (filesys,rwshare) depending on the sysplex_filesys_sharemode and the interface is (3). Otherwise, the zFS sysplex level is (admin-only) and the interface is (3), as shown in Figure 5-109.

```
$> zfsadm configquery -syslevel | grep sysplex
sysplex(filesys,norwshare) interface(3)
$> sudo /usr/sbin/mount -f hering.test.zfs test
$> zfsowner hering.test.zfs
zFS Owner    : SC70     - Aggregate read-only=N, sysplex-aware=N
$> cn SC65 @usscmd "'df -v test | grep Client='"
 @USSCMD 'df -v test | grep Client='
 File System Owner : SC70       Automove=Y      Client=Y
$> cn SC65 @usscmd "'rxdowner -d test | grep local-client'"
 @USSCMD 'rxdowner -d test | grep local-client'
 Local Sysname: SC65     - File System local-client=Y
```

Figure 5-109 zFS aggregate mounted R/W sysplex-unaware

Some explanation of the commands used in Figure 5-110 on page 288:

- **sudo** is a utility running just one command in super-user mode.
- **rxdowner** is a hardlink of the other utility zfsowner.
- **cn** is a utility to run MVS system commands.
- **SC65** is an abbreviation for specifying ROUTE SC65.
- **USSCMD** is a SYSREXX running UNIX commands.

Most of these utilities are available to be downloaded from the ITSO website, as follows:

Important: The sample REXX procedures are available in softcopy on the Internet from the Redbooks web server.

Point your browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246580/

Note that the SG part of SG246580 must be uppercase.

Alternatively, you can go to:

http://www.redbooks.ibm.com

Select **Redbooks Online**, and then **Additional Materials**.

```
$> sudo /usr/sbin/unmount test
$> sudo /usr/sbin/mount -f HERING.TEST.ZFS -t zFS -o rwshare test
$> zfsowner hering.test.zfs
zFS Owner    : SC70     - Aggregate read-only=N, sysplex-aware=Y
$> cn SC65 @usscmd "'df -v test | grep Client='"
 @USSCMD 'df -v test | grep Client='
 File System Owner : SC70       Automove=Y      Client=N
$> cn SC65 @usscmd "'rxdowner -d test | grep local-client'"
 @USSCMD 'rxdowner -d test | grep local-client'
 Local Sysname: SC65     - File System local-client=N
$>
```

Figure 5-110 zFS aggregate mounted R/W sysplex-aware

Automatic movement of file systems

Figure 5-111 shows again the three systems with zFS sysplex-aware on a file system basis (`sysplex=filesys`). However, this time the sysplex-aware zFS has two different owners. Beginning with z/OS V1R11, as a part of supporting read-write mounted file systems that are accessed as sysplex-aware, zFS automatically moves zFS ownership of a zFS file system to the system that has more read-write activity coming from it compared to the total amount from all systems.

This is because zFS can move its ownership of zFS read-write sysplex-aware file systems dynamically based on system usage among zFS systems where the file systems are locally mounted. If one sysplex member is a better fit for ownership, because more requests to the file system are being made there than any other sysplex member, then zFS dynamically moves the zFS ownership to that system.

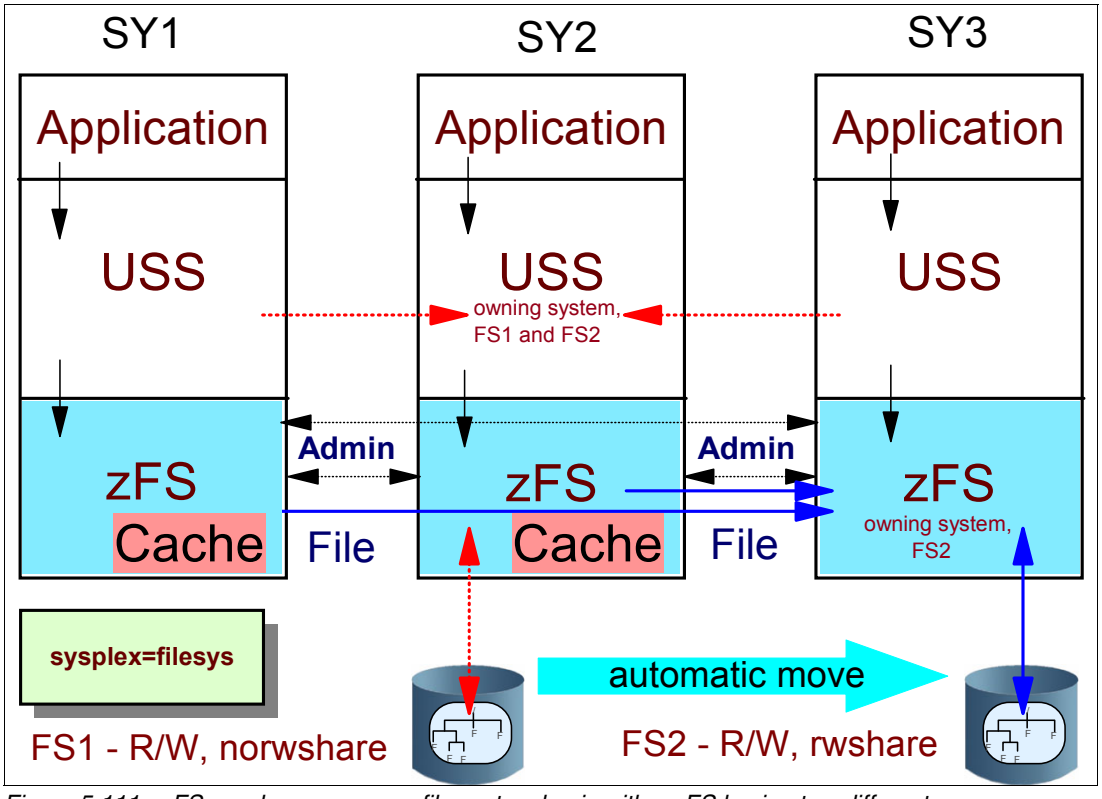


Figure 5-111 zFS sysplex-aware on a file system basis with a zFS having two different owners

Other ownership movement

zFS will also move ownership when the owning system is shut down, or an abnormal outage occurs on the zFS owning system.

Thus the z/OS UNIX owner and the zFS owner can be two entirely different systems depending on the sequence of events. This is normal, and for a sysplex that is using the zFS `sysplex=filesys` support on all sysplex members this should have no negative effects.

zFS does not have any external commands to move zFS ownership between systems. It does provide query commands to show the zFS owner, such as zfsadm lsaggr, the pfsctl() APIs or the utilities rxdowner and zfsowner as shown in Figure 5-112.

```
$> df -v test | grep Client=
File System Owner : SC70        Automove=Y      Client=N
$> zfsadm lsaggr | grep HERING.TEST.ZFS
HERING.TEST.ZFS                                 SC64      R/W
$> rxlsaggr | grep HERING.TEST.ZFS
HERING.TEST.ZFS                                 SC64      R/W
$> rxdowner -d test | grep local-client
Local Sysname: SC70     - File System local-client=N
$> zfsowner hering.test.zfs
zFS Owner    : SC64     - Aggregate read-only=N, sysplex-aware=Y
$>
```

Figure 5-112 zFS aggregate mounted R/W sysplex-aware and having two different owners

Note the following statement.

Important: A zFS ownership movement based on performance criteria is non-disruptive.

5.11.10 Benefits of and recommendations for the new support

The new support has several advantages over the zFS base functions in z/OS V1R11, as follows:

- The new support eliminates performance problems that can occur when running in a mixed environment.
- It allows better coexistence with servers that cannot handle zFS read-write sysplex-aware file systems.
- z/OS SMB server and Fast Response Cache Accelerator support of the IBM HTTP server.

  The SMB server cannot export any zFS read-write file systems that are sysplex-aware. To export zFS read-write file systems using the SMB server, the file system must be non-sysplex aware. Ensure that you have SMB APAR OA31112 for full support.

  When the IBM HTTP Server Version 5.3 for z/OS is configured to use Fast Response Cache Accelerator, static web pages are cached to improve performance. Part of this support uses register file interest, a function of z/OS UNIX, to get notified if the files representing the web pages are changed.

  Important: Running zFS sysplex-aware on a file system basis, using `sysplex=filesys`, is the preferred and recommended option for a shared file system environment.

- If you are currently running zFS with a setting sysplex=off:
  - Roll all your systems to sysplex=filesys.
  - Schedule an unmount and mount of zFS file systems you want to be sysplex-aware and specify RWSHARE on the MOUNT command.
- If you are currently running zFS with option sysplex=on:
  - Roll all your systems to sysplex=filesys with sysplex_filesys_sharemode=rwshare.

Schedule an unmount and new mount of zFS file systems that you do not want to be sysplex-aware and specify NORWSHARE on the MOUNT command.

zFS R/W sysplex-aware test

Here we demonstrate that with the current implementation the sysplex-aware state of a file system can be changed only on unmounting and mounting it again or doing an unmount remount for reading and again an unmount remount for write mode. This is shown in Figure 5-113.

```
$> zfsadm configquery -syslevel | grep sysplex
sysplex(filesys,norwshare) interface(3)
$> rxdowner -d test

MP Directory : /u/hering/test
File System  : HERING.TEST.ZFS
PFS Type     : ZFS
Local Sysname: SC70     - File System local-client=N
USS Owner    : SC70     - File System read-only=N
zFS Owner    : SC70     - Aggregate read-only=N, sysplex-aware=N

$> sudo zfsadm config -sysplex_filesys_sharemode rwshare
IOEZ00300I Successfully set -sysplex_filesys_sharemode to rwshare
$> sudo /usr/sbin/chmount -s test
$> zfsowner hering.test.zfs
zFS Owner    : SC70     - Aggregate read-only=N, sysplex-aware=N
$> sudo /usr/sbin/chmount -r test
$> sudo /usr/sbin/chmount -w test
$> sudo zfsadm config -sysplex_filesys_sharemode norwshare
IOEZ00300I Successfully set -sysplex_filesys_sharemode to norwshare
$> zfsowner hering.test.zfs
zFS Owner    : SC70     - Aggregate read-only=N, sysplex-aware=Y
$>
```

Figure 5-113 zFS R/W sysplex-aware file system state kept until doing a major remount

zFS toleration in z/OS V1R9 and V1R10

zFS R9 and R10 require APAR OA29786 being applied before the sysplex=filesys function can be enabled in zFS of z/OS V1R11.

Table 5-4 zFS Toleration in z/OS V1R9 and V1R10

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R9 (390)	UA50338	OA29786
z/OS V1R10 (3A0)	UA50336	OA29786

- It is verified that code recognizes function before it can be used.
- The zFS MOUNT parameters `RWSHARE` and `NORWSHARE` are ignored in z/OS V1R9 and V1R10 and the MOUNT is allowed to succeed.

z/OS UNIX APAR OA29712

The USS SPE OA29712 is a prerequisite for the zFS SPE OA29619.

z/OS SMB APAR OA31112

Before this new support, the z/OS SMB server does not export a zFS read-write mounted file system that is on a system running zFS sysplex-aware. SMB checks if the zFS sysplex_state is 2 (this is `sysplex=on`) or greater (sysplex_state 3 is `sysplex=filesys`).

With SMB new function APAR OA31112, if the local zFS is not running sysplex=on, the SMB server checks the sysplex-aware state of the file system. If the file system is not sysplex-aware, then the SMB server attempts to export it.

Table 5-5 SMB support for zFS sysplex-aware on a file system basis

z/OS release (SMB release)	PTF number	APAR number
z/OS V1R11 (3B0)	UA52995 (PE)	OA31112
z/OS V1R11 (3B0)	UA53952	OA32648

Note: APAR OA32648 solves the problem with the PTF for APAR OA31112.

5.11.11 z/OS UNIX directory caching display tool

This tool displays whether z/OS UNIX directory caching is active. It is important for helping to resolve performance issues when z/OS UNIX directory caching is lost due to file system movement. This normally only happens when the shared file system environment is mixed.

You can get the tool from:

`ftp://ftp.software.ibm.com/s390/zos/tools/wjsip/wjsipndc.txt`

Note: Be sure to convert it from ISO8859-1 to IBM-1047 or IBM-037 when placing the file to your z/OS system.

Exceptions are listed for file systems where the system is a z/OS UNIX client and while no caching is active, as shown in Figure 5-114 on page 292.

```
$> sudo wjsipndc
zFS file systems with a z/OS directory cache exception
HFS.ZOSR1B.Z1BRC1.ROOT
OMVS.DB2V9.SDSN5HFS.D081006
IMS11B.JMK1106.HFS

If a zFS file system was once read-write and sysplex aware but
is no longer sysplex aware due to moving ownership to a zFS
sysplex=off system or migrating back to sysplex=off from
sysplex=on, the z/OS UNIX System Services directory cache
will be disabled for that file system.  It is possible
performance of that file system can be improved by unmounting
that file system and mounting it back.
```

Figure 5-114 Running the z/OS UNIX directory caching display tool

Attention: Note that once a file system's z/OS UNIX directory cache is disabled, it does not become enabled again unless you do an explicit unmount and mount.

z/OS directory caching for a zFS being sysplex-unaware

Figure 5-115 shows a situation for loosing directory caching.

```
$> sudo zfsadm config -sysplex_filesys_sharemode rwshare -system SC65
IOEZ00300I Successfully set -sysplex_filesys_sharemode to rwshare
$> sudo /usr/sbin/mount -f HERING.TEST.ZFS -t zFS -d SC65 test
$> zfsowner hering.test.zfs
zFS Owner    : SC65     - Aggregate read-only=N, sysplex-aware=Y
$> rxdowner -d test | grep local-client=
Local Sysname: SC70     - File System local-client=N
$> sudo /usr/sbin/chmount -r test
$> sudo zfsadm config -sysplex_filesys_sharemode norwshare -system SC65
IOEZ00300I Successfully set -sysplex_filesys_sharemode to norwshare
$> sudo /usr/sbin/chmount -w test
$> zfsowner hering.test.zfs
zFS Owner    : SC65     - Aggregate read-only=N, sysplex-aware=N
$> rxdowner -d test | grep local-client=
Local Sysname: SC70     - File System local-client=Y
$> sudo wjsipndc
zFS file systems with a z/OS directory cache exception
HERING.TEST.ZFS
OMVS.DB2V9.SDSN5HFS.D081006
```

Figure 5-115 Loosing z/OS directory caching for a zFS being sysplex-unaware

Note: If you are sure to use a file system sysplex-aware when R/W is mounted all the time until you unmount it again, you should mount it using option `rwshare` as shown in Figure 5-110 on page 288.

5.11.12 File system monitoring tool FSMON

This wjsfsmon utility is mentioned in APAR OA29619 and can help you to determine which systems are accessing your zFS read-write file systems and whether these systems are accessed from multiple systems.

See the z/OS UNIX Tools and Toys website to download the tool and its documentation:

http://www.ibm.com/servers/eserver/zseries/zos/unix/tools

You can also directly go to the UNIX Tools site:

http://www-03.ibm.com/systems/z/os/zos/features/unix/bpxa1ty2.html

or retrieve it from:

ftp://ftp.software.ibm.com/s390/zos/tools/wjsfsmon/wjsfsmon.txt

Note: The documentation is available as a PDF. The tool itself is available in ASCII. Be sure to convert it from ISO8859-1 to IBM-1047 or IBM-037 when placing the file to your z/OS system.

5.12 zFS Direct I/O and zFS installation changes (R13)

The zFS Direct I/O (DIO) is a performance improvement for the USS shared file system environment using zFS read-write sysplex-aware file systems.

In a sysplex shared file system environment, it should not matter where an application runs, nor which system owns a file system. However, in current systems, location is important due to function shipping performance and file system ownership needs to be managed.

This new support provides the ability to directly read and write zFS data from any system in a sysplex shared file system environment.

Note: This especially means much less concern about file system ownership and where applications execute in a sysplex shared file system environment.

5.12.1 zFS sysplex support

We now provide a short history of recent enhancements.

zFS in z/OS V1R11 introduced support for zFS read-write sysplex-aware file systems with parameter setting **sysplex=on**. This provided a performance improvement, especially in the area of read performance. It also made file system ownership less important.

zFS R11 APAR OA29619 improved the granularity of using zFS read-write sysplex-aware file systems with the new setting of **sysplex=filesys**. This allowed you to individually choose which zFS read-write file systems are sysplex-aware and which are not.

Important: Running zFS sysplex-aware on a file system basis, using **sysplex=filesys**, is the preferred and recommended option for a shared file system environment.

zFS in z/OS V1R13 introduces the new Direct I/O function. This enhances support for zFS read-write sysplex-aware file systems. When all systems are at level z/OS V1R13, zFS can

directly read and write sysplex-aware file systems from all systems in the shared file system environment.

This makes file system ownership much less important.

Note: zFS in z/OS V1R13 always runs with setting **`sysplex=filesys`** in a USS shared file system environment.

5.12.2 Sysplex unaware read-write file system

Figure 5-116 shows the original behavior of clients accessing a zFS file system through USS function shipping using XCF and the XPFS file system interface. This is also the case if a zFS is started with sysplex=off in previous releases.

Note: Because this means there is much overhead, it is important that the system doing the most accesses is the owner of the file system.

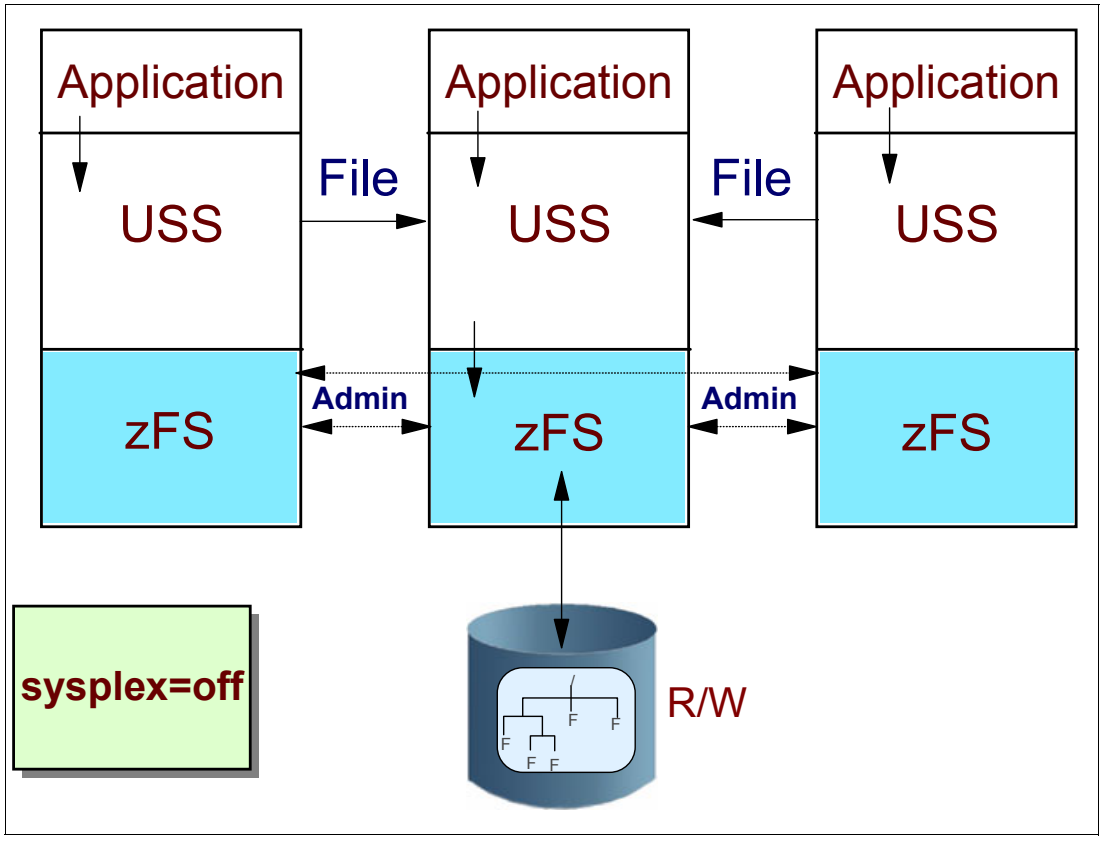


Figure 5-116 Sysplex-unaware read-write file system

5.12.3 z/OS V1R11 sysplex-aware read-write file system

Figure 5-117 on page 295 shows the behavior as it was introduced with zFS in z/OS V1R11 with zFS started as sysplex-aware by using the setting **`sysplex=on`**.

- When zFS runs sysplex-aware (for read-write) on all systems, a read-write mounted sysplex-aware file system is locally mounted on all systems.

- There is still a z/OS UNIX owning system but there is no z/OS UNIX function shipping to the owner.
- Instead, requests from applications on any system are sent directly to the local zFS on each system. This means it is now the responsibility of the local zFS to determine how to access the file system.
- One of the systems is known as the zFS owning system. This is the system where all I/O to the file system is done. zFS uses function shipping to the zFS owning system to access the file system.
- However, each zFS client system has a local cache where it keeps the most recently read file system information.
- Therefore, in many cases, when the data is still in the cache, zFS can avoid the zFS function shipping and satisfy the request locally.

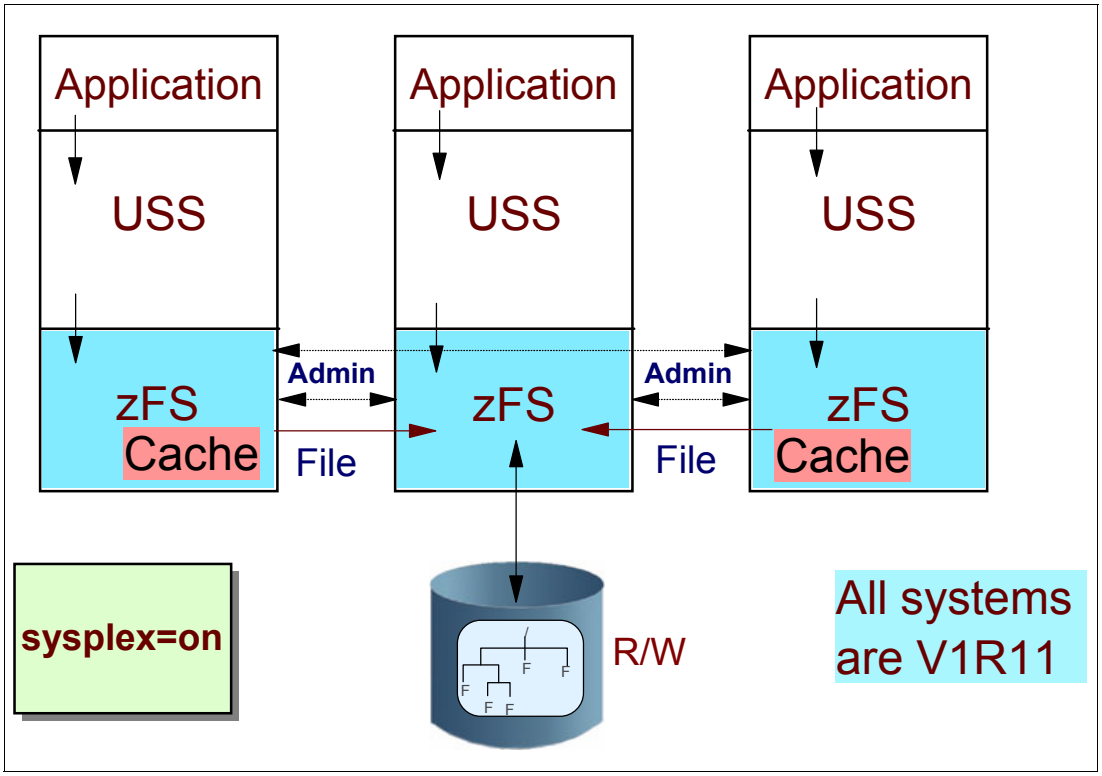


Figure 5-117 z/OS V1R11 sysplex-aware read-write file system

5.12.4 zFS sysplex-aware on a file system basis

Figure 5-118 on page 296 shows the new support of zFS being sysplex-aware on a file system basis that was introduced with APAR OA29619. Based on starting zFS with sysplex=filesys, this allows to decide whether a specific read-write file system is mounted sysplex-aware or not.

This is also known as the recommended way of starting zFS when the support is added.

- You see two zFS read-write file systems on a sysplex running zFS sysplex-aware on a file system basis on all members.
- File system FS1 is mounted NORWSHARE and FS2 is mounted RWSHARE. They are both z/OS UNIX-owned on SY2.

- The sysplex-unaware file system FS1 is only locally mounted on the z/OS UNIX owner SY2 and requests from z/OS UNIX clients SY1 and SY3 are function shipped to the z/OS UNIX owner SY2 by z/OS UNIX. A **`df –v`** command for file system FS1 from SY1 would display **`Client=Y`**.
- The sysplex-aware file system FS2 is locally mounted on all systems and z/OS UNIX never function ships requests to the z/OS UNIX owner SY2. A **`df –v`** command for the file system FS2 from SY1 would display **`Client=N`**.
- When you run zFS sysplex-aware on a file system basis on all your members, the zFS Physical File System initializes as sysplex-aware but it can individually determine which file system is sysplex-aware and which is not based on the system default setting for the file system sharemode (sysplex_filesys_sharemode) or the mount parameter RWSHARE or NORWSHARE.
- Based on file system activity, the zFS ownership of a sysplex-aware zFS file system may move to another system and so become different from the z/OS UNIX ownership.

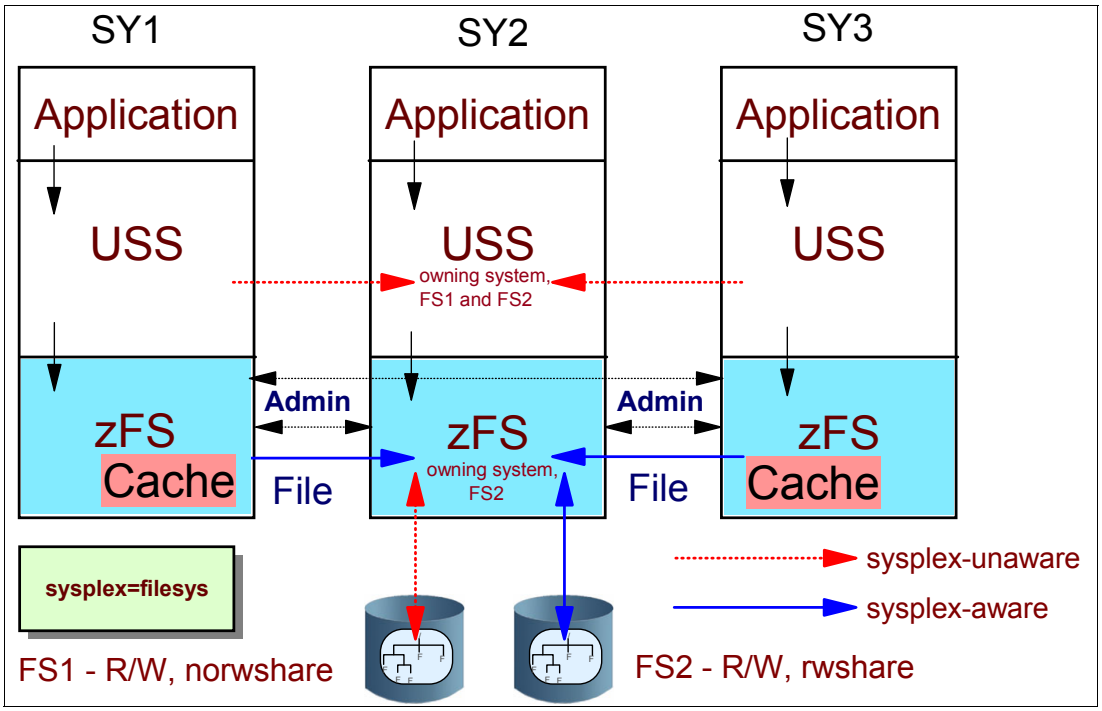


Figure 5-118 zFS sysplex-aware on a file system basis

5.12.5 z/OS V1R13 Direct I/O sysplex-aware read-write file system

Finally, Figure 5-119 on page 297 shows that with this new support in z/OS V1R13 and if the zFS file system is mounted sysplex-aware, any client zFS can directly access the file system for reading and writing data and partially also reading metadata.

Note: Metadata updates are always sent to the zFS owning system.

So, any user or application in a z/OS V1R13 environment can exploit the functionality for zFS read-write sysplex-aware file systems.

Note: The client cache is no longer used in z/OS V1R13 if the owner supports DIO. The client reply storage is still used and the new default for client_reply_storage is 10M.

Nevertheless, if a zFS in z/OS V1R11 or R12 is accessing data in zFS file system owned at a R13 system it still uses the client cache and also cannot do DIO. This is also true if the file system is owned by a z/OS V1R11 or R12 system.

Important notes:

- Aggregate movement is expected to happen less often. Only access operations to metadata, especially updates, are taken into account.
- In z/OS V1R13 with DIO being active zFS does not automatically move a zFS to a down-level system for performance reasons.

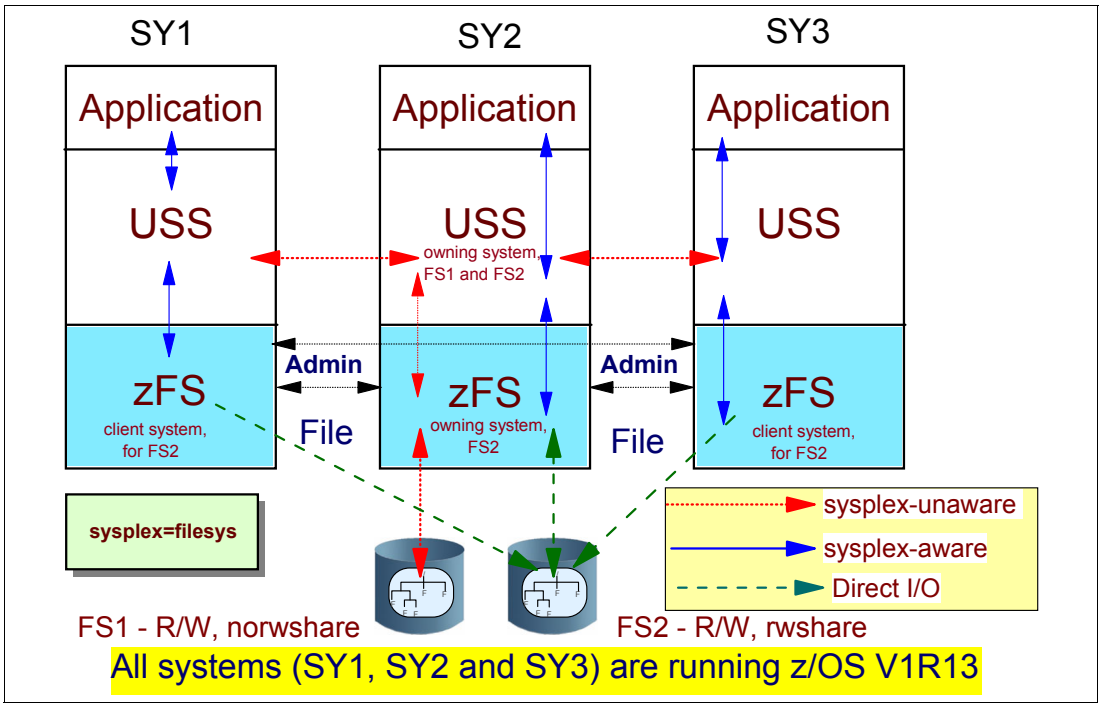


Figure 5-119 z/OS V1R13 Direct I/O sysplex-aware read-write file system

5.12.6 Migration considerations

The migration to zFS R13 is a two-step process:

- Install toleration APAR OA32925 (PTF UA55765) on all zFS R11 and R12 systems and make it active with a rolling IPL, for example.
- Change your zFS IOEFSPRM file to specify **`sysplex=filesys`** on all systems and make it active with a rolling IPL, if not used already as it was recommended to do so as soon as this new setting was introduced via APAR OA29619.
- At this point you can bring in z/OS V1R13 and zFS R13.

Unlike previous releases, zFS R13 does not store data in 1K fragments; rather, it is stored in 8K blocks. zFS R13 can read data stored in fragments; however, when the data is updated, it is moved into 8 K blocks.

Note: zFS user data is stored inline when the file is 52 bytes or less. Otherwise 8 K blocks are used.

Because previous releases of zFS can read an 8 K block that is not full, no toleration support is required on those systems. Also, in previous releases, when zFS stored data in fragments, data from multiple files typically resided in separate 8 K blocks.

However, there are certain cases when zFS R13 will require more DASD space than zFS in previous releases. This is a main reason for changing the default for aggrgrow.

Attention: The default for aggrgrow has been changed from **aggrgrow=off** to **aggrgrow=on**.

5.12.7 zFS health check for sysplex=filesys

As mentioned in “Migration considerations” zFS R13 has a migration action. Both zFS in z/OSV1R11 and R12 must run **sysplex=filesys** before zFS R13 can be brought into the shared file system environment.

- Therefore, a migration health check has been provided in z/OS V1R11 and z/OS V1R12 to check whether zFS **sysplex=filesys** has been specified.
- This zFS R13 migration health check is provided as service to zFS in V1R11 and z/OS V1R12. The zFS APAR OA35465 and PTF UA59383 for zFS release 3B0 (this is zFS R11 and R12) provides the health check.
- This helps to ensure that the migration action is executed and to avoid a problem bringing zFS R13 into the shared file system environment.

5.12.8 Sample calculations for DASD space

As mentioned in “Migration considerations” zFS R13 does not use 1 K fragments anymore.

The data for 1000 small files uses 1000 K in zFS R11 if assuming they are perfectly packed into 8 K blocks. The data for 1000 small files uses 8000 K in R13. This means that you need about 9.5 cylinders more in case of zFS R13, if you take into account that there is 720 K data per cylinder.

Note: DASD space is also used for the directory names, the anodes, etc. which is the same in R13 as in prior releases.

Nevertheless, there is one positive effect that results from this change of 1 K fragments no longer used by zFS R13. It was always important to look for the aggregate full percentage and not the file system full numbers. The latter could be misleading just because of the usage of the 1 K fragments.

Note: If a zFS aggregate has been completely rewritten or defined in z/OS V1R13 the information provided by a file system full percentage is equivalent to an aggregate full percentage. Also, the z/OS UNIX command **df -kP *mp_directory*** will then provide a correct capacity percentage.

There is a tool available in text mode at

```
ftp://public.dhe.ibm.com/s390/zos/tools/zfsspace/zfsspace.txt
```

A sample call of `zfsspace` is shown in Figure 5-120 on page 299.

```
tso zfsspace /u/hering/b1kfill
HERING.ONEK.FILES
  small files: 1000
  extents....: yes
  aggrgrow...: yes
```

Figure 5-120 Listing information about small files less 7 K

6

Performance and tuning

This chapter provides performance and tuning information for using zFS file systems versus using HFS file systems.

It discusses the following topics:

- File system access
- User file cache
- Metadata file cache
- Log file cache
- Directory cache
- Performance monitoring APIs
- Comparison of HFS and zFS file systems
- Performance comparisons with sysplex-sharing (R11)
- Performance comparisons with sysplex-sharing (R13)

6.1 File system access

The performance of zFS can be influenced by controlling the size of the caches used to hold file system and log data. There are three caches that can be monitored and controlled to reduce I/O rates, as follows:

- User file cache

  This cache is used for all user files and performs I/O for all user files greater than 7 KB.

- Metadata cache

  User files that are smaller than 7 KB have the I/O from this cache.

- Log file cache

  This cache is used to write file record transactions that describe changes to the file system.

Figure 6-1 shows the cache prior to z/OS V1R4.

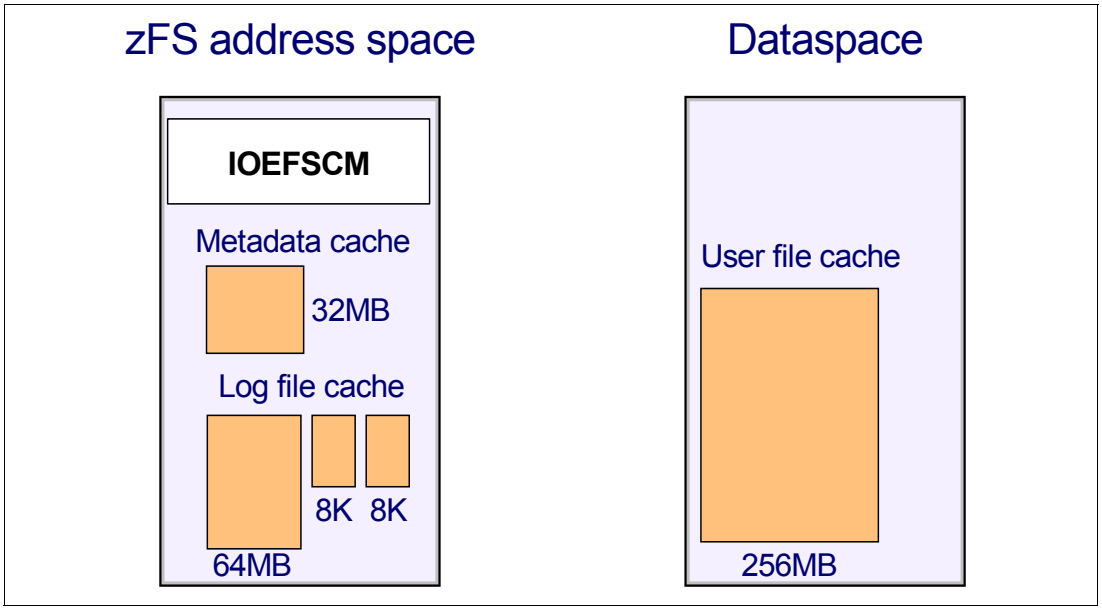


Figure 6-1 Default cache sizes

6.1.1 IOEFSPRM parameter file

This optional parameter file contains the tuning options that can be adjusted to tune the performance of the zFS file systems. Operator commands can be used to display the options that affect performance. Therefore, if you want to do performance tuning, you need to have an IOEFSPRM file specified. The following parameters affect the tuning of the file system:

user_cache_size This specifies the size, in bytes, of the cache used to contain file data. You can also specify an option to page fix the pages for performance.

Default=256 MB.

meta_cache_size This specifies the size of the cache used to contain metadata. You can also specify a page fix option that indicates that the pages are permanently fixed for performance.

Default=32 MB.

log_cache_size This specifies the size of the cache used to contain buffers for log file pages. You can also specify a page fix option that indicates that the pages are permanently fixed for performance.

Default=64 MB.

tran_cache_size This specifies the initial number of transactions in the transaction cache.

Default=2000.

Page fix option

The page fix option reserves real storage for use by zFS only. Note the following examples of page fix options:

```
user_cache_size=64M,fixed
meta_cache_size=64M,fixed
log_cache_size=32M,fixed
```

6.2 User file cache

The user file cache is used to cache all user files. It caches any file, no matter what its size, and performs the following:

- Write-behind and asynchronous read-ahead for files
- I/O for all files that are 7 K or larger

> **Note:** For files smaller than 7 K, I/O is normally performed through the metadata cache.

The user file cache is allocated in data spaces. Its size by default is 256 MB, as shown in Figure 6-1 on page 302, and can be tailored to meet your performance needs based on your overall system memory. The maximum size is 65536 MB (64 GB).

The general rule for any cache is to ensure a good hit ratio. Additionally, for a user file cache, it should be large enough to allow write-behind activity (if the cache is too small, you need to recycle buffers more frequently, which could degrade write-behind performance).

You can monitor the user file cache by using the **`f zfs,query,all`** command; see Figure 6-2 on page 304. You need to look for the *fault ratio*. To get the hit ratio, subtract the fault ratio from 100%, as shown:

cache hit ratio = (100% - fault ratio).

A value over 90% usually gives good performance.

```
                  User File (VM) Caching System Statistics
                  ----------------------------------------

External Requests:
------------------
Reads           1534     Fsyncs             0     Schedules          0
Writes             0     Setattrs           5     Unmaps            53
Asy Reads         67     Getattrs         866     Flushes            0

File System Reads:
------------------
Reads Faulted         233     (Fault Ratio   15.189%)
Writes Faulted          0     (Fault Ratio    0.00%)
Read Waits             37     (Wait Ratio    2.411%)
Total Reads           439

File System Writes:
-------------------
Scheduled Writes             0     Sync  Waits            0
Error Writes                 0     Error Waits            0
Scheduled deletes            0
Page Reclaim Writes          0     Reclaim Waits          0
Write Waits                  0     (Wait Ratio    0.00%)
```

Figure 6-2 Output from the f zfs,query,all command

6.3 Metadata cache

The metadata cache is used to contain all file system metadata, which includes the following:

- All directory contents
- File status information, which includes atime, mtime, size, permission bits, and so on
- File system structures
- Caching of data for files smaller than 7 K

6.3.1 Metadata cache monitoring

Metadata is referred to and updated very frequently for most zFS file operations, so 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 better, depending on the workload.

You can monitor the metadata cache by using the **f zfs,query,all** command. Look for the output shown in Figure 6-3.

```
                        Metadata Caching Statistics

Buffers    (K bytes)  Requests     Hits    Ratio  Updates
 ---------- --------- ---------- ---------- ----- ----------
       4096     32768     721635     721635 100.0%      77000
```

Figure 6-3 Metadata cache statistics

These statistics for the metadata cache are from the test results shown in 6.15.2, “Comparison results” on page 327.

Note: The complete output from the command is provided in Example D-1 on page 480.

6.3.2 Metadata cache storage

The metadata cache is used to contain all file system metadata, which includes the following:

- All directory contents
- File status information, which includes atime, mtime, size, permission bits, and so on
- File system structures
- Caching of data for files smaller than 7 K

New metadata backing cache

An optional metadata backing cache that contains an extension to the meta cache can be specified. This new backing cache resides in a data space and is used as a “paging” area for metadata. Therefore, it allows a larger meta cache for workloads that need large amounts of metadata.

Note: This optional cache is only needed if the metadata cache is constrained.

This option can be enabled by specifying a metaback_cache_size option in the IOEFSPRM configuration file. A value between 1 MB and 2048 MB is allowed. You can specify values by using a K (indicating kilobytes) or an M (indicating megabytes) following the value. It is also possible to specify a fixed option that indicates the pages are permanently fixed for performance, as follows:

```
metaback_cache_size=64M,fixed
```

6.4 Log file cache

Prior to z/OS V1R4, the log file cache is shared among all aggregates and is stored in the primary address space. With z/OS V1R4, the log file cache is moved to a data space, as shown in Figure 6-4 on page 306. This cache defaults to 64 MB.

For each aggregate that is attached, the log file cache is grown dynamically by adding one 8 K buffer. Each aggregate always has one 8 K buffer to record its most recent changes to file system metadata.

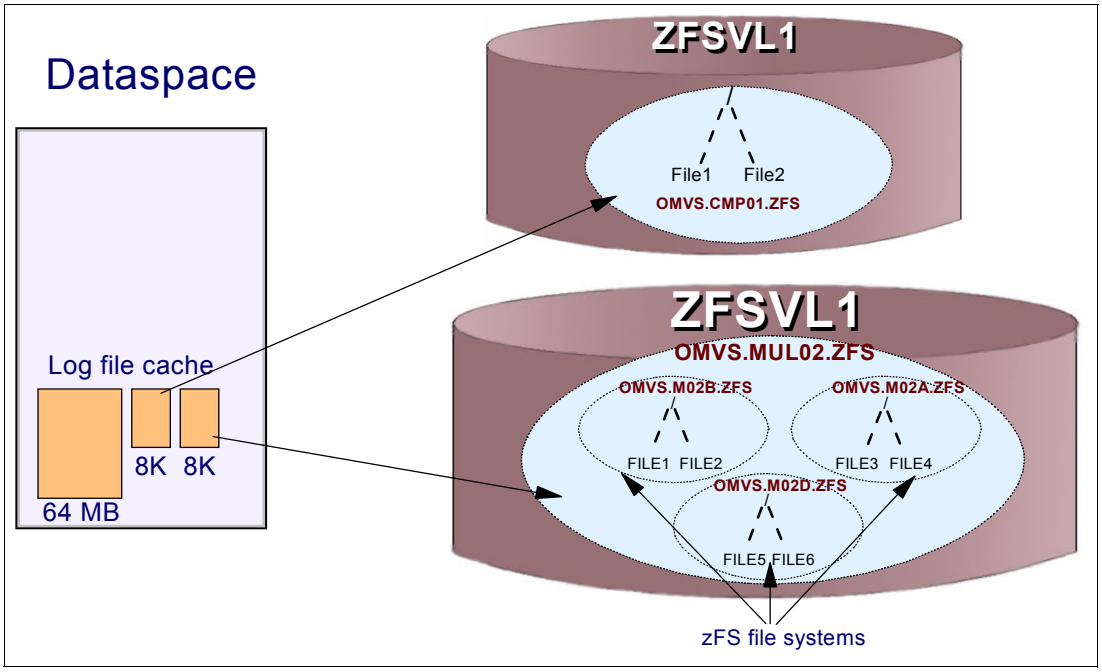


Figure 6-4 Log file cache grows dynamically

Log file cache in a data space

The log file cache is now moved into a data space, which frees up space in the zFS address space and allows for space for the other caches.

6.4.1 zFS log files

When an aggregate is formatted, the log file is created for that aggregate. The messages for the formatting indicate the size of the log if you have not specified the log size as a format parameter. Sample messages from the format routine are shown in Figure 6-5.

```
IOEZ00004I Loading dataset 'omvs.cmp01.zfs'.
IOEZ00005I Dataset 'omvs.cmp01.zfs' loaded successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 17999
*** Defaulting to 179 log blocks(maximum of 19 concurrent transactions).
Done.  /dev/lfs1/omvs.cmp01.zfs is now an zFS aggregate.
IOEZ00071I Attaching aggregate OMVS.CMP01.ZFS to create hfs-compatible file system
IOEZ00074I Creating file system of size 140487K, owner id 0, group id 2, permissions x1ED
IOEZ00048I Detaching aggregate OMVS.CMP01.ZFS
IOEZ00077I HFS-compatibility aggregate OMVS.CMP01.ZFS has been successfully created

IOEZ00004I Loading dataset 'omvs.mul01.zfs'.
IOEZ00005I Dataset 'omvs.mul01.zfs' loaded successfully.
*** Using default initialempty value of 1.
*** Using default number of (8192-byte) blocks: 17999
*** Defaulting to 179 log blocks(maximum of 19 concurrent transactions).
Done.  /dev/lfs1/omvs.mul01.zfs is now an zFS aggregate.
```

Figure 6-5 Messages from the format routine for two aggregates

Messages from format routines

Always check the messages from the format routine because they show the default number of log blocks is always 1% of the total number of 8 KB blocks that are allocated in the aggregate. This controls, as the messages state, the number of concurrent transactions allowed to file systems in the aggregate. You can control the number of log blocks and transactions by specifying the log block size as a format parameter, as shown in Figure 6-6; see the format parameters in 2.7, “Formatting zFS aggregates” on page 25.

```
zfsadm format -aggregate OMVS.CMPO1.ZFS -logsize 300
```

Figure 6-6 Format parameters to format an aggregate

6.4.2 Log file usage

To be able to determine if the log file cache is sufficient for the aggregate, you can use the operator command **f zfs,query,all** to show log file I/O rates and I/O waits. It is really necessary 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, depending on the file system usage.

However, every workload is different. For example, workloads that issue fsync operations force zFS to synchronize the log file more frequently. The log file cache is a write-only cache, so a read hit ratio is not relevant.

6.5 Directory cache and large directories

In z/OS V1R7, a new zFS IOEPRMxx parameter file statement is available to set the directory cache size. The minimum and default size was 2 M at the time that the new parameter was introduced. The maximum value that can be set is 512 M. Changing this cache size may be useful if you expect to encounter situations with mass file creates or deletes.

This setting is available via APAR OA10136 back to z/OS V1R4 (PTF UA16125 for z/OS V1R4, UA16123 for z/OS V1R5, and UA16124 for z/OS V1R6). The new IOEPRMxx parameter is:

dir_cache_size This specifies the size of the directory buffer cache. The value can be a between 2 M to 512 M.
Example: dir_cache_size=4M

Before applying the service to have the new function available, we saw in one specific situation, with mass file creates involved, a very undesirable directory cache hit ratio and further poor performance values as shown in Figure 6-7.

```
zfs_lookups avg. time: 1257.479 ms
zfs_creates avg. time: 3284.339 ms
LFS Vnode Cache Ratio: 79.961%
Directory Cache Ratio:  5.514%
```

Figure 6-7 Undesirable directory cache hit ratio on mass file creates

We applied the service and restarted zFS with higher cache settings, specifically:

```
dir_cache_size = 32M
meta_cache_size = 256M
```

Important: The directory cache has been removed in z/OS V1R13. The meta data cache took over this function in addition. (See also “Changes in IOEPRMxx configuration options” on page 125.)

After this, the performance numbers dramatically improved and the application did not encounter further delays. The new performance numbers are shown in Figure 6-8.

```
zfs_lookups avg. time:    4.950 ms
zfs_creates avg. time:   14.175 ms
LFS Vnode Cache Ratio: 90.391%
Directory Cache Ratio: 99.952%
```

Figure 6-8 Improved directory cache hit ratio after exploiting the new function

Furthermore, a much lower number of waits for locks was noticed.

Restriction: Setting the directory cache size is currently not supported dynamically by using **`zfsadm config`**. However, you can set the values as mentioned in the example to generally avoid such problems.

6.5.1 Changing the default setting for the zFS dir_cache_size

To avoid further problems and to enhance performance, APAR OA20180 changes the default directory cache size from 2 M to 32 M, as listed in Table 6-1.

Table 6-1 Changed default setting for zFS dir_cache_size

z/OS release (zFS release)	PTF number	APAR number
z/OS V1R7 (370)	UA35890	OA20180
z/OS V1R8 (380)	UA35891	OA20180
z/OS V1R9 (390)		included in base code

6.5.2 Large directories

Although the problems with the directory cache are resolved, a possible performance problem remains based on the current implementation of accessing directory data.

Tip: If you have single directories with very large numbers of files, it may be more useful to use multiple directories instead. Problems may be seen when the number of entries grows to hundreds of thousands.

Refer to “Minimum and maximum file system sizes” in the zFS Administration guide for further information about problems and effects that may be encountered when using very large directories.

6.6 Performance monitoring APIs

z/OS V1R6 zFS provides six new pfsctl application programming interfaces to retrieve performance counters: locks, storage, user data cache, iocounts, iobyaggr, and iobydasd.

z/OS V1R7 completes the set of performance monitoring capabilities by adding the performance monitoring APIs, phase 2. Now, all zFS performance counters can be retrieved by an application. The z/OS V1R7 pfsctl APIs are:

- Directory Cache
- Kernel
- Log Cache
- Metadata Cache
- Transaction Cache
- Vnode Cache

In addition, the pfsctl APIs provide a reset time stamp for the last time the counters were reset. When reset is requested, the counters are provided with the previous reset time stamp (or you can ignore the counters). The next time the counters are retrieved, the new time stamp is provided.

These APIs are implemented as subcommands of the new **ZFSCALL_STATS** command. See 2.19, “zFS application programming interfaces” on page 97 for more information about pfsctl and how to code your own programs to exploit this interface.

Note: RMF™ exploits these APIs to show zFS-related performance information in z/OS V1R7.

New zfsadm query command

In addition, zFS provides a new **zfsadm query** command with z/OS V1R6 to query and reset the performance counters. Table 6-2 shows this new **zfsadm** command.

The issuer of the **zfsadm** command must have READ authority to the data set that contains the IOEFSPRM file.

To issue the command shown in Table 6-2, you must have RACF authorization:

IOEFSPRM The command issuer must have access to a RACF data set profile to the IOEFSPRM data set.

SU mode The command issuer must have superuser authority.

Table 6-2 The new zfsadm query command for z/OS V1R6

Command/subcommand	Command description	IOEFSPRM	SU mode
zfsadm query	Query or reset the performance counter	Read	No

The format of the **zfsadm query** command is shown in Figure 6-9 on page 310.

```
zfsadm query [-locking]
         [-reset]
         [-storage]
         [-usercache]
         [-trancache] R7
         [-iocounts]
         [-iobyaggregate]
         [-iobydasd]
         [-knpfs] R7
         [-metadata] R7
         [-dircache] R7
         [-vnodecache] R7
         [-logcache] R7
         [-system system_name] R7
         [-level]
         [-help]
```

Figure 6-9 zfsadm query command parameters

Note: The options marked with **R7** are new with z/OS V1R7. All other options were introduced with z/OS V1R6. All **pfsctl** or **zfsadm query** commands can only be sent from one z/OS V1R7 system to another z/OS V1R7 system.

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

The example shown in Figure 6-10 resets the locking statistics report counters to zero without listing any further information.

```
$> zfsadm query -locking -reset > /dev/null
$>
```

Figure 6-10 Command to reset the locking statistics report counters to zero

An example of displaying the directory cache statistics in a z/OS V1R7 system is shown in Figure 6-11.

```
$> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.07.00
$> zfsadm query -dircache
                  Directory Backing Caching Statistics

Buffers    (K bytes)  Requests     Hits    Ratio   Discards
---------- --------- ---------- ---------- ------ ----------
       256      2048    1216679    1216628  99.9%         50
```

Figure 6-11 Displaying the directory cache counters report

Further examples showing the output of a **zfsadm query -usercache** command and a **zfsadm query -knpfs** command (which displays the kernel counters report) are provided in D.2, “Output data of a zfsadm query command” on page 484 and D.3, “Output data of a z/OS V1R7 zfsadm query command” on page 486.

6.7 RMF support for zFS

In addition to the zFS performance monitoring APIs in z/OS V1R7, RMF support has been added for zFS.

When looking at zFS performance, you have to consider the zFS components that are involved in I/O processing to or from a zFS file system. The performance of zFS can be influenced by controlling the size of the caches used to hold file system and log data.

There are new zFS summary and activity reports which provide data on:

- zFS response times and wait times
- zFS cache activity
- zFS activity and capacity by aggregate
- zFS activity and capacity by filesystem

This data helps to control the zFS environment according to:

- Cache sizes
- I/O balancing
- Capacity control for zFS aggregates

The zFS data reported by RMF Monitor III can be categorized into:

- zFS performance data
- zFS capacity data

6.7.1 zFS cache monitoring

The following caches can be monitored by RMF Monitor III:

User file cache This cache is used for all user files. It performs I/O for all user files greater than 7 KB. It is allocated in data spaces.

zFS has a structure for each file currently cached. Each cached file is broken into 64 K. Each segment is broken into 4 K pages.

Vnode cache Vnode is a virtual inode, an object in a file system that represents a file. The vnode cache resides in the zFS primary address space.

Metadata cache User files that are smaller than 7 KB have the I/O from this cache. The metadata cache resides in the zFS primary address space. For performance reasons, the allocated storage can be fixed.

Log file cache This cache is used to write file record transactions that describe changes to the file system. The log file cache is allocated in a data space.

Transaction cache Data structures representing transactions that change metadata. The transaction cache is stored in the zFS primary address space. zFS dynamically increases the cache based on the number of concurrent pending transactions.

Figure 6-12 on page 312 provides an overview of zFS address spaces and data spaces.

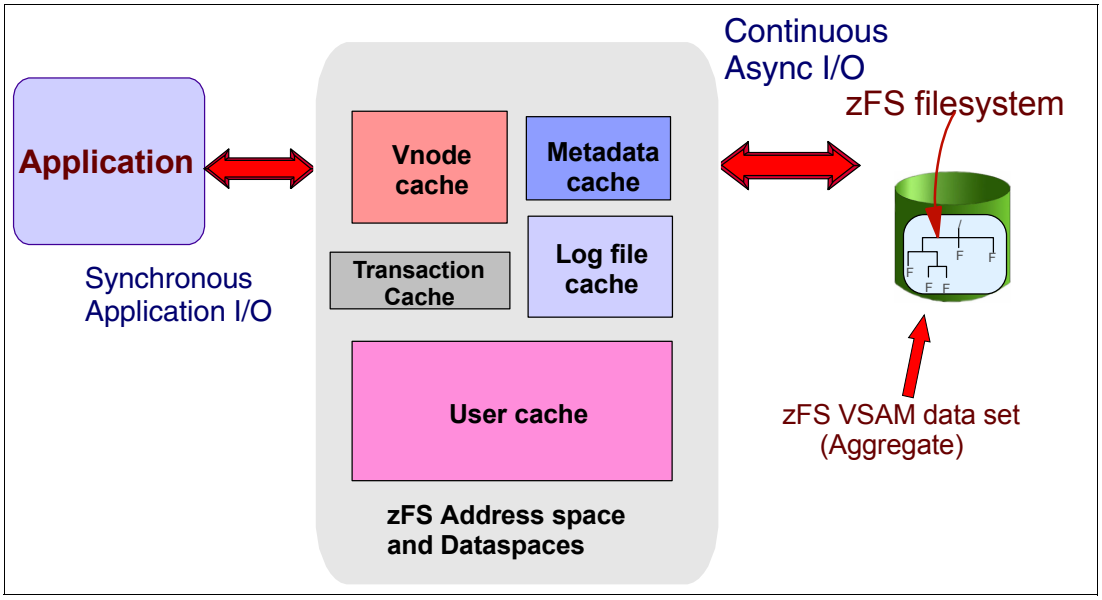


Figure 6-12 zFS address space and data spaces overview

Invoking zFS reports

zFS reports are single-system reports. They can be invoked from the RMF Overview Report Selection Menu, shown in Figure 6-13, or by using the following commands:

`ZFSSUM` or `ZFSS`: zFS summary report

`ZFSACT` or `ZFSA`: zFS activity report

Gathering zFS activity data

Gathering zFS activity data is controlled by a new Monitor III gatherer option:

`NOZFS | ZFS` (default: ZFS)

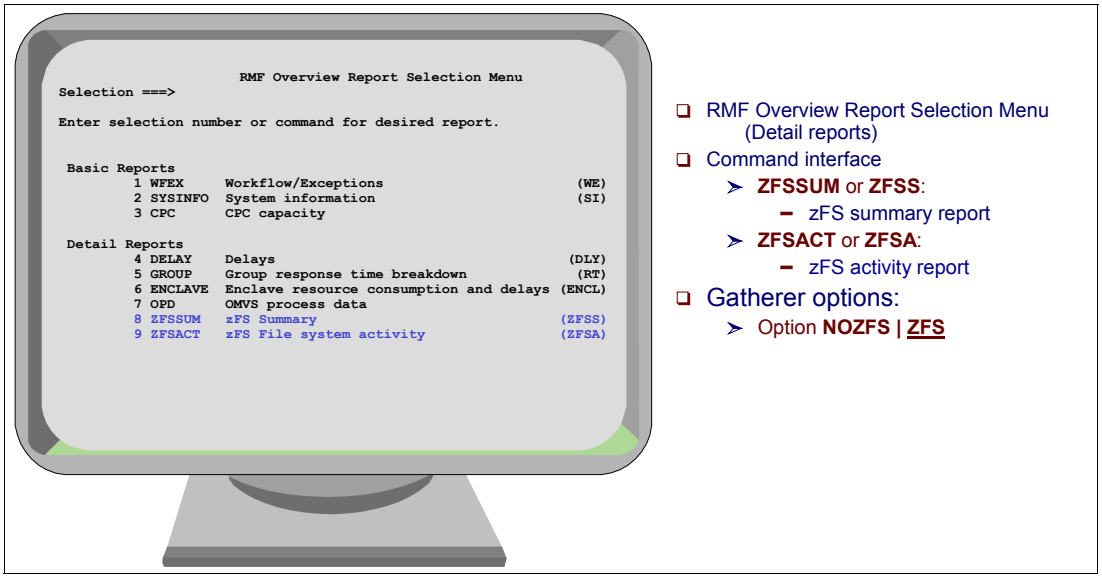


Figure 6-13 RMF Overview Report Selection Menu

6.8 RMF zFS Summary Report

The zFS Summary Report shown in Figure 6-14 on page 314 displays the zFS activity and capacity data for the following three areas:

- Summary for overall zFS response time as request response time and wait times.
- Cache activity for the following four cache types:
  - User cache
  - Vnode cache
  - Metadata cache
  - Transaction cache
- Aggregate activity and capacity data. The aggregate name is a cursor-sensitive field that shows the file systems statistics for the selected aggregate in the zFS activity report.

6.8.1 Report field descriptions

The important areas of the Response Summary Report are:

- In the response time section, the I/O, lock, and sleep wait percentages are workload-dependent. A possible reason for high values is if the caches are too small.

  Small log files (small aggregates) that are heavily updated may result in I/Os, as follows:

  Total Average time in milliseconds to complete a zFS request.

  I/O wait% Percentage of time a zFS request has to wait for an I/O completion.

  Lock wait% Percentage of time a zFS request has to wait for locks.

  Sleep wait% Percentage of time a zFS request has to wait for events to occur.

- The Cache Activity section is as follows:

  User cache rate Number of requests per second made to the user file cache.

  User cache hit% Percentage of requests to the user file cache that completed without accessing the DASD.

  User cache read% Percentage of read requests to the user file cache, based on the sum of read and write requests.

  User cache dly% Percentage of requests to the user file cache that was delayed.

  Vnode rate Number of requests per second made to the vnode cache.

  Vnode hit% Percentage of requests to the vnode cache that completed without accessing the DASD.

  Metadata rate Number of requests per second made to the metadata cache.

  Metadata hit% Percentage of requests to the metadata cache that completed without accessing the DASD.

  Trx rate Number of transactions per second that started in the transaction cache.

- The Aggregate Name section, as follows:

  Aggregate name Name of the zFS aggregate. The name of the aggregate is the VSAM Linear data set name.

  Size Size of the aggregate.

  Use% Percentage of space used in the aggregate.

Mode	Aggregate mode: R/O CP R/W CP (CP: compatibility mode = aggregate contains one file system). R/O MS R/W MS (MS: multi-file mode = aggregate can contain multiple file systems).
FS	Number of file systems in the aggregate (one for CP mode aggregates).
Read (B/sec)	Read data transfer rate (in bytes per second) for the aggregate.
Write (B/sec)	Write data transfer rate (in bytes per second) for the aggregate.

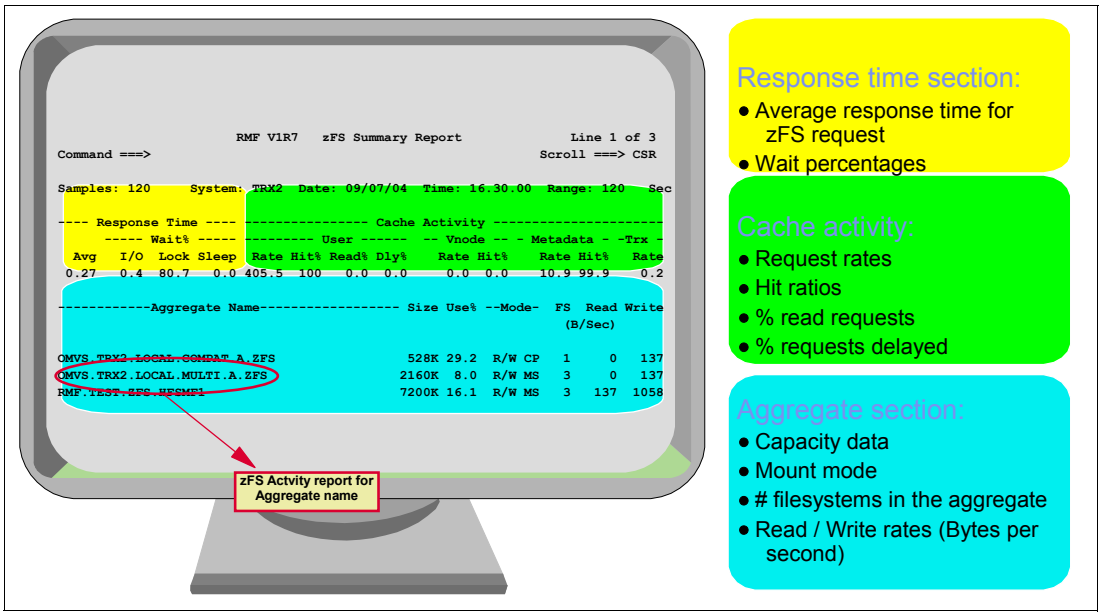


Figure 6-14 zFS Summary Report

6.9 RMF Detail reports

From the zFS Summary Report, you can obtain detail reports by placing the cursor on the cursor-sensitive report fields.

6.9.1 I/O details report

Place the cursor on the response time section (on the I/O value). This will display the I/O details panel, as shown in Figure 6-15 on page 315.

The report displays a breakdown of I/O requests for the following I/O request types:

- I/O for file system metadata
- I/O for log data
- I/O for user data

Field descriptions

Count	Total number of requests.
Waits	Number of requests waiting on an I/O completion of this I/O type.

Cancels Number of requests canceled (for example, a user deletes a file with pending I/O).

Merges Number of times two I/O requests are merged to one because of better performance.

Type Type of I/O request.

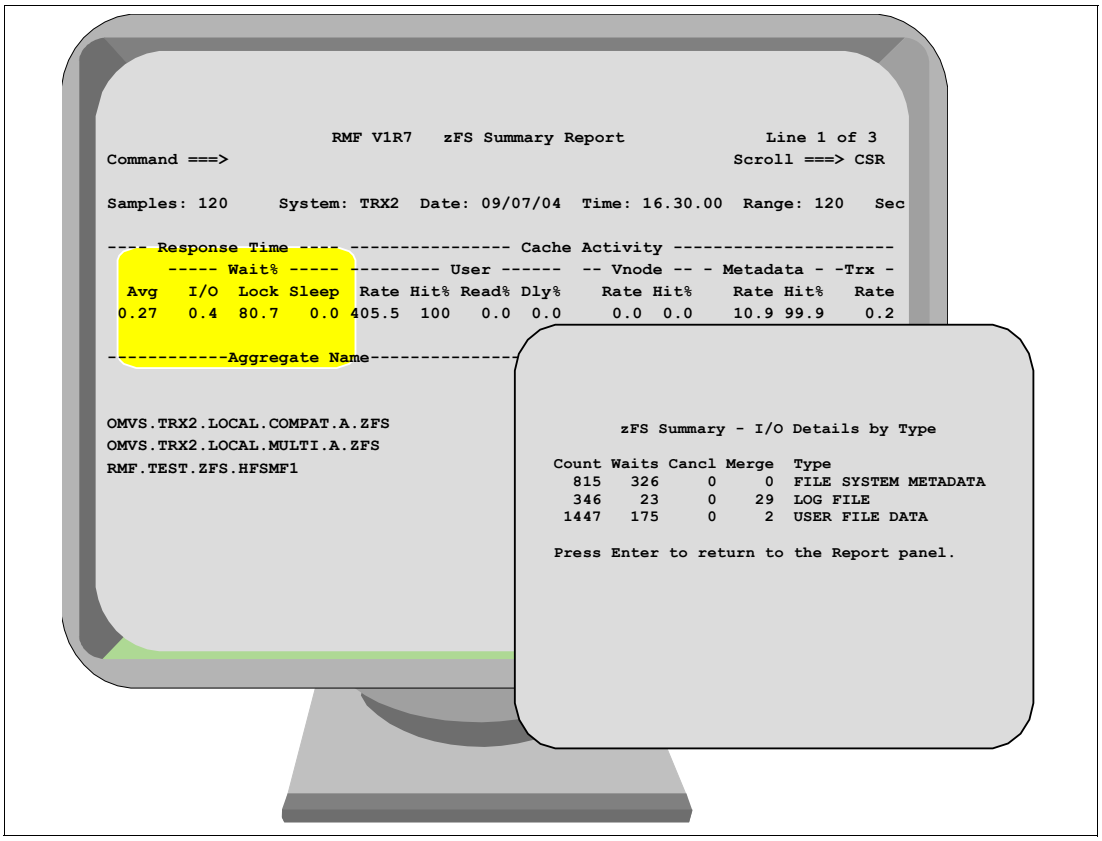


Figure 6-15 zFS Summary: I/O details

6.9.2 User and vnode cache detail reports

Figure 6-16 on page 316 shows the user and vnode cache details. Place the cursor on the User section under Cache Activity to display the user cache details panel.

Field descriptions for user cache

The fields in the User Cache Details report have the following meanings:

Read Delay% Percentage of read requests to the user file cache that was delayed.

The following reason is counted as a read request delay:

Read wait: Read has to wait for pending I/O. (For example, a read of a file found the file data pending read because of asynchronous read-ahead from DASD to the user file cache.)

Write Delay% Percentage of write requests to the user file cache that was delayed.

The following reasons are counted as write request delays:

Write wait: Write has to wait because of pending I/O.

Write faulted: Write to a file needs to perform a read from DASD. If a write only updates a portion of a page of a file and that page is not in

the user file cache, then the page needs to be read from DASD before the new data is written to the user file cache.

Async read rate Total number of read-aheads per second. Each read-ahead action for a file reads in one segment (up to 64 K) from DASD to user file cache.

Scheduled write rate Total number of scheduled writes per second. Each scheduled write action for a file writes one segment (up to 64 K) from user file cache to DASD.

Page reclaim writes Total number of page reclaim writes performed in the range time. A page reclaim write action writes one segment of a file from user file cache to DASD. Page reclaim writes are performed to reclaim space in the user file cache.

Fsyncs Shows how often applications requested that zFS synchronize a file's data to disk.

Vnode cache details

Cursor sensitivity in the Vnode section of the cache activity data displays the Vnode Cache Details panel.

The open count is important because an open file requires a UNIX System Services and zFS vnode. If there are more open files than the zFS vnode cache size, zFS is forced to allocate more vnodes to meet the requirements of the system.

zFS will never have more than vnode_cache_size extended vnodes, but it will have more than vnode_cache_size vnodes if it is forced to allocate above the vnode_cache_size due to UNIX System Services requirements.

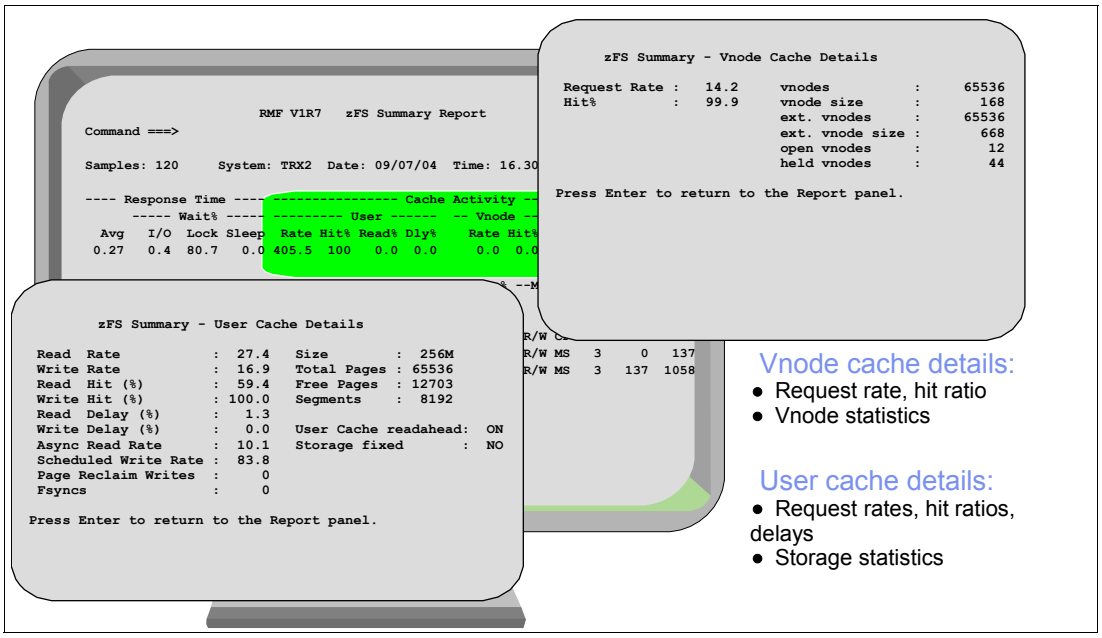


Figure 6-16 User and Vnode cache detail

6.9.3 Metadata and transaction cache detail reports

Figure 6-17 on page 317 shows the metadata and transaction cache details.

Metadata cache details

Cursor sensitivity in the metadata section of the cache activity data displays the metadata Cache Details panel.

The metadata backing cache is:

- Optional
- Can be used as an extension to the metadata cache
- Resides in a data space

Transaction cache details

Cursor sensitivity in the Trx section of the cache activity data displays the Transaction Cache Details pop-up panel. In this display, EC merge rate is the number of transaction class merges per second.

zFS decides when a transaction is related to, or dependent on, another transaction. When this determination is made, the transactions are grouped into an *equivalence class*. Any transactions in the same equivalence class are committed together or backed out together in the event of a system crash.

By using equivalence classes, threads running transactions simply run in parallel without added serialization between the two (other than locks if they hit common structures). They simply add their associated transactions to the same class. This increases throughput.

The merge of equivalence classes occurs when two transactions that need to be made equivalent are both already in equivalence classes. In this case, both classes are merged.

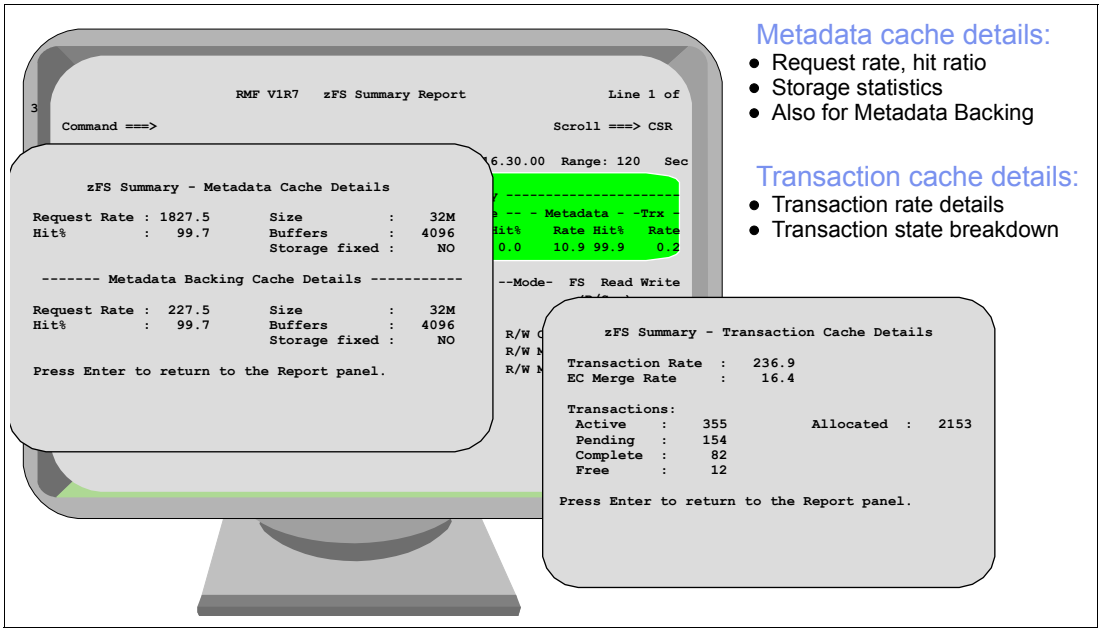


Figure 6-17 Metadata and transaction cache details

6.10 RMF zFS Activity Report

The zFS Activity Report, shown in Figure 6-18 on page 318, displays information about zFS file systems. When invoked from the RMF Overview Report selection menu or by the ZFSACT/ZFSA command, the file systems for *all* attached aggregates are reported.

When the report is invoked by cursor-sensitive control from the zFS summary report, the file systems for *one* aggregate name are reported.

6.10.1 Field descriptions

Note the following descriptions of the fields.

Aggregate name Set by cursor-sensitive control on an aggregate name in the zFS summary report. `ALL` indicates that file systems for all attached aggregates are reported.

File System Name Name of the file system or UNIX System Services file system name.

Mount Point Mount point of the file system.

Mode File system mount mode:

R/W: read-write

R/O: read-only

N/M: not mounted

QSC: file system not available because the aggregate is quiesced

Quota Limit Maximum size of the file system (known as the *quota*). The quota is a logical number. When the quota is reached, the file system indicates that it is full.

Quota Usg% Percentage of the quota currently used by the file system.

Operation Rate Total number of vnode operations per second for this file system.

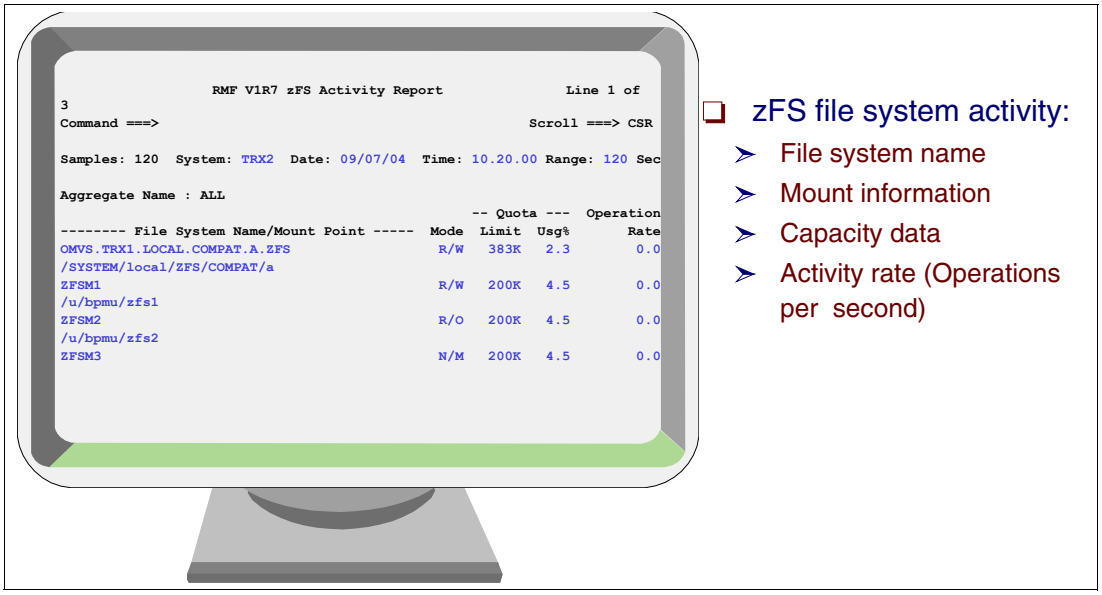


Figure 6-18 zFS Activity Report

6.11 RMF new messages

zFS support introduces new RMF Monitor III reporter messages:

```
ERB944I - Report is not available, reason code x.
1 = OMVS is not active (or not available)
2 = zFS is not active or shutting down
```

```
3 = backlevel data or no data from zFS interface

ERB945I – No aggregate found
Within the current report interval, RMF did not detect any zFS data for the
aggregate.
```

6.12 New RMF PM resources and metrics

zFS support introduces new resources and metrics, as illustrated in Figure 6-19, that can be used by the RMF PM client.

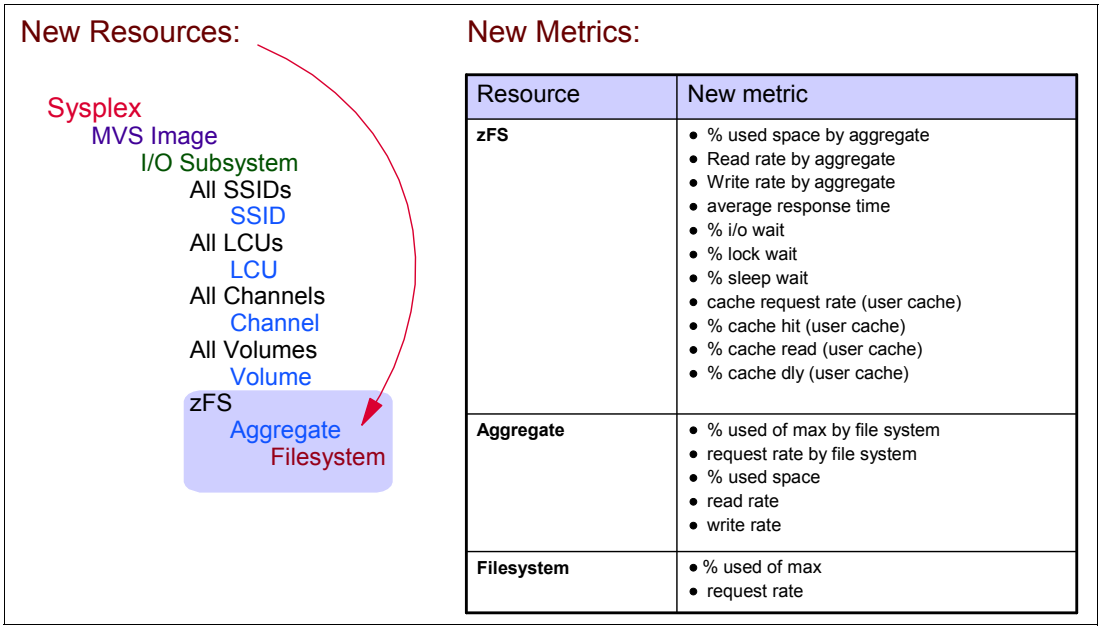


Figure 6-19 New RMF PM resources and metrics

6.13 RMF migration and coexistence considerations

With z/OS V1R7, RMF by default gathers the zFS activity data. You can suppress the gathering of zFS data by specifying the new Monitor III gatherer option `NOZFS`.

6.14 zFS performance and other parameter settings

This section discusses important zFS parameter settings and provides general suggestions about how to change default values and monitor the effects.

6.14.1 zFS user cache

The most important zFS parameter setting is *user_cache_size* for caching zFS file data. The default setting is 256 M. If you are using zFS more intensively, this value should be increased; use at least 1 GB. Specify the following value in the parameter file for the beginning:

```
user_cache_size=1024M
```

This value can be changed dynamically by using the **zfsadm config** command, as shown in Figure 6-20 on page 320.

```
#> zfsadm config -user_cache_size 2048M
IOEZ00300I Successfully set -user_cache_size to 2048M
#>
```

Figure 6-20 Dynamically changing the zFS user cache size

You can increase (and also lower again) the setting. In general, if more storage is available for user cache data, then the read hit ratio will be higher. The more zFS file systems you are using, the more user cache is desirable.

It is especially useful to monitor user cache hit ratio statistics to see whether your changes produce the desired effects. Beginning with z/OS V1R7, all the performance counters that are retrieved by the **F ZFS,QUERY,ALL** command can be displayed in sections. Refer to 6.6, “Performance monitoring APIs” on page 309 for more information about this topic.

To display user cache information, use the command **zfsadm query -usercache** as shown in Figure 6-21.

```
$> zfsadm query -usercache
User File (VM) Caching System Statistics
                  ---------------------------------------

                  Direct Statistics
                  -----------------

External Requests:
------------------
Reads          47301     Fsyncs             3     Schedules       2979
Writes          4840     Setattrs        2265     Unmaps          2413
Asy Reads      12324     Getattrs       51985     Flushes            0

File System Reads:
------------------
Reads Faulted        3851     (Fault Ratio    8.141%)
Writes Faulted          0     (Fault Ratio    0.00%)
Read Waits            171     (Wait Ratio    0.361%)
Total Reads          4320

File System Writes:
-------------------
Scheduled Writes          2517     Sync  Waits            1
Error Writes                 0     Error Waits            0
Scheduled deletes            0
Page Reclaim Writes          0     Reclaim Waits          0
Write Waits                  0     (Wait Ratio    0.00%)
...
$>
```

Figure 6-21 Displaying zFS user cache statistics

If a value has been identified to meet the installation requirements, place it into the IOEPRMxx parmlib member.

In parallel, you can also lower the amount of virtual storage allowed for HFS buffering. The defined or default HFS limits and statistics can be displayed by using the command shown in Figure 6-22 on page 321.

```
$> /usr/sbin/confighfs -ql
HFS Limits
 Maximum virtual storage: ______2010(MB)
 Minimum fixed storage:   _________0(MB)

HFS Statistics
( 04/22/08 7:30pm )
 Virtual storage:    _______172    (pages)
                     _________0.672(MB)
 Fixed storage:      _________0    (pages)
                     _________0.000(MB)
...
$>
```

Figure 6-22 Listing HFS limits and statistics

Figure 6-23 shows an example used to lower the HFS virtual storage limit to 512 MB and keep the fixed storage size at 0 MB.

```
#> /usr/sbin/confighfs -v 512 -f 0
#>
```

Figure 6-23 Lowering HFS buffer limits

You can set these values by using the HFS FILESYSTYPE definition shown in Figure 6-24.

```
 FILESYSTYPE TYPE(HFS)
             ENTRYPOINT(GFUAINIT)
             PARM('SYNCDEFAULT(60), VIRTUAL(512)')
```

Figure 6-24 Modified HFS filesystype statement in BPXPRMxx

If you are migrating more and more HFS data sets to zFS aggregates, then enlarge the zFS user cache size and lower the HFS buffer size in parallel again.

6.14.2 zFS vnode cache

The value specified for vnode_cache_size is the initial number of vnodes that will be cached by zFS. The number of vnodes with vnode extensions will not exceed this number. The default value is 32768.

Important: Double the number of vnodes to 65536 at least when running in a z/OS UNIX shared file system environment and sysplex-aware.

In an I/O intensive environment we recommend to set the value even higher, for example:

```
vnode_cache_size=80000
```

You can also dynamically enlarge the value as follows:

```
zfsadm config -vnode_cache_size 80000
```

Furthermore, there is a hiper APAR OA37950 reporting a problem in vnode handling.

Important: We recommend to apply fixes for APAR OA37950 in zFS releases 3B0 (z/OS V1R11 and V1R12) and 3D0 (z/OS V1R13) when they are available.

6.14.3 Automatic aggregate growing

To allow the automatic growth of zFS aggregates, add the following statement to the zFS parameter file:

```
aggrgrow=on
```

Refer to 2.13.2, “Dynamic aggregate extension” on page 63 for more detailed information about this topic.

The value can also be set or changed dynamically by using the **zfsadm config** command.

Note: Beginning with z/OS V1R13 the value **on** is the default aggrgrow setting.

6.14.4 zFS sync interval

zFS writes cache data to disk asynchronously to other processing. This is different compared to HFS. Therefore, when the sync interval is reached in case of zFS, there is normally not much left do because most of the data is on disk already. Therefore, a useful setting for the sync interval time is as follows:

```
sync_interval=60
```

The value can also be set or changed dynamically by using the **zfsadm config** command.

6.14.5 New block security

The setting for nbs specifies whether new block security is used generally. If a file was extended or new blocks were allocated for a file, but the user data was not written to disk when a failure occurred, then the newly allocated blocks are shown as all binary zeros (0) and not whatever was on disk in those blocks at time of failure.

Setting this value to OFF gives some performance benefit:

```
nbs=off
```

The value can also be set or changed dynamically by using the **zfsadm config** command.

Attention: In z/OS V1R13 the zFS configuration option **-nbs** is obsolete and no longer used. The new block security is without any disadvantage independant of the setting.

6.14.6 Read ahead of user file data

Depending on whether your applications are using more sequential access to file system data or more random access, set the following parameter setting either ON or OFF. Here is the default setting:

```
user_cache_readahead=on
```

The value can also be set or changed dynamically by using the **zfsadm config** command.

6.14.7 zFS directory and metadata cache

Suggestions for setting or taking default values for the directory cache and the metadata cache size are provided in 6.5, “Directory cache and large directories” on page 307. Figure 6-25 shows these values.

```
dir_cache_size=32M
meta_cache_size=256M
```

Figure 6-25 zFS directory and metadata cache

Note: The directory cache has been removed in z/OS V1R13. The metadata cache took over this function in addition. (See also “Changes in IOEPRMxx configuration options” on page 125.)

We recommend to use a small metaback cache in addition to the meta cache. The metaback cache is a backing cache used to contain metadata. This resides in a data space and can optionally be used to extend the size of the metadata cache.

Restriction: In order to be able to use and change the metaback cache ensure that there is a metaback_cache_size statement in your parm file on starting or restarting zFS.

Therefore, we recommend at least the settings shown in Figure 6-26.

```
meta_cache_size=256M
metaback_cache_size=32M
```

Figure 6-26 zFS metadata and metaback cache settings in IOEPRMxx

6.15 HFS and zFS file system comparison

We created a simple test scenario to compare large file access behavior between HFS and zFS. The REXX procedures we used are listed in B.9, “LARGEFIL procedure” on page 399. These tests are performed in single systems and not in a UNIX System Services file system sharing environment.

Refer to the following sections for the JCL needed to perform this comparison:

- To create the file systems
  - C.6.1, “Define the HFS data set” on page 471
  - C.6.2, “Create the zFS aggregate” on page 471
- To mount the file systems

  C.6.3, “Mount the file systems” on page 472
- To fill the file systems with data

  C.6.4, “Create a large file in a file system” on page 473
- To access the data with a variety of processes

  C.6.5, “Test with a specified amount of large random I/Os” on page 473

6.15.1 Description of the test environment

We prepared and ran the test environment as described here.

zFS cache sizes

zFS was started with the cache sizes shown in Figure 6-27. The results of using this cache size are shown as ZFS1 in Table 6-3 on page 328.

```
**********************************************************************
* zSeries File System (zFS) Sample Parameter File:  ioefsprm
...
*adm_threads=5
*auto_attach=ON
user_cache_size=256M
log_cache_size=64M
sync_interval=60
*vnode_cache_size=5000
nbs=off
*fsfull(85,5)
*aggrfull(90,5)
...
```

Figure 6-27 zFS cache sizes defined in IOEFSPRM

HFS cache size

We started with the HFS cache size value shown in Figure 6-28.

```
$> /usr/lpp/dfsms/bin/confighfs -l
HFS Limits
 Maximum virtual storage: _______751(MB)
 Minimum fixed storage:   _________0(MB)
```

Figure 6-28 Querying the HFS cache limit

zFS cache size increased

We increased the zFS user file cache to 384 MB later, which provided better numbers for zFS. We also increased the cache size value for HFS and found no benefit in our test scenario. Nevertheless, we used this higher cache size during the tests that we discuss here. We did not use fixed storage for HFS and zFS. Figure 6-29 shows the increased zFS cache size used during the test as ZFS2.

```
#> /usr/lpp/dfsms/bin/confighfs -v 1500
#> /usr/lpp/dfsms/bin/confighfs -l
HFS Limits
 Maximum virtual storage: ______1500(MB)
 Minimum fixed storage:   _________0(MB)
```

Figure 6-29 Enlarging the HFS cache size

Defining the HFS file system

We defined an HFS file system large enough to hold a 500 MB file. The JCL used is shown in Figure 6-30.

```
//ZFSJOB   JOB ,'Define HFS File',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Define HFS File
//* Property of IBM  (C) Copyright IBM Corp. 1999, 2002
//* ------------------------------------------------------------------
//*
//  SET  HFSNAME=OMVS.LARGE.HFS            <=== HFS data set name
//  SET   VOLSER=TOTZF2                    <=== Volume
//  SET   HFSPRM=750                       <=== HFS primary allocation
//  SET   HFSSEC=50                        <=== HFS secon'y allocation
//*
//* ------------------------------------------------------------------
//CREATE   EXEC PGM=IEFBR14
//HFS      DD DSN=&HFSNAME.,DISP=(NEW,CATLG,DELETE),UNIT=SYSALLDA,
//            DCB=(DSORG=PO),SPACE=(CYL,(&HFSPRM.,&HFSSEC.,0)),
//            DSNTYPE=HFS,VOL=SER=&VOLSER.
//* ------------------------------------------------------------------
```

Figure 6-30 Defining the HFS file system

Defining and formatting a zFS aggregate

Then we created a zFS aggregate on the same disk, as shown in Figure 6-31.

```
zfsadm define -aggregate OMVS.LARGE.ZFS -volume TOTZF2 -megabytes 550 50
IOEZ00248E VSAM linear dataset OMVS.LARGE.ZFS successfully created.
zfsadm format -aggregate OMVS.LARGE.ZFS -compat -owner HERING -perms o755
IOEZ00077I HFS-compatibility aggregate OMVS.LARGE.ZFS has been successfully created
```

Figure 6-31 Creating the zFS aggregate

Mounting the file systems

We mounted the file systems at directory location `/u/zfs`, as shown in Figure 6-32 on page 326.

```
mkdir -m 755 /u/zfs/largehfs
/usr/sbin/mount -o sync(60) -f OMVS.LARGE.HFS -t HFS /u/zfs/largehfs
chmod 755 /u/zfs/largehfs
ls -Ed /u/zfs/largehfs
drwxr-xr-x        2 HERING   SYS1        8192 Apr 19 23:24 /u/zfs/largehfs
df -kvP /u/zfs/largehfs
Filesystem         1024-blocks        Used  Available  Capacity Mounted on
OMVS.LARGE.HFS     540000               20     539908        1% /u/zfs/largehfs
HFS, Read/Write, Device:89, ACLS=Y
Filetag : T=off   codeset=0

mkdir -m 755 /u/zfs/largezfs
/usr/sbin/mount -o noreadahead -f OMVS.LARGE.ZFS -t ZFS /u/zfs/largezfs
ls -Ed /u/zfs/largezfs
drwxr-xr-x        2 HERING   SYS1         256 Apr 19 23:30 /u/zfs/largezfs
df -kvP /u/zfs/largezfs
Filesystem         1024-blocks        Used  Available  Capacity Mounted on
OMVS.LARGE.ZFS     552343                9     552334        1% /u/zfs/largezfs
ZFS, Read/Write, Device:90, ACLS=Y
Filetag : T=off   codeset=0
```

Figure 6-32 Mounting the file systems

Filling the file systems with data

Afterwards we used a REXX procedure in a job to create and fill the file systems with a size of 500 MB in the HFS and the zFS file systems. Refer to B.9, “LARGEFIL procedure” on page 399, and in C.6.4, “Create a large file in a file system” on page 473 for the REXX procedure and the JCL we used.

It took 85.4 seconds to fill the zFS file system and 155.9 seconds to fill the HFS file system.

Accessing the file systems

This section describes how the I/O test was performed and what the selection criteria were to test access to the file systems.

- Choose either the zFS or the HFS file system.
- Define the number of processes that run in parallel to read or write 1 MB blocks of data.
- Specify the number of I/Os that each process has to perform.
- Select the percentage of reads among all I/Os. R70, shown in Figure 6-33 on page 327, means that 70% of the I/Os are reads and 30% are writes.
- Provide a seed value. This is used to create predictable random numbers for offsets into the large file when doing an I/O access.

The JCL for the job used, shown in Figure 6-33 on page 327, is where you specify the selection criteria for accessing the file system.

```
//ZFSJOB   JOB ,'LARGEIOS',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Run test with doing a specified amount of large random I/Os
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET    FDIR='/u/zfs/largezfs'         <=== Directory of large_file
//  SET   PROCS=10         <=== Number of parallel processes to start
//  SET  BLKIOS=200        <=== Number of 1MB block I/Os to perform
//  SET  TYPIOS=R70        <=== Type of IOs, Rxx: xx% Rs, (100-xx)% Ws
//  SET    SEED=88888      <=== Seed value for predictable random nbrs
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,
// PARM='LARGESCD &TIMEOUT &REXXLIB &PROCS &BLKIOS &TYPIOS &SEED &FDIR'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* -------------------------------------------------------------------
```

Figure 6-33 Job to run a specified amount of large random I/Os

6.15.2 Comparison results

We always used TYPEIOS=R70 and tested with 10, 20, and 40 processes running in parallel. With the HFS, we did not change the cache at all during the tests. With zFS, the cache still had good data in it after doing a number of I/Os. Therefore, we decided to always run a test twice for zFS to demonstrate how this may change response times.

Afterwards we stopped and restarted zFS to be sure the cache was invalidated. It turned out that good cache data may help for a second, similar test with up to about 20 processes involved.

Note: We did not include the test results for when the cache was invalidated in the table.

Random number specification

For the predictable random numbers, we used a seed value of 88888 for the 10 processes tests, a value of 76543 for the 20 processes tests, and a value of 66666 for the 40 processes tests just to produce some different patterns.

During the tests we examined CPU usage and found these effects:

- For both HFS and zFS, all processes use about the same CPU resources. In the case of zFS, however, the value is about 100% higher.
- Because the zFS processes only used 50% of the time that the HFS processes did, both types used about the same amount of CPU resources. Refer to Chapter 7, “Domino performance with zFS” on page 339 for further information about this topic.

Comparison results

In Table 6-3 on page 328, ZFS1 lists the results with a user cache size of 256 MB, which is the default. ZFS2 indicates the results using a cache size of 384 MB. The zFS address space was restarted to clear or invalidate the cache before running with a new cache size.

Table 6-3 Comparison results

	Processes	AVG secs	MIN secs	MAX secs
HFS	10	504.42	499.26	506.04
ZFS1	10	254.21	239.78	261.59
ZFS2	10	193.15	185.33	197.44
HFS	20	992.09	960.84	1003.27
ZFS1	20	486.69	465.67	499.93
ZFS2	20	311.28	284.40	325.54
HFS	40	1973.64	1893.17	2005.07
ZFS1	40	939.01	848.78	979.64
ZFS2	40	590.29	516.23	623.42

6.16 Performance comparisons with sysplex-sharing (R11)

This section describes a small, simple test scenario that you can easily set up and run in a UNIX System Services sysplex file system sharing environment, and then use to learn about the functions provided in zFS to improve performance when accessing a file system. The results will demonstrate the effectiveness of the enhancements added in z/OS V1R11.

Attention: With z/OS V1R11 we saw the first step towards direct I/O to eliminate the performance disadvantage for a client system on behalf of a zFS file system. See 5.12, “zFS Direct I/O and zFS installation changes (R13)” on page 293 for more information about direct I/O. Because the tests done here are very similar to what is described in 6.17, “Performance comparisons with sysplex-sharing (R13)” on page 332 with much better results there, we will probably remove this chapter the next time this book is released.

6.16.1 Test environments - description and setup

For this scenario we used two systems with UNIX System Services sysplex file system sharing, SC74 and SC75. First we created two file systems on the same volume, one of type HFS and one of type zFS. Then we created two files of 750 MB and filled them with data in both file systems.

Refer to the following sections for the JCL needed to perform this comparison:

- To create the file systems
  - C.8.1, “Define the HFS data set” on page 476
  - C.6.2, “Create the zFS aggregate” on page 471
- To mount the file systems

  C.6.3, “Mount the file systems” on page 472
- To fill the file systems with data

  C.6.4, “Create a large file in a file system” on page 473

- To access the data with a variety of processes

  C.8.5, “Test with a specified amount of large random I/Os” on page 478

The main actions are simply listed as a sequence of commands in Figure 6-34 on page 329.

```
#> mkdir -m 755 /u/hering/r11test/hfs
#> /usr/sbin/mount -o "sync(60)" -f OMVS.R11TEST.LARGE.HFS -t HFS \
> /u/hering/r11test/hfs
#> chmod 755 /u/hering/r11test/hfs
#> mkdir -m 755 /u/hering/r11test/zfs
#> /usr/sbin/mount -o noreadahead -f OMVS.R11TEST.LARGE.ZFS -t ZFS \
> /u/hering/r11test/zfs
#> /usr/sbin/mount -qv /u/hering/r11test/
----A- OMVS.R11TEST.LARGE.ZFS /u/hering/r11test/zfs
----A- OMVS.R11TEST.LARGE.HFS /u/hering/r11test/hfs
----A- HERING.ZFS          /u/hering
#> # Now creating and filling the files...
#> ls -l /u/hering/r11test/hfs
total 3072000
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 17:43 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 17:48 large_file_2
#> ls -l /u/hering/r11test/zfs
total 3073568
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 22:35 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 May  6 22:35 large_file_2
#> rexx 'Say "File size=" 786432000/(1024*1024)||"MB"'
File size= 750MB
```

Figure 6-34 Creating and filling the two HFS and the two zFS files

User and client cache with V1R11

The user_cache and client_cache are shown in Figure 6-35 on page 330. In z/OS V1R11, the cache is in two parts, as follows:

1. For requests at the zFS owner (also called the server), the user_cache_size cache is used.

   The default size is 256 MB, and the maximum is 65536 MB (which is 64 GB).

2. For requests at a system that is not the zFS owner (these systems are called *clients*), the client_cache_size cache is used.

   The default size is 128 MB, and the maximum is 65536 MB.

   In addition there is the client_replay_cache used to handle sysplex server replies.

   The default size is 40 MB, and the maximum is 128 MB.

HFS environment

In this scenario, the HFS file system was owned by system SC74 and the client system was SC75. The files were filled on system SC74. HFS was used with the default cache setting.

Sysplex-unaware zFS environment

The first zFS environment was set up to be sysplex-unaware as in past; the file system owner was SC74 as for HFS. The files were filled on SC74, as well. The user cache size was set to 512 MB.

Sysplex-aware zFS environment

The second zFS environment was set up to be sysplex-aware as it is by default in a z/OS V1R11 UNIX System Services sysplex file system sharing environment.

Attention: In z/OS V1R11, by default zFS runs sysplex-unaware for read-write mounted file systems. To change this, specify **`sysplex=on`** in the parameter file.

We used the same zFS files as before and the same user cache size of 512 MB. The client cache size was set to 512 MB, as well. Figure 6-35 illustrates this environment.

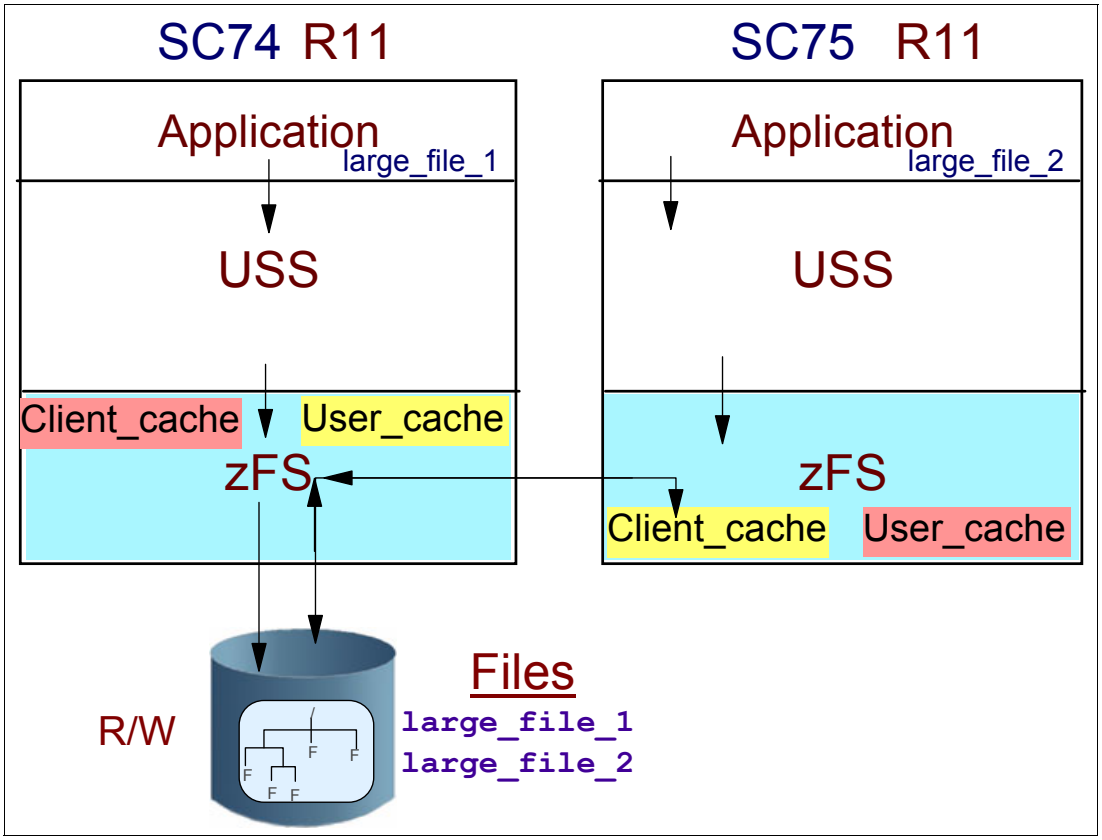


Figure 6-35 Sysplex-aware zFS environment with owner SC74 and client SC75

Accessing the file system files in the tests

The tests done were set up as follows.

- We set up two jobs running in parallel on different systems for every environment tested. One job accessed large_file_1 in system SC74. The other job accessed large_file_2 in system SC75.
- Each job locally started 20 processes running in parallel and performed 200 I/Os to read or write 1 MB blocks.
- 70% of the I/Os were reads and 30% were writes. This percentage of reads is seen to be realistic for applications.
- The sequence of I/Os was based on a set of predictable random numbers for offsets into the large files when performing an I/O access. The expression “predictable” means that all tests are using exactly the same set of random numbers and make the results directly comparable.

6.16.2 Test results

The following single tests provide the necessary information to clearly identify what is set in the environment.

HFS numbers

Table 6-4 lists the results for the HFS tests, locally in SC74 and remote on SC75. There are no useful numbers for the client system because all client requests must be forwarded to the owning system for access to the file system.

Table 6-4 HFS test results

	AVG sec per process	MIN sec per process	MAX sec per process
HFS on owning system	68.44	67.24	68.93
HFS on client system	353.91	351.73	354.81

Results for zFS being sysplex-unaware

Table 6-5 lists the results for the old zFS behavior and the setup as described in “Sysplex-unaware zFS environment” on page 329. The results on the zFS owning system are better. However, the numbers for the zFS client are slightly better than for HFS but still not good due to sending function shipping requests to the owning system.

Table 6-5 zFS test results with sysplex=no

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	13.19	7.77	15,90
zFS on client system	206,80	205,23	207,62

Results for zFS being sysplex-aware

Table 6-6, “zFS test results with sysplex=yes” on page 331 lists the results if zFS is running with **`sysplex=on`**. In this test, both important caches (the user cache and the client cache) are set to 512 MB. In addition we changed the client reply cache to 128 MB, to avoid bottlenecks. The zFS client system now had much better performance because it could access data in the cache on the same system and did not always have to go to the owning system for data.

Important: The user cache is used for caching on the zFS owning system. The client cache is used for client access.

In our scenario, even the client results improved. Note that in this test we did not switch ownership, so these numbers really reflect access from a remote system. However, depending on the sequence of data access, switching the zFS ownership may provide a benefit.

Table 6-6 zFS test results with sysplex=yes

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	14.14	6.96	16.20
zFS on client system	128.27	124.40	136.54

Tests exploiting automatic move of zFS aggregate ownership

Table 6-7 lists the test results on a system that is the zFS client initially, when the test starts.

Important: In z/OS V1R11 with zFS being sysplex-aware, a read-write zFS file system's (zFS) ownership is automatically moved to the system that is most using the file system if it is initially mounted on the wrong system.

These results show that, probably close to the moment when the first 200 I/Os were completed, zFS decided to switch the ownership. Based on the number of I/Os performed, the example illustrates the effectiveness of this zFS behavior.

Table 6-7 zFS test results with sysplex=yes on initially client system

No of 1MB I/Os	AVG sec per process	MIN sec per process	MAX sec per process
200	120.02	86.25	132.17
2000	373.34	278.57	381.16

From the numbers seen in this test and behavior, we can say that the zFS implementation with automatically changing the zFS owner resulted in the most effective change for usage of read-write zFS file systems.

Further tests and findings

In our case, we performed further testing and we provide a brief overview of the results in this section.

- We performed a special test that only read data on the client side. We found no essential difference to a case with a low amount of writes.
- In the tests described, we still saw client cache read fault ratios of about 30%. Therefore, we ran further tests with a client cache size of 1024 M and experienced response times that were comparable to the server side because most of the data could be handled or was found in the cache already.
- As long as writes do not enlarge a file, we found that there is no need to move data back at once to the owning system, and that all I/O operations can be performed with the client cache.

Note: A high client cache size, in combination with low (or no) file increasing on I/Os, can provide very acceptable performance numbers on client systems.

6.17 Performance comparisons with sysplex-sharing (R13)

As in the past, provided here is a small and simple test scenario that can be easily set up and run in a USS sysplex file system sharing environment to get an idea about the functions provided in zFS to improve performance when accessing a file system. The results show how effective the enhancements that have been added in z/OS V1R13, named "the direct I/O" are.

6.17.1 Setup and description of the test environments

We used two systems with USS sysplex file system sharing, SC74 and SC75. First we created two file systems on the same volume, one of type HFS and one of type zFS. Then two

files of size 750 MB were created and filled with data in both file systems, as shown in Figure 6-36.

```
#> mkdir -pm 755 /u/hering/r13test/hfs
#> /usr/sbin/mount -o "sync(60)" -f OMVS.R13TEST.LARGE.HFS -t HFS \
> /u/hering/r13test/hfs
#> chmod 755 /u/hering/r13test/hfs
#> mkdir -m 755 /u/hering/r13test/zfs
#> /usr/sbin/mount -o noreadahead -f OMVS.R13TEST.LARGE.ZFS -t ZFS \
> /u/hering/r13test/zfs
#> /usr/sbin/mount -qv /u/hering/r13test/
----A- OMVS.R13TEST.LARGE.ZFS /u/hering/r13test/zfs
----A- OMVS.R13TEST.LARGE.HFS /u/hering/r13test/hfs
----A- HERING.ZFS          /u/hering
#> # Now creating and filling the files...
#> ls -l /u/hering/r13test/hfs
total 3072000
-rw-rw-rw-   1 HERING   SYS1     786432000 Mai  5 13:54 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 Mai  5 14:06 large_file_2
#> ls -l /u/hering/r13test/zfs
total 3073568
-rw-rw-rw-   1 HERING   SYS1     786432000 Mai  5 23:24 large_file_1
-rw-rw-rw-   1 HERING   SYS1     786432000 Mai  5 23:24 large_file_2
#> rexx 'Say "File size=" 786432000/(1024*1024)||"MB"'
File size= 750MB
```

Figure 6-36 Creating and filling the two HFS and the two zFS files

User and client cache with V1R13

The user_cache and the old client_cache are shown in Figure 6-35 on page 330. In z/OS V1R13, the cache is back to be one single cache for all situations, as follows:

- For any requests (at the zFS owner, also called the server, and at zFS clients), the user_cache_size cache is used.

  The default is 256 MB, the maximum size is 65536 MB (which is 64 GB).

- For requests at a system that is not the zFS owner (these systems are called clients), the client cache that was used in z/OS V1R11 and V1R12 is no longer needed and has been removed. It is the user cache that is used in this case now, too.

For reference, Figure 6-37 on page 334 shows a basic picture for zFS cache structures in z/OS V1R13.

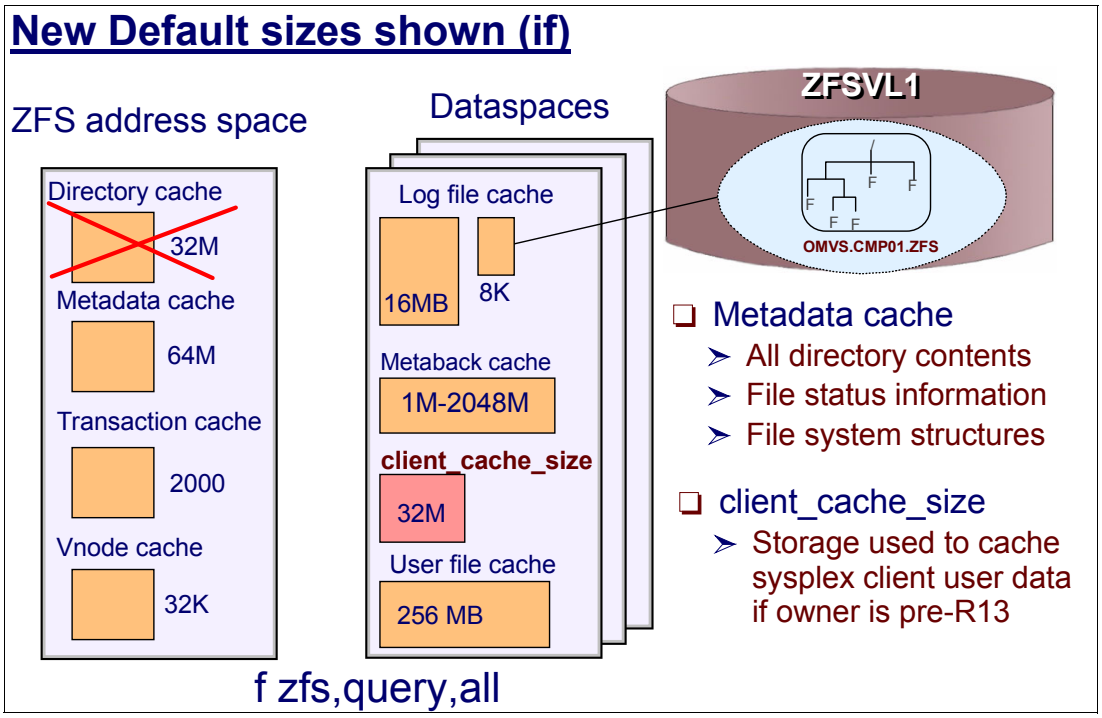


Figure 6-37 R13 zFS cache types

The HFS environment

The HFS file system was owned by system SC74, and SC75 was the client system. Filling the files was performed on system SC74. HFS was used with the default cache setting.

Sysplex-unaware zFS file system mount

In the first case the zFS file system was mounted sysplex-unaware. The file system owner was SC74, as for HFS. Filling the files was performed on SC74 as well. For both systems, the user cache size was set to the default value, which was 256 MB.

Sysplex-aware zFS file system mount

The second zFS environment was set up to use the file system mounted sysplex-aware.

Attention: In z/OS V1R13 zFS is mounting file systems sysplex-aware (for read-write mounted file systems) by default if the actual setting of `sysplex_filesys_sharemode` is rwshare.

We used the same zFS files as before and the same user cache size of 256 MB. Figure 6-38 on page 335 shows a picture of this environment.

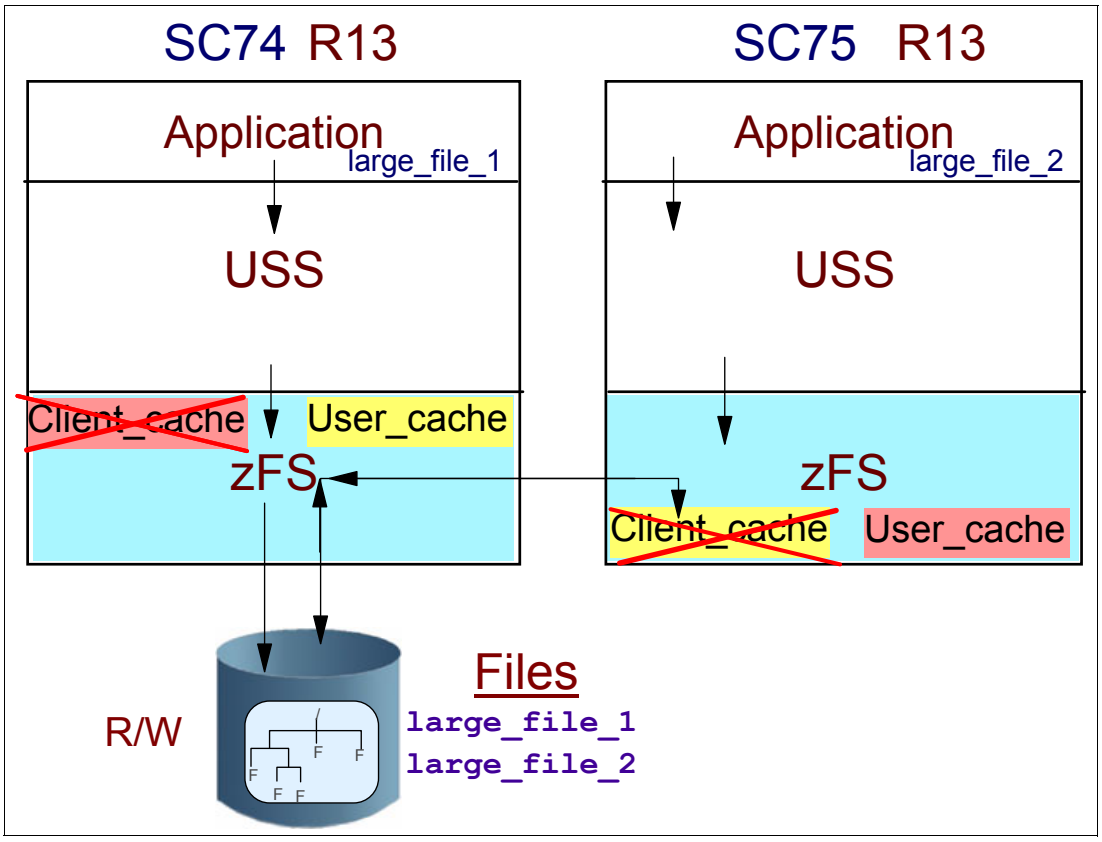


Figure 6-38 Sysplex-aware zFS file system with owner SC74 and client SC75

Accessing the file system files in the tests

The tests done were generally set up as follows:

- We set up two jobs running in parallel on different systems for every environment tested. One was accessing large_file_1 in system SC74 and the other large_file_2 in system SC75.
- Each job locally starts 20 processes running in parallel and doing 200 I/Os to read or write 1 MB blocks.
- 70% of the I/Os are reads and 30% are writes. This is a percentage of reads that is seen to be realistic for applications.
- The sequence of I/Os is based on a set of predictable random numbers for offsets into the large files when doing an I/O access. The expression “predictable” means that all tests are using exactly the same set of random numbers, making the results directly comparable.

6.17.2 Test results

The following single tests provide the necessary information to clearly identify what is set in the environment.

HFS numbers

In Table 6-8 on page 336 the results for the HFS tests, locally in SC74 and remote on SC75, are shown. There are no good numbers for the client system since all client requests must be forwarded to the owning system for access to the file system.

Table 6-8 HFS test results

	AVG sec per process	MIN sec per process	MAX sec per process
HFS on owning system	67.39	66.61	68.28
HFS on client system	780.41	778.60	781.93

Results for zFS being sysplex-unaware

In Table 6-9 the results for the old zFS behavior and the setup as described in "Sysplex-unaware zFS environment" on page 329 are shown. The results on the zFS owning system are good. The numbers for the zFS client are slightly better than for HFS but still bad due to sending function shipping requests to the owning system.

Table 6-9 zFS test results with file system mounted sysplex-unaware

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	12.30	11.18	12.87
zFS on client system	481.75	479.26	483.30

Results for zFS being sysplex-aware

First in Table 6-10 we list the results if the zFS is mounted sysplex-aware. In this test both user cache sizes are set to 256 MB. The zFS client system now has better performance since it can access and change data without having to go to the owning system for data.

Important: The local user cache is used for caching if it is the zFS owning system and also if it is a client system.

In this situation there is no difference between the owner and the client results.

Table 6-10 zFS test results with file system mounted sysplex-aware

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	21.45	18.87	22.25
zFS on client system	21.45	20.21	22.39

If you compare the owner numbers, the sysplex-unaware case looks better but this is a result of the client now doing well the same way. See the numbers shown in Table 6-11 for reference.

Table 6-11 zFS owner-only test results with file system mounted sysplex-aware

	AVG sec per process	MIN sec per process	MAX sec per process
zFS on owning system	11.91	11.30	12.42

Based on these results we can clearly state the following:

You can now share zFS file systems mounted in read-write mode in a USS sysplex file system sharing environment with no performance disadvantage for a client system on data access. Therefore, the old requirement MR1211023911 addressed to zFS is resolved.

zFS metadata considerations

While data access from a client system is no longer a problem, restrictions still apply to complex access operations to metadata.

Attention:

- We still recommend to run commands resulting in complex access to metadata, for example `find $MPDIR -xdev -exec ls -Ed {} ";"`, on the zFS owning system.
- If the UNIX file structure is large enough, you may see the zFS ownership moving if starting the command on a client system in such a situation.

7

Domino performance with zFS

This chapter provides performance information about the Domino® Server environment, comparing an HFS file system to a zFS file system.

It discusses the following topics:

- The Domino Server environment
- Tasks performed by the Domino Server
- Workloads used for the test environment
- Domino Server test results

7.1 Domino and zFS performance

DFSMS has provided the only local physical file system (HFS, available for OS/390 and z/OS since UNIX System Services was first released), that hardens data to DASD. It was rewritten with the DFSMS 1.5 release that shipped with OS/390 V1R6 to enhance performance, and this provided a substantial performance improvement over previous releases. It did not, however, cache reads or writes for Domino data because the design point for DFSMS 1.5 was to cache only those files that are 1 MB or less in size, and practically all Domino databases are larger than this limit.

zFS has no such restrictions on caching, and it has mechanisms that allow administrators to control how the file system caches I/O. It provides the administrator with more flexibility to apply more system resources (primarily storage) to any operations that are performance-sensitive. This chapter explains these benefits to help administrators make informed decisions about zFS in their environments.

The Domino server uses UNIX files for many different purposes, and the characteristics of the I/O for each of these purposes are different. We examine the performance of the two file systems for each of these types of I/O to illustrate the characteristics of each. We also show how these differences manifest themselves to the user.

7.2 The Domino server environment

The test environment we used simulated an enterprise server domain. The file system structure for the environment is shown in Figure 7-1.

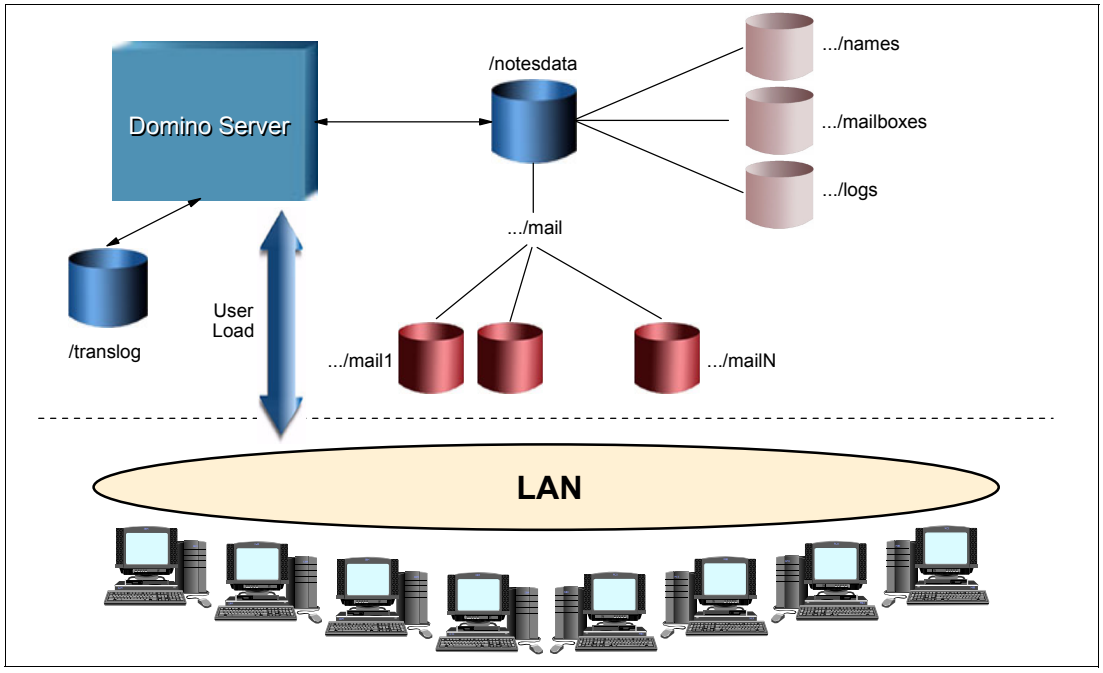


Figure 7-1 File system structure for the test environment

Many of the file systems in a Domino environment have a common purpose and can be grouped together according to the usage patterns that the server drives against them, as explained here:

/translog This is the Domino transaction log1. All I/O to a Domino database passes through the transaction log in a two-phased commit that provides a modest performance advantage and better recovery characteristics for the server. I/O to and from the transaction log is sequential, with a small number of logger threads performing all of the I/O.

The transaction log resides in a file system as a series of files, all about 64 MB in size, plus a small control file. The logger processes these files using a small number of threads (2 or 3, depending on the configuration of the logger). Transaction logging can run in circular, linear, or archive mode, but these modes have largely the same I/O characteristics. That is, data is written sequentially to one of the logging files until the file fills up, when the logger moves on to the next file. A separate thread reads these logger files and writes the data it reads to the appropriate target database.

Although contention for these logger files is low, the I/O rates to these files are very high. We strongly recommend that you put the transaction log in a separate file system on a separate DASD device, and have plenty of channel bandwidth to the device.

/names This is the Name and Address Book (NAB, now called the Domino Directory) for the domain. This one database completely describes the Domino domain, from user and server identities to ACLs to data routing information. I/O to this file tends to be random, with a potentially large number of users attempting to perform I/O at the same time. I/O is primarily read, with few write operations.

The Domino Directory is used to authenticate every client request made of the server, every server-to-server interaction, and the execution of all Domino applications that run on the server. It is the common resource for all access control requests required of the server. This usually makes the Directory the common resource most contended for by the server. For this reason, it is often advisable to treat it like the transaction log, that is, isolate it from the rest of the /notesdata directory and file system.

/notesdata This is the notes data directory, which is the home directory for the Domino server and contains most of the common resources used for Domino server operation. I/O to files in this file system tends to be random, with potentially many threads attempting to perform I/O simultaneously.

This file system usually contains the central mail boxes (mail.box*) used as a clearinghouse for all mail routed to users on that server. It also often contains the Name and Address Book for the server as well. Our tests have the NAB residing in both a separate file system and the Notes data directory, depending on the object of the test.

/logs These are the server logs. These contain console data from the running server. I/O here tends to be sequential in nature with a small number of writers, and occasional readers. This data is often separated into a discrete file system because of space considerations, and not to avoid contention issues. I/O is primarily write, with relatively few read operations.

/mailx This are the mail file systems. They hold databases for mail users, the workload most often supported by a Domino server. I/O here is random, generally with a 1-to-1 relationship between a thread or user and database. Contention for a mail database is not often an issue.

These databases tend to range in size from 50 MB to 250 MB, although some power users have mail databases in the gigabyte range. Because of space management issues for these databases, spreading these files over several file systems is the most practical option. Reads and writes generally tend to be about evenly proportioned.

7.2.1 Tasks performed by the Domino server

The Domino server performs different tasks that fall into two broad classifications: offline housekeeping functions, and servicing client requests. Each class of tasks causes a different type of stress on the server.

Offline housekeeping functions

The offline housekeeping functions are used to perform maintenance and migration operations on Domino databases. These functions are usually performed when the server is down, either during scheduled maintenance windows or during migrations from other servers. They generally require a single thread of execution, and operate on a single database. The functions we measured include:

- Compact
  - Copy-style compaction (**-c**)

    A temporary copy of the database is made during the compact process, and any required structural changes are performed. This type of compact recovers space internally within the database, and reduces the size of the file.
  - In-place compaction without file reduction (**-b**)

    Compact a database “in place.” No temporary copy is required. Space is recovered internally within the database, but no reduction in actual file size occurs.
  - In-place compaction with file reduction (**-B**)

    Compact the database “in place.” No temporary copy is required. Space is recovered internally within the database, and the size of the file is reduced.
- Updall

  Rebuild all full-text indexes and all used and unused views in the database from scratch (-RC).
- Create a full-text index (FTI)

  Create a new full text index to facilitate keyword searches in a target database.

Note: For more detailed explanations of these functions, see the Domino 5 Administration Help database.

These functions were chosen for this test because they are key migration steps whenever you move from one feature release of the Domino server to another. They often provide the first impression of performance that an administrator has of the platform, and they have historically been areas where performance has been an issue.

We measured the performance of these functions while they were being performed against the IBM US Directory (NAB). When compacted to bring this database to On Disk Structure 41 format (the standard R5 ODS level), this database is approximately 1.24 GB in size, with 191,000 person documents, 1850+ groups, and 1900+ connection documents for other purposes.

Client-driven workloads

These workloads are primarily standard benchmark applications that simulate different flavors of client access to mail. They create many virtual threads of execution to drive client access to many different databases served by Domino. As with the common housekeeping functions, there is largely a 1-to-1 relationship between virtual thread and target database. The difference is that there are large numbers of these virtual threads running simultaneously.

Virtual threads are a Domino abstraction, used to represent each client request being serviced. They provide a context within which the server takes actions on behalf of the client against a specified database. Rather than allocating a UNIX pthread for each virtual Domino thread, the server maintains a relatively small pool of pthreads that are all driven by a small dispatch routine. This routine essentially causes each pthread to sleep until a client request arrives for servicing. When this happens, one of the pthreads is bound to a virtual Domino thread, and the conversation between client and server proceeds.

What all this means from an I/O standpoint is that although the server may appear to be servicing thousands of clients simultaneously, there are generally not more than about 50 physical threads of execution within the core Domino server that are being managed by z/OS at any one instant in time. Still, these physical threads that are running tend to be very active, and there are sufficient numbers of them to drive interesting amounts of I/O, particularly to common files, such as the Domino directory, and the files that make up the transaction log. During our testing we allocated 100 pthreads to the Domino thread pool.

The workloads used for this test that can be considered client-driven workloads included R5Mail, WebMail, and iNotes®, as explained here.

- **R5Mail** - This consists of standard R5 mail clients accessing their mail.

  This is a simulation of what is considered “light” mail users. Each user performs the following tasks in an average 15-minute interval:

  - Opens the inbox view.
  - Reads five documents.
  - Categorizes two documents.
  - Sends a message of 4000 bytes to six recipients.
  - Adds two documents to the inbox.
  - Schedules an appointment with a description of 4000 bytes.
  - Sends a meeting invitation of 4000 bytes to six recipients.
  - Deletes two documents from the inbox.
  - Responds to a meeting invitation.
  - Closes the view

  Table 7-1 lists and describes the workload parameters for R5Mail.

Table 7-1 Workload parameters for R5Mail

Parameter	Default value	Value for this test	Description
NormalMessageSize	1,000	4,000	Size of a message or invitation, in bytes
NumMessageRecipients	3	6	Number of users to send mail or invitations to others
NthIteration	6	1	Number of loops through test before sending mail or invitations to others

By setting these three parameters to values that are higher than the defaults, as shown in Table 7-1 on page 343, we substantially increased the amount of I/O required to support a single user, and therefore the stress on the underlying file system.

- **Webmail** - This consists of Web users accessing their mail.

  This is a simulation of “light” mail users browsing their mail using a regular Web browser. The first difference to note here is that no calendaring and scheduling functions (meetings) are exploited. Like the R5mail workload, this test works on a 15-minute interval to:

  – Send a message of 1000 bytes to 5 recipients
  – Delete a document
  – Read 5 documents

  As with the R5Mail workload, we tuned the parameters of the test to make the workload more I/O-intense; see Table 7-2.

Table 7-2 Workload parameters for Webmail

Parameter	Default value	Value for this test	Description
NormalMessageSize	1,000	1,000	Size of a message, in bytes
NumMessageRecipients	3	5	Number of users to send mail to others
NthIteration	6	1	Number of loops through the test before sending mail to others

- **iNotes** - This consists of Web users accessing their mail using the iNotes Web Access interface.

  This is an improved interface over the usual Web mail interface. It provides more features as well as a measurably lighter load on the server, both in terms of CPU consumption and I/O required. The functionality performed for this test is identical to the Webmail workload, except that the functions drive a different path through the Domino HTTP server when exploiting the iNotes Web Access interface. Given the lighter I/O workload of this test, we changed the size of the messages read and sent; see Table 7-3.

Table 7-3 Workload parameters for Inotes

Parameter	Default value	Value for this test	Description
NormalMessageSize	1,000	4,000	Size of a message, in bytes
NumMessageRecipients	3	5	Number of users to send mail to others
NthIteration	6	1	Number of loops through test before sending mail to others

Production Domino applications

During this study, we also tested two custom Domino applications to see how zFS would impact their response times and throughput characteristics. These applications were from two Domino for S/390® client production environments.

Our interest in the first application was to simply measure the elapsed time of a single thread of execution performing operations on a database. The second application had a requirement to support both a given throughput and response time for a set of concurrent users processing a single database. We generated a workload of 70 concurrent users against the database containing the application using LoadRunner1, and a custom workload script.

7.2.2 Test results

The test results indicate that in virtually all scenarios, zFS outperforms HFS by a substantial margin. We believe that this is due in large part to the fact that zFS is able to cache Domino data where HFS cannot. We understand from the zFS and HFS development teams that the design point for HFS (DFSMS 1.5) was to cache files of up to 1 MB, but larger files would not be cached. Because all Domino databases are larger than 1 MB in size, HFS is at an inherent performance disadvantage relative to zFS.

Offline housekeeping functions

Table 7-4 shows the measurements for all of the offline housekeeping functions, which are described in “Offline housekeeping functions” on page 342.

Table 7-4 Measurements for the offline housekeeping functions including elapsed time

Function	HFS Elapsed Time (secs)	zFS Elapsed Time (secs)	Difference	HFS:zFS
compact -c	2,631	1,065	-60%	2.47:1
compact -b	405	166	-59%	2.44:1
compact -B	3,179	1,548	-51%	2.05:1
updall -RC	17,553	5,490	-69%	3.20:1
create FTI	4,463	3,300	-26%	1.35:1

Although we do not have specific numbers to compare in this study, we made measurements showing that Domino with zFS will complete these housekeeping functions about 10% faster than a similarly configured Intel machine running Windows NT4.

Note: The CPU and storage requirements for the two file systems varied considerably, as shown in Table 7-5 and Table 7-6.

Table 7-5 Measurements for the offline housekeeping functions - CPU usage

Function	HFS CPU Usage (percent)	zFS CPU Usage (percent)	Difference	HFS:zFS
compact -c	11.38	18.63	+64%	1:1.64
compact -b	3.24	5.30	+64%	1:1.64
compact -B	4.23	12.02	+184%	1:2.84
updall -RC	10.07	27.22	+170%	1:2.70
create FTI	28.50	15.83	-44%	1.80:1

Table 7-6 Measurements for the offline housekeeping functions - central storage usage

Function	HFS Central Storage Usage (MB)	zFS Central Storage Usage (MB)	Difference	HFS:zFS
compact -c	704	1620	+130%	1:2.30
compact -b	623	1599	+157%	1:2.57
compact -B	639	1614	+153%	1:2.53

Function	HFS Central Storage Usage (MB)	zFS Central Storage Usage (MB)	Difference	HFS:zFS
updall -RC	858	1949	+127%	1:2.27
create FTI	712	1822	+156%	1:2.56

Although zFS resource usage is significantly higher here, it did not have to be. This test explored the best performance measurements achievable with both types of file systems. zFS could have been tuned to match the storage resources of HFS, and we could have gathered more CPU and elapsed time measurements for comparison. The key point illustrated by this data is that although HFS and zFS can be configured to use the same resources, zFS is capable of applying more available system resources to the tasks than it is asked to perform to enhance performance.

The data showing I/O to DASD, shown in Figure 7-2, indicates how effective the zFS caching is at preventing I/O requests from traveling all the way out to the hardware.

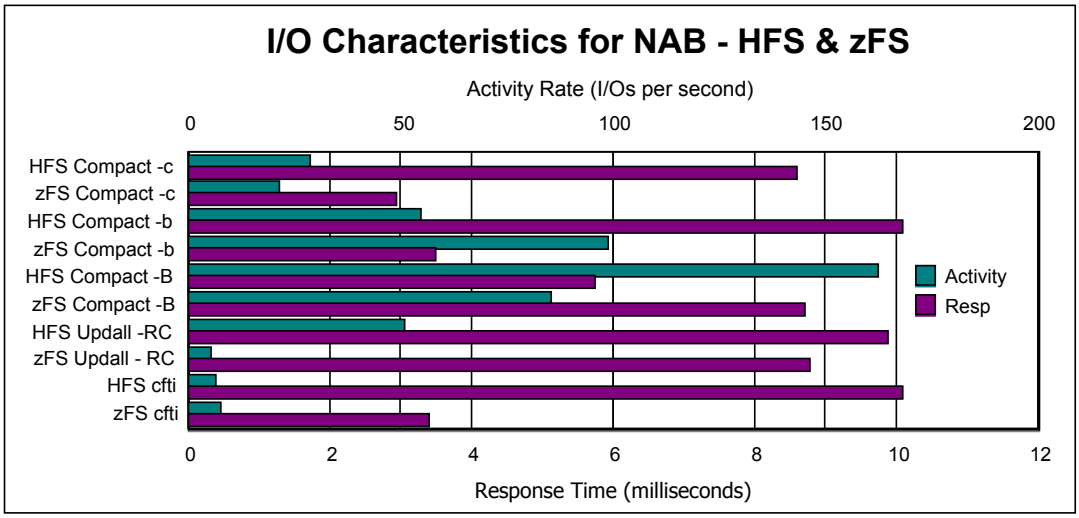


Figure 7-2 I/O characteristics for NAB between HFS and zFS

Each pair of bars indicates the activity rate and response time for the DASD supporting the file system containing the NAB. For instance, the chart shows that when running updall to rebuild views and indexes for the IBMUS NAB, the DASD had much less I/O activity (51.1 I/Os per sec. for HFS versus 5.6 for zFS), and marginally better response time (9.9 msec for HFS versus 8.8 msec for zFS).

What is interesting is that for any given task, either the activity rate or the response time for the DASD associated with the zFS measurement may actually be higher than for zFS. However, in all cases, the overall I/O intensity (activity rate multiplied by response time) is better for zFS than for HFS; see Table 7-7 on page 346.

Table 7-7 Measurements for the offline housekeeping function - I/O intensity

Function	HFS I/O Intensity	zFS I/O Intensity	Difference	HFS:zFS
compact -c	281	76	-73%	3.70:1
compact -b	557	347	-38%	1.61:1

Function	HFS I/O Intensity	zFS I/O Intensity	Difference	HFS:zFS
compact -B	843	641	-24%	1.32:1
updall -RC	552	52	-91%	10.62:1
create FTI	66	28	-58%	2.36:1

A closer look at the DASD response time reveals details about how HFS and zFS perform their I/O to the physical device; see Figure 7-3.

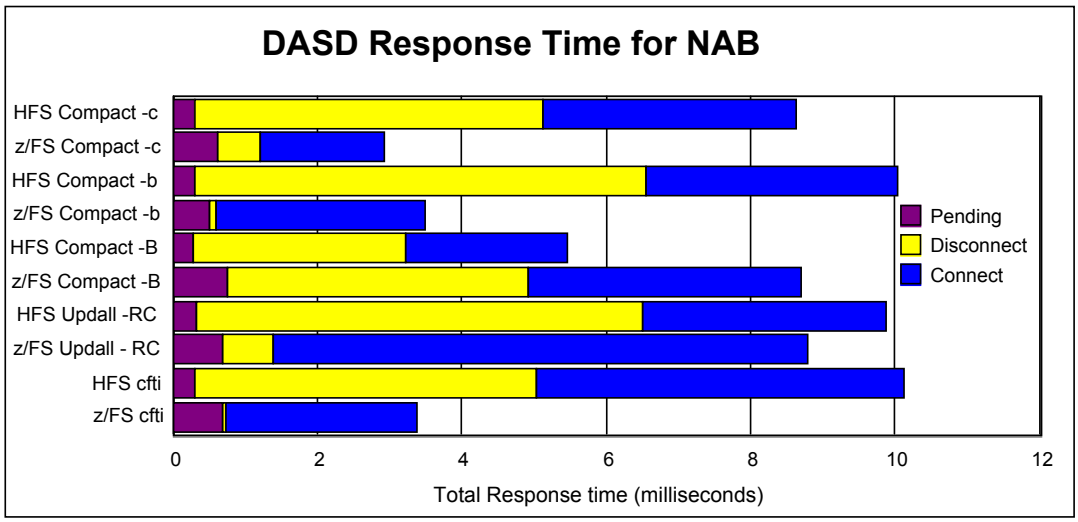


Figure 7-3 DASD response time for NAB

There was no I/O service queue (IOSQ) time component to the response time for either the HFS or zFS measurements, which was to be expected because there was only one thread of execution trying to use this file and its associated file system at a time. Several general patterns emerge from this chart:

- zFS has a slightly higher pending time in all cases, indicating longer waits in the path out to the device. This was likely due to a configuration reality of the environment because the HFS and zFS file systems were defined on different DASD devices, with different channel path characteristics. We do not believe this to be caused by zFS.
- In all cases except for the compact in place with file size reduction (compact `-B`), disconnect time for zFS is much better than for HFS. Disconnect indicates the amount of time that an I/O request is freed from the channel due to seek, latency, or rotational delays.

Connect time, which is the portion of an I/O request where data is actually transferred to the device, is a higher portion of the total response time for zFS in all cases. It appears that when zFS does have to do I/O to the device, it has the ability to do so more efficiently than HFS.

7.2.3 Client-driven workloads

As described, we used three different benchmark workloads to simulate users for this portion of the evaluation: R5Mail, Webmail, and iNotes Web Access. R5Mail users drive the server through the traditional Notes RPC interface. Webmail and iNotes Web Access users pass through Domino's HTTP server.

The Notes RPC interface is inherently more efficient, because the server and client cooperate to provide functionality to the user through a robust set of remote procedure calls. Web access to the server is somewhat more primitive in comparison, because the server is required to perform all formatting and rendering of the Web page before delivering it to the Web browser. As a result, Web-based clients are generally sized to be 3 to 4 times more CPU-intensive on the server size than Notes clients.

We included both Webmail and iNotes Web Access workloads here because both drive large amounts of I/O on the server in different and important ways.

R5Mail workload

We simulated 2,000 users for this test, all running the R5Mail workload. All of our results here were collected on a 5-minute interval for 1 hour after the steady state of the workload had been reached (all users ramped up and driving work on the server). We ran this test using zFS in three different configurations so that we could demonstrate the benefits of zFS when used for different purposes:

- All HFS - HFS was used for the transaction log, Notes data, and mail file systems.
- Mixed - zFS was used for the mail file systems, but the transaction log and Notes data remain on HFS.
- All zFS - zFS was used for the transaction log, Notes data, and mail file systems.

The response times, shown in Table 7-8, are average response times calculated by the benchmark application at the client.

Table 7-8 R5Mail workload with 2000 users

	CPU Usage %	Central Storage (MB)	Expanded Storage (MB)	End-User Response Time (msec)
All HFS	52.1	1,590	9*	902
Mixed	60.4	2,063	633	308
All zFS	63.4	2,063	657	211
Diff, All HFS versus All zFS	+22%	+30%	n/a	-77%

Note: * There is always a small amount of expanded storage in use; 9 MB in use for HFS is effectively zero.

In this configuration, we had 2 GB of main storage and 1 GB of expanded storage installed. HFS used about three-quarters of available main storage, and no significant expanded storage. zFS made use of all available main storage, and a substantial amount of expanded storage, presumably for caching purposes. The payoff is a large reduction in user response time. As with the offline housekeeping functions, we could have configured zFS to match the HFS resource usage, and traded off some user response time dividend for it.

DASD I/O characteristics for all of the file systems used for this test are as you might expect, with zFS driving much less I/O to the device. Figure 7-4 shows the I/O characteristics for the mail file systems.

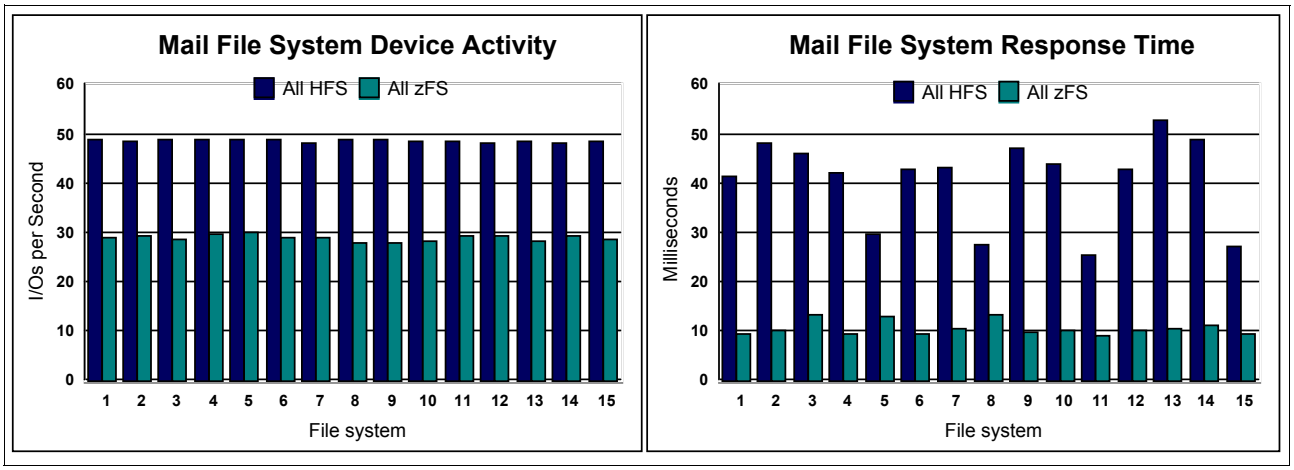


Figure 7-4 Mail file system device activity and response time

For this portion of the test, we spread the 2,000 simulated users across 15 mail file systems for a density of 133 users per file system. With this user density, and the tuning of the workload via the parameters listed (NormalMessageSize, NumMessageRecipients, NthIteration), we were able to drive the HFS to relatively long average DASD response times, and substantial I/O rates. HFS drove substantially less I/O, and allowed the DASD to respond with a generally acceptable response time. Note that there is no real difference between the Mixed and All zFS configurations, because the mail file systems were zFS file systems in both cases.

Figure 7-5 shows the components of the response time for a representative mail file system in both HFS, and zFS configurations. The first thing to note is that there is no I/O service queue time during any time interval for the zFS file system. There was no contention between threads simultaneously trying to reach the DASD with I/O requests. The second thing to note is that although disconnect time is substantial for the zFS case, it is still significantly smaller than that for HFS.

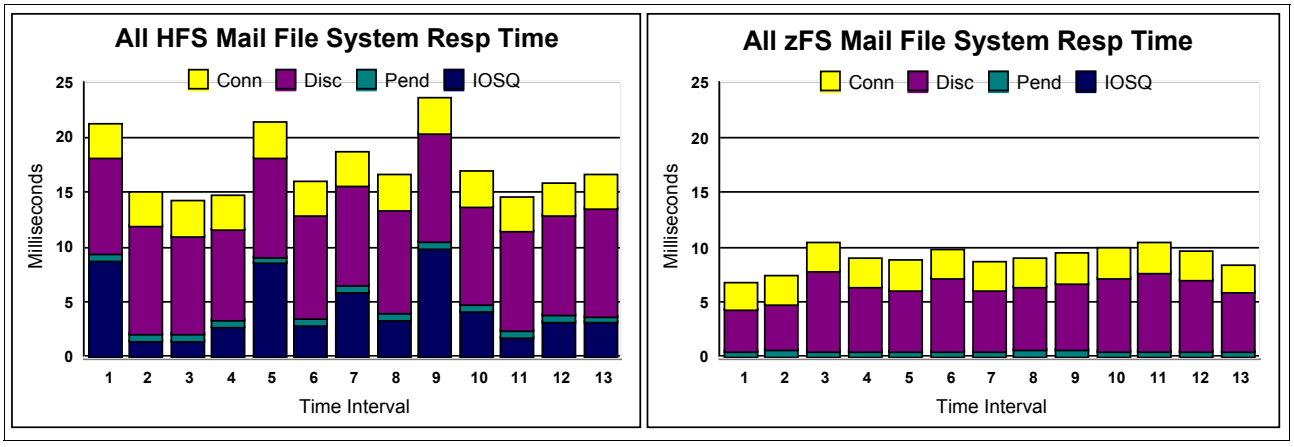


Figure 7-5 Mail response time comparison between HFS and zFS

The I/O characteristics for the Notes data shown in Figure 7-6 on page 350, and transaction log file systems shown in Figure 7-7 on page 350, are very interesting.

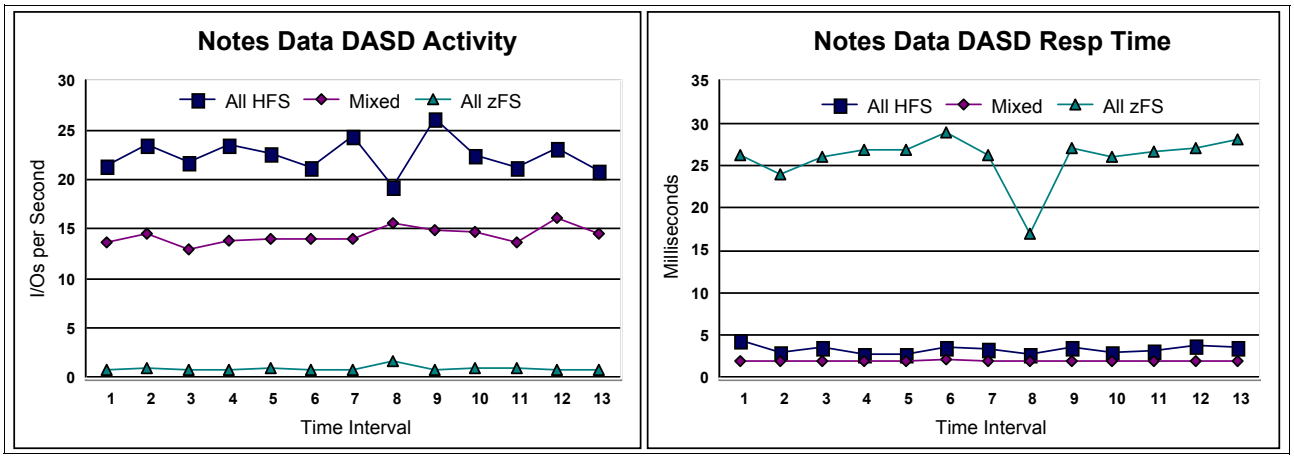


Figure 7-6 I/O characteristics for Notes data

DASD activity for the Notes data directory actually fell when the only change to the configuration was to move the target mail databases for the simulated users to zFS. This is not intuitively obvious, and we do not really have a good explanation for it. The other thing to note is that although the response time for the zFS configuration is several times higher than that for the HFS configuration, zFS was performing almost no I/O to DASD at the time. These long response times are inconsequential, given the very low DASD activity.

One detail not shown here is that the Notes data file system spanned two volumes in the HFS configuration, and four volumes in the zFS configuration. I/O was only performed to the first volume of each file system, so all of the DASD data presented here is for the first volume only.

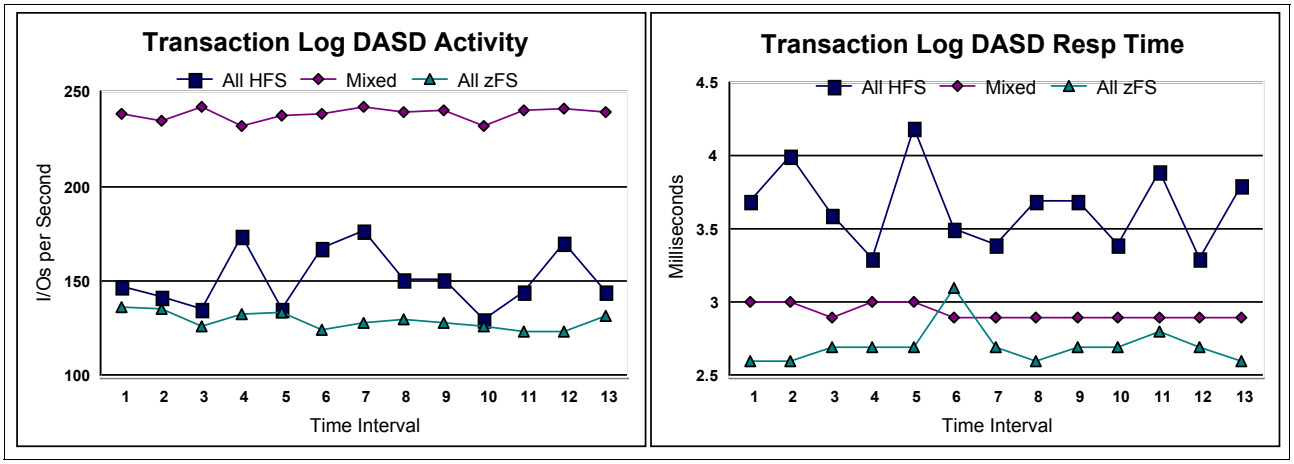


Figure 7-7 I/O characteristics for the transaction log

This data shows the high activity rates inherent in the use of the transaction log, and the relatively low response times for all three configurations. I/O to this file system is highly sequential, so it is not surprising that both file systems were able to handle the load very well. I/O service queue times were either absolutely, or practically, zero for all configurations. Similarly, pending and disconnect times were very low, thus making the connect time the largest component of the I/O response time in all cases.

As with the Notes data directory, shown in Figure 7-6 on page 350, we cannot account for the difference between the All HFS configuration, and Mixed file system configuration. The benchmark workload runs at a constant transaction rate, so differences in the DASD rates cannot be attributed to the workload stressing the server more or less heavily between the different tests.

The Webmail workload

For this workload we simulated 500 users, all running the Webmail workload previously described. All of our results here were collected on a 5-minute interval for 1 hour after the steady state of the workload had been reached (all users ramped up, and were driving work on the server). We used only two configurations for this test, All HFS, and All zFS.

From a storage usage and user response time standpoint, the results are very similar to those for R5Mail.

Table 7-9 Webmail workload with 500 users

	CPU Usage %	Central Storage (MB)	Expanded Storage (MB)	End-User Response Time (Msec)
All HFS	69.9	1,614	9*	1,229
All zFS	69.4	2,063	651	544
Difference	-1%	+28%	n/a	-56%

Note: * There is always a small amount of expanded storage in use; 9 MB in use for HFS is effectively zero.

What is substantially different here is the CPU consumption, which shows essentially no difference between the HFS and zFS configurations.

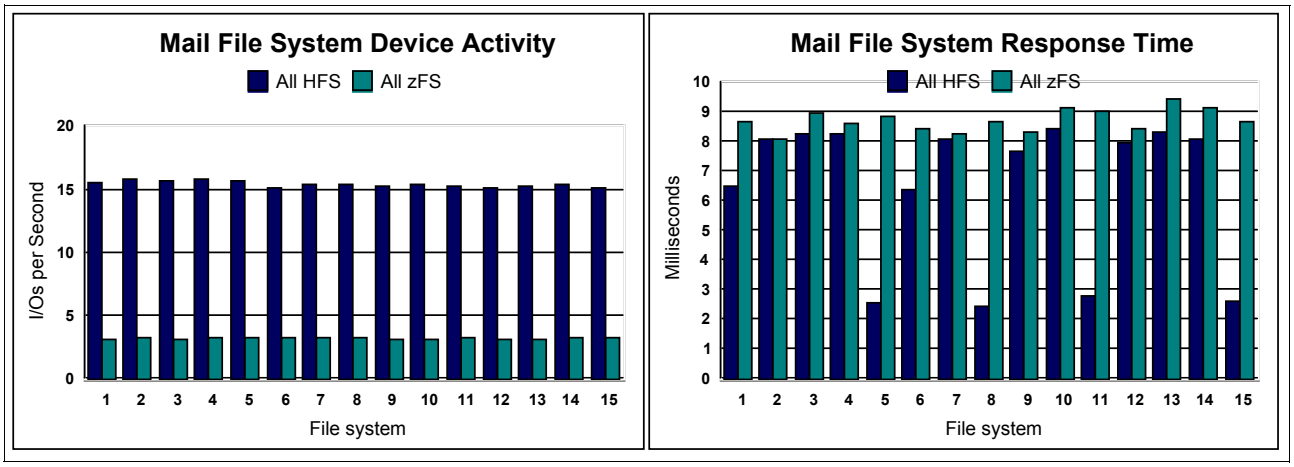


Figure 7-8 Mail file system device activity and response time

The I/O details for the mail file systems, shown in Figure 7-8, demonstrate the differences between the R5Mail and Webmail benchmark workloads in terms of the amount of stress that they drive against the mail file systems. For the Webmail-based workloads, the Domino server spends more time and resource assembling and rendering data than it does actually acquiring that data from the database. This is the fundamental difference between Web-based and Notes client-based access to a database.

As can be seen in Figure 7-8 on page 351, HFS and zFS were not stressed heavily to service I/O requests to the mail file systems for this workload. DASD response times for HFS dropped back down into the acceptable range, and were comparable to those of zFS. What is worth noting is that the DASD activity gap between HFS and zFS grew under this lighter workload. For R5Mail, HFS drove about 1.7 times as much I/O as zFS. For the lighter Webmail workload, this gap was about 4.9.

Where zFS provides its real performance benefit for this workload is in the I/O performed to the Notes data and transaction log file systems, as shown in Figure 7-9.

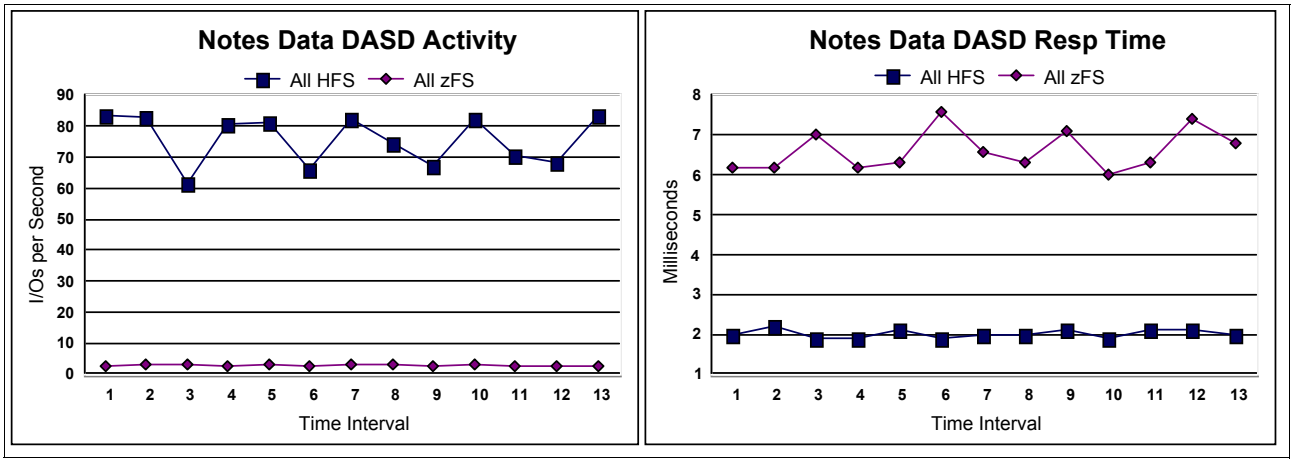


Figure 7-9 I/O characteristics for Notes data and the transaction log

You can see that HFS has to drive much more I/O to the Notes data file system in order to gather the things that it needs to build Web pages from the documents that it has read from the user's mail database. These parts include things like GIF files and HTML templates, and they reside in a subdirectory off the Notes data directory. zFS was able to cache this data, and keep its I/O to DASD low and very flat. The difference in response time between HFS and zFS was not significant.

The difference in traffic to the transaction log also follows the same pattern as for the R5Mail workload, only the absolute amount of I/O required for both file system types is lower than for the R5Mail workload, as shown in Figure 7-10.

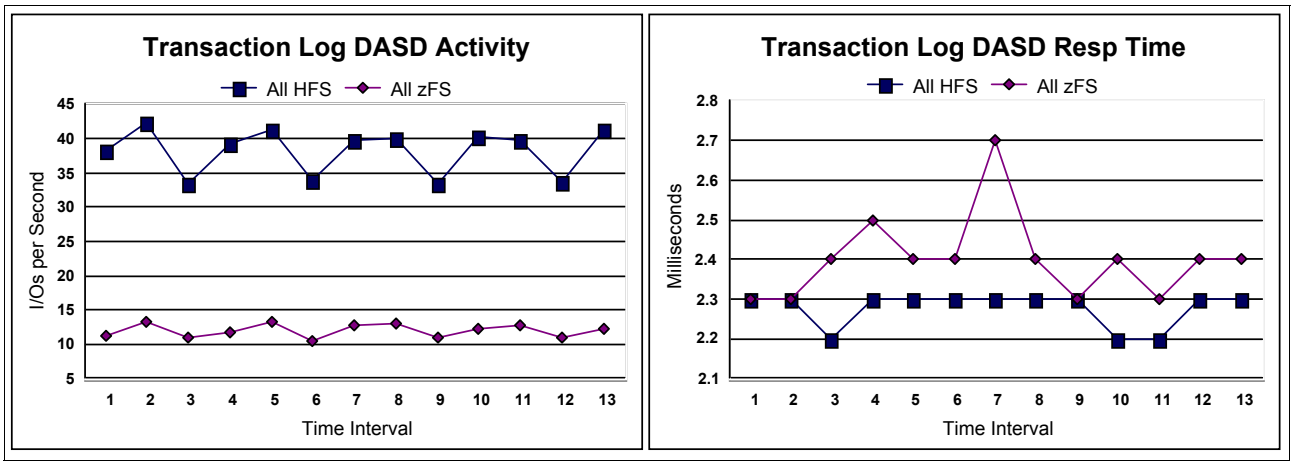


Figure 7-10 Transaction log device activity and response time

What is significant here again is the gap between the I/O performed by HFS and zFS. For the R5Mail workload, where the volume of traffic to the actual mail databases was much higher, the difference in DASD utilization between the two file system types was smaller. As with the other workloads, the higher resource usage of zFS reflects how it is able to apply more available system resource to service the loads required of the file system.

The iNotes workload

This workload was configured to run the same way as the Webmail workload, with two important differences:

- 750 users were simulated instead of 500.
- The size of the message sent was 4,000 bytes instead of 1,000 bytes.

The reason for these differences is to make the CPU and I/O usages higher, as shown in Figure 7-11 on page 354, so that the measurements would show differences more clearly.

Table 7-10 iNotes workload with 750 users

	CPU Usage %	Central Storage (MB)	Expanded Storage (MB)	End-User Response Time (Msec)
All HFS	88.5	1,582	7*	731
All zFS	91.3	2,063	647	709
Difference	+3%	+30%	n/a	-3%

Note: *There is always a small amount of expanded storage in use; 7 MB in use for HFS is effectively zero.

One of the primary benefits of iNotes relative to the “old” Webmail interface tested is that iNotes performs substantially better. You can see from the data in Figure 7-11 on page 354 that an important reason for this is that iNotes requires much less I/O to be performed to accomplish the same amount of work. For this test, even though we simulated 50% more users sending notes that are four times larger, the DASD I/O rates to the mail databases for both HFS and zFS tests were lower for iNotes than for Webmail.

This smaller I/O component within the overall cost to support a given user is the reason why there is so little difference in the CPU usage and response time measurements between the HFS and zFS tests. Enhancing the performance of a small part of the cost yields a small overall cost reduction.

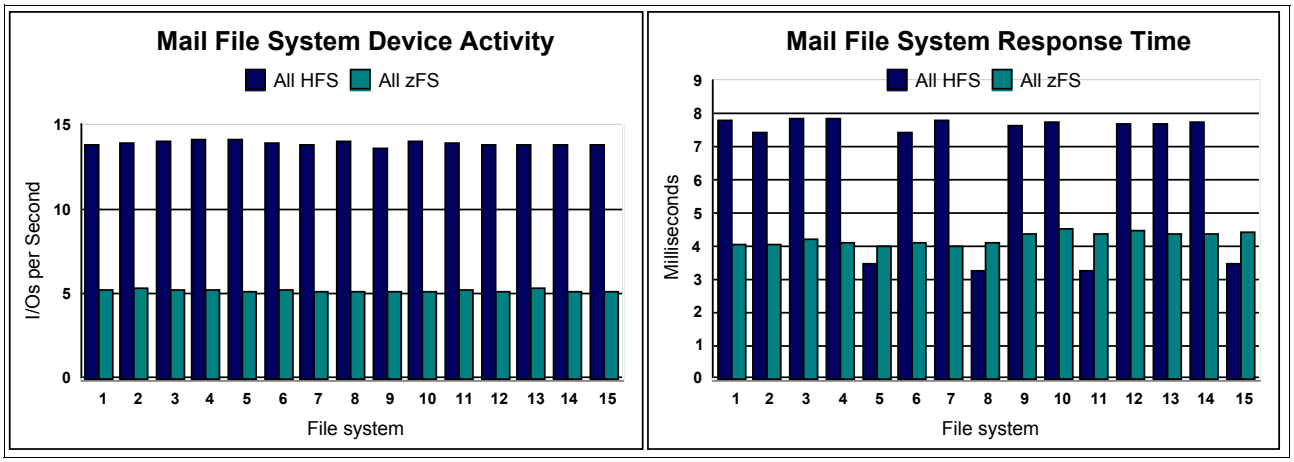


Figure 7-11 I/O usages for the iNotes workload with 750 users

Our efforts to drive the server utilization to higher levels by increasing the parameters of the iNotes workload did cause the Domino server to do substantially more work. Although we saw the I/O to the mail databases above actually drop relative to Webmail, both in terms of the total I/O required and the I/O per simulated user, the heavier workload did manifest itself as higher I/O to the transaction log.

These results, shown in Figure 7-12 and Figure 7-13 on page 355, remain consistent with those of the other workloads. The point here, however, is to understand the components of overall user cost and performance metrics. zFS usage provides more advantage in those environments where local file system I/O is a bigger part of the cost to support a workload.

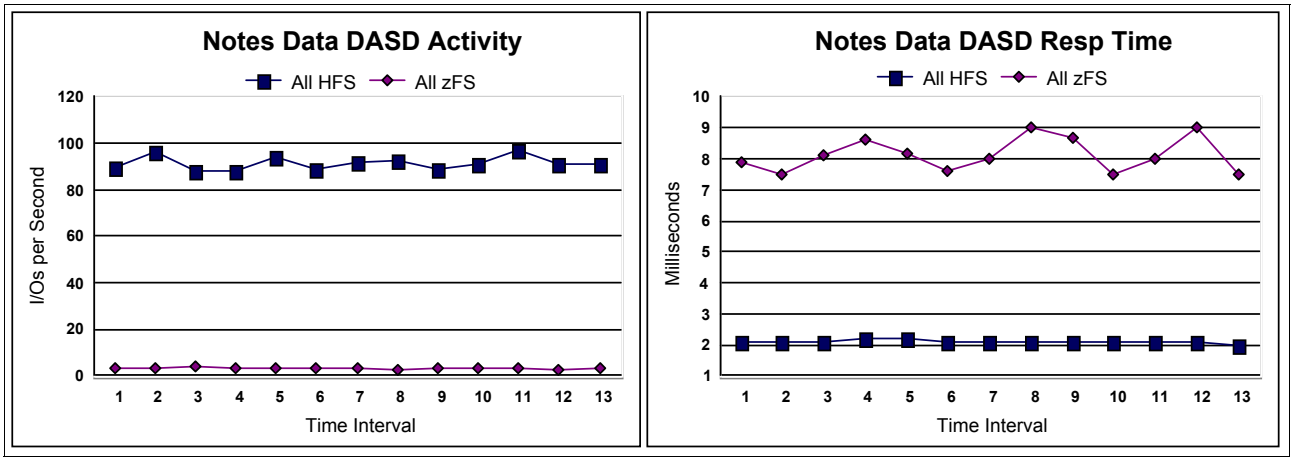


Figure 7-12 I/O usage for Notes data showing device activity and response time

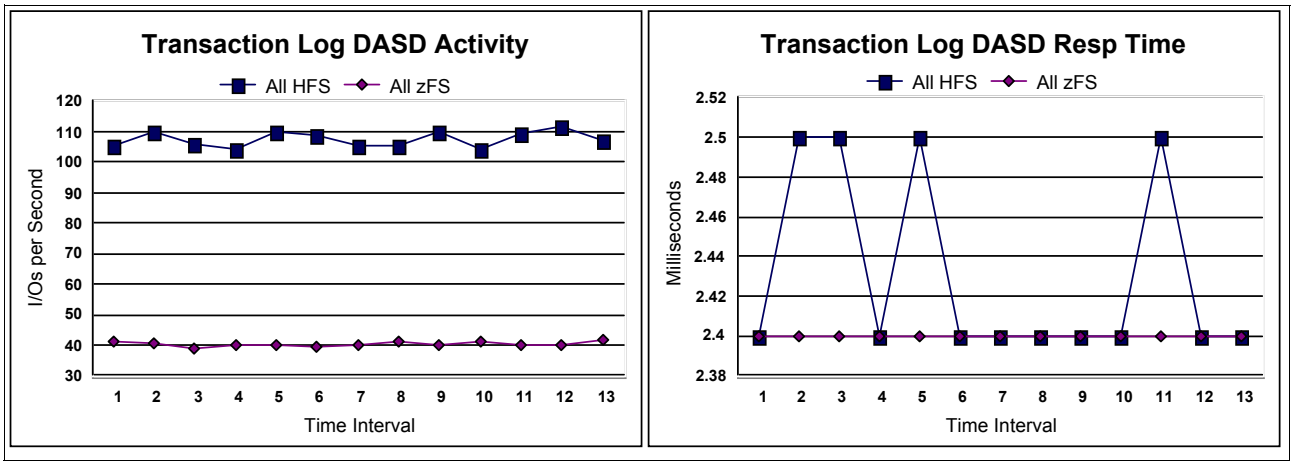


Figure 7-13 I/O usage for the transaction log for device activity and response time

Production Domino applications

We measured two production environments to see what benefits zFS might bring to a Domino application (not mail).

The first application is from a state agency and is used to trace case histories for all types of government business (driver's license applications, taxes, human services, and so on). The application is implemented as a collection of agents that are used to process the data in the database. For the purposes of this test, the client implemented a simple test agent that performed three types of operations:

- Run a query built from twelve separate conditional clauses, and count the number of matching documents.
- Explicitly loop through all documents, looking for those having a field with a specific value, and updating a count field in that document.
- Loop through all case types, and generate a monthly report.

The database containing this agent is 34 MB in size, and contains more than 37,000 documents. The point of this test is simply to measure the elapsed time required for a single thread of execution to complete running the test agent. Measurements were made with the test database in the Notes data directory.

Table 7-11 Elapsed time differences for the five test runs

Test Run	Elapsed Time HFS (seconds)	Elapsed Time zFS (seconds)	% Difference
1	50	19	-62%
2	22	18	-18%
3	22	18	-18%
4	22	18	-18%
5	22	18	-18%

The second application that we evaluated with zFS was from a different state agency that is interested in the performance of multiple Web users simultaneously performing operations on the same set of databases. For this test, all databases were located in a single file system. The tests were driven through LoadRunner, using test scripts created by the client who wrote the application. Table 7-12 shows the results generated by LoadRunner.

Table 7-12 LoadRunner test results

	HFS	zFS	% Difference
Number of simulated users	70	70	0%
Total throughput (MB)	212	239	12.7%
Throughput (bytes/second)	78,776	89,121	13.1%
Total hits	40,172	45,539	13.4%
Hits per second	14.27	16.22	13.7%
Total passed	8,979	10,222	13.8%
Total failed	174	162	-6.9%
Total abort	164	157	-4.3%
Minimum (seconds)	165	160.5	-2.7%
Average (seconds)	262.6	233.4	-11.1%
Maximum (seconds)	579.9	555.3	-4.2%
90% (seconds)	416.4	400.6	-3.8%

As can be seen in Table 7-12, throughput is improved significantly, both in terms of the number of bytes passed between server and browser, and the number of Web hits serviced. This had a positive effect on the number of tests that LoadRunner deemed to be successful, presumably because there were fewer timeouts during the test. Average end-user response time was also substantially reduced.

Although the relatively short duration of this test, and the limited number of simulated users, make drawing any conclusions about contention between the multiple threads somewhat questionable, the higher throughput numbers would seem to indicate that zFS allowed the workload to run with greater freedom, and less waiting for the common resources of the application.

7.2.4 Domino performance conclusions

Domino servers perform better when deployed with zFS instead of HFS in all of the operations that we tested. This manifests itself at the client in terms of higher throughput, and lower elapsed times to perform any client-initiated operation that requires a significant amount of I/O to be driven on the server. This is true for both Notes client access, and access via Web browser. Non-client-driven server operations and housekeeping functions are similarly improved.

The caching capabilities of zFS appear to be the key to providing this performance gain. This is supported by the fact that zFS drives only about one-third of the amount of I/O to DASD that HFS drives when put under an identical load. This reduction in DASD traffic is realized for both highly random and highly sequential I/O patterns. Although we were not able to directly determine the cause for improvement in this test, our end-user throughput measurements indicate that zFS also provides significantly higher throughput rates than HFS.

This performance improvement comes with a higher cost in terms of the amount of storage required for the particular zFS configurations that we tested. It was not a goal of this testing to tune caching values so that the storage required by zFS matched that for HFS when the server was under a given workload. The key observation is that zFS gives the administrator as much or as little storage to apply to the file system as needed to achieve the performance levels required.

It is also important to note that server CPU consumption does not significantly change for a given workload, regardless of file system type. In fact, when converting an existing Domino environment to zFS, it is possible that you could actually see CPU increase marginally, depending on how much the server is currently bottlenecked by HFS. Users who are converting from HFS to zFS in an environment that has high levels of I/O to DASD should monitor CPU consumption and be prepared to tune zFS to keep resource consumption, both CPU and storage, within required operating ranges.

zFS has provided us with many performance benefits in the lab, both for our internal benchmarking efforts and for various client environments that we recreated in the lab. To date, zFS has been deployed internally within IBM for limited production over the last several months, and its performance and stability have been very good. zFS is a viable alternative to HFS that anyone should seriously consider for their Domino environment.

A

zFS configuration file sample

This appendix contains the sample zFS configuration file provided in the hlq.SIOESAMP library.

A.1 IOEFSPRM configuration file

Figure A-1 is the sample IOEFSPRM member provided in IOE.SIOESAMP.

```
************************************************************************
* 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_output_dsn parameter defines the optional output
* message data set. If this parameter is not specified, or if the data
* set is not found, messages will be written to the system log.
* You must delete the * from a line to activate the parameter.
************************************************************************
* msg_output_dsn=data.set.name
************************************************************************
* The following msg_input_dsn parameter is ONLY required if the optional
* NLS feature (e.g JOH232J) 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
*debug_settings_dsn=data.set.name(membername)
*trace_table_size=32M
*storage_details=ON
*storage_details_dsn=data.set.name
```

Figure A-1 Sample IOEFSPRM member

B

REXX utility procedures

This appendix contains all of the REXX procedures created to make it easier to implement zFS-related tasks, either by using the TSO foreground or JCL. Most of the procedures use JCL, but some are also useful from the foreground. A detailed description is also provided explaining how they work and why they were created.

This appendix includes the following REXX utility procedures:

- RXIOE
- RXZFS
- COPYPAX
- ADDMNTPS
- CN
- CNZFS
- SU
- LARGEFIL, LARGEIOS and LARGESCD (for comparison of HFS and zFS)
- UNIX System Services script zfsgrow
- RXLSAGGR
- RXZFSMON
- MOVEAGGR
- RXBATCH
- RXDOWNER and ZFSOWNER
- LARGEFIL, LARGEIOS, and LARGESCD (Version 2 for comparison in UNIX System Services sysplex file system sharing environments)

B.1 Creating the tools

All the tools and REXX procedures were created to simplify performing zFS-related tasks and to make it easy to document and save the information created. These procedures are useful with zFS commands, and they may also be used for different types of commands and situations.

To understand the environment where you need to use these tools, read the following explanation regarding superusers, superuser mode, and authorizations.

Important: The sample REXX procedures are available in softcopy on the Internet from the Redbooks Web server.

Point your browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246580/

Note that the SG part of SG246580 must be upper case.

Alternatively, you can go to:

http://www.redbooks.ibm.com

Select **Redbooks Online**, and then **Additional Materials**.

B.1.1 z/OS UNIX superuser mode

For a user who needs to perform administration tasks in the system, a recommended way to give authorization to use the zFS commands is to use one of the following methods (rather than just assigning the user a UID of 0):

- The UNIXPRIV class profiles

  Using this class can reduce the number of people who have superuser authority at your installation by defining profiles in the UNIXPRIV class that grant RACF authorization for certain z/OS UNIX privileges.

- The BPX.SUPERUSER FACILITY profile

  A user having read access to BPX.SUPERUSER is often just called a BPX.SUPERUSER.

B.1.1.1 Giving superuser authority

If a user needs access to many or almost all superuser functions in z/OS UNIX, making that user a BPX.SUPERUSER is much simpler than providing access to all the different types of profiles in the UNIXPRIV class. However, with BPX.SUPERUSER authority, the user must first switch to superuser mode before using his extended authorization. This is easy when working in a shell interactively because you just use the **su** command to achieve it. But there are several situations where it is difficult or not obvious how to do that.

The following commands support the BPX.SUPERUSER directly by switching to SU mode (superuser mode) automatically:

- TSO MOUNT and UNMOUNT commands
- BPXCOPY

  BPXCOPY copies MVS data to the z/OS UNIX file structure. It is used by SMP/E and allows a BPX.SUPERUSER to perform SMP/E install and service tasks.

Note: Not many functions automatically switch to SU mode. In all other situations it is up to you to perform the switch first.

In the TSO foreground, you are not allowed to switch to a real UID of 0. However, you may switch the real and the effective UID to 0. This is what the ISHELL does when you run the ISHELL command **su** as a BPX.SUPERUSER. This gives you the same possibilities.

You can execute z/OS UNIX commands in a shell environment or a BPXBATCH job and set the real UID to 0. You provide the input data through STDIN and receive the output data in the STDOUT and STDERR output data files. Note that you may have to add additional job steps to create the STDIN input data file and to access the STDOUT and STDERR output files.

B.1.2 Benefits of using the tools

The tools or REXX procedures described in this appendix were created to make it easier to use the zFS commands when a user has to be authorized to use the z/OS UNIX-authorized commands. In summary, these REXX procedures do the following:

- Exploit access to BPX.SUPERUSER when needed by:
  - Switching the effective UID to 0 in TSO foreground when starting.
  - Switching back the effective UID to the original value in TSO foreground when ending.
  - Switching the real UID to 0 in TSO batch jobs.
- Avoid additional job steps to provide input and obtain z/OS UNIX output data.
- Avoid using temporary files in the z/OS UNIX file structure. If you use BPXBATCH to submit jobs, temporary files in the z/OS UNIX file structure are necessary and are sometimes accidentally kept afterwards if something goes wrong during processing.

Note: In addition to explaining functionality, the usage information for each REXX procedure describes additional benefits that are available with the procedure.

B.2 RXIOE

REXX procedure RXIOE allows a user to execute the zFS **ioezadm** command in TSO batch jobs by switching to superuser mode by setting the UID of the submitting user to 0, and then executing the command.

Note: RXIOE is not supported in the TSO foreground. If you are a BPX.SUPERUSER and want to run a **ioezadm** command in SU mode from TSO, you can use the REXX utility SU. It is described and listed in B.8, “SU procedure” on page 397.

B.2.1 Using RXIOE

In addition to the benefits mentioned in B.1.2, “Benefits of using the tools” on page 363, RXIOE provides the following functions and it works according to the following rules here:

- Command lines ending with a dash (-) may be continued on the next line.
- Leading blanks of a continuation line are removed.
- Commands ending with a plus sign (+) are not displayed within the command output data.
- Lines starting with a pound sign (#) in column 1 are treated as comments.

- Blank lines are displayed as blank lines in the output.
- Text entered in lines following a forward slashmark plus asterisk (/*) is displayed in the output starting with the first non-blank character. This allows easy comments in output data.
- All the command output data is available with DD name SYSPRINT.

The use of this REXX procedure is shown in the JCL samples in C.2.1, “Display information for zFS aggregates” on page 452.

B.2.1.1 IOEZADM utility program

When using **ioezadm** as a command in a REXX procedure, the output messages cannot be trapped by the REXX procedure. The command output is sent to the SYSPRINT DD and not the standard data set for TSO, SYSTSPRT. Only TSO input command errors are found with DD name SYSTSPRT. So this output is normally empty except for the READY and END lines.

B.2.2 RXIOE code

Example B-1 displays the contents of RXIOE.

Example: B-1 RXIOE REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: RXIOE                                                */
/*  Description: Switch to SU mode if possible and run IOEZADM cmds   */
/*               Property of IBM  (C) Copyright IBM Corp. 2002        */
/*    Format is: rxioe                                                */
/*********************************************************************/

Trace O
Parse Source . . myname .
final_rc = 0
no_msgs = 1
noexit_on_error = 1
If Sysvar("SYSENV")="FORE" Then Do
  Say "RXIOE001E is only supported to be used within TSO batch jobs."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "RXIOE002E The SYSCALL environment could not be established."
  Exit 8
End

"EXECIO * DISKR SYSIN (STEM INPUTD. FINIS"
If rc<>0 Then Exit 8

/* ----------------------------------------------------------------- */
/*  Switch UID to zero if possible                                   */
/* ----------------------------------------------------------------- */

Call Syscall_Cmd "getuid"
cur_uid = retval
If cur_uid<>0 Then
  Call Syscall_Cmd "setuid 0", noexit_on_error, no_msgs
```

```
/* ---------------------------------------------------------------- */
/*  Run the TSO commands                                            */
/* ---------------------------------------------------------------- */

tso_cmd = ""
Do i=1 To inputd.0
  indata = inputd.i
  If Left(indata,1)="#" Then Iterate
  indata = Strip(indata)
  If Right(indata,1)="-" Then Do
    no_cont = 0
    indata = Left(indata,Length(indata)-1)
  End
  Else no_cont = 1
  tso_cmd = tso_cmd||indata
  If no_cont Then Do
    If Right(tso_cmd,1)<>"+" Then display_cmd = 1
    Else Do
      tso_cmd = Left(tso_cmd,Length(tso_cmd)-1)
      display_cmd = 0
    End
    If Left(tso_cmd,2)="/*" | tso_cmd="" Then
      Call Send2sysp Strip(Substr(tso_cmd,3))
    Else Do
      If display_cmd Then Call Send2sysp tso_cmd
      tso.0 = 0
      Call Outtrap "TSO."
      Address TSO tso_cmd
      retc=rc
      Call Outtrap "OFF"
      Do j=1 To tso.0
        Call Send2sysp tso.j
      End
      If retc<>0 Then Do
        Call Send2sysp "*** non-zero return code: Rc("||retc||")"
        final_rc = 8
      End
    End
    tso_cmd = ""
  End
End
Exit final_rc

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
```

```
    Call Send2sysp "SYSCALL Service:" syscall_cmd
    Call Send2sysp "Syscall Return Code=" rc
    Call Send2sysp "OMVS Return Value =" retval
    Call Send2sysp "OMVS Return Code =" errno
    Call Send2sysp "OMVS Reason Code =" errnojr
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      Call Send2sysp ""
      Call Send2sysp "OMVS Return Code Explanation -"
      Call Send2sysp err.se_errno
      Call Send2sysp "OMVS Reason Code Explanation -"
      Call Send2sysp err.se_reason
      Call Send2sysp ""
    End
    If exit_on_error Then Exit 8
  End
Return

Send2sysp:
  Parse Arg sysprint.1
  "EXECIO 1 DISKW SYSPRINT (STEM SYSPRINT."
  If rc<>0 Then Exit 8
Return
```

B.3 RXZFS

REXX procedure RXZFS allows the user to execute z/OS UNIX and zFS-related **zfsadm** commands in TSO batch jobs, or interactively in the TSO foreground.

B.3.1 Usage rules and benefits of RXZFS

In addition to the benefits mentioned in B.1.2, “Benefits of using the tools” on page 363, RXZFS provides the following functions and uses the following rules:

- In a batch job, environment variables can be set to be used in the commands.
- Command lines ending with a dash (-) may be continued on the next line.
- Leading blanks of a continuation line are removed.
- Commands ending with a plus sign (+) are not displayed in the output data.
- Lines starting with a pound sign (#) in column 1 are treated as comments.
- RXZFS allows the user to specify a timeout value to automatically cancel a command if it does not end within the specified amount of time. A value of 0 indicates no timeout setting is used.
- RXZFS allows a “one-line” command call if the optional timeout value and the command are specified with RXZFS.
- The commands are run in a z/OS UNIX shell environment as follows:
  - With no expensive shell initialization (simply **sh,** not **sh -L**)
  - Using the root directory (/) as the home and working directory

The use of this REXX procedure is illustrated in the JCL samples provided in Appendix C, “JCL samples” on page 451. Also read the header section of the procedure to obtain more information about specifying timeout values.

The procedure now has the identical functions as REXX procedure RXSUSH, which is available with the USSTools package. Refer to “SU procedure” on page 397 for more information about this package.

Attention: Envvar settings provided in the JCL replace the default settings. If you define the PATH environment variable, remember to add `/bin`. Otherwise, the programs in /bin cannot be used by their names.

There two additional “pseudo” environment variables for controlling the behavior of RXZFS.

- `_PROCESS_INFO_MSGS=0`

  In batch processing, this avoids displaying the messages seen otherwise about the main process ID that can used to smoothly stop the job. Any value other than zero (0) will keep showing the messages.

- `_EXIT_ON_FIRST_ERROR=1`

  This forces to stop the job as soon as the first command has an return code not equal to zero. The default behavior is to run all commands in the list and provide RC=8 at end of job to indicate that a command was not successful. Any value other than 1 results in the default behavior again.

B.3.2 RXZFS code

Example B-2 displays the contents of RXZFS.

Example: B-2 RXZFS REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: RXZFS                                                */
/*  Description: Run UNIX commands/run UNIX commands in SU mode        */
/*               Property of IBM  (C) Copyright IBM Corp. 2003-2008    */
/*               Robert Hering (robert.hering@de.ibm.com)              */
/*    Format is: rxzfs <<R>timeout_value> <single_cmd>                 */
/*               Timeout values may be specified as follows.           */
/*               R0 - No automatic timeout, output messages are shown */
/*                    as they appear. This is the default.             */
/*               0  - No automatic timeout, output messages are shown */
/*                    when single command is finished or the process   */
/*                    is cleanly stopped.                              */
/*               nn - Single command will be canceled after nn secs    */
/*                    if not yet finished.                             */
/*                                                                     */
/*********************************************************************/

Trace O
Parse Source . calltype myname .
swsu = 1 /* forces SU switching */
Parse Arg timeout_value single_cmd
do_wait = 1
not_do_wait = 0
exit_on_first_error = 0
```

```
process_info_msgs = 1 /* display information on main process in batch */
If timeout_value="" Then timeout_value = "R0"
If Translate(Left(timeout_value,1))="R" Then Do
  do_wait = 0
  not_do_wait = 1
  timeout_value = Substr(timeout_value,2)
End
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  do_wait = 0
  not_do_wait = 1
  timeout_value = 0
  Parse Arg single_cmd
End

If Syscalls("ON")>4 Then Do
  Say "RXSHX002E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "RXSHX003E The SIGNAL interface could not be established."
  Exit 8
End

switched = 0
final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ------------------------------------------------------------------ */
/*  Switch effective and/or real UID to zero if needed and possible   */
/* ------------------------------------------------------------------ */

If swsu Then Do
  If background Then Do
    Call Syscall_Cmd "getuid"
    cur_uid = retval
    If cur_uid<>0 Then
      Call Syscall_Cmd "setuid 0", noexit_on_error, no_msgs
  End
  Else Do /* foreground */
    Call Syscall_Cmd "geteuid"
    cur_euid = retval
    Call Syscall_Cmd "getuid"
    cur_uid = retval
    If MVSVar("SYSMVS")>="SP7.0.3" Then Do
      If cur_euid<>0 | cur_uid<>0 Then Do
        Call Syscall_Cmd "setreuid 0 0", noexit_on_error, no_msgs
```

```
        If OK Then Do
          Say "UID setting switched to 0..."
          switched = 1
        End
      End
    End
    Else Do
      If cur_euid<>0 Then Do
        Call Syscall_Cmd "seteuid 0", noexit_on_error, no_msgs
        If OK Then Do
          Say "Effective UID switched to 0..."
          switched = 1
        End
      End
    End
  End
End

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars, open /dev/null   */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "getpid"
mainprocess_pid = retval
If swsu Then Do
  Call Syscall_Cmd "getcwd cur_cwd"
  home = "/"
  Call Syscall_Cmd "chdir" home
End
Else Do
  Call Syscall_Cmd "getpwnam" Userid() "omvs."
  home = omvs.pw_dir
End
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
envvars = 5
If background Then Do
  "EXECIO * DISKR STDENV (STEM ENVVAR. FINIS"
  If rc<>0 Then Call Final_Exit 8
  Do i=1 To envvar.0
    envvar_line = Strip(envvar.i)
    If envvar_line="" | Left(envvar_line,1)="#" Then Iterate
    pos_equal = Pos("=",envvar_line)
    If pos_equal=0 Then Iterate
    If Left(envvar_line,pos_equal-1)="_EXIT_ON_FIRST_ERROR" Then Do
      If Substr(envvar_line,pos_equal+1)="1" Then
        exit_on_first_error = 1
      Else
        exit_on_first_error = 0
      Iterate i
    End
    If Left(envvar_line,pos_equal-1)="_PROCESS_INFO_MSGS" Then Do
```

```
      If Substr(envvar_line,pos_equal+1)="0" Then
        process_info_msgs = 0
      Else
        process_info_msgs = 1
      Iterate i
    End
    envvars = envvars+1
    __environment.envvars = envvar_line
  End
End
__environment.0 = envvars
Call Syscall_Cmd "open /dev/null (o_rdonly)"
fd.0 = retval
If background & calltype="COMMAND" & process_info_msgs Then Do
  dash_line = Copies("----",19)
  Say dash_line
  Say "The main process ID for this job is" mainprocess_pid||".",
    "If you should need to stop"
  If do_wait Then signal_type = "SIGALRM"
  Else signal_type = "SIGTERM"
  Say "processing use the following UNIX command to do this smoothly:"
  Say "kill -s" signal_type mainprocess_pid
  Say dash_line
  Say ""
End

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"
If not_do_wait Then
  Call Syscall_Cmd "sigaction" sigterm sig_cat 0 "xhdl xflg"

/* ---------------------------------------------------------------- */
/*  Read in all the commands                                        */
/* ---------------------------------------------------------------- */

If single_cmd="" Then Do
  If background Then Do
    "EXECIO * DISKR STDIN (STEM INPUTD. FINIS"
    If rc<>0 Then Call Final_Exit 8
    in_lines = inputd.0
  End
  Else in_lines = 0
End
Else Do
  inputd.1 = single_cmd
  in_lines = 1
End

/* ---------------------------------------------------------------- */
```

```
/*  Run all the commands                                              */
/* ------------------------------------------------------------------ */

stdin_line = ""
ln = 0
Do Forever
  If in_lines>0 | background Then Do
    If ln=in_lines Then Leave
    ln = ln+1
    indata = inputd.ln
  End
  Else Do
    Say "Enter command input data or ""exit"" to exit processing:"
    Parse Pull indata
    If indata="exit" Then Leave
  End
  If Left(indata,1)="#" Then Iterate
  indata = Strip(indata)
  If Right(indata,1)="-" Then Do
    no_cont = 0
    indata = Left(indata,Length(indata)-1)
  End
  Else no_cont = 1
  stdin_line = stdin_line || indata
  If no_cont Then Do
    If Right(stdin_line,1)<>"+" Then Say stdin_line
    Else stdin_line = Left(stdin_line,Length(stdin_line)-1)
    If stdin_line="exit" Then Leave
    Call Shell_Cmd stdin_line
    stdin_line = ""
  End
End

/* ------------------------------------------------------------------ */
/*  End of processing                                                 */
/* ------------------------------------------------------------------ */

Call Final_Exit final_rc
Exit 9999

/* ------------------------------------------------------------------ */
/*  Subroutines                                                       */
/* ------------------------------------------------------------------ */

Final_Exit:
  Parse Arg final_rc
  If swsu Then Call Syscall_Cmd "chdir" cur_cwd
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.i)", noexit_on_error
  End
  If pp.1<>-1 Then Call Syscall_Cmd "close (pp.1)", noexit_on_error
```

```
  If swsu & switched & foreground Then Do
    If MVSVar("SYSMVS")>="SP7.0.3" Then Do
      Call Syscall_Cmd "setreuid" cur_uid cur_euid, noexit_on_error
      If not_OK Then Say "UID settings could not be switched back..."
    End
    Else Do
      Call Syscall_Cmd "seteuid" cur_euid, noexit_on_error
      If not_OK Then Say "Effective UID could not be switched back..."
    End
  End
Exit final_rc

Shell_Cmd:
  Parse Arg shell_cmd
  parm.0 = 3
  parm.1 = "sh"
  parm.2 = "-c"
  parm.3 = shell_cmd
  shell_rc = -1
  shell_termsig = -1
  shell_stopsig = -1
  time_out = 0
  Call Syscall_Cmd "pipe pp."
  fd.1 = pp.2
  If not_do_wait Then Call Syscall_Cmd "f_setfl (pp.1) (o_nonblock)"
  Call Syscall_Cmd "dup (fd.1)"
  fd.2 = retval
  Call Syscall_Cmd "spawnp (parm.1) 3 fd. parm. __environment."
  pid = retval
  all_output = ""
  If do_wait Then Do
    Call Syscall_Cmd "alarm" timeout_value /* timeout setting in secs */
    Call Syscall_Cmd "waitpid (pid) stat. 0", info_on_timeout
  End
  Else Do
    time_beg = Time("E")
    Do Forever
      Call Syscall_Cmd "waitpid (pid) stat. (w_nohang)"
      If timeout_value<>0 & Time("E")-time_beg>=timeout_value Then Do
        time_out = 1
        Leave
      End
      If stat.w_ifexited Then Leave
      Call Get_Output_Data
      Call Syscall_Cmd "sigpending sigset"
      If Substr(sigset,sigterm,1)=1 Then Do
        Say "RXSHX006I Signal SIGTERM received, terminating..."
        Say "RXSHX007I Command:" shell_cmd
        If background Then Call Final_Exit 8
        Else Leave
      End
      Call Syscall_Cmd "sleep 1"
    End
  End
  Call Syscall_Cmd "alarm 0"
```

```
  If time_out Then Do
    Call Syscall_Cmd "kill (pid)" sigkill, noexit_on_error
    Call Syscall_Cmd "waitpid (pid) stat. 0"
  End
  Select
    When stat.w_ifexited Then shell_rc = stat.w_exitstatus
    When stat.w_ifsignaled Then shell_termsig = stat.w_termsig
    When stat.w_ifstopped Then shell_stopsig = stat.w_stopsig
    Otherwise Nop
  End
  Do j=1 To 2
    If fd.j<>-1 Then Call Syscall_Cmd "close (fd.j)"
    fd.j = -1
  End
  Do Until retval<1000
    Call Syscall_Cmd "read (pp.1) output_data 1000"
    all_output = all_output||output_data
  End
  Do While Length(all_output)>0
    Parse Var all_output output_line (esc_n) all_output
    Say output_line
  End
  Call Syscall_Cmd "close (pp.1)"
  pp.1 = -1
  If time_out Then Do
    Say "RXSHX004I Signal SIGALRM received or timeout occurred,",
      "terminating..."
    Say "RXSHX005I Command:" shell_cmd
    If background Then Call Final_Exit 8
  End
  If shell_rc<>0 Then Do
    Say "*** non-zero return code: Rc("||shell_rc||")."
    final_rc = 8
    If exit_on_first_error Then Do
      Say "RXSHX008I Stopping on first error as requested..."
      Call Final_Exit 8
    End
  End
Return

Get_Output_Data:
  Do Until retval<1000
    output_data = ""
    Call Syscall_Cmd "read (pp.1) output_data 1000"
    all_output = all_output||output_data
  End
  Do While Length(all_output)>0 & Pos(esc_n,all_output)>0
    Parse Var all_output output_line (esc_n) all_output
    Say output_line
  End
Return

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
```

```
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  If rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then Do
    not_OK = 0
    retval = 0
  End
  Else not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
      errno_save = errno
      errnojr_save = errnojr
      If errno<>"A3" & errno<>"A4" & is_omvs_range Then
        show_reason = 1
      Else
        show_reason = 0
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        If err.se_errno<>"" Then
          Say "OMVS Return Code Explanation -" err.se_errno
        If show_reason & err.se_reason<>"" Then
          Say "OMVS Reason Code Explanation -" err.se_reason
      End
      errno = errno_save
      errnojr = errnojr_save
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.4 COPYPAX

REXX procedure COPYPAX allows the user to use the **pax** command in copy mode. It can be used from the TSO foreground but should be used in a TSO batch job.

B.4.1 Benefits of COPYPAX

The **pax** utility had two unusual problems:

- During a copy or archive process all external links were destroyed. They were just replaced by invalid symbolic links. This problem has been solved by PTF UW51713 of FMID HOT1180.
- The second problem still exists. The **pax** utility does not correctly handle file descriptor files with a minor device setting above 255. This is not a severe problem, but it is nevertheless still an error. The COPYPAX procedure corrects all invalid file descriptors that are found.

COPYPAX uses the **pax** command in copy mode to copy a source to a target structure with the following functions and additional benefits:

- A test is done to verify that the source and the target structure are located in different file systems. This avoids problems that may occur if the target structure is located within the source structure because **pax** could get into a never-ending loop.
- If desired, a test is done to verify that the target directory is empty. However, you can also do copying even if the target structure is not empty to allow merging of file systems.
- The **pax** option `-X` is used to avoid skipping over mount point boundaries.
- If desired, invalid file descriptor files are corrected when **pax** has finished its copy processing. You should skip this step if you know that the file system does not contain any character special files. (Normally, only directory `/dev` contains such entries.)
- You can get clear advice in the job log (just when the job starts) about how to stop processing, if this should ever be needed. Therefore, it is suggested to run the job with no timeout being specified.
- You can switch on verbose "pax" output listing all entries copied or created. This includes the definition of missing mount points.

B.4.2 COPYPAX code

Example B-3 displays the contents of COPYPAX.

Example: B-3 COPYPAX REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: COPYPAX - Version 2.3                                */
/*  Description: COPY (clone) a file system structure using "pax"     */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: copypax <time_out> source_dir target_dir             */
/*********************************************************************/

Trace O
Parse Source . . myname .
no_args = Words(Arg(1))
Select
  When no_args=2 Then Do
    Parse Arg source_dir target_dir .
    timeout_value = "R0"
  End
  When no_args=3 Then Parse Arg timeout_value source_dir target_dir .
  Otherwise Do
    Say "CPPAX001E Syntax:" myname "< time_out > source_dir target_dir"
    Exit 8
```

```
  End
End /* Select */

If timeout_value="" Then timeout_value = "R0"
If Translate(Left(timeout_value,1))="R" Then
  timeout_value = Substr(timeout_value,2)
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "CPPAX002E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "CPPAX003E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "CPPAX004E The SIGNAL interface could not be established."
  Exit 8
End

swsu = 1        /* always try to switch to superuser mode          */
do_wait = 0     /* generally use the new interface, independent of */
not_do_wait = 1 /* specifying Rxxxx or xxxx as timeout value       */
sig_enabled = 0
switched = 0
final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
Signal On Novalue Name Novalue_Error

/* ---------------------------------------------------------------- */
/*  Read and process COPYPAX parameters from CPPXPARM DD name       */
/* ---------------------------------------------------------------- */

Call Bpxwdyn "INFO DD(CPPXPARM)"
If Pos(Left(Right(D2x(result),8,"0"),4),"0438 0440")<>0 Then Do
  /* IKJ56247I FILE INFO-RETRIEVAL NOT PERFORMED, NOT ALLOCATED */
  cppxvar.0 = 0
End
Else Do
  "EXECIO * DISKR CPPXPARM (STEM CPPXVAR. FINIS"
  If rc<>0 Then Do
    Say  "CPPAX005E Error on reading COPYPAX variables from CPPXPARM",
      "DD name, Rc="||rc
    Call Final_Exit 8
  End
End
```

```
   process_info_msgs = 1 /* display batch main process information    */
target_must_be_empty = 1 /* Target structure must be empty            */
correct_inv_fd_files = 1 /* Correct invalid fd files                  */
    copy_pax_verbose = 0 /* List all objects copied                   */
   no_file_overwrite = 0 /* Prevent overwriting of existing files     */

invalid_cppxvar = 0
Do i=1 To cppxvar.0
  cppxvar_line = Translate(Strip(cppxvar.i))
  If cppxvar_line="" | Left(cppxvar_line,1)="#" Then Iterate i
  pos_equal = Pos("=",cppxvar_line)
  If pos_equal=0 Then Iterate i
  Select
    When Left(cppxvar_line,pos_equal-1)="PROCESS_INFO_MSGS" Then Do
      Parse Value Substr(cppxvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test="Y" Then process_info_msgs = 1
        When opt_test="N" Then process_info_msgs = 0
        Otherwise Do
          Say "The value specified for PROCESS_INFO_MSGS is invalid:"
          Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
          invalid_cppxvar = 1
        End
      End
    End
    When Left(cppxvar_line,pos_equal-1)="TARGET_MUST_BE_EMPTY" Then Do
      Parse Value Substr(cppxvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test="Y" Then target_must_be_empty = 1
        When opt_test="N" Then target_must_be_empty = 0
        Otherwise Do
          Say "The value specified for TARGET_MUST_BE_EMPTY is invalid."
          Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
          invalid_cppxvar = 1
        End
      End
    End
    When Left(cppxvar_line,pos_equal-1)="CORRECT_INV_FD_FILES" Then Do
      Parse Value Substr(cppxvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test="Y" Then correct_inv_fd_files = 1
        When opt_test="N" Then correct_inv_fd_files = 0
        Otherwise Do
          Say "The value specified for CORRECT_INV_FD_FILES is invalid."
          Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
          invalid_cppxvar = 1
        End
      End
    End
    When Left(cppxvar_line,pos_equal-1)="COPY_PAX_VERBOSE" Then Do
      Parse Value Substr(cppxvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test="Y" Then copy_pax_verbose = 1
        When opt_test="N" Then copy_pax_verbose = 0
        Otherwise Do
```

```
          Say "The value specified for COPY_PAX_VERBOSE is invalid."
          Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
          invalid_cppxvar = 1
        End
      End
    End
    When Left(cppxvar_line,pos_equal-1)="NO_FILE_OVERWRITE" Then Do
      Parse Value Substr(cppxvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test="Y" Then no_file_overwrite = 1
        When opt_test="N" Then no_file_overwrite = 0
        Otherwise Do
          Say "The value specified for COPY_PAX_VERBOSE is invalid."
          Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
          invalid_cppxvar = 1
        End
      End
    End
    Otherwise Do
      Say "An unknown COPYPAX parameter has been specified."
      Say "CPPXPARM line" Right(i,2)||":" cppxvar.i
      invalid_cppxvar = 1
    End
  End /* Select */
End i
If invalid_cppxvar Then Do
  Say "CPPAX006E An invalid COPYPAX parameter has been found in the",
    "CPPXPARM data specification."
  Call Final_Exit 8
End

/* ---------------------------------------------------------------- */
/*  Start processing                                                */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "getpid"
mainprocess_pid = retval
If background & process_info_msgs Then Do
  dash_line = Copies("----",19)
  Say dash_line
  Say "The main process ID for this job is" mainprocess_pid||".",
    "If you should need to stop"
  If do_wait Then signal_cmd = "kill -s ALRM"
  Else signal_cmd = "kill" /* "-s TERM" is the default */
  Say "processing use the following UNIX command in authorized mode to",
    "do this:"
  Say signal_cmd mainprocess_pid
  Say dash_line
  Say ""
End

/* ---------------------------------------------------------------- */
/*  Switch effective and real UID to zero if needed and possible    */
/* ---------------------------------------------------------------- */
```

```
If swsu Then Do
  If background Then Do
    Call Syscall_Cmd "getuid"
    cur_uid = retval
    If cur_uid<>0 Then
      Call Syscall_Cmd "setuid 0", noexit_on_error, no_msgs
  End
  Else Do /* foreground */
    Call Syscall_Cmd "geteuid"
    cur_euid = retval
    Call Syscall_Cmd "getuid"
    cur_uid = retval
    If cur_euid<>0 | cur_uid<>0 Then Do
      Call Syscall_Cmd "setreuid 0 0", noexit_on_error, no_msgs
      If OK Then Do
        Say "UID setting switched to 0..."
        switched = 1
      End
    End
  End
End

/* ------------------------------------------------------------------ */
/*  Test whether source and target structure are in different devices */
/* ------------------------------------------------------------------ */
Say Time()||": Verifying existence of source directory" source_dir||,
  "..."
Call Syscall_Cmd "stat (source_dir) sst."
If sst.st_type<>s_isdir Then Do
  Say "CPPAX007E The source specification does not resolve to be a",
    "directory."
  Call Final_Exit 8
End
Say Time()||": Verifying existence of target directory" target_dir||,
  "..."
Call Syscall_Cmd "stat (target_dir) tst."
If tst.st_type<>s_isdir Then Do
  Say "CPPAX008E The target specification does not resolve to be a",
    "directory."
  Call Final_Exit 8
End
If sst.st_dev=tst.st_dev Then Do
  Say "CPPAX009E Source and target structure cannot be located in the",
    "same device."
  Call Final_Exit 8
End

/* ------------------------------------------------------------------ */
/*  Test whether target_dir is empty if requested                     */
/* ------------------------------------------------------------------ */
If target_must_be_empty Then Do
  target_not_empty = 0
  Call Syscall_Cmd "opendir (target_dir)"
  targ_fd = retval
  Call Syscall_Cmd "rddir (targ_fd) uss_data 4096"
```

```
  trg_entries = retval
  Select
    When trg_entries>3 Then target_not_empty = 1
    When trg_entries=3 Then Do
      target_not_empty = 1
      Do i=1 To 3
        len_entry = C2d(Left(uss_data,2))
        uss_data = Substr(uss_data,len_entry+1)
        If len_entry=4 Then target_not_empty = 0
      End
    End
    Otherwise Nop
  End
  Call Syscall_Cmd "closedir (targ_fd)"
  If target_not_empty Then Do
    Say "CPPAXO1OE The target directory structure is not empty, but",
      "this is requested."
    Call Final_Exit 8
  End
End

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars, open /dev/null   */
/* ---------------------------------------------------------------- */

If swsu Then Do
  Call Syscall_Cmd "getcwd cur_cwd"
  home = "/"
  Call Syscall_Cmd "chdir" home
End
Else Do
  Call Syscall_Cmd "getpwnam" Userid() "omvs."
  home = omvs.pw_dir
End
Call Syscall_Cmd "chdir" source_dir
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
envvars = 5
Call Bpxwdyn "INFO DD(STDENV)"
If Pos(Left(Right(D2x(result),8,"0"),4),"0438 0440")<>0 Then Do
  /* IKJ56247I FILE INFO-RETRIEVAL NOT PERFORMED, NOT ALLOCATED */
  envvar.0 = 0
End
Else Do
  "EXECIO * DISKR STDENV (STEM ENVVAR. FINIS"
  If rc<>0 Then Do
    Say  "CPPAXO11E Error on reading environment variables from STDENV",
      "DD name, Rc="||rc
    Call Final_Exit 8
  End
End
Do i=1 To envvar.0
```

```
  envvar_line = Strip(envvar.i)
  envvars = envvars+1
  __environment.envvars = envvar_line
End
__environment.0 = envvars
Call Syscall_Cmd "open /dev/null (o_rdonly)"
fd.0 = retval

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"
If not_do_wait Then
  Call Syscall_Cmd "sigaction" sigterm sig_cat 0 "xhdl xflg"

/* ---------------------------------------------------------------- */
/*  Running pax in copy mode                                        */
/* ---------------------------------------------------------------- */

copy_msg = "Copying the source to the target structure using pax"
pax_cmd_start = "pax -rw"
If no_file_overwrite Then Do
  copy_msg = copy_msg "and do not overwrite existing files"
  pax_cmd_start = pax_cmd_start||"k"
End
If copy_pax_verbose Then Do
  copy_msg = copy_msg "and listing all entries copied..."
  pax_cmd_start = pax_cmd_start||"v"
End
Else copy_msg = copy_msg||"..."
Say Time()||":" copy_msg
Call Shell_Cmd pax_cmd_start "-peW -XCM ." target_dir
If shell_rc<>0 Then Do
  Say "CPPAX012E The pax command ended with return code",
    "Rc("||shell_rc||")."
  Call Final_Exit 8
End

/* ---------------------------------------------------------------- */
/*  Running find command for csfs and correct fdfs if needed        */
/* ---------------------------------------------------------------- */

If correct_inv_fd_files Then Do
  Say Time()||": Searching for and correcting invalid target file",
    "descriptor entries..."
  fd_files = 0
  Call Shell_Cmd "find . -type c -xdev"
  fdfile.0 = fd_files

  If shell_rc<>0 Then Do
    Say "CPPAX013E The find command ended with return code",
```

```
      "Rc("||shell_rc||")."
    Say "Begin of output provided by find command..."
    Do i=1 To fdfile.0
      Say fdfile.i
    End
    Say "End of output provided by find command..."
    Call Final_Exit 8
  End

  corrected = 0
  Do i=1 To fdfile.0
    source_csf = fdfile.i
    Call Syscall_Cmd "stat (source_csf) sst."
    If sst.st_major=5 & sst.st_minor>255 Then Do
      target_csf = Strip(target_dir,"T","/")||Substr(source_csf,2)
      Call Syscall_Cmd "stat (target_csf) tst."
      If tst.st_minor<>sst.st_minor Then Do
        Call Syscall_Cmd "unlink (target_csf)"
        Call Syscall_Cmd "mknod (target_csf)" sst.st_mode 5 sst.st_minor
        corrected = corrected+1
      End
    End
  End
  If corrected<>0 Then
    Say Time()||": Corrected" corrected "file descriptor entries..."
End

/* ------------------------------------------------------------------ */
/*  End of processing                                                 */
/* ------------------------------------------------------------------ */

Call Final_Exit final_rc
Exit 9999

/* ------------------------------------------------------------------ */
/*  Subroutines                                                       */
/* ------------------------------------------------------------------ */

Final_Exit: Trace O
  Parse Arg final_rc
  Say Time()||": Processing is ending..."
  If swsu & Symbol("cur_cwd")="VAR" Then
    Call Syscall_Cmd "chdir" cur_cwd
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.i)", noexit_on_error
  End
  If pp.1<>-1 Then Call Syscall_Cmd "close (pp.1)", noexit_on_error
  If swsu & switched & foreground Then Do
    Call Syscall_Cmd "setreuid" cur_uid cur_euid, noexit_on_error
    If not_OK Then Say "UID settings could not be switched back..."
  End
```

```
  Call Syscalls "OFF"
Exit final_rc

Novalue_Error: Trace O
  Say "CPPAXO14W REXX error in sourceline" sigl "of" myname
  Say "CPPAXO15I Line" sigl||":" Strip(Sourceline(sigl))
  Say "CPPAXO16E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Shell_Cmd: Trace O
  Parse Arg shell_cmd
  parm.O = Words(shell_cmd)
  Do i=1 To parm.O
    parm.i = Word(shell_cmd,i)
  End
  shell_rc = -1
  shell_termsig = -1
  shell_stopsig = -1
  time_out = O
  Call Syscall_Cmd "pipe pp."
  fd.1 = pp.2
  If not_do_wait Then Call Syscall_Cmd "f_setfl (pp.1) (o_nonblock)"
  Call Syscall_Cmd "dup (fd.1)"
  fd.2 = retval
  Call Syscall_Cmd "spawnp (parm.1) 3 fd. parm. __environment."
  pid = retval
  all_output = ""
  If do_wait Then Do
    Call Syscall_Cmd "alarm" timeout_value /* timeout setting in secs */
    Call Syscall_Cmd "waitpid (pid) stat. O", info_on_timeout
  End
  Else Do
    time_beg = Time("E")
    Do Forever
      Call Syscall_Cmd "waitpid (pid) stat. (w_nohang)"
      If timeout_value<>O & Time("E")-time_beg>=timeout_value Then Do
        time_out = 1
        Leave
      End
      If stat.w_ifexited Then Leave
      Call Get_Output_Data
      Call Syscall_Cmd "sigpending sigset"
      If Substr(sigset,sigterm,1)=1 | Substr(sigset,sigalrm,1)=1 Then Do
        time_out = 1
        Leave
      End
      Call Syscall_Cmd "sleep 1"
    End
  End
  Call Syscall_Cmd "alarm O"
  If time_out Then Do
    Call Syscall_Cmd "kill (pid)" sigkill, noexit_on_error
    Call Syscall_Cmd "waitpid (pid) stat. O"
  End
```

```
  Select
    When stat.w_ifexited Then shell_rc = stat.w_exitstatus
    When stat.w_ifsignaled Then shell_termsig = stat.w_termsig
    When stat.w_ifstopped Then shell_stopsig = stat.w_stopsig
    Otherwise Nop
  End
  Do j=1 To 2
    If fd.j<>-1 Then Call Syscall_Cmd "close (fd.j)"
    fd.j = -1
  End
  Call Get_Output_Data
  Call Syscall_Cmd "close (pp.1)"
  pp.1 = -1
  If time_out Then Do
    Say "CPPAX017I Signal SIGALRM or SIGTERM received or timeout",
      "occurred, terminating..."
    Say "CPPAX018I Command:" shell_cmd
    If background Then Call Final_Exit 8
  End
Return

Get_Output_Data: Trace O
  Do Until retval<1000
    output_data = ""
    Call Syscall_Cmd "read (pp.1) output_data 1000"
    all_output = all_output||output_data
  End
  Do While Length(all_output)>0 & Pos(esc_n,all_output)>0
    Parse Var all_output output_line (esc_n) all_output
    If Symbol("fd_files")="VAR" Then Do
      fd_files = fd_files+1
      fdfile.fd_files = output_line
    End
    Else Say output_line
  End
Return

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  If rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then Do
    not_OK = 0
    retval = 0
  End
  Else not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
```

```
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
      errno_save = errno
      errnojr_save = errnojr
      If errno<>"A3" & errno<>"A4" & is_omvs_range Then
        show_reason = 1
      Else
        show_reason = 0
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        If err.se_errno<>"" Then
          Say "OMVS Return Code Explanation -" err.se_errno
        If show_reason & err.se_reason<>"" Then
          Say "OMVS Reason Code Explanation -" err.se_reason
      End
      errno = errno_save
      errnojr = errnojr_save
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.5 ADDMNTPS

REXX procedure ADDMNTPS can be used in the TSO foreground or TSO batch jobs to create all the missing mount points in a target structure after using COPYPAX to copy a source structure that has file systems mounted on it during the time of copying the data.

The logic of the procedure is as follows:

- It looks for all the mount points of all the file systems mounted in the system.
- If the file system is the source file system or the mount point is the root (/), no action is taken.
- If the parent directory of the mount point belongs to the source file system, then this mount point is one of the mount point directories that needs to exist in the target structure.
- A test is done to determine whether the corresponding directory in the target file system is already there (it could have been defined in the meantime). If it does not exist, it is created with owner UID 0 and permission bits set to 700.

Note: Routine ADDMNTPS is no longer really needed because the function is included in routine COPYPAX, and z/OS UNIX command **pax** supports adding missing mount points since z/OS V1R7.

B.5.1 ADDMNTPS code

Example B-4 displays the contents of ADDMNTPS.

Example: B-4 ADDMNTPS REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: ADDMNTPS                                             */
/*  Description: Add missing mount point mounts after clone process   */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2010   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: addmntps source_dir target_dir                       */
/*********************************************************************/

Trace O
Parse Source . . myname .
Parse Arg timeout_value source_dir target_dir junk
If target_dir="" | junk<>"" Then Do
  Say "ADMNTOO1E Syntax:" myname "time_out source_dir target_dir"
  Exit 8
End

If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "ADMNTOO2E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "ADMNTOO3E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "ADMNTOO4E The SIGNAL interface could not be established."
  Exit 8
End

switched = 0
final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ----------------------------------------------------------------- */
/*  Switch effective and real UID to zero if possible                */
/* ----------------------------------------------------------------- */

If background Then Do
  Call Syscall_Cmd "geteuid" /* effective uid 0 is sufficient here */
  cur_uid = retval
```

```
  If cur_uid<>0 Then
    Call Syscall_Cmd "seteuid 0", noexit_on_error, no_msgs
End
Else Do /* foreground */
  Call Syscall_Cmd "geteuid"
  cur_euid = retval
  If cur_euid<>0 Then Do
    Call Syscall_Cmd "seteuid 0", noexit_on_error, no_msgs
    If OK Then Do
      If foreground Then Say "Effective UID switched to 0..."
      switched = 1
    End
  End
End

/* ---------------------------------------------------------------- */
/*  Test whether source and target structure are in different devices */
/* ---------------------------------------------------------------- */
source_dir = Strip(source_dir,"T","/")||"/"
target_dir = Strip(target_dir,"T","/")||"/"
Call Syscall_Cmd "stat (source_dir) sst."
Call Syscall_Cmd "stat (target_dir) tst."
If sst.st_dev=tst.st_dev Then Do
  Say "ADMNT005E Source and target structure cannot be located in the",
    "same device."
  Call Final_Exit 8
End

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars                   */
/* ---------------------------------------------------------------- */

home = "/"
Call Syscall_Cmd "chdir" source_dir
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.0 = 5

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"

/* ---------------------------------------------------------------- */
/*  Looking for and creating the missing directory mount points     */
/* ---------------------------------------------------------------- */

Say "Looking for and creating the missing directory mount points..."
```

```
Call Syscall_Cmd "stat (source_dir) st."
source_devid = st.st_dev
Call Syscall_Cmd "getmntent mnt."
Do i=1 To mnt.0
  If mnt.mnte_dev.i=source_devid Then Iterate i /* is source */
  source_path = mnt.mnte_path.i
  If source_path="/" Then Iterate i /* is root */
  pp_slash_pos = Lastpos("/",Strip(source_path,"T","/"))
  If Left(source_path,1)<>"/" | pp_slash_pos=0 Then Do
    Say "ADMNT006E Unexpected mount point value:" source_path
    Call Final_Exit 8
  End
  parent_path = Left(source_path,pp_slash_pos)
  Call Syscall_Cmd "stat (parent_path) st."
  If st.st_dev=source_devid & Pos(source_dir,parent_path) Then Do
    target_path = target_dir||Substr(source_path,Length(source_dir)+1)
    Call Syscall_Cmd "stat (target_path) st.", noexit_on_error, no_msgs
    Select
      When errno=81 Then Do
        Call Syscall_Cmd "mkdir (target_path) 700"
        Say "Mount point" target_path "has been created..."
      End
      When OK Then
        Say "Mount point" target_path "exists already..."
      Otherwise Do
        Call Syscall_Cmd "stat (target_path) st."
        Say "Unexpected error on ""stat"" command..."
        Call Final_Exit 8
      End
    End
  End
End

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit:
  Parse Arg final_rc
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.i)", noexit_on_error
  End
  If pp.1<>-1 Then Call Syscall_Cmd "close (pp.1)", noexit_on_error
  If cur_euid<>0 & switched & foreground Then Do
```

```
    Call Syscall_Cmd "seteuid" cur_euid, noexit_on_error
    If not_OK Then Say "Effective UID could not be switched back..."
    Else Say "Effective UID switched back..."
  End
Exit final_rc

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        Say ""
        Say "OMVS Return Code Explanation -"
        Say err.se_errno
        Say "OMVS Reason Code Explanation -"
        Say err.se_reason
        Say ""
      End
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.6 CN procedure

The REXX procedure CN allows the user to execute MVS system commands and trap the syslog output if used in the TSO foreground or in TSO batch jobs.

B.6.1 TSO CONSOLE command facility

CN uses the TSO CONSOLE command facility to execute MVS system commands. Because a user acting as a system administrator will probably have appropriate OPERPARM settings, you may simply need to permit user access for the TSO CONSOLE command, as shown in Figure B-1.

```
SETROPTS CLASSACT(TSOAUTH)
SETROPTS RACLIST(TSOAUTH)
RDEFINE TSOAUTH CONSOLE UACC(NONE)
PERMIT CONSOLE CLASS(TSOAUTH) ID(HERING) ACCESS(READ)
SETROPTS RACLIST(TSOAUTH) REFRESH
```

Figure B-1 Defining and permitting access to the TSO CONSOLE command facility

When using this interface while a console with your user ID's name is already active (for example, an extended MCS console in SDSF or (E)JES), this can cause some trouble. To avoid this problem, create an alternate console and define the same OPERPARM settings that you have. This is shown in Figure B-2.

```
ADDUSER HERICONS DFLTGRP(SYS1) NAME('HERING ALT CONSOLE') OPERPARM(AUTH(MASTER)
  AUTO(YES) ...) NOPASSWORD NOOIDCARD
PERMIT CONSOLE CLASS(TSOAUTH) ID(HERICONS) ACCESS(READ)
SETROPTS RACLIST(TSOAUTH) REFRESH
```

Figure B-2 Defining an alternate console

By specifying NOOIDCARD and NOPASSWORD, you define a protected user ID that cannot be used to enter the system by any means that require a password to be specified, such as a TSO logon.

Attention: In the active MPFLSTxx parmlib member, or in general in the active message processing facility (MPF) settings, you need to have the following line included and active:

```
.NO_ENTRY,SUP(NO),RETAIN(YES),AUTO(YES)
```

This assures that an EMCS console used with the appropriate setup (such as the TSO console command facility) really catches all messages.

Permit access to this console for the user, as shown in Figure B-3. For reference purposes, all the OPERCMDS definitions are displayed, starting from class activation.

```
SETROPTS CLASSACT(OPERCMDS)
SETROPTS GENERIC(OPERCMDS)
SETROPTS RACLIST(OPERCMDS)
RDEFINE OPERCMDS MVS.MCSOPER.HERICONS UACC(NONE)
PERMIT MVS.MCSOPER.HERICONS CLASS(OPERCMDS) ID(HERING) ACCESS(READ)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure B-3 Permitting access to the new alternate console

Important: If the OPERCMDS class is active, you may need to define a profile such as MVS.MCSOPER.HERING for the user and permit read access to it. This is necessary if the user does not already have read access to a generic profile (for example, MVS.MCSOPER.*). Otherwise, the console may only get the default authorization of INFO instead of MASTER.

B.6.2 Usage rules and benefits of CN

The REXX procedure CN is useful whenever you execute an MVS system command that will complete within some seconds and you want to trap the output provided without a need to go to a console or search for the output data in the syslog.

Functions provided are:

- CN executes an MVS system command and as does the following:
  - Uses the user ID as the name of the console by default.
  - Uses the name specified by the first parameter C=console_name.
- According to the timeout values specified in the header area of the procedure, CN traps messages written to the syslog for a while.
- In the TSO foreground, the trapped data is collected in a temporary data set and automatically displayed in BROWSE mode.
- In TSO batch jobs. the output is available with DD name SYSTSPRT.
- Long message output lines are kept in one line and are readable without any need to edit the data.
- CN supports restarting of zFS by reacting to pseudo MVS command RESTART ZFS. C.5.3, “Restart zFS” on page 469 shows a sample JCL that exploits this function.

B.6.3 CN code

Example B-5 displays the contents of CN.

Example: B-5 CN REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: CN                                                   */
/*  Description: Run a console command and get the output displayed   */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2005   */
/*    Format is: cn <c=console_name> console_cmd                      */
/*********************************************************************/

first_msg_timeout = 5    /* timeout for the first message to display  */
inter_msg_timeout = 2    /* timeout between single messages           */
total_msg_timeout = 20   /* timeout for the last message to display   */

Trace O
Parse Source . . myname .
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
ispf_active = (Sysvar("SYSISPF")="ACTIVE")
ispf_inactv = (ispf_active=0)
If foreground & ispf_inactv Then Do
  Call Disp_Msg "You need to have ISPF active if run in foreground."
  Exit 8
End

Parse Arg console_name console_cmd 1 argument_line
Upper console_name
If Left(console_name,2)="C=" Then console_name = Substr(console_name,3)
Else Do
  console_cmd = argument_line
```

```
  console_name = ""
End
If console_name="" Then console_name = Userid()
If console_cmd="" Then Do
  Call Disp_Msg "Syntax:" myname "< C=console_name > console_cmd"
  Exit 4
End

If foreground Then Do
  msg_save = Msg("OFF")
  "ALLOC DD(SYSTSPRT) LRECL(255) RECFM(V B) CYLINDERS SPACE(1,1)",
    "RELEASE DSORG(PS) NEW REUSE UNIT(SYSALLDA)"
  Call Msg msg_save
End
"CONSPROF SOLDISPLAY(NO) SOLNUM(4000) UNSOLDISPLAY(NO) UNSOLNUM(4000)"
If rc<>0 Then Exit rc

tso.0 = 0
Call Outtrap "TSO."
"CONSOLE ACTIVATE NAME("||console_name||")"
Call Outtrap "OFF"
If rc<>0 Then Do
  retc = rc
  message = "CONSOLE COMMAND RC("||retc||")."
  If retc=40 Then Do
    msg_id = "AN ERROR OCCURRED DURING CONSOLE INITIALIZATION."
    Do i=1 To tso.0
      If Pos(msg_id,tso.i)>0 Then Do
        Parse Var tso.i "THE MCSOPER RETURN CODE WAS X'" mcs_rc,
          "' AND THE REASON CODE WAS X'" mcs_rsn "'."
        Leave i
      End
    end
    If mcs_rc="00000004" Then
      message = "CONSOLE command has terminated. A console named",
        console_name "is already active."
    Else
      message = "MCSOPER returned Rc("||mcs_rc||") and Rsn("||,
        mcs_rsn||"). See 'MVS Auth Assm Services Reference LLA-SDU'",
        "for further information."
  End
  Else Do i=1 To tso.0
    message = message tso.i
  End
  Call Disp_Msg message
  Exit 8
End

cnn = 0
If Translate(console_cmd)="RESTART ZFS" Then Do
  Address CONSOLE "D R,R"
  zfs_smsg = "BPXF032D FILESYSTYPE ZFS TERMINATED."
  zfs_rmsg = zfs_smsg " REPLY 'R' WHEN"
  zfs_rnr = -1
  Call Get_Msgs ""
```

```
  Do i=1 To cnn
    message = cn.i
    If Pos(zfs_rmsg,message)<>0 Then Do
      Parse Var message "*" reply_nr (zfs_rmsg)
      reply_nr = Strip(reply_nr,"T")
      If Verify(reply_nr,"1234567890")=0 Then Do
        zfs_rnr = reply_nr
        Leave i
      End
    End
  End
  If zfs_rnr<>-1 Then Do
    Address CONSOLE "R" zfs_rnr||",R"
    Call Get_Msgs ""
  End
  Else Do
    cnn = cnn+1
    cn.cnn = " *** Message """||zfs_smsg||""" not found ***"
  End
End
Else Do
  Address CONSOLE console_cmd
  Call Get_Msgs ""
End
"CONSOLE DEACTIVATE"
"CONSPROF SOLDISPLAY(YES) UNSOLDISPLAY(YES)"
If cnn>0 Then Do
  cn.0 = cnn
  "EXECIO" cnn "DISKW SYSTSPRT (STEM CN. FINIS"
End
If foreground Then Do
  If cnn>0 Then Do
    Address ISPEXEC "ISPEXEC LMINIT DATAID(TEMPFILE) DDNAME(SYSTSPRT)"
    Address ISPEXEC "ISPEXEC BROWSE DATAID("||tempfile||")"
  End
  Call Msg "OFF"
  "FREE DD(SYSTSPRT)"
  Call Msg msg_save
End
Exit

Get_Msgs:
  Parse Arg msg_type
  time = first_msg_timeout
  Call Time "R"
  OK = 0
  Do Until (result<>0 | OK)
    consdata.0 = 0
    Call Getmsg "CONSDATA.",msg_type,,,time
    time = inter_msg_timeout
    Do i=1 to consdata.0
      If cnn=0 & Word(consdata.i,1)="IEA630I" Then Iterate i
      cnn = cnn+1
      cn.cnn = consdata.i
    End
```

```
    If Time("E")>total_msg_timeout Then OK = 1
  End
Return

Disp_Msg:
  Parse Arg zedlmsg
  If foreground & ispf_active Then Do
    zedsmsg =""
    Address ISPEXEC "SETMSG MSG(ISRZ001)"
  End
  Else Say zedlmsg
Return
```

B.7 CNFZFS

The REXX procedure CNFZFS is useful whenever you execute an F ZFS command and want to trap the output provided without going to a console or searching for the output data in the syslog.

B.7.1 Usage rules and benefits of CNFZFS

CNFZFS provides the following functions:

- CNFZFS executes an MVS system command with the given zFS procname, and it:
  - Uses the user ID as the name of the console by default
  - Uses the name specified by the first parameter C=console_name
- According to the timeout values specified in the header area of the procedure, CN traps messages written to the syslog for a while.
- In the TSO foreground, the trapped data is collected in a temporary data set and automatically displayed in BROWSE mode.
- In TSO batch jobs, the output is available with DD name SYSTSPRT.
- Long message output lines are kept in one line and are more likely readable without the need to edit the data.

B.7.2 CNFZFS code

Example B-6 displays the contents of CNFZFS.

Example: B-6 CNFZFS REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: CNFZFS                                               */
/*  Description: Run a "F ZFS" command and get the output displayed   */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2005   */
/*    Format is: cnfzfs <c=console_name> zfsproc zfsparms             */
/*               Resulting "F" command: f zfsproc,zfsparms            */
/*********************************************************************/

first_msg_timeout = 5    /* timeout for the first message to display  */
inter_msg_timeout = 2    /* timeout between single messages           */
total_msg_timeout = 20   /* timeout for the last message to display   */
```

```
Trace O
Parse Source . . myname .
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
ispf_active = (Sysvar("SYSISPF")="ACTIVE")
ispf_inactv = (ispf_active=0)
If foreground & ispf_inactv Then Do
  Call Disp_Msg "You need to have ISPF active if run in foreground."
  Exit 8
End

Parse Arg console_name console_cmd 1 argument_line
Upper console_name
If Left(console_name,2)="C=" Then console_name = Substr(console_name,3)
Else Do
  console_cmd = argument_line
  console_name = ""
End
If console_name="" Then console_name = Userid()
Parse Var console_cmd zfs_proc zfs_parms
zfs_proc = Translate(Strip(zfs_proc))
If zfs_parms="" Then Do
  Call Disp_Msg "Syntax:" myname "<C=console_name> zfsproc zfsparms"
  Exit 4
End

console_cmd = "F" zfs_proc||","||zfs_parms
If foreground Then Do
  msg_save = Msg("OFF")
  "ALLOC DD(SYSTSPRT) LRECL(255) RECFM(V B) CYLINDERS SPACE(1,1)",
    "RELEASE DSORG(PS) NEW REUSE UNIT(SYSALLDA)"
  Call Msg msg_save
End
"CONSPROF SOLDISPLAY(NO) SOLNUM(4000) UNSOLDISPLAY(NO) UNSOLNUM(4000)"
If rc<>0 Then Exit rc

tso.0 = 0
Call Outtrap "TSO."
"CONSOLE ACTIVATE NAME("||console_name||")"
Call Outtrap "OFF"
If rc<>0 Then Do
  retc = rc
  message = "CONSOLE COMMAND RC("||retc||")."
  If retc=40 Then Do
    msg_id = "AN ERROR OCCURRED DURING CONSOLE INITIALIZATION."
    Do i=1 To tso.0
      If Pos(msg_id,tso.i)>0 Then Do
        Parse Var tso.i "THE MCSOPER RETURN CODE WAS X'" mcs_rc,
          "' AND THE REASON CODE WAS X'" mcs_rsn "'."
        Leave i
      End
    end
    If mcs_rc="00000004" Then
      message = "CONSOLE command has terminated. A console named",
        console_name "is already active."
```

```
    Else
      message = "MCSOPER returned Rc("||mcs_rc||") and Rsn("||,
        mcs_rsn||"). See 'MVS Auth Assm Services Reference LLA-SDU'",
        "for further information."
  End
  Else Do i=1 To tso.0
    message = message tso.i
  End
  Call Disp_Msg message
  Exit 8
End

cnn = 1
cn.1 = "" console_cmd
Address CONSOLE console_cmd
Call Get_Msgs ""
exit_rc = result
"CONSOLE DEACTIVATE"
"CONSPROF SOLDISPLAY(YES) UNSOLDISPLAY(YES)"
If cnn>0 Then Do
  cn.0 = cnn
  "EXECIO" cnn "DISKW SYSTSPRT (STEM CN. FINIS"
End
If foreground Then Do
  If cnn>0 Then Do
    Address ISPEXEC "ISPEXEC LMINIT DATAID(TEMPFILE) DDNAME(SYSTSPRT)"
    Address ISPEXEC "ISPEXEC BROWSE DATAID("||tempfile||")"
  End
  Call Msg "OFF"
  "FREE DD(SYSTSPRT)"
  Call Msg msg_save
End
Exit exit_rc

Get_Msgs:
  Parse Arg msg_type
  retc = 8
  time = first_msg_timeout
  Call Time "R"
  OK = 0
  Do Until (result<>0 | OK)
    consdata.0 = 0
    Call Getmsg "CONSDATA.",msg_type,,,time
    time = inter_msg_timeout
    Do i=1 to consdata.0
      If i=1 Then Do
        Parse Upper Var consdata.1 word_one .
        If Left(word_one,6)<>"IOEZ00" Then Leave i
        /* ------------------------------------------------------- *
         IOEZ00025I ZFS: MODIFY command - "Parm" completed successfully.
         IOEZ00024E ZFS: MODIFY command - "Parm" failed.
         * ------------------------------------------------------- */
        If word_one="IOEZ00025I" | word_one="IOEZ00024E" Then OK = 1
        If word_one="IOEZ00025I" Then retc = 0
      End
```

```
      cnn = cnn+1
      cn.cnn = consdata.i
    End
    If Time("E")>total_msg_timeout Then OK = 1
  End
Return retc

Disp_Msg:
  Parse Arg zedlmsg
  If foreground & ispf_active Then Do
    zedsmsg =""
    Address ISPEXEC "SETMSG MSG(ISRZ001)"
  End
  Else Say zedlmsg
Return
```

B.8 SU procedure

The REXX procedure SU allows you to execute a TSO or zFS-related IOEZADM command in SU mode if you are a BPX.SUPERUSER. It has been taken from the z/OS UNIX System Services Tools and Toys (USSTools) on the Internet, which is available at the following URL:

http://www-1.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html

Simply append the desired command to SU.

B.8.1 SU code

Example B-7 displays the contents of SU.

Example: B-7 SU REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: SU                                                   */
/*  Description: Switch to SU mode if possible and run TSO commands   */
/*               Property of IBM  (C) Copyright IBM Corp. 2000-2009   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: su tso_cmd                                           */
/**********************************************************************/

Trace O
Parse Source . . myname .
Parse Arg tso_cmd
If tso_cmd="" Then Do
  Say myname||": No TSO command has been provided."
  Exit 4
End

If Syscalls("ON")>4 Then Do
  Say myname||": The SYSCALL interface could not be established."
  Exit 8
End

foreground = (Sysvar("SYSENV")="FORE")
```

```
background = (foreground=0)

switched = 0
no_msgs = 1
noexit_on_error = 1

/* ---------------------------------------------------------------- */
/*  Switch effective and real UID to zero if needed and possible    */
/* ---------------------------------------------------------------- */

If background Then Do
  Call Syscall_Cmd "getuid"
  cur_uid = retval
  If cur_uid<>0 Then Do
    Call Syscall_Cmd "setuid 0", noexit_on_error, no_msgs
    If not_OK Then Say "The UID could not be switched to 0..."
  End
End
Else Do /* foreground */
  Call Syscall_Cmd "geteuid"
  cur_euid = retval
  If cur_euid<>0 Then Do
    Call Syscall_Cmd "getpwnam" Userid() "pw."
    init_uid = pw.pw_uid
    Call Syscall_Cmd "setreuid" init_uid init_uid
    Call Syscall_Cmd "setreuid 0 0", noexit_on_error, no_msgs
    /* ------------------------------------------------ *
    Call Syscall_Cmd "seteuid 0", noexit_on_error, no_msgs
     * ------------------------------------------------ */
    If OK Then switched = 1
    Else Say "Effective UID could not be switched to 0..."
  End
End

/* ---------------------------------------------------------------- */
/*  Run TSO Command                                                 */
/* ---------------------------------------------------------------- */

Address TSO tso_cmd
final_rc = rc

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

If cur_euid<>0 & switched & foreground Then Do
  Call Syscall_Cmd "setreuid" cur_euid cur_euid, noexit_on_error
  /* ------------------------------------------- *
  Call Syscall_Cmd "seteuid" cur_euid, noexit_on_error
   * ------------------------------------------- */
  If not_OK Then Say "UID could not be switched back..."
End
If final_rc<>0 Then Say "Rc("||final_rc||")"
Exit final_rc
```

```
/* ---------------------------------------------------------------- */
/*  Subroutine                                                      */
/* ---------------------------------------------------------------- */

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(2OFF)
    syscall_rc = rc
    If errno<>"A3" & errno<>"A4" & is_omvs_range Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Say "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Say "OMVS Reason Code Explanation -" err.se_reason
    End
    If exit_on_error Then Call Final_Exit 8
  End
Return
Return not_OK
```

B.9 LARGEFIL procedure

The REXX procedure LARGEFIL is used in the comparison test between HFS and zFS. It fills 500 MB of data into one big file in each of the file systems.

B.9.1 LARGEFIL code

Example B-8 displays the contents of LARGEFIL.

Example: B-8 LARGEFIL REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: LARGEFIL                                             */
/*  Description: Create large file in a file system and fill it       */
/*               Property of IBM  (C) Copyright IBM Corp. 2002        */
/*    Format is: largefil directory                                   */
/**********************************************************************/

Trace O
```

```
Parse Source . . myname .
Parse Arg directory
If directory="" Then Do
  Say "LRGFL001E Syntax:" myname "directory"
  Exit 8
End

timeout_value = 0
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "LRGFL002E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "LRGFL003E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "LRGFL004E The SIGNAL interface could not be established."
  Exit 8
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars                   */
/* ---------------------------------------------------------------- */

home = "/"
Call Syscall_Cmd "chdir" directory
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.0 = 5

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"
```

```
/* ------------------------------------------------------------------ */
/*  Create and fill the new large file in the file system             */
/* ------------------------------------------------------------------ */
large_file = "large_file"
Call Syscall_Cmd "open (large_file)" o_creat+o_excl+o_wronly 666
fd_large = retval
one_meg = 1024*1024
Say "About to filling the file with data..."
Say ""
Call Time "R"
Do i=1 To 500
  block_data = Copies(Right(i,4,0),4)  /* 16 byte */
  block_data = Copies(block_data,64)   /* 1KB     */
  block_data = Copies(block_data,1024) /* 1MB     */
  block_data = Overlay(Time("L")||"-"||Right(i,10,0)||"--",block_data)
  Call Syscall_Cmd "write (fd_large) block_data" one_meg
  If i//50=0 Then Do
    Say "Blocks written and time used in seconds up to now:" Right(i,3),
      Right(Time("E"),12)
  End
End
Call Syscall_Cmd "close (fd_large)"
Say ""
Say "Finished filling the file with data..."
Say "Time used for whole filling process:" Time("E") "sec"

/* ------------------------------------------------------------------ */
/*  End of processing                                                 */
/* ------------------------------------------------------------------ */

Call Final_Exit final_rc
Exit 9999

/* ------------------------------------------------------------------ */
/*  Subroutines                                                       */
/* ------------------------------------------------------------------ */

Final_Exit:
  Parse Arg final_rc
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.i)", noexit_on_error
  End
  If pp.1<>-1 Then Call Syscall_Cmd "close (pp.1)", noexit_on_error
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
```

```
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        Say ""
        Say "OMVS Return Code Explanation -"
        Say err.se_errno
        Say "OMVS Reason Code Explanation -"
        Say err.se_reason
        Say ""
      End
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.10 LARGEIOS procedure

The REXX procedure LARGEIOS is used in the comparison test between HFS and zFS to perform the specified number of random I/Os to read or write 1 MB of data.

B.10.1 LARGEIOS code

Example B-9 displays the contents of LARGEFIL.

Example: B-9 LARGEIOS REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: LARGEIOS                                             */
/*  Description: Do a specied amount of large random I/Os             */
/*               Property of IBM  (C) Copyright IBM Corp. 2002        */
/*    Format is: largeios blk_ios type_ios seed lf_dir                */
/*********************************************************************/

Trace O
Parse Source . call_type myname . . . . omvs .
Parse Arg blk_ios type_ios seed lf_dir
Upper type_ios
If lf_dir="" Then Do
  Say "Syntax:" myname "blk_ios type_ios seed lf_dir"
```

```
  Exit 8
End

Parse Var type_ios 1 read_id 2 read_pc
If read_id<>"R" | Length(read_pc)>2 & read_pc<>100 |,
  Verify(read_pc,"1234567890")<>0 Then Do
  Say "IO type value must be set to Rxx, where xx is a number",
    "between 0 and 100."
  Exit 8
End

If omvs<>"OMVS" Then Do
  If Syscalls("ON")>4 Then Do
    Say "The SYSCALL environment could not be established."
    Exit 8
  End
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
fd = -1
one_meg = 1024*1024
minus_one_meg = (-1)*one_meg

/* ---------------------------------------------------------------- */
/*  Do the specified amount of large random I/Os                    */
/* ---------------------------------------------------------------- */

rnr.1 = Random(1,500,seed)
Do i=2 To blk_ios
  rnr.i = Random(1,500)
End
Call Time "R"
large_file = Strip(lf_dir,"T","/")||"/large_file"
Call Syscall_Cmd "open (large_file) (o_rdwr)"
fd = retval
rs = 0
ws = 0
Do i=1 To blk_ios
  rnr = rnr.i
  file_offset = (rnr-1)*one_meg
  Call Syscall_Cmd "lseek (fd) (file_offset) (seek_set)"
  Select
    When i=1 & read_pc<50 Then io_type = "W"
    When i=1 Then io_type = "R" /* read_pc>=50 */
    When (rs/(rs+ws))*100>read_pc Then io_type = "W"
    Otherwise io_type = "R" /* (rs/(rs+ws))*100<=read_pc */
  End
  If io_type="R" Then Do
    Call Syscall_Cmd "read (fd) block_data (one_meg)"
    Say "Read  block" Right(rnr,3) Right(Time("E"),12) "-",
      Left(block_data,32)
    rs = rs+1
  End
```

```
  Else Do
    block_data = Copies(Right(rnr,4,0),4)/* 16 byte */
    block_data = Copies(block_data,64)   /* 1KB     */
    block_data = Copies(block_data,1024) /* 1MB     */
    block_data = Overlay(Time("L")||"-"||Right(i,10,0)||"--",block_data)
    Call Syscall_Cmd "write (fd) block_data" one_meg
    Say "Write block" Right(rnr,3) Right(Time("E"),12) "-",
      Left(block_data,32)
    ws = ws+1
  End
End

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit:
  Parse Arg final_rc
  If fd<>-1 Then Call Syscall_Cmd "close (fd)", noexit_on_error
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      Say ""
      Say "OMVS Return Code Explanation -"
      Say err.se_errno
      Say "OMVS Reason Code Explanation -"
      Say err.se_reason
      Say ""
    End
    If exit_on_error Then Call Final_Exit 8
  End
Return
```

B.11 LARGESCD procedure

The REXX procedure LARGESCD is used in the comparison test between HFS and zFS. It starts the specified number of processes that are running in parallel, and provides the needed parameters to finally collect all the statistical information.

B.11.1 LARGESCD code

Example B-10 displays the contents of LARGESCD.

Example: B-10 LARGESCD REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: LARGESCD                                             */
/*  Description: Large I/O test "scheduler"                           */
/*               Property of IBM  (C) Copyright IBM Corp. 2002        */
/*    Format is: largescd timeout idir procs blkios typeios seed fdir */
/*********************************************************************/

Trace O
Parse Source . . myname .
Parse Arg timeout_value rexxlib procs blkios typeios seed fdir
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "LRGSCO01E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "LRGSCO02E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "LRGSCO03E The SIGNAL interface could not be established."
  Exit 8
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ----------------------------------------------------------------- */
/*  Set home and working directory, setup envvars                    */
/* ----------------------------------------------------------------- */

home = "/"
Call Syscall_Cmd "chdir" home
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
```

```
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.6 = "TSOALLOC=sysexec"
__environment.7 = "sysexec='"||rexxlib||"'"
__environment.0 = 7

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"

/* ---------------------------------------------------------------- */
/*  Start all largeios processes and finally collect the results    */
/* ---------------------------------------------------------------- */

parm.0 = 3
parm.1 = "sh"
parm.2 = "-c"
Do i=1 To procs
  parm.3 = "tso 'largeios" blkios typeios seed+i-1 fdir||"'"
  shell_rc = -1
  shell_termsig = -1
  shell_stopsig = -1
  time_out = 0
  Call Syscall_Cmd "open /dev/null (o_rdonly)"
  fd.0 = retval
  fd.0.i = fd.0
  Call Syscall_Cmd "pipe pp."
  fd.1 = pp.2
  fd.1.i = fd.1
  pp.1.i = pp.1
  Call Syscall_Cmd "dup (fd.1)"
  fd.2 = retval
  fd.2.i = fd.2
  Call Syscall_Cmd "spawnp (parm.1) 3 fd. parm. __environment."
  pid.i = retval
End
avg_time = 0
min_time = 0
max_time = 0
proc_cnt = 0
Do i=1 To procs
  Call Syscall_Cmd "alarm" timeout_value /* timeout setting in secs */
  Call Syscall_Cmd "waitpid (pid.i) stat. 0", info_on_timeout
  Call Syscall_Cmd "alarm 0"
  If time_out Then Do
    Call Syscall_Cmd "kill (pid.i)" sigkill, noexit_on_error
    Call Syscall_Cmd "waitpid (pid.i) stat. 0"
  End
  Select
```

```
    When stat.w_ifexited Then shell_rc = stat.w_exitstatus
    When stat.w_ifsignaled Then shell_termsig = stat.w_termsig
    When stat.w_ifstopped Then shell_stopsig = stat.w_stopsig
    Otherwise Nop
  End
  pid = pid.i
  pid.i = -1
  Do j=0 To 2
    If fd.j.i<>-1 Then Call Syscall_Cmd "close (fd.j.i)"
    fd.j.i = -1
  End
  hdr = "Results of process no." Right(i,5) "with process id" pid
  Say hdr
  Say Copies("-",Length(hdr))
  Say ""
  all_output = ""
  Do Until retval<1000
    Call Syscall_Cmd "read (pp.1.i) output_data 1000"
    all_output = all_output||output_data
  End
  first_line = 1
  Do While Length(all_output)>0
    Parse Var all_output output_line (esc_n) all_output
    If first_line Then first_line = 0
    Else Say output_line
  End
  Parse Var output_line . . . act_time .
  If Verify(act_time,"1234567890.")=0 & Datatype(act_time,"N") Then Do
    act_time = Format(act_time,,2)
    proc_cnt = proc_cnt+1
    avg_time = avg_time+act_time
    max_time = Max(max_time,act_time)
    If proc_cnt=1 Then min_time = act_time
    Else min_time = Min(min_time,act_time)
  End
  Call Syscall_Cmd "close (pp.1.i)"
  pp.1.i = -1
  If time_out Then Do
    Say "LRGSC004E Timeout occurred during command processing..."
    Say "LRGSC005I Command:" shell_cmd
    final_rc = 8
  End
  If shell_rc<>0 Then Do
    Say "*** non-zero return code: Rc("||shell_rc||")."
    final_rc = 8
  End
  Say ""
End
Say ""
Say "Summary for all processes run successfully"
Say ""
Say "Average time:" Right(Format(avg_time/proc_cnt,,2),12)
Say "Minimal time:" Right(min_time,12)
Say "Maximal time:" Right(max_time,12)
Say ""
```

```
/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit:
  Parse Arg final_rc
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=1 To procs
    Do j=0 To 2
      If fd.j.i<>-1 Then
        Call Syscall_Cmd "close (fd.j.i)", noexit_on_error
    End
    If pid.i<>-1 Then Do
      Call Syscall_Cmd "kill (pid.i)" sigkill, noexit_on_error
      Call Syscall_Cmd "waitpid (pid.i) stat. 0"
    End
    If pp.1.i<>-1 Then
      Call Syscall_Cmd "close (pp.1.i)", noexit_on_error
  End
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        Say ""
        Say "OMVS Return Code Explanation -"
        Say err.se_errno
```

```
        Say "OMVS Reason Code Explanation -"
        Say err.se_reason
        Say ""
      End
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.12 UNIX script zfsgrow

The UNIX REXX script zfsgrow makes it possible to grow a zFS aggregate several times by specifying the number of grows and the aggregate name. A secondary extent value must be set for the VSAM data set.

B.12.1 zfsgrow code

Example B-11 displays the contents of zfsgrow.

Example: B-11 UNIX REXX script zfsgrow

```
/* REXX ***************************************************************/
/*    Procedure: zfsgrow                                              */
/*  Description: Run several zfsadm grow commands in a loop           */
/*               Property of IBM  (C) Copyright IBM Corp. 2004        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: zfsgrow grows aggregate_name                         */
/*********************************************************************/

Trace O
Parse Source . . myname . . . . omvs .
myname = Substr(myname,Lastpos("/",myname)+1)
Parse Arg grows aggregate_name .
If Verify(grows,"1234567890")<>0 | aggregate_name="" | grows=0 Then Do
  Say "Syntax:" myname "grows aggregate_name"
  Say "grows          - number of grow runs to be performed"
  Say "aggregate_name - name of the aggregate to be grown"
  Exit 2
End

Do i=1 To grows
  Say "Performing grow run" i||"..."
  "zfsadm grow -aggregate" aggregate_name "-size 0"
  If rc<>0 Then Do
    Say ""
    Say "The zfsadm grow was not successful; rc="||rc
    Say "No further grow run will be processed, stopping..."
    Exit rc
  End
End
```

```
Exit 0
```

B.13 RXLSAGGR procedure

The REXX procedure rxlsaggr allows you to display a list of the names of all attached aggregates on a system, together with the system name. It may also be used from a UNIX shell environment.

B.13.1 RXLSAGGR code

Example B-12 displays the contents of RXLSAGGR.

Example: B-12 REXX procedure RXLSAGGR

```
/* REXX ***************************************************************/
/*    Procedure: rxlsaggr V1.1                                        */
/*  Description: Listing attached aggregates with attributes using    */
/*               List attached aggregate names (version 2)/opcode 140 */
/*               and List aggregate status (Version 2)/opcode 146     */
/*               for pfsctl API ZFSCALL_AGGR command                  */
/*               Property of IBM  (C) Copyright IBM Corp. 2004-2010   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: rxlsaggr <aggr_name|-axd|-nax|-qsd|F8K|-sa|-all >    */
/**********************************************************************/

Trace O
Parse Source . . myname . . . . omvs .
myname = Substr(myname,Lastpos("/",myname)+1)
omvs = (omvs="OMVS")
If omvs Then retc = 1
Else Do
  retc = 8
  If Syscalls("ON")>4 Then Do
    Say "The SYSCALL environment could not be established."
    Exit retc
  End
End

Parse Upper Arg rx_arg .
list_axd = 0
list_nax = 0
list_qsd = 0
list_sa = 0
list_f8k = 0
list_all = 0
list_one = 0
info.0 = "Disabled"
info.1 = "Enabled"
Select
  When rx_arg="?" Then Do
    Say myname "          - lists one information line per aggregate"
    Say myname "-axd      - lists the aggregates with aggrgrow=on"
    Say myname "-nax      - lists the aggregates with aggrgrow=off"
    Say myname "-qsd      - lists the aggregates that are quiesced"
```

```
    Say myname "-f8k      - lists the number of free 8K blocks"
    Say myname "-sa       - lists the aggregates that are sysplex-aware"
    Say myname "-all      - lists important data for all aggregates"
    Say myname "aggr_name - lists important data for given aggregate"
    Exit 0
  End
  When rx_arg="-AXD" Then list_axd = 1 /* aggregates with autogrow=on */
  When rx_arg="-NAX" Then list_nax = 1 /* autogrow=off aggregates     */
  When rx_arg="-QSD" Then list_qsd = 1 /* quiesced aggregates         */
  When rx_arg="-F8K" Then list_f8k = 1 /* number of free 8K blocks    */
  When rx_arg="-SA"  Then list_sa = 1  /* aggregates with sysplex-aware*/
  When rx_arg="-ALL" Then list_all = 1 /* list important information  */
  When rx_arg<>"" Then Do /* list important infos for given aggregate */
    aggr_name = rx_arg
    list_one = 1
    aggr_nfnd = 1
  End
  Otherwise Nop
End
If MVSVar("SYSMVS")<"SP7.0.4" Then Do
  Say "The level of the BCP must be at z/OS V1R4 or higher to use this",
    "function." /* APAR OA11602: R6 function rollback to R4 and R5 */
  Exit retc
End
If MVSVar("SYSMVS")<"SP7.0.7" Then Do
  Say "Listing the number of free 8K blocks is supported only with",
    "z/OS V1R7 or above."
  Exit retc
End

Signal On Syntax
Numeric Digits 10
final_rc = 0
no_msgs = 1
noexit_on_error = 1
get_size_needed = 0

zero_parm = D2c(0,4)
zero_01 = "00"x
zero_33 = Copies(zero_01,33)
zero_st = Copies(zero_01,164) /* aggr_status initialization part */
zfs_cmd = X2d("40000005")/* ZFSCALL_AGGR */
op_code = D2c(140,4) /* Listing attached aggregate names (Version 2) */
parm_len = 32
aggr_struct_size = 84 /* length of aggregate structure */
parm_3 = zero_parm /* not used */
parm_4 = zero_parm /* not used */
parm_5 = zero_parm /* not used */
parm_6 = zero_parm /* not used */
parm_3456 = parm_3 || parm_4 || parm_5 || parm_6
size_needed = zero_parm /* size needed for buffer, set to zero */

parm_0 = zero_parm /* no buffer is provided */
parm_1 = zero_parm /* no buffer is provided */
parm_2 = D2c(parm_len,4) /* offset to size */
```

```
zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
  size_needed
get_size_needed = 1
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
buf_size = C2d(Substr(zfs_buf,parm_len+1,4))
If buf_size<>0 Then Do
  parm_0 = D2c(buf_size,4) /* buffer length */
  parm_1 = D2c(parm_len,4) /* offset to first aggregate structure */
  parm_2 = D2c(buf_size+parm_len,4) /* offset to size */
  buffer = Left("",buf_size) /* buffer */
  zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
    buffer || size_needed
  Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
  buf_size = C2d(Substr(zfs_buf,buf_size+parm_len+1,4))
End
aggrs = buf_size/aggr_struct_size
aggr_off = parm_len
If list_axd | list_nax | list_qsd | list_sa | list_one Then Nop
Else Say "A total of" aggrs "aggregates are attached."
op_code = D2c(146,4) /* List aggregate status (Version 2), next calls */
  parm_2 = zero_parm /* not used next */
  parm_23456 = parm_2 || parm_3 || parm_4 || parm_5 || parm_6
Do i=1 To aggrs
  aggr_struct = Substr(zfs_buf,aggr_off+1,aggr_struct_size)
  aggr_eyecat = Left(aggr_struct,4)
  aggr_len = C2d(Substr(aggr_struct,5,1)) /* "84", not tested */
  aggr_ver = C2d(Substr(aggr_struct,6,1)) /* "2", not tested */
  If aggr_eyecat<>"AGID" Then Do
    Say "Unexpected data returned for current aggregate entry..."
    Call Final_Exit retc
  End
  aggr_name_c = Strip(Substr(aggr_struct,7,45),"T","00"x)
  aggr_system = Strip(Substr(aggr_struct,52,9),"T","00"x)
  aggr_include = 1
  If list_one & aggr_name_c<>aggr_name Then aggr_include = 0
  Else aggr_name = aggr_name_c
  If aggr_include Then Do
    Call List_aggr_info
    If list_one Then aggr_nfnd = 0
  End
  aggr_off = aggr_off+aggr_struct_size
End
If list_one Then Do
  If aggr_nfnd Then Say "Aggregate named" aggr_name "cannot be found."
End
Call Final_Exit 0
Exit 9999

List_aggr_info:
  aggr_id = "AGID" || "0001"x || Left(aggr_name,45,zero_01) || zero_33
  aggr_id = Overlay(D2c(Length(aggr_id),1),aggr_id,5)
  aggr_len = Length(aggr_id)
  aggr_status = "AGST" || "00000200"x || zero_st
  aggr_status = Overlay(D2c(Length(aggr_status),2),aggr_status,5)
  parm_0 = D2c(parm_len,4) /* offset to aggr_id */
```

```
parm_1 = D2c(parm_len+aggr_len,4) /* offset to aggr_status2 */
zfs_buf2 = op_code || parm_0 || parm_1 || parm_23456 ||,
  aggr_id || aggr_status
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf2" Length(zfs_buf2)
aggr_status = Substr(zfs_buf2,parm_len+aggr_len+1)
If Left(aggr_status,4)<>"AGST" Then Do
  Say "Unexpected data for aggregate" aggr_name||"..."
  Call Final_Exit retc
End
flag = Substr(aggr_status,19,1)
m4full  = Bitand(flag,'80'x)="80"x      /* monitored for full     */
rdonly  = Bitand(flag,'40'x)="40"x      /* read only              */
nbsecu  = Bitand(flag,'20'x)="20"x      /* new block security     */
compat  = Bitand(flag,'10'x)="10"x      /* HFS compatible         */
aggrow  = Bitand(flag,'08'x)="08"x      /* dynamic grow           */
aggmov  = Bitand(flag,'04'x)="04"x      /* aggrmove               */
qusced  = Bitand(flag,'01'x)="01"x      /* quiesced               */
flag2 = Substr(aggr_status,20,1)
dsable  = Bitand(flag2,'80'x)="80"x     /* disabled               */
splxaw  = Bitand(flag2,'40'x)="40"x     /* sysplex-aware          */
free_8K = Right(C2d(Substr(aggr_status,57,4))/8,10)
list_details = 0
Select
  When list_axd Then Do
    If aggrow Then Say aggr_name
  End
  When list_nax Then Do
    If aggrow Then Nop
    Else Say aggr_name
  End
  When list_qsd Then Do
    If qusced Then Say aggr_name
  End
  When list_sa Then Do
    If splxaw Then Say aggr_name
  End
 When list_f8K Then Do
    Say Left(aggr_name,47) free_8K
  End
  Otherwise Do
    If rdonly Then accmode = "R/O"
    Else accmode = "R/W"
    If qusced Then accmode = accmode "QUIESCE"
    If splxaw Then accmode = accmode "Sysplex-aware"
    Say Left(aggr_name,47) Left(aggr_system,9) accmode
    If list_all | list_one Then list_details = 1
  End
End
If list_details Then Do
  monitor_info = info.m4full
  If m4full Then monitor_info = monitor_info || ", (" ||,
    C2d(Substr(aggr_status,17,1)) || "," ||,
    C2d(Substr(aggr_status,18,1)) || ")"
  Say "Monitored for full  . . :" monitor_info
  Say "New block security  . . :" info.nbsecu
```

```
    compat_info = info.compat
    If compat Then Nop
    Else compat_info = compat_info ||"," C2d(Substr(aggr_status,13,4)),
      "file systems"
    Say "HFS compatibility . . . :" compat_info
    Say "Auto-extend . . . . . :" info.aggrow
    /*  "Number of file systems  :",
      Right(C2d(Substr(aggr_status,13,4)),10)
        "Threshold for aggr full :",
      Right(C2d(Substr(aggr_status,17,1)),10)
        "Increment for aggr full :",
      Right(C2d(Substr(aggr_status,18,1)),10) */
    Say "Number of fragments . . :",
      Right(C2d(Substr(aggr_status,21,4)),10)
    /*  "Fragment size . . . . . :",
      Right(C2d(Substr(aggr_status,25,4)),10)
        "Block size  . . . . . :",
      Right(C2d(Substr(aggr_status,29,4)),10) */
    Say "8K Blocks available . . :",
      Right(C2d(Substr(aggr_status,33,4))/8,10)
    Say "Aggregate free space KB :",
      Right(C2d(Substr(aggr_status,37,4)),10)
    /*  "Minimum fragments free  :",
      Right(C2d(Substr(aggr_status,41,4)),10)
        "Reserved  . . . . . :" Substr(aggr_status,45,12) */
    If MVSVar("SYSMVS")>="SP7.0.7" Then Do
      Say "Number free 8K blocks . :" free_8K
      Say "Number free 1K fragments:",
        Right(C2d(Substr(aggr_status,61,4)),10)
      Say "Log file size in KB . . :",
        Right(C2d(Substr(aggr_status,65,4)),10)
      /*  "Indirect log size in KB :",
        Right(C2d(Substr(aggr_status,69,4)),10) */
      Say "File system table in KB :",
        Right(C2d(Substr(aggr_status,73,4)),10)
      Say "Bitmap file size in KB  :",
        Right(C2d(Substr(aggr_status,77,4)),10)
      major_version = C2d(Substr(aggr_status,81,4))
      minor_version = C2d(Substr(aggr_status,85,4))
      Say "Disk format version . . :" major_version||"."||minor_version
    End
    /*  "Reserved  . . . . . :" Substr(aggr_status,89,84) */
    Say ""
  End
Return

Final_Exit:
  Parse Arg final_rc
Exit final_rc

Syntax:
  Say Errortext(rc) "in line" sigl "- Rc("||rc||")"
  Say "This is probably caused by invalid data that was received."
  Call Final_Exit retc
Exit 999
```

```
Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  If rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then not_OK = 0
  Else not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  If get_size_needed & errno="91" & errnojr="EF176274" Then Do
    not_OK = 0
    get_size_needed = 0
  End
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
    errno_save = errno
    errnojr_save = errnojr
    If errno<>"A3" & errno<>"A4" & is_omvs_range Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Say "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Say "OMVS Reason Code Explanation -" err.se_reason
    End
    errno = errno_save
    errnojr = errnojr_save
    If exit_on_error Then Call Final_Exit retc
  End
Return
```

B.14 RXZFSMON procedure

The REXX procedure rxzfsmon can be used to monitor the status of all or a specific zFS aggregate, especially including freespace information. It may also be used from a UNIX shell environment.

B.14.1 RXZFSMON code

Example B-13 displays the contents of RXZFSMON.

Example: B-13 REXX procedure RXZFSMON

```
/* REXX ***************************************************************/
/*    Procedure: rxzfsmon V1.1                                        */
```

```
/*  Description: Listing attached aggregates with attributes using    */
/*               List attached aggregate names (version 2)/opcode 140 */
/*               and List aggregate status (Version 2)/opcode 146     */
/*               for pfsctl API ZFSCALL_AGGR command                  */
/*               Property of IBM  (C) Copyright IBM Corp. 2004-2010   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: rxzfsmon <aggr_name>                                  */
/*********************************************************************/

Trace O
Parse Source . . myname . . . . omvs .
myname = Substr(myname,Lastpos("/",myname)+1)
omvs = (omvs="OMVS")
If omvs Then retc = 1
Else Do
  retc = 8
  If Syscalls("ON")>4 Then Do
    Say "The SYSCALL environment could not be established."
    Exit retc
  End
End

Parse Upper Arg rx_arg .
list_all = 0
list_one = 0
Select
  When rx_arg="?" Then Do
    Say
    Say myname "          - lists freespace information for all",
      "active aggregates"
    Say myname "aggr_name - lists freespace information for given",
      "aggregate"
    Say
    Say "Information provided is the following:"
    Say
    Say "Column 1= aggregate name"
    Say "Column 2= total number of 8K blocks in the aggregate"
    Say "Column 3= used number of 8K blocks in the aggregate"
    Say "Column 4= percentage of the used number of 8K blocks"
    Say "Column 5= X|- Q|- R|- S|- (auto-eXtend Quiesced Read-only",
      "Sysplex-aware)"
    Say
    Exit 0
  End
  When rx_arg="" Then list_all = 1 /* list infos for all aggregates   */
  When rx_arg<>"" Then Do /* list freespace infos for given aggregate */
    aggr_name = rx_arg
    list_one = 1
    aggr_nfnd = 1
  End
  Otherwise Nop
End
If MVSVar("SYSMVS")<"SP7.0.4" Then Do
  Say "The level of the BCP must be at z/OS V1R4 or higher to use this",
    "function." /* APAR OA11602: R6 function rollback to R4 and R5 */
```

```
  Exit retc
End
If MVSVar("SYSMVS")<"SP7.0.7" Then Do
  Say "Listing the number of free 8K blocks is supported only with",
    "z/OS V1R7 or above."
  Exit retc
End

Signal On Syntax
Numeric Digits 10
final_rc = 0
no_msgs = 1
noexit_on_error = 1
get_size_needed = 0

zero_parm = D2c(0,4)
zero_01 = "00"x
zero_33 = Copies(zero_01,33)
zero_st = Copies(zero_01,164) /* aggr_status initialization part */
zfs_cmd = X2d("40000005")/* ZFSCALL_AGGR */
op_code = D2c(140,4) /* Listing attached aggregate names (Version 2) */
parm_len = 32
aggr_struct_size = 84 /* length of aggregate structure */
parm_3 = zero_parm /* not used */
parm_4 = zero_parm /* not used */
parm_5 = zero_parm /* not used */
parm_6 = zero_parm /* not used */
parm_3456 = parm_3 || parm_4 || parm_5 || parm_6
size_needed = zero_parm /* size needed for buffer, set to zero */

parm_0 = zero_parm /* no buffer is provided */
parm_1 = zero_parm /* no buffer is provided */
parm_2 = D2c(parm_len,4) /* offset to size */
zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
  size_needed
get_size_needed = 1
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
buf_size = C2d(Substr(zfs_buf,parm_len+1,4))
If buf_size<>0 Then Do
  parm_0 = D2c(buf_size,4) /* buffer length */
  parm_1 = D2c(parm_len,4) /* offset to first aggregate structure */
  parm_2 = D2c(buf_size+parm_len,4) /* offset to size */
  buffer = Left("",buf_size) /* buffer */
  zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
    buffer || size_needed
  Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
  buf_size = C2d(Substr(zfs_buf,buf_size+parm_len+1,4))
End
aggrs = buf_size/aggr_struct_size
aggr_off = parm_len
no_hdrlines = 1
op_code = D2c(146,4) /* List aggregate status (Version 2), next calls */
  parm_2 = zero_parm /* not used next */
  parm_23456 = parm_2 || parm_3 || parm_4 || parm_5 || parm_6
Do i=1 To aggrs
```

```
  aggr_struct = Substr(zfs_buf,aggr_off+1,aggr_struct_size)
  aggr_eyecat = Left(aggr_struct,4)
  aggr_len = C2d(Substr(aggr_struct,5,1)) /* "84", not tested */
  aggr_ver = C2d(Substr(aggr_struct,6,1)) /* "2", not tested */
  If aggr_eyecat<>"AGID" Then Do
    Say "Unexpected data returned for current aggregate entry..."
    Call Final_Exit retc
  End
  aggr_name_c = Strip(Substr(aggr_struct,7,45),"T","00"x)
  aggr_system = Strip(Substr(aggr_struct,52,9),"T","00"x)
  aggr_include = 1
  If list_one & aggr_name_c<>aggr_name Then aggr_include = 0
  Else aggr_name = aggr_name_c
  If aggr_include Then Do
    Call List_aggr_info
    If list_one Then aggr_nfnd = 0
  End
  aggr_off = aggr_off+aggr_struct_size
End
If list_one Then Do
  If aggr_nfnd Then Say "Aggregate named" aggr_name "cannot be found."
  Else Say ""
End
If list_one Then Nop
Else Do
  Say
  Say "A total of" aggrs "aggregates are attached."
  Say
End
Call Final_Exit 0
Exit 9999

List_aggr_info:
  aggr_id = "AGID" || "0001"x || Left(aggr_name,45,zero_01) || zero_33
  aggr_id = Overlay(D2c(Length(aggr_id),1),aggr_id,5)
  aggr_len = Length(aggr_id)
  aggr_status = "AGST" || "00000200"x || zero_st
  aggr_status = Overlay(D2c(Length(aggr_status),2),aggr_status,5)
  parm_0 = D2c(parm_len,4) /* offset to aggr_id */
  parm_1 = D2c(parm_len+aggr_len,4) /* offset to aggr_status2 */
  zfs_buf2 = op_code || parm_0 || parm_1 || parm_23456 ||,
    aggr_id || aggr_status
  Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf2" Length(zfs_buf2)
  aggr_status = Substr(zfs_buf2,parm_len+aggr_len+1)
  If Left(aggr_status,4)<>"AGST" Then Do
    Say "Unexpected data for aggregate" aggr_name||"..."
    Call Final_Exit retc
  End
  flag = Substr(aggr_status,19,1)
  m4full  = Bitand(flag,'80'x)="80"x      /* monitored for full     */
  rdonly  = Bitand(flag,'40'x)="40"x      /* read only              */
  nbsecu  = Bitand(flag,'20'x)="20"x      /* new block security     */
  compat  = Bitand(flag,'10'x)="10"x      /* HFS compatible         */
  aggrow  = Bitand(flag,'08'x)="08"x      /* dynamic grow           */
  aggmov  = Bitand(flag,'04'x)="04"x      /* aggrmove               */
```

```
  qusced  = Bitand(flag,'01'x)="01"x      /* quiesced               */
  flag2 = Substr(aggr_status,20,1)
  dsable  = Bitand(flag2,'80'x)="80"x     /* disabled               */
  splxaw  = Bitand(flag2,'40'x)="40"x     /* sysplex-aware          */
  If splxaw Then xqrs = "S"; Else xqrs = "-"
  If rdonly Then xqrs = "R"||xqrs; Else xqrs = "-"||xqrs
  If qusced Then xqrs = "Q"||xqrs; Else xqrs = "-"||xqrs
  If aggrow Then xqrs = "X"||xqrs; Else xqrs = "-"||xqrs
  num_8K_blks_total = C2d(Substr(aggr_status,33,4))/8
  num_8K_blks_used = num_8K_blks_total - C2d(Substr(aggr_status,57,4))/8
  num_8K_blks_quote = num_8K_blks_used/num_8K_blks_total
  num_8K_blks_pcntg = Format(100*num_8K_blks_quote,3,1)||"%"
  If no_hdrlines Then Do
    Say
    Say Left("Aggregate name",44) Left("8K blk tot",10),
      Left("8K blk use",10) "Percent  XQRS"
    Say Copies("-",44) Copies("-",10) Copies("-",10) Copies("-",7) "",
      "----"
    no_hdrlines = 0
  End
  Say Left(aggr_name,44) Right(num_8K_blks_total,10),
    Right(num_8K_blks_used,10) "="||num_8K_blks_pcntg "" xqrs
Return

Final_Exit:
  Parse Arg final_rc
Exit final_rc

Syntax:
  Say Errortext(rc) "in line" sigl "- Rc("||rc||")"
  Say "This is probably caused by invalid data that was received."
  Call Final_Exit retc
Exit 999

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  If rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then not_OK = 0
  Else not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  If get_size_needed & errno="91" & errnojr="EF176274" Then Do
    not_OK = 0
    get_size_needed = 0
  End
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
    errno_save = errno
    errnojr_save = errnojr
```

```
    If errno<>"A3" & errno<>"A4" & is_omvs_range Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Say "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Say "OMVS Reason Code Explanation -" err.se_reason
    End
    errno = errno_save
    errnojr = errnojr_save
    If exit_on_error Then Call Final_Exit retc
  End
Return
```

B.15 MOVEAGGR procedure

The REXX procedure MOVEAGGR enables you to overcome the problems that might be encountered when you take backups of zFS aggregates, if the backup job is run on a system that is not the owner of a mounted zFS file system. As long as the zfsadm interface is not working sysplex-wide (this means that not all the involved systems are running at least at z/OS V1R7 level), a **`zfsadm quiesce`** command can only work locally on the system that owns the aggregate.

Important: This utility MOVEAGGR is no longer needed because all your systems in the UNIX System Services sysplex sharing environment should be running at level z/OS V1R7 or later. We will probably remove this utility the next time this book is updated.

B.15.1 Usage rules and benefits of MOVEAGGR

This procedure is designed to be run from a job before and after taking UNIX System Services file system backups. It has the following functions:

- When called before doing a backup, a test is done to determine whether it is of type zFS or not. If it is not a zFS file system, no further action is taken.
- If it is a zFS file system, a further test is done to see whether the local system is the owner of the zFS aggregate. If the answer is `No`, it is moved to the local system. (This works if the aggregate is a compat mode aggregate and no R/O clone is mounted at that time. This is a known restriction for zFS in a UNIX System Services sysplex sharing group.)
- When the backup is done, if it is a zFS file system and the original owner is different from the local system, the zFS aggregate is moved back to that original system.

The procedure is called with parameter GET before performing the backup, and with PUT after performing the backup. This can be seen from the JCL samples provided in C.8, “Comparison in UNIX System Services sysplex sharing environments” on page 476.

B.15.2 MOVEAGGR code

Example B-14 displays the contents of MOVEAGGR.

Example: B-14 MOVEAGGR REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: MOVEAGGR                                             */
/*  Description: Move compat mode aggregate to local system or back   */
/*               Property of IBM  (C) Copyright IBM Corp. 2004-2005   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: moveaggr fsname < Get | Put >                        */
/**********************************************************************/

Trace O
Parse Source . . myname .
Parse Upper Arg fsname get_put .
get_put = Left(get_put,1)
Select
  When get_put="G" Then function = "GET"
  When get_put="P" Then function = "PUT"
  Otherwise Do
    Say "Syntax :" myname "fsname < Get | Put >"
    Say
    Say "fsname : zFS compat mode aggregate name to move"
    Say "Get    : Move zfs aggregate to the local system if needed"
    Say "Put    : Move zfs aggregate back to the original system"
    Exit 4
  End
End

If Sysvar("SYSENV")="FORE" Then Do
  Say "This procedure is designed to run in a specific batch job only."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "The SYSCALL environment could not be established."
  Exit 8
End

no_msgs = 1
noexit_on_error = 1
switched = O

Call Syscall_Cmd "geteuid"
cur_euid = retval
If cur_euid<>O Then
  Call Syscall_Cmd "seteuid O", noexit_on_error, no_msgs

Call Syscall_Cmd "statfs (fsname) st.", noexit_on_error, no_msgs
If errno="79" & errnojr="567OO2E" Then Do
  Say "The file system is not mounted currently."
  Exit O
End

devno = st.stfs_fsid
Call Syscall_Cmd "getmntent mnt." devno
cur_sysname = Strip(mnt.mnte_sysname.1)
If mnt.mnte_fstype.1<>"ZFS" Then Do
```

```
  Say "The file system is not a zFS, no action is needed."
  Exit 0
End

If function="GET" Then sysname = MVSVar("SYSNAME")
Else Do
  "EXECIO 1 DISKR ZFSOWNER (STEM ZFSOWNER. FINIS"
  If rc<>0 Then Do
    Say "The name of the original owning system could not be retrieved."
    Exit 8
  End
  Parse Var zfsowner.1 sysname .
End

If sysname="" Then Do
  Say "The name of the original owning system is empty by mistake."
  Exit 8
End

If cur_sysname<>sysname Then Do
  m. = ""
  m.mnte_fsname = fsname
  m.mnte_sysname = sysname
  m.mnte_rflags = 1
  Call Syscall_Cmd "mount m."
  Say fsname "has been moved from" cur_sysname "to" sysname||"."
End
Else Say fsname "is mounted on" sysname "already."
If function="GET" Then Do
  zfsowner.1 = cur_sysname
  zfsowner.0 = 1
  "EXECIO 1 DISKW ZFSOWNER (STEM ZFSOWNER. FINIS"
  If rc<>0 Then Do
    Say "The name of the original system could not be saved."
    Exit 8
  End
End

Exit 0

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(2OFF)
    errno_save = errno
```

```
    errnojr_save = errnojr
    If errno<>"A3" & errno<>"A4" & is_omvs_range Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Say "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Say "OMVS Reason Code Explanation -" err.se_reason
    End
    errno = errno_save
    errnojr = errnojr_save
    If exit_on_error Then Exit 8
  End
Return
```

B.16 RXBATCH procedure

The REXX procedure RXBATCH provides enhanced usability for jobs using BPXBATCH or BPXBATSL. It is used in several JCL samples. RXBATCH can be used as an alternative to RXZFS.

B.16.1 Usage rules and benefits of RXBATCH

Using RXBATCH to run BPXBATCH from TSO in batch mode provides the following benefits over using BPXBATCH native:

- You can use DD name STDIN for STDIN data as desired. You do not and should not use DD name STDPARM and do not have to provide the data as one long UNIX command.
- It allows you to use BPXBATSL with parameter SH to process a login shell if you are a BPX.SUPERUSER.

  This allows you to work with DD names in UNIX commands because you stay in the same ASID.
- You can code your UNIX STDIN data using your preferred code page. This makes it easier to write the commands, and they are much more readable.
- You can use the backslash (\) to continue a command on the following line even if you are using record format FB for your JCL library.
- The combination of the preceding benefits allows you to use MVS data sets or members, created in your code page as `/bin/sh` scripts for usage as STDIN data.

  Doing so (and different from using the STDPARM DD name), you can switch to other subshells or interactive command processors, such as using **`su`**, **`su -s`**, or the interactive **`rexx`**.

The procedure supports a set of four RXBATCH variables to set some specific control switches for processing. Note that these settings are provided with DD name STDENV, but are not activated as UNIX environment variables.

_RXBATCH_SWSU Do (value 1) or do not (value 0) try to switch to superuser mode. The default is 0 (do not switch).

_RXBATCH_LOGIN	Use a login shell (value 1) or do not (value 0). The default is 1.
_RXBATCH_SL	Use BPXBATSL if running in superuser mode (value 0) or use it anyway (value 1). The default is 0.
_RXBATCH_CP	Specify a code page that all the STDIN data is created in. The default code page is IBM-1047.

"Create the zFS aggregate" on page 476 or "Mount the file systems" on page 477 provides an example that shows how to use this routine.

B.16.2 RXBATCH code

Example B-15 displays the contents of RXBATCH.

Example: B-15 RXBATCH REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: RXBATCH V1.2                                         */
/*  Description: Run UNIX commands in batch mode (TSO batch)          */
/*               Property of IBM  (C) Copyright IBM Corp. 2009-2010   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: rxbatch                                              */
/* ------------------------------------------------------------------ */
/* Thanks for functional extensions, suggestions and corrections to   */
/*               Manfred Lotz (manfred.lotz@de.ibm.com)               */
/* ------------------------------------------------------------------ */
/* Error corrections and new functions added to initial base version: */
/* ------------------------------------------------------------------ */
/*  - Fixed DD:STDIN problem in case of stdin_cp being IBM-1047 (set  */
/*    or by default) -- ML V1.1                                       */
/*  - Introduction of DD:RXBPARM for possible separation of pseudo    */
/*    envvars from real environment variables -- ML V1.1              */
/*  - Allowing free combination of DD:RXBPARM (only for pseudo        */
/*    envvars) and DD:STDENV (pseudo and real envvars) or even none   */
/*    (taking all defaults) -- RH V1.1                                */
/*  - Only set home directory to "/" in super user mode if defined    */
/*    home directory is not available (if needed set envvar HOME to   */
/*    "/" via DD:STDENV); set HOME to "/" in all cases if set home    */
/*    directory does not exist - RH V1.1                              */
/*  - Added 'Call Syscalls "OFF"' at the end of processing in order   */
/*    to avoid possible SA03 abends - RH V1.2                         */
/**********************************************************************/

Trace O
Parse Source . . myname .
switched = 0
final_rc = 0
no_msgs = 1
noexit_on_error = 1

swsu = 0               /* do not try to switch to superuser mode      */
login_shell = 1        /* use a login shell by default                */
use_rxbatsl = 0        /* use BPXBATSL only if in superuser mode      */
stdin_cp = "IBM-1047"  /* STDIN data is composed in IBM-1047          */

If Sysvar("SYSENV")="FORE" Then Do
```

```
  Say "This procedure is not supported in foreground."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Call RxSay "The SYSCALL environment could not be established."
  Call Final_Exit 8
End

/* ---------------------------------------------------------------- */
/*  Read in all STDENV and STDIN lines, free DDs and reallocate them  */
/* ---------------------------------------------------------------- */

Call Bpxwdyn "INFO DD(RXBPARM)"
If Pos(Left(Right(D2x(result),8,"0"),4),"0438 0440")<>0 Then Do
  /* IKJ56247I FILE INFO-RETRIEVAL NOT PERFORMED, NOT ALLOCATED */
  rxbvar.0 = 0
End
Else Do
  "EXECIO * DISKR RXBPARM (STEM RXBVAR. FINIS"
  If rc<>0 Then Do
    Call RxSay "Error on reading pseudo environment variables from",
      "RXBPARM DD name, Rc="||rc
    Call Final_Exit 8
  End
End

Call Bpxwdyn "INFO DD(STDENV) MSG(RXINFO.)"
If Pos(Left(Right(D2x(result),8,"0"),4),"0438 0440")<>0 Then
  envvar.0 = 0
Else Do
  "EXECIO * DISKR STDENV (STEM ENVVAR. FINIS"
  If rc<>0 Then Do
    Call RxSay "Error on reading environment variables from initial",
      "STDENV DD name, Rc="||rc
    Call Final_Exit 8
  End
  Call Bpxwdyn "FREE DD(STDENV) MSG(WTP)"
  If result<>0 Then Do
    Call RxSay "Error occurred on freeing initial STDENV DD, Rc="||resul
    Call Final_Exit 8
  End
End

Call Syscall_Cmd "realpath /tmp tmp_dir"
stdenv_file = tmp_dir || "/RXBATCH."||Userid()||".STDENV." ||,
  DelStr(Delstr(Time("L"),3,1),5,1)
Call Bpxwdyn "ALLOC DD(STDENV) MSG(WTP) FILEDATA(TEXT)",
  "PATH('"||stdenv_file||"') PATHDISP(DELETE,DELETE)",
  "PATHMODE(SIRUSR,SIWUSR) PATHOPTS(OEXCL,OCREAT,ORDONLY)"
If result<>0 Then Do
  Call RxSay "Error occurred on allocating a new STDENV file,",
    "Rc="||result
  Call Final_Exit 8
End
```

```
"EXECIO * DISKR STDIN (STEM STDIN. FINIS"
If rc<>0 Then Do
  Call RxSay "Error on reading complete data from initial STDIN DD",
    "name, Rc="||rc
  Call Final_Exit 8
End

Call Bpxwdyn "FREE DD(STDIN) MSG(WTP)"
If result<>0 Then Do
  Call RxSay "Error occurred on freeing initial STDIN DD name,",
    "Rc=" result
  Call Final_Exit 8
End

stdin_file = tmp_dir || "/RXBATCH."||Userid()||".STDIN." ||,
  DelStr(Delstr(Time("L"),3,1),5,1)
Call Bpxwdyn "ALLOC DD(STDIN) MSG(WTP) FILEDATA(TEXT)",
  "PATH('"||stdin_file||"') PATHDISP(DELETE,DELETE)",
  "PATHMODE(SIRUSR,SIWUSR) PATHOPTS(OEXCL,OCREAT,ORDONLY)"
If result<>0 Then Do
  Call RxSay "Error occurred on allocating a new STDIN file,",
    "Rc=" result
  Call Final_Exit 8
End

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars                   */
/* ---------------------------------------------------------------- */

envvars = 6
rxbparms = rxbvar.0
stdenvs = envvar.0
is_pseudo = 1
is_stdenv = 0
Do i=1 To rxbparms+stdenvs
  If i=rxbparms+1 Then Do
    is_pseudo = 0
    is_stdenv = 1
  End
  If is_pseudo Then envvar_line = Strip(rxbvar.i)
  Else Do
    j = i-rxbparms
    envvar_line = Strip(envvar.j)
  End
  If envvar_line="" | Left(envvar_line,1)="#" Then Iterate i
  pos_equal = Pos("=",envvar_line)
  If pos_equal=0 Then Iterate i
  Select
    When Left(envvar_line,pos_equal-1)="_RXBATCH_SWSU" Then Do
      Parse Value Substr(envvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test=0 Then swsu = 0
        When opt_test=1 Then swsu = 1
        Otherwise Nop /* invalid value, simply ignore it */
```

```
      End
    End
    When Left(envvar_line,pos_equal-1)="_RXBATCH_LOGIN" Then Do
      Parse Value Substr(envvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test=0 Then login_shell = 0
        When opt_test=1 Then login_shell = 1
        Otherwise Nop /* invalid value, simply ignore it */
      End
    End
    When Left(envvar_line,pos_equal-1)="_RXBATCH_SL" Then Do
      Parse Value Substr(envvar_line,pos_equal+1) With opt_test .
      Select
        When opt_test=0 Then use_rxbatsl = 0
        When opt_test=1 Then use_rxbatsl = 1
        Otherwise Nop /* invalid value, simply ignore it */
      End
    End
    When Left(envvar_line,pos_equal-1)="_RXBATCH_CP" Then Do
      Parse Value Substr(envvar_line,pos_equal+1) With opt_test .
      If opt_test<>"" Then stdin_cp = opt_test
    End
    When is_pseudo Then Nop /* only pseudo envvars used from RXBPARM */
    Otherwise Do
      envvars = envvars+1
      __environment.envvars = envvar_line
    End
  End /* Select */
End i
__environment.0 = envvars

If swsu Then Do /* Switch to SU mode if possible */
  Call Syscall_Cmd "geteuid"
  cur_euid = retval
  Call Syscall_Cmd "getuid"
  cur_uid = retval
  If cur_euid<>0 | cur_uid<>0 Then Do
    Call Syscall_Cmd "setreuid 0 0", noexit_on_error, no_msgs
    If OK Then switched = 1
  End
End
Call Syscall_Cmd "getpwnam" Userid() "omvs."
home = omvs.pw_dir
Call Syscall_Cmd "getcwd cur_cwd", noexit_on_error, no_msgs
If not_OK | cur_cwd="" Then Do
  Call Syscall_Cmd "chdir" home, noexit_on_error, no_msgs
  If not_OK Then Do
    home = "/"
    Call Syscall_Cmd "chdir" home
  End
End
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
```

```
__environment.5 = "PWD="||home
__environment.6 = "_EDC_ADD_ERRNO2=1"
Call Syscall_Cmd "writefile (stdenv_file) 600 __environment."

/* ---------------------------------------------------------------- */
/*  Strip STDIN trailing blanks, convert if needed, write new STDIN */
/* ---------------------------------------------------------------- */

Do i=1 To stdin.0
  If stdin_cp<>"IBM-1047" Then stdins.i = Strip(stdin.i,"T")
  Else stdin.i = Strip(stdin.i,"T")
End
If stdin_cp<>"IBM-1047" Then Do
  stdins.0 = stdin.0
  Drop stdin.
  Call Bpxwunix "iconv -f" stdin_cp "-t IBM-1047", "STDINS.", "STDIN.",,
    "STDERR."
  retc = result
  If retc<>0 Then Do
    Do i=1 To stderr.0
      Call RxSay stderr.i
    End
    Call RxSay "Error on converting STDIN data lines to IBM-1047, Rc=",
      retc
    Call Final_Exit 8
  End
End

Call Syscall_Cmd "writefile (stdin_file) 600 stdin."

/* ---------------------------------------------------------------- */
/*  Run the commands using BPXBATCH or BPXBATSL                     */
/* ---------------------------------------------------------------- */

If cur_euid=0 & cur_uid=0 | switched | use_rxbatsl Then
  batch_module = "BPXBATSL"
Else
  batch_module = "BPXBATCH"
If login_shell Then batch_parm = "SH"
Else batch_parm = "PGM /bin/sh"
batch_module batch_parm
final_rc = (rc<>0)*8
bpx_rc = rc
If rc//256=0 Then uss_stat = rc/256
Else uss_stat = -1
If rc<>0 Then Do
  Call RxSay ""
  Call RxSay "BPXBATCH processing ended with RC=" bpx_rc
  If uss_stat>0 Then
    Call RxSay "Last UNIX status retrieved is RC=" uss_stat
End

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */
```

```
Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit: Trace O
  Parse Arg final_rc
  Call Syscalls "OFF"
Exit final_rc

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Call RxSay "SYSCALL Service:" syscall_cmd
    Call RxSay "Syscall Return Code=" rc
    Call RxSay "OMVS Return Value =" retval
    Call RxSay "OMVS Return Code =" errno
    Call RxSay "OMVS Reason Code =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
    errno_save = errno
    errnojr_save = errnojr
    If errno<>"A3" & errno<>"A4" & is_omvs_range Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Call RxSay "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Call RxSay "OMVS Reason Code Explanation -" err.se_reason
    End
    errno = errno_save
    errnojr = errnojr_save
    If exit_on_error Then Call Final_Exit 8
  End
Return

RxSay: Trace O
  Parse Arg message.1
  "EXECIO 1 DISKW STDOUT (STEM MESSAGE."
  If rc<>0 Then Do
    Say message.1
    Say "Error on writing a message to the STDOUT DD name, Rc="||rc
  End
Return
```

B.17 REXX

REXX procedure REXX is supported in ISPF/TSO (using command **tso rexx**) and UNIX System Services shell environments (using **rexx**). It allows you to run TSO, SYSCALL, UNIX System Services Shell and REXX commands interactively. It was originally provided with the USSTools package. We provide the most current version here for your reference.

B.17.1 Usage rules and benefits of REXX

If you enter an empty or blank line in interactive mode, help information is displayed describing the usage as follows.

The REXX command input syntax is: `[ prefix ] cmd_input`

The prefix can be specified as one of the following:

SH	The input data cmd_input is run as a UNIX command.
Syscall	The input data cmd_input is run as a SYSCALL command.
TSO	The input data cmd_input line is run as a TSO command.
MVS	The command is run with addressing environment MVS.

If you do not specify a prefix, then the command input is only interpreted by REXX. The command input data may contain several commands, separated by a semi-colon (;**)**. You can extend the amount of data to multiple lines by appending a dash (**-**) at the end of the current line.

Important: You must use an explicit **Exit** statement to end processing. This is needed especially in batch mode; otherwise, the REXX will not stop.

B.17.2 REXX code

Example B-16 displays the contents of REXX.

Example: B-16 REXX REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: REXX                                                 */
/*  Description: Run REXX commands interactively...                   */
/*               Property of IBM  (C) Copyright IBM Corp. 1998-2009   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: rexx < only_one_cmd >                                */
/**********************************************************************/

Trace O

no_rexx_func = 1
If MVSVar("SYSMVS")<"SP7.0.4" Then Do
  Signal On Syntax Name No_REXX_Func
  Call Bpxwunix ""
End
no_rexx_func = 0
No_REXX_Func:
Signal Off Syntax
```

```
Parse Arg first_cmd, sec_parm, thr_parm
Parse Source . call_type name . . . . omvs .
omvs = (omvs="OMVS")
not_omvs = (omvs=0)
not_sysc = 0
If omvs Then Do
  Address SYSCALL "ttyname 2 stderr"
  If Left(stderr,5)="/dev/" Then foreground = 1
  Else foreground = 0
End
Else foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
If not_omvs Then Do
  If Syscalls("ON")>4 Then not_sysc = 1
End
only_once = (first_cmd<>"")
disp_retc = (call_type="COMMAND") | ^only_once
Call Run_Loop
Exit 998

Run_Loop:
  Signal On Syntax
  Do Forever
    If only_once Then cmdline = first_cmd
    Else Do
      If foreground Then Do
        rexx_prompt = Address()||"> "
        If omvs Then Do
          prmpt_cont = Right("> ",Length(rexx_prompt))
          Address SYSCALL "write 2 rexx_prompt"
        End
        Else Do
          prmpt_cont = ">"
          Say rexx_prompt
        End
      End
      cmdline = ""
      Do Forever
        Parse External cl_line
        If cmdline="" Then cl_line = Strip(cl_line,"T")
        Else cl_line = Strip(cl_line,"B")
        If Right(cl_line,1)="-" | Right(cl_line,1)="," Then Do
          cmdline = cmdline || Substr(cl_line,1,Length(cl_line)-1)
          If foreground Then Do
            If omvs Then Address SYSCALL "write 2 prmpt_cont"
            Else Say prmpt_cont
          End
        End
        Else Do
          cmdline = cmdline || cl_line
          Leave
        End
      End
    End
    rc = 0
```

```
If background Then Say "Rx>" cmdline
Parse Var cmdline env env_cmd
Upper env
Select
  When cmdline="" & foreground Then Do
    Say ""
    Say "REXX command input syntax: < prefix > cmd_input"
    Say ""
    Say "The prefix can be one of the following:"
    Say "SH      - the input line is run as a UNIX command."
    Say "Syscall - the input line is run as a SYSCALL command."
    Say "TSO     - the input line is run as a TSO command."
    Say "MVS     - the command is run with addressing environment",
      "MVS."
    Say "Without a prefix the command input is only interpreted",
      "by REXX."
    Say ""
    Say "The command input data may contain several commands,",
      "separated by "";"". You can"
    Say "extend the amount of data to multiple lines by",
      "appending a dash (-) at the end"
    Say "of the current line."
    Say ""
    Say "Important: You must use an explicit ""Exit"" statement to",
      "end processing. This is needed"
    Say "especially in batch mode; otherwise the REXX will not",
      "stop."
    Say ""
  End
  When Abbrev("SYSCALL",env,1) & not_sysc Then
    Say "Environment ""SYSCALL"" is not available."
  When Abbrev("SYSCALL",env,1) & env_cmd="" Then Nop
  When Abbrev("SYSCALL",env,1) Then Do
    Interpret "Address SYSCALL" env_cmd
    not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
    OK = (not_OK=0)
    Select
      When rc>=0 Then Do
        Say "OMVS Return Value (retval) =" retval
        If errno<>0 | errnojr<>0 Then Do
          Say "OMVS Return Code (errno )  =" errno
          Say "OMVS Reason Code (errnojr) =" errnojr
          errno_save = errno
          errnojr_save = errnojr
          is_omvs_range =,
            X2d(Left(Right(errnojr,8,"0"),4))<=X2d(2OFF)
          syscall_rc = rc
          If errno<>"A3" & errno<>"A4" & is_omvs_range Then
            show_reason = 1
          Else
            show_reason = 0
          Address SYSCALL "strerror" errno errnojr "err."
          If rc=0 & retval>=0 Then Do
            If err.se_errno<>"" Then
              Say "OMVS Return Code Explanation -" err.se_errno
```

```
                If show_reason & err.se_reason<>"" Then
                  Say "OMVS Reason Code Explanation -" err.se_reason
              End
              errno = errno_save
              errnojr = errnojr_save
            End
          End
          When rc=-20 Then Say "Command not recognized or improper",
            "number of parameters specified..."
          When rc<-20 Then Say "Parameter number" (-20-rc) "of the",
            "SYSCALL command is in error."
          When rc<0 Then Say "The command did not finish successfully."
          Otherwise Nop
        End
        If not_OK & rc=0 Then rc = -1
        If OK & rc>0 Then rc = 0
      End
      When env="SH" & not_omvs Then Do
        Say "Environment ""SH"" is not supported from TSO/ISPF."
        rc = -1
      End
      When env="SH" Then Interpret "Address SH" env_cmd
      When env="TSO" & omvs & no_rexx_func Then Do
        Say "Environment ""TSO"" is not supported here from the Shell."
        Say "Please use: sh ""tso ..."""."
        rc = -1
      End
      When env="TSO" Then Interpret "Address TSO" env_cmd
      When env="MVS" Then Interpret "Address MVS" env_cmd
      When Abbrev("REXX",env) Then Interpret cmdline
      Otherwise Interpret cmdline
    End
    If rc<>0 & disp_retc Then Say "Rc("||rc||")"
    If only_once Then Call Final_Exit rc
  End
Return

Syntax:
  Say Errortext(rc)
  Say "Rc("||rc||")"
  If only_once Then Call Final_Exit rc
  Else Call Run_Loop
Exit 999

Final_Exit:
Parse Arg final_rc
If final_rc="" Then final_rc = 0
Exit final_rc
```

B.18 RXDOWNER and ZFSOWNER

REXX procedures RXDOWNER and ZFSOWNER can be used to display owner-related information about UNIX System Services file systems and especially zFS aggregates. With

zFS they use using several commands and operation codes of the **pfsctl** interface. They are supported in ISPF/TSO (using the commands **tso rxdowner** and **tso zfsowner**) and UNIX System Services shell environments (using **rxdowner** and **zfsowner**).

B.18.1 Usage rules and benefits of RXDOWNER

The syntax to run the RXDOWNER routine is as follows:

```
rxdowner -d uss_direntry | -f file_system | -a zfs_aggrname
```

Figure 7-14 Syntax of routine rxdowner

The parameters appended with a blank in between after the possible options have the following meaning:

uss_direntry This is a UNIX System Services file system directory entry.

file_system This is the name a currently mounted UNIX System Services file system.

zfs_aggrname This the name of a currently active zFS aggregate.

As a result, the owner and owner-related information is displayed.

B.18.2 Usage rules and benefits of ZFSOWNER

The syntax to run the ZFSOWNER routine is as follows:

```
zfsowner zfs_aggrname
```

Figure 7-15 Syntax of routine zfsowner

The parameter has the following meaning:

zfs_aggrname This the name of a currently active zFS aggregate.

As a result, the owner and owner-related information for the aggregate is displayed.

Important: After copying **RXDOWNER** into a REXX library for use in TSO and into a UNIX directory named **rxdowner**, you simply define **ZFSOWNER** as an ALIAS in the REXX library and **zfsowner** as a hard link of rxdowner in UNIX.

B.18.3 RXDOWNER code

Example B-17 displays the contents of RXDOWNER.

Example: B-17 RXDOWNER REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: rxdowner V1.1                                        */
/*  Description: List owner and further information for a USS file    */
/*               system, zFS aggregate or a USS directory entry       */
/*               Property of IBM  (C) Copyright IBM Corp. 2009        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*  Format 1 is: rxdowner rx_option rx_argument                       */
/*  Format 2 is: zfsowner zfs_aggrname                                */
```

```
/*********************************************************************/

Trace O
Parse Source . . myname . . . . omvs .
myname = Substr(myname,Lastpos("/",myname)+1)
omvs = (omvs="OMVS")
If omvs Then retc = 1
Else Do
  retc = 8
  If Syscalls("ON")>4 Then Do
    Say "The SYSCALL environment could not be established."
    Exit retc
  End
End

If MVSVar("SYSMVS")<"SP7.0.7" Then Do
  Say "The level of the BCP must be at z/OS V1R7 or higher to use this",
    "function."
  Exit retc
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
get_size_needed = 0
is_fsn = 0
is_agg = 0
is_uss = 0

If Translate(myname)="ZFSOWNER" Then Do
  Parse Upper Arg zfs_aggrname junk
  If zfs_aggrname="" | junk<>"" Then Do
    Say ""
    Say "Syntax:" myname "zfs_aggrname"
    Say ""
    Say "Parameter ""zfs_aggrname"" is the name of a currently active",
      "zFS aggregate. As a"
    Say "result owner and owner related information for the aggregate",
      "is displayed."
    Say ""
    Call Final_Exit retc
  End
  Else Do
    rx_opt = "-a"
    rx_argument = zfs_aggrname
  End
End
Else Parse Arg rx_opt rx_argument

Parse Var rx_argument . junk
rx_opt = Translate(rx_opt)
Select
  When rx_opt="-F" & rx_argument<>"" & junk="" Then Do
    is_fsn = 1
    Parse Var rx_argument file_system .
```

```
  End
  When rx_opt="-A" & rx_argument<>"" & junk="" Then Do
    is_agg = 1
    Parse Upper Var rx_argument zfs_aggrname .
  End
  When rx_opt="-D" & rx_argument<>"" Then Do
    is_uss = 1
    uss_direntry = Strip(rx_argument)
  End
  Otherwise Do
    Say ""
    Say "Syntax:" myname "-d uss_direntry | -f file_system | -a",
      "zfs_aggrname"
    Say ""
    Say "Parameter ""uss_direntry"" is an USS file system directory",
      "entry, ""file_system"""
    Say "is the name a currently mounted USS file system and",
      """zfs_aggrname"" is the name"
    Say "of a currently active zFS aggregate. As a result appropriate",
      "owner and owner"
    Say "related information is displayed."
    Say ""
    Call Final_Exit retc
  End
End /* Select */

Select
  When is_uss Then Do
    Call Syscall_Cmd "stat (uss_direntry) st.", no_exit_on_error
    If not_OK Then Do
      Say "USS directory entry """||uss_direntry||""" cannot be found."
      Call Final_Exit retc
    End
    devno = X2d(st.st_dev)
  End
  When is_fsn Then Do
    Call Syscall_Cmd "statfs (file_system) st.", no_exit_on_error
    If not_OK Then Do
      Say "The file system" file_system "cannot be found to be mounted."
      Call Final_Exit retc
    End
    devno = st.stfs_fsid
  End
  Otherwise Nop /* is_agg */
End

If is_uss | is_fsn Then Do
  lcl_sysname = MVSVar("SYSNAME")
  Call Syscall_Cmd "getmntent mnt." devno
  mnt_dirname = mnt.mnte_path.1
  mnt_fsname = Strip(mnt.mnte_fsname.1)
  mnt_sysname = Left(mnt.mnte_sysname.1,8)
  mnt_aggrname = Strip(mnt.mnte_aggrname.1)
  mnt_fstype = Left(mnt.mnte_fstype.1,8)
  mnt_mode = D2c(mnt.mnte_mode.1,4)
```

```
  mnt_rdonly = D2c(mnt_mode_rdonly,4)
  If Bitand(mnt_mode,mnt_rdonly)=mnt_rdonly Then mnt_rdonly_info = "Y"
  Else mnt_rdonly_info = "N"
  mnt_client = D2c(mnt_mode_client,4)
  If Bitand(mnt_mode,mnt_client)=mnt_client Then mnt_client_info = "Y"
  Else mnt_client_info = "N"

  Say ""
  Say "MP Directory :" mnt_dirname
  Say "File System  :" mnt_fsname
  If mnt_fstype="ZFS" Then Do
    /* ---------------------------------------------------------- *
    Say "Aggregate    :" mnt_aggrname  - remove this information *
     * ---------------------------------------------------------- */
    zfs_aggrname = mnt_aggrname
  End
  Say "PFS Type     :" mnt_fstype
  Say "Local Sysname:" Left(lcl_sysname,8) "- File System",
    "local-client=" || mnt_client_info
  Say "USS Owner    :" mnt_sysname "- File System read-only=" ||,
    mnt_rdonly_info
  If mnt_fstype<>"ZFS" Then Do
    Say ""
    Call Final_Exit 0
  End
End

Signal On Syntax
Numeric Digits 10
not_zfsaggr_found = 1

zero_parm = D2c(0,4)
zfs_cmd = X2d("40000005")/* ZFSCALL_AGGR */
op_code = D2c(140,4) /* Listing attached aggregate names (Version 2) */
parm_len = 32
aggr_struct_size = 84 /* length of aggregate structure */
parm_3 = zero_parm /* not used */
parm_4 = zero_parm /* not used */
parm_5 = zero_parm /* not used */
parm_6 = zero_parm /* not used */
parm_3456 = parm_3 || parm_4 || parm_5 || parm_6
size_needed = zero_parm /* size needed for buffer, set to zero */

parm_0 = zero_parm /* no buffer is provided */
parm_1 = zero_parm /* no buffer is provided */
parm_2 = D2c(parm_len,4) /* offset to size */
zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
  size_needed
get_size_needed = 1
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
buf_size = C2d(Substr(zfs_buf,parm_len+1,4))
If buf_size<>0 Then Do
  parm_0 = D2c(buf_size,4) /* buffer length */
  parm_1 = D2c(parm_len,4) /* offset to first aggregate structure */
  parm_2 = D2c(buf_size+parm_len,4) /* offset to size */
```

```
  buffer = Left("",buf_size) /* buffer */
  zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
    buffer || size_needed
  Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
  buf_size = C2d(Substr(zfs_buf,buf_size+parm_len+1,4))
End
aggrs = buf_size/aggr_struct_size
aggr_off = parm_len
Do i=1 To aggrs
  aggr_struct = Substr(zfs_buf,aggr_off+1,aggr_struct_size)
  aggr_eyecat = Left(aggr_struct,4)
  aggr_len = C2d(Substr(aggr_struct,5,1)) /* "84", not testet */
  aggr_ver = C2d(Substr(aggr_struct,6,1)) /* "2", not testet */
  If aggr_eyecat<>"AGID" Then Do
    Say ""
    Say "Unexpected data returned for current aggregate entry..."
    Call Final_Exit retc
  End
  aggr_nam = Strip(Substr(aggr_struct,7,45),"T","00"x)
  If aggr_nam=zfs_aggrname Then Do
    aggr_sys = Strip(Substr(aggr_struct,52,9),"T","00"x)
    not_zfsaggr_found = 0
    Leave i
  End
  aggr_off = aggr_off+aggr_struct_size
End
If not_zfsaggr_found Then Do
  Say ""
  Say "Aggregate" zfs_aggrname "cannot be found." /* Error message */
  If final_rc=0 Then final_rc = retc
  Call Final_Exit final_rc
End

op_code = D2c(146,4) /* List aggregate status (Version 2) */
aggr_id = "AGID" || "0001"x || Left(zfs_aggrname,45,"00"x) ||,
  Copies("00"x,33)
aggr_id = Overlay(D2c(Length(aggr_id),1),aggr_id,5)
aggr_len = Length(aggr_id)
aggr_status = "AGST" || "00000200"x || Copies("00"x,164)
aggr_status = Overlay(D2c(Length(aggr_status),2),aggr_status,5)
parm_0 = D2c(parm_len,4) /* offset to aggr_id */
parm_1 = D2c(parm_len+aggr_len,4) /* offset to aggr_status2 */
parm_2 = zero_parm /* not used next */
zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
  aggr_id || aggr_status
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
aggr_status = Substr(zfs_buf,parm_len+aggr_len+1)
If Left(aggr_status,4)<>"AGST" Then Do
  Say ""
  Say "Unexpected data for aggregate" aggr_name||"..."
  Call Final_Exit retc
End
flag = Substr(aggr_status,19,1)
is_rdonly = Bitand(flag,"40"x)="40"x        /* read only             */
is_compat = Bitand(flag,"10"x)="10"x        /* HFS compatible        */
```

```
flag2 = Substr(aggr_status,20,1)
is_sysplexaware = Bitand(flag2,"40"x)="40"x /* sysplex-aware state    */

/* This works at all z/OS releases and zFS R11 service levels based   *
 * on the fact that zFS R11 base GA level includes APAR OA29607, PTF  *
 * UA49766 for release 3B0 providing the correct sysplex-aware flag   *
 * for zFS R11 and simply results in zero for all old zFS levels      */

zfs_cmd = X2d("40000006")/* Configuration command code ZFSCALL_CONFIG */
op_code = D2c(215,4)     /* Subcommand: Query sysplex_state (215)     */
parm_0 = D2c(parm_len,4) /* offset to cfg_option */
parm_2 = zero_parm /* not used */
co_eye = "CFOP"
co_len = D2c(0,2)
co_ver = "01"x
co_slen = 80
co_soff = Length(co_eye||co_len||co_ver)
co_string = Copies("00"x,co_slen+1)
co_value_reserved = Copies(zero_parm,4)
co_reserved = D2c(0,24)
cfg_option = co_eye || co_len || co_ver || co_string ||,
  co_value_reserved || co_reserved
cfg_optlen = Length(cfg_option)
cfg_option = Overlay(D2c(cfg_optlen,2),cfg_option,Length(co_eye)+1)
If aggr_sys=lcl_sysname Then parm_1 = zero_parm /* no system name */
Else parm_1 = D2c(parm_len+cfg_optlen,4) /* offset to system name */
zfs_buf = op_code || parm_0 || parm_1 || parm_2 || parm_3456 ||,
  cfg_option
If parm_1<>zero_parm Then zfs_buf = zfs_buf || Left(aggr_sys,9,"00"x)
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf)
sysplex_state = Substr(zfs_buf,parm_len+co_soff+1,co_slen+1)
sysplex_state = Strip(sysplex_state,"T","00"x)
Select
  When sysplex_state=0 | is_compat=0 Then zfs_sysplex_aware = "-"
  /* left for reference to the initial system wide sysplex-awareness  *
   * When sysplex_state=2 | is_rdonly Then zfs_sysplex_aware = "Y"    *
   * ---------------------------------------------------------------- */
  When is_sysplexaware Then zfs_sysplex_aware = "Y"
  When is_rdonly Then zfs_sysplex_aware = "-"
  Otherwise /* sysplex_state=1 and/or flag=0 */ zfs_sysplex_aware = "N"
End
zFS_owner = "zFS Owner    :" Left(Strip(aggr_sys,"T","00"x),8),
  "- Aggregate read-only=" || Substr("NY",is_rdonly+1,1) || ","
If is_compat Then Nop; Else zFS_owner = zFS_owner "compat-mode=N,"
Say zFS_owner "sysplex-aware=" || zfs_sysplex_aware
/* ---------------------------------------------------------- *
Say "zFS Owner    :" Left(Strip(aggr_sys,"T","00"x),8),
  "- Aggregate read-only=" || Substr("NY",is_rdonly+1,1) || ",",
  "compat-mode=" || Substr("NY",is_compat+1,1) || ",",
  "sysplex-aware=" || zfs_sysplex_aware
 * ---------------------------------------------------------- */
If is_fsn | is_uss Then Say ""

Call Final_Exit 0
Exit 9999
```

```
Final_Exit:
  Parse Arg final_rc
Exit final_rc

Syntax:
  Say ""
  Say Errortext(rc) "in line" sigl "- Rc("||rc||")"
  Say "This is probably caused by invalid data that was received."
  Call Final_Exit retc
Exit 999

Syscall_Cmd: Trace O
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  If rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then not_OK = 0
  Else not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  If get_size_needed & errno="91" & errnojr="EF176274" Then Do
    not_OK = 0
    get_size_needed = 0
  End
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say ""
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    is_omvs_range = X2d(Left(Right(errnojr,8,"0"),4))<=X2d(20FF)
    is_zfs = (Left(Right(errnojr,8,"0"),2)="EF")
    errno_save = errno
    errnojr_save = errnojr
    If errno<>"A3" & errno<>"A4" & (is_omvs_range | is_zfs) Then
      show_reason = 1
    Else
      show_reason = 0
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      If err.se_errno<>"" Then
        Say "OMVS Return Code Explanation -" err.se_errno
      If show_reason & err.se_reason<>"" Then
        Say "OMVS Reason Code Explanation -" err.se_reason
    End
    Say ""
    errno = errno_save
    errnojr = errnojr_save
    If exit_on_error Then Call Final_Exit retc
  End
Return
```

B.19 LARGEFIL procedure (version 2)

Version 2 of the REXX procedure LARGEFIL is used in UNIX System Services file system sharing test scenarios. It is used to fill 750 MB of data into two big files in each of the file systems.

B.19.1 LARGEFIL code (version 2)

Example B-18 displays the contents of LARGEFIL (version 2).

Example: B-18 LARGEFIL REXX procedure (version 2)

```
/* REXX ***************************************************************/
/*    Procedure: LARGEFIL (version 2)                                 */
/*  Description: Create large file in a file system and fill it       */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2009   */
/*    Format is: largefil directory number                            */
/*********************************************************************/

Trace O
Parse Source . . myname .
Parse Arg directory number
If number="" Then Do
  Say "LRGFLOO1E Syntax:" myname "directory number"
  Exit 8
End

timeout_value = 0
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "LRGFLOO2E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "LRGFLOO3E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "LRGFLOO4E The SIGNAL interface could not be established."
  Exit 8
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ------------------------------------------------------------------ */
/*  Set home and working directory, setup envvars                     */
```

```
/* ---------------------------------------------------------------- */

home = "/"
Call Syscall_Cmd "chdir" directory
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.0 = 5

/* ---------------------------------------------------------------- */
/*  Enable for timer interrupts                                     */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"

/* ---------------------------------------------------------------- */
/*  Create and fill the new large file in the file system           */
/* ---------------------------------------------------------------- */
large_file = "large_file_"||number
Call Syscall_Cmd "open (large_file)" o_creat+o_excl+o_wronly 666
fd_large = retval
one_meg = 1024*1024
Say "About to filling the file with data..."
Say ""
Call Time "R"
Do i=1 To 750
  block_data = Copies(Right(i,4,0),4)  /* 16 byte */
  block_data = Copies(block_data,64)   /* 1KB     */
  block_data = Copies(block_data,1024) /* 1MB     */
  block_data = Overlay(Time("L")||"-"||Right(i,10,0)||"--",block_data)
  Call Syscall_Cmd "write (fd_large) block_data" one_meg
  If i//50=0 Then Do
    Say "Blocks written and time used in seconds up to now:" Right(i,3),
      Right(Time("E"),12)
  End
End
Call Syscall_Cmd "close (fd_large)"
Say ""
Say "Finished filling the file with data..."
Say "Time used for whole filling process:" Time("E") "sec"

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
```

```
/* ---------------------------------------------------------------- */

Final_Exit:
  Parse Arg final_rc
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.i)", noexit_on_error
  End
  If pp.1<>-1 Then Call Syscall_Cmd "close (pp.1)", noexit_on_error
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        Say ""
        Say "OMVS Return Code Explanation -"
        Say err.se_errno
        Say "OMVS Reason Code Explanation -"
        Say err.se_reason
        Say ""
      End
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

B.20 LARGEIOS procedure (version 2)

Version 2 of the REXX procedure LARGEIOS is used in the UNIX System Services file system sharing test scenarios. It performs the specified number of random I/Os to read or write 1 MB of data.

B.20.1 LARGEIOS code (version 2)

Example B-19 displays the contents of LARGEIOS (version 2).

Example: B-19 LARGEIOS REXX procedure (version 2)

```
/* REXX ***************************************************************/
/*    Procedure: LARGEIOS (version 2)                                 */
/*  Description: Do a specied amount of large random I/Os             */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2009   */
/*    Format is: largeios blk_ios type_ios seed lf_dir                */
/*********************************************************************/

Trace O
Parse Source . call_type myname . . . . omvs .
Parse Arg blk_ios type_ios seed lf_dir
Upper type_ios
If lf_dir="" Then Do
  Say "Syntax:" myname "blk_ios type_ios seed lf_dir"
  Exit 8
End

Parse Var type_ios 1 read_id 2 read_pc
If read_id<>"R" | Length(read_pc)>2 & read_pc<>100 |,
  Verify(read_pc,"1234567890")<>0 Then Do
  Say "IO type value must be set to Rxx, where xx is a number",
    "between 0 and 100."
  Exit 8
End

If omvs<>"OMVS" Then Do
  If Syscalls("ON")>4 Then Do
    Say "The SYSCALL environment could not be established."
    Exit 8
  End
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
fd = -1
one_meg = 1024*1024
minus_one_meg = (-1)*one_meg

/* ----------------------------------------------------------------- */
/*  Do the specified amount of large random I/Os                     */
/* ----------------------------------------------------------------- */

rnr.1 = Random(1,750,seed) /* 500 */
Do i=2 To blk_ios
```

```
  rnr.i = Random(1,750)    /* 500 */
End
Call Time "R"
large_file = Strip(lf_dir,"T","/") /*||"/large_file"*/
Call Syscall_Cmd "open (large_file) (o_rdwr)"
fd = retval
rs = 0
ws = 0
Do i=1 To blk_ios
  rnr = rnr.i
  file_offset = (rnr-1)*one_meg
  Call Syscall_Cmd "lseek (fd) (file_offset) (seek_set)"
  Select
    When i=1 & read_pc<50 Then io_type = "W"
    When i=1 Then io_type = "R" /* read_pc>=50 */
    When (rs/(rs+ws))*100>read_pc Then io_type = "W"
    Otherwise io_type = "R" /* (rs/(rs+ws))*100<=read_pc */
  End
  If io_type="R" Then Do
    Call Syscall_Cmd "read (fd) block_data (one_meg)"
    Say "Read  block" Right(rnr,3) Right(Time("E"),12) "-",
      Left(block_data,32)
    rs = rs+1
  End
  Else Do
    block_data = Copies(Right(rnr,4,0),4)/* 16 byte */
    block_data = Copies(block_data,64)   /* 1KB     */
    block_data = Copies(block_data,1024) /* 1MB     */
    block_data = Overlay(Time("L")||"-"||Right(i,10,0)||"--",block_data)
    Call Syscall_Cmd "write (fd) block_data" one_meg
    Say "Write block" Right(rnr,3) Right(Time("E"),12) "-",
      Left(block_data,32)
    ws = ws+1
  End
End

/* ------------------------------------------------------------------ */
/*  End of processing                                                 */
/* ------------------------------------------------------------------ */

Call Final_Exit final_rc
Exit 9999

/* ------------------------------------------------------------------ */
/*  Subroutines                                                       */
/* ------------------------------------------------------------------ */

Final_Exit:
  Parse Arg final_rc
  If fd<>-1 Then Call Syscall_Cmd "close (fd)", noexit_on_error
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
```

```
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    Say "SYSCALL Service:" syscall_cmd
    Say "Syscall Return Code=" rc
    Say "OMVS Return Value  =" retval
    Say "OMVS Return Code   =" errno
    Say "OMVS Reason Code   =" errnojr
    Address SYSCALL "strerror" errno errnojr "err."
    If rc=0 & retval>=0 Then Do
      Say ""
      Say "OMVS Return Code Explanation -"
      Say err.se_errno
      Say "OMVS Reason Code Explanation -"
      Say err.se_reason
      Say ""
    End
    If exit_on_error Then Call Final_Exit 8
  End
Return
```

B.21 LARGESCD procedure (version 2)

Version 2 REXX procedure LARGESCD is used in the UNIX System Services file system sharing test scenarios. It starts the specified number of processes that are running in parallel, and provides the needed parameters to finally collect all the statistical information.

B.21.1 LARGESCD code (version 2)

Example B-20 displays the contents of LARGESCD (version 2).

Example: B-20 LARGESCD REXX procedure (version 2)

```
/* REXX ***************************************************************/
/*    Procedure: LARGESCD (version 2)                                 */
/*  Description: Large I/O test "scheduler"                           */
/*               Property of IBM  (C) Copyright IBM Corp. 2002-2009   */
/*    Format is: largescd timeout idir procs blkios typeios seed fdir */
/*********************************************************************/

Trace O
Parse Source . . myname .
Parse Arg timeout_value rexxlib procs blkios typeios seed fdir
If timeout_value="" Then timeout_value = 0
If Verify(timeout_value,"1234567890")<>0 Then Do
  Say "LRGSCO01E The timeout value specified is invalid."
  Exit 8
End

If Syscalls("ON")>4 Then Do
  Say "LRGSCO02E The SYSCALL environment could not be established."
  Exit 8
```

```
End

If Syscalls("SIGON")<>0 Then Do
  Say "LRGSCO03E The SIGNAL interface could not be established."
  Exit 8
End

final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)

/* ------------------------------------------------------------------ */
/*  Set home and working directory, setup envvars                     */
/* ------------------------------------------------------------------ */

home = "/"
Call Syscall_Cmd "chdir" home
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH=/bin"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.6 = "TSOALLOC=sysexec"
__environment.7 = "sysexec='"||rexxlib||"'"
__environment.0 = 7

/* ------------------------------------------------------------------ */
/*  Enable for timer interrupts                                       */
/* ------------------------------------------------------------------ */

Call Syscall_Cmd "sigaction" sigalrm sig_cat 0 "ohdl oflg"
sig_enabled = 1
Call Syscall_Cmd "sigprocmask" sig_unblock,
  Sigaddset(Sigsetempty(),sigalrm) "mask"

/* ------------------------------------------------------------------ */
/*  Start all largeios processes and finally collect the results      */
/* ------------------------------------------------------------------ */

parm.0 = 3
parm.1 = "sh"
parm.2 = "-c"
Do i=1 To procs
  parm.3 = "tso 'largeios" blkios typeios seed+i-1 fdir||"'"
  shell_rc = -1
  shell_termsig = -1
  shell_stopsig = -1
  time_out = 0
  Call Syscall_Cmd "open /dev/null (o_rdonly)"
```

```
  fd.0 = retval
  fd.0.i = fd.0
  Call Syscall_Cmd "pipe pp."
  fd.1 = pp.2
  fd.1.i = fd.1
  pp.1.i = pp.1
  Call Syscall_Cmd "dup (fd.1)"
  fd.2 = retval
  fd.2.i = fd.2
  Call Syscall_Cmd "spawnp (parm.1) 3 fd. parm. __environment."
  pid.i = retval
End
num_cases = 0
Do i=1 To procs
  Call Syscall_Cmd "alarm" timeout_value /* timeout setting in secs */
  Call Syscall_Cmd "waitpid (pid.i) stat. 0", info_on_timeout
  Call Syscall_Cmd "alarm 0"
  If time_out Then Do
    Call Syscall_Cmd "kill (pid.i)" sigkill, noexit_on_error
    Call Syscall_Cmd "waitpid (pid.i) stat. 0"
  End
  Select
    When stat.w_ifexited Then shell_rc = stat.w_exitstatus
    When stat.w_ifsignaled Then shell_termsig = stat.w_termsig
    When stat.w_ifstopped Then shell_stopsig = stat.w_stopsig
    Otherwise Nop
  End
  pid = pid.i
  pid.i = -1
  Do j=0 To 2
    If fd.j.i<>-1 Then Call Syscall_Cmd "close (fd.j.i)"
    fd.j.i = -1
  End
  hdr = "Results of process no." Right(i,5) "with process id" pid
  Say hdr
  Say Copies("-",Length(hdr))
  Say ""
  all_output = ""
  Do Until retval<1000
    Call Syscall_Cmd "read (pp.1.i) output_data 1000"
    all_output = all_output||output_data
  End
  first_line = 1
  cur_rc = 0
  Do While Length(all_output)>0
    Parse Var all_output output_line (esc_n) all_output
    If first_line Then first_line = 0
    Else Say output_line
  End
  Call Syscall_Cmd "close (pp.1.i)"
  pp.1.i = -1
  If time_out Then Do
    Say "LRGSC004E Timeout occurred during command processing..."
    Say "LRGSC005I Command:" shell_cmd
    final_rc = 8
```

```
    cur_rc = 8
  End
  If shell_rc<>0 Then Do
    Say "*** non-zero return code: Rc("||shell_rc||")."
    final_rc = 8
    cur_rc = 8
  End
  If cur_rc=0 Then Do
    If Subword(output_line,1,2)="Write block" |,
       Subword(output_line,1,2)="Read  block" Then Do
      e_time = Word(output_line,4)
      If Verify(e_time,"1234567890.")=0 Then Do
        num_cases = num_cases + 1
        If num_cases=1 Then Do
          sum_etime = e_time
          min_etime = e_time
          max_etime = e_time
        End
        Else Do
          sum_etime = sum_etime + e_time
          min_etime = Min(min_etime,e_time)
          max_etime = Max(max_etime,e_time)
        End
      End
    End
  End
  Say ""
End
avg_header = "AVG time per process"
min_header = "MIN time per process"
max_header = "MAX time per process"
num_dashes = Length(avg_header)
avg_etime = Right(Format(sum_etime/num_cases,6,6),num_dashes)
min_etime = Right(Format(min_etime,6,6),num_dashes)
max_etime = Right(Format(max_etime,6,6),num_dashes)
Say avg_header min_header max_header
Say Copies("-",num_dashes) Copies("-",num_dashes) Copies("-",num_dashes)
Say avg_etime min_etime max_etime
Say ""

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit:
  Parse Arg final_rc
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
```

```
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do i=1 To procs
    Do j=0 To 2
      If fd.j.i<>-1 Then
        Call Syscall_Cmd "close (fd.j.i)", noexit_on_error
    End
    If pid.i<>-1 Then Do
      Call Syscall_Cmd "kill (pid.i)" sigkill, noexit_on_error
      Call Syscall_Cmd "waitpid (pid.i) stat. 0"
    End
    If pp.1.i<>-1 Then
      Call Syscall_Cmd "close (pp.1.i)", noexit_on_error
  End
Exit final_rc

Syscall_Cmd:
  Parse Arg syscall_cmd, call_type, no_display
  display_msgs = (no_display<>"1")
  exit_on_error = (call_type="")
  Address SYSCALL syscall_cmd
  not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  OK = (not_OK = 0)
  If not_OK & display_msgs Then Do
    If call_type=info_on_timeout & errno=78 Then Do
      time_out = 1
      final_rc = 8
    End
    Else Do
      Say "SYSCALL Service:" syscall_cmd
      Say "Syscall Return Code=" rc
      Say "OMVS Return Value  =" retval
      Say "OMVS Return Code   =" errno
      Say "OMVS Reason Code   =" errnojr
      Address SYSCALL "strerror" errno errnojr "err."
      If rc=0 & retval>=0 Then Do
        Say ""
        Say "OMVS Return Code Explanation -"
        Say err.se_errno
        Say "OMVS Reason Code Explanation -"
        Say err.se_reason
        Say ""
      End
      If exit_on_error Then Call Final_Exit 8
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

C

JCL samples

This appendix contains sample jobs for many zFS-related tasks. They are intended to be used as skeletons for your own JCL and to provide information and ideas about how to use the commands, functions, and utilities in addition to what is shown in the main chapters in this book. A detailed description of how they work is found with each JCL example.

Most of the tests and commands were performed using these and other JCL statements because this makes it very easy to run similar tasks, perform other commands, and provide the necessary documentation in the job output.

This appendix describes and contains the following JCL:

- RXZFS and RXIOE samples
- AMS, TSO and IOEAGFMT samples
- Migration JCL samples
- MVS system command samples
- JCL used for the comparison between HFS and zFS
- JCL for backup and restore
- JCL used for comparison in UNIX System Services sysplex file system sharing environments

C.1 Using the JCL examples

To use the JCL examples provided, you must store the REXX procedures into a library. In our JCL examples, this library is:

```
HERING.ZFS.REXX.EXEC
```

In our JCL examples, replace the following statement with your library name:

```
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
```

Otherwise, you can place the library into your TSO logon procedure.

These JCL examples also allow you to set a timeout value. This can be used to stop processing (cancel the UNIX process) automatically after the number of seconds specified. A value of 0 indicates that no timeout setting is used. This value may be useful sometimes. A time limit may also be specified in the ISHELL when executing a command with the EX command on the main panel. The timeout value is set as follows:

```
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
```

Note: The sample JCL jobs shown in this appendix are available in softcopy on the Internet from the Redbooks Web server.

Point your browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246580/
```

Note: SG246580 must be upper case (the SG part).

Alternatively, you can go to:

```
http://www.redbooks.ibm.com
```

and select **Redbooks Online**, and then **Additional Materials**.

C.2 JCL examples using RXZFS and RXIOE

This section provides a collection of zFS-related JCL jobs that use the REXX procedures RXZFS or RXIOE shown in Appendix B, “REXX utility procedures” on page 361.

C.2.1 Display information for zFS aggregates

Example: C-1 Job that uses the AGGRINFO subcommand

```
//ZFSJOB   JOB ,'AGGRINFO',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Display information for zFS aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//IOEZADM  EXEC PGM=IKJEFT01,PARM=RXIOE
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSIN    DD DATA,DLM=##
# -------------------------------------------------------------------
/* Display information for all aggregates...
/* ---------------------------------------+
ioezadm aggrinfo+
```

```
/* Display information for a specific aggregate...
/* ---------------------------------------------+
ioezadm aggrinfo -aggregate OMVS.CMP02.ZFS
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSPRINT DD SYSOUT=*,LRECL=260,RECFM=V
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

Note: Lines starting with /* can span more than one line. If they end with a dash (-), you should append a plus sign (+) to avoid having the command in the next line treated as a continuation.

Example C-2 shows the output from the submitted job.

Example: C-2 AGGRINFO job output

```
Display information for all aggregates...
------------------------------------------
OMVS.CMP04.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
OMVS.MUL01.ZFS (R/W MULT): 18615 K free out of total 18872 (2000 reserved)
OMVS.ROOTCOMP.ZFS (R/W COMP): 112465 K free out of total 2030048 (20504 reserved)
OMVS.MULTUSER.ZFS (R/W MULT): 9548 K free out of total 9720 (1072 reserved)
OMVS.CMP03.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
OMVS.CMP02.ZFS (R/W COMP): 4981 K free out of total 5184 (568 reserved)
OMVS.CMP01.ZFS (R/W COMP): 6326 K free out of total 6480 (712 reserved)
OMVS.MUL02.ZFS (R/W MULT): 40111 K free out of total 40472 (2000 reserved)

Display information for a specific aggregate...
-----------------------------------------------
ioezadm aggrinfo -aggregate OMVS.CMP02.ZFS
OMVS.CMP02.ZFS (R/W COMP): 4981 K free out of total 5184 (568 reserved)
```

C.2.2 Attach a compatibility mode aggregate

Example C-3 is a test to unmount a compatibility mode aggregate and attach it as a multifile system aggregate.

Example: C-3 ATTCHCMP JCL to attach a compatibility mode aggregate

```
//ZFSJOB   JOB ,'ATTCHCMP',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Attaching a compatibility mode aggreagte
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ------------------------------------------------------------------
 AGGREGATE=OMVS.CMP04.ZFS
 MOUNTPOINT=/u/zfs/c04
# ------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
```

```
# ---------------------------------------------------------------------
CMD="zfsadm aggrinfo -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="/usr/sbin/unmount $MOUNTPOINT"; echo $CMD;$CMD+
CMD="zfsadm attach -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="zfsadm aggrinfo -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="zfsadm detach -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="/usr/sbin/mount -f $AGGREGATE -t ZFS $MOUNTPOINT"; echo $CMD;$CMD+
CMD="zfsadm aggrinfo -aggregate $AGGREGATE"; echo $CMD;$CMD+
# ---------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ---------------------------------------------------------------------
```

Example C-4 displays the ATTCHCMP job output.

Example: C-4 ATTCHCMP job output

```
zfsadm aggrinfo -aggregate OMVS.CMP04.ZFS
OMVS.CMP04.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
/usr/sbin/unmount /u/zfs/c04
zfsadm attach -aggregate OMVS.CMP04.ZFS
IOEZ00117I Aggregate OMVS.CMP04.ZFS attached successfully
zfsadm aggrinfo -aggregate OMVS.CMP04.ZFS
OMVS.CMP04.ZFS (R/W MULT): 4962 K free out of total 5184 (568 reserved)
zfsadm detach -aggregate OMVS.CMP04.ZFS
IOEZ00122I Aggregate OMVS.CMP04.ZFS detached successfully
/usr/sbin/mount -f OMVS.CMP04.ZFS -t ZFS /u/zfs/c04
zfsadm aggrinfo -aggregate OMVS.CMP04.ZFS
OMVS.CMP04.ZFS (R/W COMP): 4962 K free out of total 5184 (568 reserved)
```

Tip: The input command lines look unusual, but defining the command as a variable allows you to display the command with all the envvars being resolved before executing it.

C.2.3 Attach zFS aggregates and mount file systems

Example: C-5 ATTCHMNT JCL

```
//ZFSJOB   JOB ,'ATTCHMNT',NOTIFY=&SYSUID.,REGION=0M
//* ---------------------------------------------------------------------
//* Attach zFS aggregates and mount file systems
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ---------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ---------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ---------------------------------------------------------------------
# ---------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ---------------------------------------------------------------------
zfsadm attach -aggregate OMVS.MUL01.ZFS
zfsadm attach -aggregate OMVS.MUL02.ZFS
/usr/sbin/mount -f OMVS.CMP01.ZFS -t ZFS /u/zfs/c01
/usr/sbin/mount -f OMVS.CMP02.ZFS -t ZFS /u/zfs/c02
/usr/sbin/mount -f OMVS.CMP03.ZFS -t ZFS /u/zfs/c03
```

```
/usr/sbin/mount -f OMVS.MO1.ZFS -t ZFS /u/zfs/mO1
/usr/sbin/mount -f OMVS.MO2.ZFS -t ZFS /u/zfs/mO2
/usr/sbin/mount -f OMVS.mO3.ZFS -t ZFS /u/zfs/mO3
/usr/sbin/mount -f OMVS.MO4.ZFS -t ZFS /u/zfs/mO4
# ---------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=26O,RECFM=V
//* ---------------------------------------------------------------------
```

C.2.4 Define an automount aggregate and create user file systems

Example: C-6 AUTOMNT JCL

```
//ZFSJOB   JOB ,'AUTOMNT',NOTIFY=&SYSUID.,REGION=OM
//* ---------------------------------------------------------------------
//* Define automount aggregate and create user file systems
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ---------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ---------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ---------------------------------------------------------------------
  AGGREGATE=OMVS.MULTUSER.ZFS
  STORCLASS=SCCOMP
  MEGABYTES=10 10
       SIZE=5000
       PERM=o700
# ---------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ---------------------------------------------------------------------
CMD="zfsadm define -aggregate $AGGREGATE -storageclass $STORCLASS -
  -megabytes $MEGABYTES"; echo $CMD;$CMD+
CMD="zfsadm format -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="zfsadm attach -aggregate $AGGREGATE"; echo $CMD;$CMD+
CMD="zfsadm create -filesystem OMVS.ALAN.ZFS -aggregate -
  $AGGREGATE -size $SIZE -owner ALAN -perms $PERM"; echo $CMD;$CMD+
CMD="zfsadm create -filesystem OMVS.TOLOD.ZFS -aggregate -
  $AGGREGATE -size $SIZE -owner TOLOD -perms $PERM"; echo $CMD;$CMD+
CMD="zfsadm create -filesystem OMVS.YVES.ZFS -aggregate -
  $AGGREGATE -size $SIZE -owner YVES -perms $PERM"; echo $CMD;$CMD+
# ---------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=26O,RECFM=V
//* ---------------------------------------------------------------------
```

C.2.5 Define, format, and mount a compatibility mode aggregate

Example: C-7 COMPAGGR JCL

```
//ZFSJOB   JOB ,'COMPAGGR',NOTIFY=&SYSUID.,REGION=OM
//* ---------------------------------------------------------------------
//* Define, format and mount a compatibility mode aggregate
```

```
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ---------------------------------------------------------------------
//  SET TIMEOUT=10         <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ---------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ---------------------------------------------------------------------
  AGGREGATE=OMVS.CMP05.ZFS
    VOLUMES=TOTZF1
  MEGABYTES=5 1
       OWNER=888
 PERMISSION=o755
 MOUNTPOINT=/u/hering/test5
# ---------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ---------------------------------------------------------------------
echo Define and format aggregate $AGGREGATE+
zfsadm define -aggregate $AGGREGATE -volumes $VOLUMES -
  -megabytes $MEGABYTES+
zfsadm format -aggregate $AGGREGATE -compat -owner $OWNER -
  -perms $PERMISSION+

CMD="mkdir -m 755 $MOUNTPOINT"; echo $CMD;$CMD+
CMD="chown $OWNER $MOUNTPOINT"; echo $CMD;$CMD+
CMD="/usr/sbin/mount -f $AGGREGATE -t ZFS $MOUNTPOINT"; echo $CMD;$CMD+

echo Display information about the new aggregate and file system+
zfsadm aggrinfo -aggregate $AGGREGATE+
df -kP $MOUNTPOINT+
# ---------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ---------------------------------------------------------------------
```

Example: C-8 COMPAGGR job output

```
Define and format aggregate OMVS.CMP05.ZFS
IOEZ00248E VSAM linear dataset OMVS.CMP05.ZFS successfully created.
IOEZ00077I HFS-compatibility aggregate OMVS.CMP05.ZFS has been successfully created

mkdir -m 755 /u/hering/test5
mkdir: FSUM6404 directory "/u/hering/test5": EDC5117I File exists.
*** non-zero return code: Rc(1).
chown 888 /u/hering/test5
/usr/sbin/mount -f OMVS.CMP05.ZFS -t ZFS /u/hering/test5

Display information about the new aggregate and file system
OMVS.CMP05.ZFS (R/W COMP): 5030 K free out of total 5184 (568 reserved)
Filesystem         1024-blocks        Used  Available  Capacity Mounted on
OMVS.CMP05.ZFS    5039           9       5030        1% /u/hering/test5
```

Note: Commands ending with a non-zero return code are flagged and the job step return code is set to 8.

C.2.6 Define and delete an aggregate

Example: C-9 DEFDELAG JCL

```
//ZFSJOB   JOB ,'DEFDELAG',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Define and delete an aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
  AGGREGATE=OMVS.CMP05.ZFS
    VOLUMES=TOTZF1
  MEGABYTES=5 1
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
usscmd="zfsadm define -aggregate $AGGREGATE -volumes $VOLUMES -
  -megabytes $MEGABYTES"; echo $usscmd; $usscmd+
tso "delete '$AGGREGATE' PURGE ERASE"+
# -------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* -------------------------------------------------------------------
```

Example: C-10 DEFDELAG job output

```
zfsadm define -aggregate OMVS.CMP05.ZFS -volumes TOTZF1 -megabytes 5 1
IOEZ00248E VSAM linear dataset OMVS.CMP05.ZFS successfully created.
delete 'OMVS.CMP05.ZFS' PURGE ERASE
IDC0550I ENTRY (D) OMVS.CMP05.ZFS.DATA DELETED
IDC0550I ENTRY (C) OMVS.CMP05.ZFS DELETED
```

Note: TSO commands that do not need to run authorized can easily be executed from a UNIX System Services shell environment using command **tso**. To do so with authorized commands, you need to use the **tsocmd** command.

C.2.7 Increase the physical size of an aggregate

Example: C-11 GROW JCL

```
//ZFSJOB   JOB ,'GROW',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Make the physical size of an aggregate larger
//* Property of IBM  (C) Copyright IBM Corp. 2002-2008
//* -------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
```

```
# ------------------------------------------------------------------
  AGGREGATE=OMVS.MUL02.ZFS
# ------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ------------------------------------------------------------------
CMD="zfsadm grow -aggregate $AGGREGATE -size 0"; echo $CMD;$CMD+
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

Example: C-12 GROW job output

```
zfsadm grow -aggregate OMVS.MUL02.ZFS -size 0
IOEZ00173I Aggregate OMVS.MUL02.ZFS successfully grown
OMVS.MUL02.ZFS (R/W MULT): 29311 K free out of total 29672 (2000 reserved)
```

C.2.8 Display help information

Example: C-13 HELPZFSA JCL

```
//ZFSJOB   JOB ,'HELPZFSA',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Display help information
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET TIMEOUT=20         <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ------------------------------------------------------------------
# Display help information about a command and an apropos topic
# ------------------------------------------------------------------
  Command_Topic=aggrinfo
  Apropos_Topic=aggregate
      Code_Page=IBM-037
# ------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ------------------------------------------------------------------
zfsadm help -topic $Command_Topic | iconv -f IBM-1047 -t $Code_Page+

echo Display help information for string $Apropos_Topic"":+
zfsadm apropos -topic $Apropos_Topic | iconv -f IBM-1047 -t $Code_Page+
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

Example: C-14 HELPZFSA job output

```
zfsadm aggrinfo: Obtain information on an attached aggregate
IOEZ00243I Usage: zfsadm aggrinfo [-aggregate <aggregate name>] [-level] [-help]
```

```
Display help information for string aggregate:
aggrinfo: Obtain information on an attached aggregate
attach: attach an aggregate
detach: detach an aggregate
format: format an aggregate
grow: grow an aggregate
lsaggr: list aggregates
lsquota: list filesystem and aggregate space usage
quiesce: quiesce an aggregate
unquiesce: unquiesce an aggregate
```

Note: Converting the output data to the code page used with the terminal emulation just makes it easier to read. Otherwise, the characters Ý-level¨ Ý-help¨ appear.

C.2.9 Get information about aggregates

Example: C-15 LSAGGRFS JCL

```
//ZFSJOB   JOB ,'LSAGGRFS',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Get information about aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* -------------------------------------------------------------------
//IOEZADM  EXEC PGM=IKJEFT01,PARM=RXIOE
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSIN    DD DATA,DLM=##
# -------------------------------------------------------------------
ioezadm lsaggr
ioezadm lsfs
ioezadm lsfs -fast
ioezadm lsfs -long
# -------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSPRINT DD SYSOUT=*,LRECL=260,RECFM=V
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
```

C.2.10 List quota information about a file system

Example: C-16 LSQUOTA JCL

```
//ZFSJOB   JOB ,'LSQUOTA',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* List quota information about a file system
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* -------------------------------------------------------------------
//IOEZADM  EXEC PGM=IKJEFT01,PARM=RXIOE
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSIN    DD DATA,DLM=##
# -------------------------------------------------------------------
ioezadm lsquota -filesystem OMVS.CMP01.ZFS
# -------------------------------------------------------------------
##
```

```
//SYSTSIN  DD DUMMY
//SYSPRINT DD SYSOUT=*,LRECL=260,RECFM=V
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

C.2.11 Mount a mixed-case file system in TSO

Example: C-17 MOUNTTSO JCL

```
//ZFSJOB   JOB ,'MOUNTTSO',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Mount mixed case file system in TSO
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//IOEZADM  EXEC PGM=IKJEFT01,PARM=RXIOE
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSIN    DD DATA,DLM=##
# ------------------------------------------------------------------
mount filesystem('''OMVS.m03.ZFS''') -
      mountpoint('/zfstest/m03') type(zfs)
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSPRINT DD SYSOUT=*,LRECL=260,RECFM=V
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

Attention: To mount file systems with mixed-case names, triple quotes are needed around the name when using the TSO MOUNT command.

Because this example is dealing with multifile system aggregates, we will remove it on a next update of this book.

C.2.12 Define a multifile system aggregate and create file systems

Example: C-18 MULTAGGR JCL

```
//ZFSJOB   JOB ,'MULTAGGR',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Define a multi-file system aggregate and create file systems
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//IOEZADM  EXEC PGM=IKJEFT01,PARM=RXIOE
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSIN    DD DATA,DLM=##
# ------------------------------------------------------------------
ioezadm define -aggregate OMVS.MUL02.ZFS -
               -volumes TOTZF1 TOTZF2 -
               -megabytes 20 10

ioezadm format -aggregate OMVS.MUL02.ZFS

ioezadm attach -aggregate OMVS.MUL02.ZFS
```

```
ioezadm create -filesystem OMVS.m01.ZFS -
               -aggregate OMVS.MUL02.ZFS -
               -size 6000 -owner 888 -perms o755
ioezadm create -filesystem OMVS.M02.ZFS -
               -aggregate OMVS.MUL02.ZFS -
               -size 6000
ioezadm create -filesystem OMVS.m03.ZFS -
               -aggregate OMVS.MUL02.ZFS -
               -size 6000
ioezadm create -filesystem OMVS.M04.ZFS -
               -aggregate OMVS.MUL02.ZFS -
               -size 6000 -perms o700

ioezadm rename -oldname OMVS.m01.ZFS -
               -newname OMVS.M01.ZFS

ioezadm aggrinfo -aggregate OMVS.MUL02.ZFS
ioezadm lsfs -aggregate OMVS.MUL02.ZFS

ioezadm lsquota -filesystem OMVS.M01.ZFS
ioezadm lsquota -filesystem OMVS.M02.ZFS
ioezadm lsquota -filesystem OMVS.m03.ZFS
ioezadm lsquota -filesystem OMVS.M04.ZFS

mkdir '/u/zfs/m01' mode(7,0,0)
mkdir '/u/zfs/m02' mode(7,0,0)
mkdir '/u/zfs/m03' mode(7,0,0)
mkdir '/u/zfs/m04' mode(7,0,0)

mount filesystem('OMVS.M01.ZFS') -
      mountpoint('/zfstest/m01') type(zfs)
mount filesystem('OMVS.M02.ZFS') -
      mountpoint('/zfstest/m02') type(zfs)
mount filesystem('''OMVS.m03.ZFS''') -
      mountpoint('/zfstest/m03') type(zfs)
mount filesystem('OMVS.M04.ZFS') -
      mountpoint('/zfstest/m04') type(zfs)

# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSPRINT DD SYSOUT=*,LRECL=260,RECFM=V
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

Attention: File system names used with **zfsadm** or **ioezadm** commands are always case sensitive and need to be spelled exactly as they have been created.

As this example is dealing with multifile system aggregates, we will remove it on a next update of this book.

C.2.13 Use the Salvager utility

Example: C-19 Salvager job JCL

```
//ZFSJOB   JOB ,'SALVAGE',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Run the Salvager...
//* Property of IBM  (C) Copyright IBM Corp. 2002
```

```
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ------------------------------------------------------------------
# The external link has been created with the following command in
# SU mode:
# ln -e IOEAGSLV /u/hering/bin/zfssalvage
  AGGREGATE=OMVS.SC43.COMPAT01.ZFS
       PATH=/u/hering/bin:/bin
# ------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ------------------------------------------------------------------
C="zfssalvage -aggregate $AGGREGATE -verbose"; echo $C;$C+
C="zfssalvage -aggregate $AGGREGATE -salvageonly"; echo $C;$C+
C="zfssalvage -aggregate $AGGREGATE -verifyonly"; echo $C;$C+
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* ------------------------------------------------------------------
```

Example: C-20 Salvager job output

```
zfssalvage -aggregate OMVS.SC43.COMPAT01.ZFS -verbose
Salvaging OMVS.SC43.COMPAT01.ZFS
Will run recovery on OMVS.SC43.COMPAT01.ZFS
 SectorSize 4096; TotalBlocks 0; BlockSize 8192;
 FragmentSize 1024; FirstBlock 1; NLogBlocks 13;
 NumBigChunks 113; minBlkSize 4096; minBlkCount 1800;
 FileSysCreateTime 1016066366;
 FileSysClean 0; FileSysEmpty 0; FileSysMountedAs ``''
/dev/lfs1/OMVS.SC43.COMPAT01.ZFS: zFS aggregate created Thu Mar 14 00:39:26 2002
Device /dev/lfs1/OMVS.SC43.COMPAT01.ZFS, major 0, minor 1; total 899 8192-byte blocks
Block size 8192, frag size 1024, firstBlock 1, nBlocks 899
Principal superblock at byte 0, nLogBlocks 65536.
Processed 1 vols 7 anodes 1 dirs 0 files 0 acls
Done.  OMVS.SC43.COMPAT01.ZFS checks out as zFS aggregate.
zfssalvage -aggregate OMVS.SC43.COMPAT01.ZFS -salvageonly
Salvaging OMVS.SC43.COMPAT01.ZFS
Processed 1 vols 6 anodes 1 dirs 0 files 0 acls
Done.  OMVS.SC43.COMPAT01.ZFS checks out as zFS aggregate.
zfssalvage -aggregate OMVS.SC43.COMPAT01.ZFS -verifyonly
Verifying OMVS.SC43.COMPAT01.ZFS
Processed 1 vols 6 anodes 1 dirs 0 files 0 acls
Done.  OMVS.SC43.COMPAT01.ZFS checks out as zFS aggregate.
READY
END
```

C.3 AMS, TSO, and IOEAGFMT examples

Following are a collection of zFS-related JCL examples using AMS and TSO services.

C.3.1 Define and format a new aggregate

Example: C-21 IOEAGFMT JCL

```
//ZFSJOB   JOB ,'ZFS NewAggr',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//DEFINE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=TOTZF1
//SYSIN    DD     *
  DEFINE CLUSTER (NAME(OMVS.MUL01.ZFS) VOLUMES(TOTZF1)  -
   LINEAR MEGABYTES(20 10) SHAREOPTIONS(2))
//* Format VSAM Linear Data Set as ZFS Multiple File System Aggregate
//FORMAT   EXEC   PGM=IOEAGFMT,REGION=0M,
//         PARM=('-aggregate OMVS.MUL01.ZFS')
//*STEPLIB DD DISP=SHR,DSN=hlq.SIOELMOD    <--LOADLIB FOR ZFS
//SYSPRINT DD   SYSOUT=*
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CEEDUMP  DD   SYSOUT=*
//*
```

C.3.2 Listcat aggregate in TSO batch mode

Example: C-22 LISTCTSO JCL

```
//ZFSJOB   JOB ,'Listcat VSAM LD',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Listcat aggregate in TSO batch mode (LISTCTSO)
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//*
//  SET CLUSTER=OMVS.MUL02.ZFS
//*
//* ------------------------------------------------------------------
//LISTCAT  EXEC PGM=IKJEFT01,
//  PARM='LISTCAT ENT(''&CLUSTER.'') ALL'
//SYSEXEC  DD DSNAME=&SYSUID..UNIX.REXX.EXEC,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

C.3.3 Rename VSAM LD using AMS services

Example: C-23 RENLDAMS JCL

```
//ZFSJOB   JOB ,'Rename VSAM LD',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Delete an aggregate in TSO batch mode
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//* All of the following steps assume that the new qualifier and the
//* old qualifier reside in the same user catalog. Access method
//* services does not allow alter where the new name does not match
//* the existing catalog structure for an sms-managed vsam data set.
//* if the data set is not managed by sms, the rename succeeds, but
```

```
//* db2 cannot allocate it as described in dfsms/mvs: access method
//* services for the integrated catalog.
//* -------------------------------------------------------------------
//RENAMELD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*,LRECL=136,RECFM=V
//SYSIN    DD DATA,DLM=##
ALTER -
    'OMVS.CMP04.ZFS' -
    NEWNAME('OMVS.CMP05.ZFS')
  ALTER -
    'OMVS.CMP04.ZFS.DATA' -
    NEWNAME('OMVS.CMP05.ZFS.DATA')
##
//* -------------------------------------------------------------------
```

Note: When renaming a compatibility mode aggregate, the file system name is automatically renamed when it is mounted the next time.

C.3.4 Rename an aggregate in TSO batch mode

Example: C-24 RENLDTSO JCL

```
//ZFSJOB   JOB ,'Rename VSAM LD',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Rename an aggregate in TSO batch mode
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//*
//  SET OLDCLSTR=OMVS.MULTTEST.ZFS
//  SET NEWCLSTR=OMVS.MULTTST2.ZFS
//*
//* -------------------------------------------------------------------
//RENAMELD EXEC PGM=IKJEFT01,
//  PARM='ALTER ''&OLDCLSTR.'' NEWNAME(''&NEWCLSTR.'')'
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
//RENAMEDA EXEC PGM=IKJEFT01,
//  PARM='ALTER ''&OLDCLSTR..DATA'' NEWNAME(''&NEWCLSTR..DATA'')'
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* -------------------------------------------------------------------
```

C.3.5 Delete an aggregate using AMS services

Example: C-25 DELLDAMS JCL

```
//ZFSJOB   JOB ,'AMS Delete',
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
//DELETE   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=*
//SYSUDUMP DD     SYSOUT=*
//AMSDUMP  DD     SYSOUT=*
//DASD0    DD     DISP=OLD,UNIT=3390,VOL=SER=TOTZF1
//SYSIN    DD     *
  DELETE OMVS.MUL01.ZFS -
    PURGE ERASE
```

```
//*
```

C.3.6 Delete an aggregate in TSO batch mode

Example: C-26 DELLDTSO JCL

```
//ZFSJOB   JOB ,'Delete VSAM LD',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Delete an aggregate in TSO batch mode (DELLDTSO)
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//*
//  SET CLUSTER=OMVS.CMP05.ZFS
//*
//* -------------------------------------------------------------------
//DELETELD EXEC PGM=IKJEFT01,
//  PARM='DELETE ''&CLUSTER.'' PURGE ERASE'
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* -------------------------------------------------------------------
```

Example: C-27 DELLDTSO job output

```
ENTRY (D) OMVS.CMP05.ZFS.DATA DELETED
ENTRY (C) OMVS.CMP05.ZFS DELETED
```

C.3.7 Format an aggregate using an external link to IOEAGFMT

Example: C-28 Format an aggregate JCL

```
//ZFSAGFMT JOB ,'zFS NewAggr',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Create a new zFS Aggregate, format it (and delete it again)
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
# The external link has been created with the following command in
# SU mode:
# ln -e IOEAGFMT /u/hering/bin/zfsformat
# -------------------------------------------------------------------
  AGGREGATE=OMVS.SC43.COMPTEST.ZFS
    VOLUMES=TOTZF1
  STORCLASS=SCCOMP
       CYLS=1 1
      OWNER=888
 PERMISSION=755
#  TSOALLOC=sysexec
#   sysexec='HERING.ZFS.REXX.EXEC'
       PATH=/u/hering/bin:/bin
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
C="tso -t DEFINE CLUSTER (NAME('$AGGREGATE') VOLUMES($VOLUMES) -
```

```
  LINEAR CYLINDERS($CYLS) SHAREOPTIONS(2))"; echo $C;$C+
C="zfsformat -aggregate $AGGREGATE -compat -owner $OWNER -
  -perms $PERMISSION"; echo $C;$C+
# --------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=V
//* --------------------------------------------------------------------
```

C.4 Migration JCL examples

Following is a collection of JCL examples dealing with copying and migrating.

C.4.1 Use copytree to copy or migrate UNIX structures

Example: C-29 COPYTREE JCL

```
//ZFSJOB   JOB ,'COPYTREE',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=SC70
//*MAIN SYSTEM=SC70
//* --------------------------------------------------------------------
//* Using copytree to copy or migrate UNIX structures
//* Property of IBM  (C) Copyright IBM Corp. 2002-2006
//* --------------------------------------------------------------------
//  SET TIMEOUT=R0      <=== Timeout value in seconds, <R>0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* --------------------------------------------------------------------
//ZFSCOPY  EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# --------------------------------------------------------------------
  source_dir=/u/hering/
  target_dir=/tmp/hering.test
# List additional reason codes on EDC messages if available (1=Y, 0=N)
  _EDC_ADD_ERRNO2=1
# --------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# --------------------------------------------------------------------
echo "Source:" $source_dir+
echo "Target:" $target_dir+
echo ""+
echo "--------------------------"+
echo "Copy processing starting..."+
echo "--------------------------"+
/samples/copytree -a $source_dir $target_dir+
echo "--------------------------"+
echo "Copy processing stopping..."+
echo "--------------------------"+
# --------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* --------------------------------------------------------------------
```

C.4.2 A simple way to use pax in copy mode

Example: C-30 COPYPAX7 JCL

```
//ZFSJOB   JOB ,'COPYPAX7',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=SC70
//*MAIN SYSTEM=SC70
//* -------------------------------------------------------------------
//* Simple way to use pax R7+ in copy mode for copying and migration
//* Property of IBM  (C) Copyright IBM Corp. 2002-2006
//* -------------------------------------------------------------------
//  SET TIMEOUT=R0      <=== Timeout value in seconds, <R>0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSCOPY  EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
  source_dir=/u/hering/
  target_dir=/tmp/hering.test
    pax_v1r7=/bin/pax
# Specify vb= or vb=v to request verbose N or Y
          vb=
# Specify kp= or kp=k to prevent overwriting of existing files N or Y
          kp=
# List additional reason codes on EDC messages if available (1=Y, 0=N)
  _EDC_ADD_ERRNO2=1
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
echo "Source:" $source_dir+
echo "Target:" $target_dir+
echo ""+
echo "---------------------------"+
echo "Copy processing starting..."+
echo "---------------------------"+
cd $source_dir && $pax_v1r7 -rw$kp$vb -peW -XCM . $target_dir+
echo "---------------------------"+
echo "Copy processing stopping..."+
echo "---------------------------"+
# -------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* -------------------------------------------------------------------
```

C.4.3 Use pax to copy a source to a target structure

Example: C-31 COPYPAX JCL

```
//ZFSJOB   JOB ,'COPYPAX',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=SC70
//*MAIN SYSTEM=SC70
//* -------------------------------------------------------------------
//* Use pax to copy source directory structure to target directory
//* Property of IBM  (C) Copyright IBM Corp. 2002-2010
//* -------------------------------------------------------------------
//  SET SOURCED='/u/hering'                     <=== Source directory
//  SET TARGETD='/tmp/hering'                   <=== Target directory
```

```
//* ------------------------------------------------------------------
//  SET TIMEOUT=R0          <=== Timeout value in seconds, R0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//COPYPAX  EXEC PGM=IKJEFT01,PARM='COPYPAX &TIMEOUT &SOURCED &TARGETD'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//CPPXPARM DD DATA,DLM=##
TARGET_MUST_BE_EMPTY=Y < Target structure must be empty      (--Y--|N)
CORRECT_INV_FD_FILES=Y < Correct invalid fd files            (--Y--|N)
    COPY_PAX_VERBOSE=N < List all objects copied or created  (Y|--N--)
   NO_FILE_OVERWRITE=N < Prevent replacing of existing files (Y|--N--)
   PROCESS_INFO_MSGS=Y < Display process data in batch mode  (--Y--|N)
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* ------------------------------------------------------------------
```

C.4.4 Add missing mount point directories after a clone process

Example: C-32 ADDMNTPS JCL

```
//ZFSJOB   JOB ,'ADDMNTPS',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Add missing mount point directories after clone process
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET SOURCED='/'                            <=== Source directoy
//  SET TARGETD='/u/zfs/clone'                 <=== Target directoy
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='ADDMNTPS &TIMEOUT &SOURCED &TARGETD'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

C.5 MVS system command samples

Following is a collection of jobs running MVS system commands and trapping the syslog output.

C.5.1 Start zFS

Example: C-33 STARTZFS JCL

```
//ZFSJOB   JOB ,'STARTZFS',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Run MVS sytem command and retrieve the output
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET CONSCMD='SETOMVS RESET=(ZS)'             <=== Console command
//  SET CONSOLE=''                               <=== Console name
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//CONSCMND EXEC PGM=IKJEFT01,PARM='CN C=&CONSOLE. &CONSCMD.'
```

```
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

Note: Parmlib member BPXPRMZS contains the ZFS FILESYSTYPE statement.

C.5.2 Stop zFS

This example has been removed as we did not add new support to the command CN to support the new interface for stopping zFS. This had been done for restarting zFS.

Note: In z/OS V1R8 the old zFS stop command was removed. You need to use the new command F OMVS,STOPPFS=ZFS now and answer a reply message first.

C.5.3 Restart zFS

Example: C-34 RSTRTZFS JCL

```
//ZFSJOB   JOB ,'RSTRTZFS',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Run MVS system command and retrieve the output
//* Property of IBM  (C) Copyright IBM Corp. 2002-2005
//* ------------------------------------------------------------------
//  SET CONSCMD='RESTART ZFS'                    <=== Console command
//  SET CONSOLE=''                               <=== Console name
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//CONSCMND EXEC PGM=IKJEFT01,PARM='CN C=&CONSOLE. &CONSCMD.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

Example: C-35 RSTRTZFS job output

```
IEA630I  OPERATOR HERING   NOW ACTIVE,   SYSTEM=SC43    , LU=ZFSJOB
 D R,R
 IEE112I 20.55.28 PENDING REQUESTS 314
 RM=3    IM=0     CEM=0     EM=0     RU=0    IR=0    AMRF
 ID:R/K     T MESSAGE TEXT
        048 R *048 BPXF032D FILESYSTYPE ZFS TERMINATED.  REPLY 'R' WHEN
               READY TO RESTART. REPLY 'I' TO IGNORE.
        907 R *907 DFS996I *IMS READY*  IMSA
        763 R *763 VAINIVPW.IEFPROC   WTR READY,CLASSES= V
 R 048,R
 IEE600I REPLY TO 048 IS;R
 IEF196I         1 //ZFS      JOB MSGLEVEL=1
 IEF196I         2 //STARTING EXEC ZFS
 IEF196I  STMT NO. MESSAGE
 IEF196I         2 IEFC001I PROCEDURE ZFS WAS EXPANDED USING SYSTEM
 IEF196I LIBRARY SYS1.PROCLIB
...
 IEF403I ZFS - STARTED - TIME=20.55.30 - ASID=01F8 - SC43
 IEF196I IEF236I ALLOC. FOR ZFS ZFS
 IEF196I IEF237I 3E17 ALLOCATED TO IOEZPRM
 IOEZ00052I zFS kernel: Initializing z/OS    zSeries File System
```

```
 Version 01.03.00 Service Level OW53952.
 Created on Wed Mar 27 17:02:19 EST 2002.
 IOEZ00178I ZFS.SC43.IOEFSZFS(IOEFSPRM) is the
 configuration dataset currently in use.
...
 IOEZ00044I Aggregate OMVS.MUL01.ZFS attached successfully.
 IOEZ00044I Aggregate OMVS.MUL02.ZFS attached successfully.
...
 IOEZ00055I zFS kernel: initialization complete.
...
```

Note: The pseudo MVS system command, **RESTART ZFS**, is used to trigger special code in the CN REXX procedure that looks for the BPXF032D ZFS termination message and replies to it with an R xx command.

C.5.4 Display all zFS counters

Example: C-36 ZFSQUERY JCL

```
//ZFSJOB   JOB ,'ZFSQUERY',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Run F ZFS (zfsproc) commands and retrieve output
//* Property of IBM  (C) Copyright IBM Corp. 2002-2005
//* ------------------------------------------------------------------
//  SET ZFSPROC=ZFS                              <=== zFS procname
//  SET ZFSPARM='QUERY,ALL'                      <=== "F" parameter
//  SET CONSOLE=''                               <=== Console name
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//CONSCMND EXEC PGM=IKJEFT01,
//  PARM='CNFZFS C=&CONSOLE. &ZFSPROC. &ZFSPARM.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//*
--------------------------------------------------------------------------
```

C.6 Compare HFS and zFS

The following jobs are used for the comparison between the HFS and zFS file systems.

C.6.1 Define the HFS data set

Example: C-37 LARGEHFS JCL

```
//ZFSJOB   JOB ,'Define HFS File',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Define HFS File
//* Property of IBM  (C) Copyright IBM Corp. 1999, 2002
//* -------------------------------------------------------------------
//*
//  SET  HFSNAME=OMVS.LARGE.HFS            <=== HFS data set name
//  SET   VOLSER=TOTZF2                    <=== Volume
//  SET   HFSPRM=750                       <=== HFS primary allocation
//  SET   HFSSEC=50                        <=== HFS secon'y allocation
//*
//* -------------------------------------------------------------------
//CREATE   EXEC PGM=IEFBR14
//HFS      DD DSN=&HFSNAME.,DISP=(NEW,CATLG,DELETE),UNIT=SYSALLDA,
//            DCB=(DSORG=PO),SPACE=(CYL,(&HFSPRM.,&HFSSEC.,0)),
//            DSNTYPE=HFS,VOL=SER=&VOLSER.
//* -------------------------------------------------------------------
```

C.6.2 Create the zFS aggregate

Example: C-38 LARGEZFS JCL

```
//ZFSJOB   JOB ,'LARGEZFS',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Define, format and mount a compatibility mode aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* -------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* -------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# -------------------------------------------------------------------
  AGGREGATE=OMVS.LARGE.ZFS
  STORCLASS=SCCOMP
     VOLUME=TOTZF2
     ZFSPRM=550
     ZFSSEC=50
      OWNER=HERING
      PERMS=o755
# -------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# -------------------------------------------------------------------
C="zfsadm define -aggregate $AGGREGATE -storageclass $STORCLASS -
  -megabytes $ZFSPRM $ZFSSEC"+
C="zfsadm define -aggregate $AGGREGATE -volume $VOLUME -
  -megabytes $ZFSPRM $ZFSSEC"; echo $C;$C+
C="zfsadm format -aggregate $AGGREGATE -compat -owner $OWNER -
  -perms $PERMS"; echo $C;$C+
```

```
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

C.6.3 Mount the file systems

Example: C-39 LARGEMNT JCL

```
//ZFSJOB   JOB ,'LARGEMNT',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Define, format and mount a compatibility mode aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='RXZFS &TIMEOUT.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ------------------------------------------------------------------
   HFS_NAME=OMVS.LARGE.HFS
   HFS_MNTP=/u/zfs/largehfs
   ZFS_NAME=OMVS.LARGE.ZFS
   ZFS_MNTP=/u/zfs/largezfs
      OWNER=HERING
      PERMS=755
# ------------------------------------------------------------------
##
//STDIN    DD DATA,DLM=##
# ------------------------------------------------------------------
C="mkdir -m 755 $HFS_MNTP"; echo $C;$C+
C="/usr/sbin/mount -o sync(60) -f $HFS_NAME -t HFS $HFS_MNTP"; -
  echo $C;$C+
C="chmod 755 $HFS_MNTP"; echo $C;$C+
C="ls -Ed $HFS_MNTP"; echo $C;$C+
C="df -kvP $HFS_MNTP"; echo $C;$C+

C="mkdir -m 755 $ZFS_MNTP"; echo $C;$C+
C="/usr/sbin/mount -o noreadahead -f $ZFS_NAME -t ZFS $ZFS_MNTP"; -
  echo $C;$C+
C="ls -Ed $ZFS_MNTP"; echo $C;$C+
C="df -kvP $ZFS_MNTP"; echo $C;$C+
# ------------------------------------------------------------------
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

Note: When you define a variable and then use this variable to run a z/OS UNIX command, you must not use quotes because they become part of the command. So, instead of using **-o 'sync(60)'**, specify **-o sync(60)**.

C.6.4 Create a large file in a file system

Example: C-40 LARGEFIL JCL (HFS sample)

```
//ZFSJOB   JOB ,'LARGEFIL',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Create large file in a file system and fill it with data
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET DIRECTRY='/u/zfs/largehfs'               <=== Source directoy
//* ------------------------------------------------------------------
//* SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='LARGEFIL &DIRECTRY.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

C.6.5 Test with a specified amount of large random I/Os

Example: C-41 LARGEIOS JCL (zFS sample)

```
//ZFSJOB   JOB ,'LARGEIOS',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Run test with doing a specied amount of large random I/Os
//* Property of IBM  (C) Copyright IBM Corp. 2002
//* ------------------------------------------------------------------
//  SET    FDIR='/u/zfs/largezfs'         <=== Directory of large_file
//  SET   PROCS=10         <=== Number of parallel processes to start
//  SET  BLKIOS=200        <=== Number of 1MB block I/Os to perform
//  SET  TYPIOS=R70        <=== Type of IOs, Rxx: xx% Rs, (100-xx)% Ws
//  SET    SEED=88888      <=== Seed value for predictable random nbrs
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,
// PARM='LARGESCD &TIMEOUT &REXXLIB &PROCS &BLKIOS &TYPIOS &SEED &FDIR'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=V
//* ------------------------------------------------------------------
```

C.7 JCL for an aggregate backup and restore

Example: C-42 Sample JCL for zFS backup of an aggregate

```
//ZFSBKUPO JOB ,'zFS Backup',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Backup a zFS aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002-2008
//* ------------------------------------------------------------------
//  SET AGGRNAME=OMVS.HERING.TEST.ZFS              <=== zFS aggregate
//  SET AGGRDUMP=OMVS.HERING.TEST.ZFS.DUMP         <=== zFS dmp ds
//  SET     DISP='NEW,CATLG,DELETE'                <=== zFS dmp disp
//  SET    SPACE='CYL,(20,10),RLSE'                <=== zFS dmp space
//* ------------------------------------------------------------------
```

```
//  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
//* ------------------------------------------------------------------
//* Copy REXX Code
//* ------------------------------------------------------------------
//REXXCOPY EXEC PGM=IKJEFT01,REGION=0M,
//           PARM='OCOPY INDD(REXXSRC) OUTDD(REXXCODE)'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//REXXSRC  DD DATA,DLM=##
/* REXX - Create SYSIN Data Sets */
Trace O
Parse Upper Arg zfs_aggregate
line.1 = "  DUMP DATASET(INCLUDE("||zfs_aggregate||")) -"
line.2 = "    COMPRESS OUTDDNAME(AGGRSEQ) TOL(ENQF)"
line.0 = 2
"EXECIO" line.0 "DISKW SYSIN (STEM LINE. FINIS"
Exit rc
##
//REXXCODE DD DSN=&&TEMPLIB(CRESYSIN),DISP=(,PASS),UNIT=SYSALLDA,
//           LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,1))
//* ------------------------------------------------------------------
//* Create SYSIN Files using REXX Code
//* ------------------------------------------------------------------
//  IF REXXCOPY.RC EQ 0 THEN
//CRESYSIN EXEC PGM=IKJEFT01,PARM='CRESYSIN &AGGRNAME'
//SYSPROC  DD DSN=*.REXXCOPY.REXXCODE,DISP=(OLD,PASS)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//SYSIN    DD DSN=&&SYSINDU,DISP=(,PASS),UNIT=SYSALLDA,LRECL=80,
//           RECFM=FB,SPACE=(TRK,(1,1,0))
//* ------------------------------------------------------------------
//* This step quiesces, dumps and unquieces the aggregate
//* ------------------------------------------------------------------
//  IF CRESYSIN.RC EQ 0 THEN
//DUMP    EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//AGGRSEQ  DD  DSN=&AGGRDUMP.,DISP=(&DISP.),SPACE=(&SPACE.)
//SYSIN    DD DSN=*.CRESYSIN.SYSIN,DISP=(OLD,DELETE)
//  ENDIF
//* ------------------------------------------------------------------
//  ENDIF
//* ------------------------------------------------------------------
```

Attention: The original Quiesce and Unquiesce steps of the job shown in Example C-42 on page 473 have been removed because of the zFS catalog support.

See 4.3.4, “Catalog indication for zFS aggregates” on page 179 for more information about zFS catalog support.

Example: C-43 Sample JCL to restore a zFS aggregate

```
//ZFSRESTO JOB ,'zFS Restore',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Backup a zFS aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002-2008
//* ------------------------------------------------------------------
//  SET AGGRORIG=OMVS.HERING.TEST.ZFS              <=== zFS original
```

```
//  SET  AGGRNEW=OMVS.HERING.TEST.RST.ZFS           <=== zFS restored
//  SET AGGRDUMP=OMVS.HERING.TEST.ZFS.DUMP          <=== zFS dmp ds
//* ------------------------------------------------------------------
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS.REXX.EXEC              <=== SYSEXEC library
//* ------------------------------------------------------------------
//* Copy REXX Code
//* ------------------------------------------------------------------
//REXXCOPY EXEC PGM=IKJEFT01,REGION=0M,
//           PARM='OCOPY INDD(REXXSRC) OUTDD(REXXCODE)'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//REXXSRC  DD DATA,DLM=##
/* REXX - Create SYSIN Data Sets */
Trace O
Parse Upper Arg zfs_aggr_orig zfs_aggr_new
line.1 = "  RESTORE DATASET(INCLUDE(**)) -"
line.2 = "    CATALOG -"
line.3 = "    RENAMEU( -"
line.4 = "    ("||zfs_aggr_orig||", -"
line.5 = "     "||zfs_aggr_new||") -"
line.6 = "    ) -"
line.7 = "  WRITECHECK -"
line.8 = "  INDDNAME(AGGRSEQ)"
line.0 = 8
"EXECIO" line.0 "DISKW SYSIN (STEM LINE. FINIS"
Exit rc
##
//REXXCODE DD DSN=&&TEMPLIB(CRESYSIN),DISP=(,PASS),UNIT=SYSALLDA,
//           LRECL=80,RECFM=FB,SPACE=(TRK,(1,1,1))
//* ------------------------------------------------------------------
//* Create SYSIN Files using REXX Code
//* ------------------------------------------------------------------
//  IF REXXCOPY.RC EQ 0 THEN
//CRESYSIN EXEC PGM=IKJEFT01,
// PARM='CRESYSIN &AGGRORIG. &AGGRNEW.'
//SYSPROC  DD DSN=*.REXXCOPY.REXXCODE,DISP=(OLD,PASS)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//SYSIN    DD DSN=&&SYSINDU,DISP=(,PASS),UNIT=SYSALLDA,LRECL=80,
//           RECFM=FB,SPACE=(TRK,(1,1,0))
//* ------------------------------------------------------------------
//* This step restores the aggregate.
//* ------------------------------------------------------------------
//  IF CRESYSIN.RC EQ 0 THEN
//RESTORE EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//AGGRSEQ  DD DSN=&AGGRDUMP.,DISP=SHR
//SYSIN    DD DSN=*.CRESYSIN.SYSIN,DISP=(OLD,DELETE)
//  ENDIF
//* ------------------------------------------------------------------
//  ENDIF
//* ------------------------------------------------------------------
```

C.8 Comparison in UNIX System Services sysplex sharing environments

The following jobs are used for the comparison in UNIX System Services sysplex file system sharing environments.

C.8.1 Define the HFS data set

Example: C-44 LARGEHFS JCL (version 2)

```
//ZFSJOB   JOB ,'Define HFS File',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Define HFS File
//* Property of IBM  (C) Copyright IBM Corp. 2002-2011
//* -------------------------------------------------------------------
//*
//  SET  HFSNAME=OMVS.R13TEST.LARGE.HFS    <=== HFS data set name
//  SET STORCLAS=OPENMVS                   <=== Storage class
//  SET   VOLSER=BH5OE2                    <=== Volume
//  SET   HFSPRM=2000                      <=== HFS primary allocation
//  SET   HFSSEC=100                       <=== HFS secon'y allocation
//*
//* -------------------------------------------------------------------
//CREATE   EXEC PGM=IEFBR14
//HFS      DD DSN=&HFSNAME.,DISP=(NEW,CATLG,DELETE),UNIT=SYSALLDA,
//            DCB=(DSORG=PO),SPACE=(CYL,(&HFSPRM.,&HFSSEC.,0)),
//            DSNTYPE=HFS,STORCLAS=&STORCLAS.,VOL=SER=&VOLSER.
//* -------------------------------------------------------------------
```

C.8.2 Create the zFS aggregate

Example: C-45 LARGEZFS JCL (version 2)

```
//ZFSJOB   JOB ,'LARGEZFS',NOTIFY=&SYSUID.,REGION=0M
//* -------------------------------------------------------------------
//* Define, format and mount a compatibility mode aggregate
//* Property of IBM  (C) Copyright IBM Corp. 2002-2011
//* -------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC             <=== SYSEXEC library
//* -------------------------------------------------------------------
//RXBATCH  EXEC PGM=IKJEFT01,PARM=RXBATCH
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDIN    DD DATA,DLM=##
echo "-> Defining zFS aggregate $AGGREGATE..."
zfsadm define -aggregate $AGGREGATE -storageclass $STORCLASS \
  -megabytes $ZFSPRM $ZFSSEC -volume $VOLUME
echo "\n-> Formatting aggregate $AGGREGATE..."
zfsadm format -aggregate $AGGREGATE -compat -owner $OWNER \
  -perms $PERMS
##
//STDENV   DD DATA,DLM=##
  AGGREGATE=OMVS.R13TEST.LARGE.ZFS
  STORCLASS=OPENMVS
     VOLUME=BH5OE2
     ZFSPRM=2000
     ZFSSEC=100
      OWNER=HERING
      PERMS=o755
##
```

```
//RXBPARM  DD DATA,DLM=##
 _RXBATCH_SWSU=1       < 0= no switch (default), 1= switch to SU mode
_RXBATCH_LOGIN=0       < 0= no login shell, 1= login shell (default)
   _RXBATCH_SL=1       < 0= use BPXBATSL if in SU mode, 1= use always
_RXBATCH_CP=IBM-1047   < code page of STDIN data, default is IBM-1047
##
//STDOUT   DD SYSOUT=*,LRECL=136,RECFM=VB
//SYSTSIN  DD DUMMY
//SYSTSPRT DD DUMMY    < Use "SYSOUT=*" instead in case of problems
//*STDPARM  DD ...     < Do not add a STDPARM DD statement to this JCL
//* --------------------------------------------------------------------
```

C.8.3 Mount the file systems

Example: C-46 LARGEMNT JCL (version 2)

```
//ZFSJOB   JOB ,'LARGEMNT',NOTIFY=&SYSUID.,REGION=0M
//* --------------------------------------------------------------------
//* Create mountpoint directories  and mount file systems
//* Property of IBM  (C) Copyright IBM Corp. 2009-2011
//* --------------------------------------------------------------------
//  SET REXXLIB=HERING.ZFS.REXX.EXEC                  SYSEXEC library
//* --------------------------------------------------------------------
//RXBATCH  EXEC PGM=IKJEFT01,PARM=RXBATCH
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDIN    DD DATA,DLM=##
echo "-> Creating directory $HFS_MNTP..."
mkdir -pm 755 $HFS_MNTP
echo "\n-> Mounting HFS file system $HFS_NAME..."
/usr/sbin/mount -o "sync(60)" -f $HFS_NAME -t HFS $HFS_MNTP
chmod 755 $HFS_MNTP
echo "\n->Listing information about the HFS mount point and"\
  "file system..."
ls -Ed $HFS_MNTP
df -kvP $HFS_MNTP
echo "\n-> Creating directory $ZFS_MNTP..."
mkdir -m 755 $ZFS_MNTP
echo "\n-> Mounting zFS file system $ZFS_NAME..."
/usr/sbin/mount -o "noreadahead rwshare" -f $ZFS_NAME \
  -t ZFS $ZFS_MNTP
echo "\n-> Listing information about the zFS mount point and"\
  "file system..."
ls -Ed $ZFS_MNTP
df -kvP $ZFS_MNTP
##
//STDENV   DD DATA,DLM=##
  HFS_NAME=OMVS.R13TEST.LARGE.HFS
  HFS_MNTP=/u/hering/r13test/hfs
  ZFS_NAME=OMVS.R13TEST.LARGE.ZFS
  ZFS_MNTP=/u/hering/r13test/zfs
     OWNER=HERING
     PERMS=755
##
//RXBPARM  DD DATA,DLM=##
 _RXBATCH_SWSU=1       < 0= no switch (default), 1= switch to SU mode
_RXBATCH_LOGIN=0       < 0= no login shell, 1= login shell (default)
   _RXBATCH_SL=1       < 0= use BPXBATSL if in SU mode, 1= use always
_RXBATCH_CP=IBM-1047   < code page of STDIN data, default is IBM-1047
##
//STDOUT   DD SYSOUT=*,LRECL=136,RECFM=VB
```

```
//SYSTSIN  DD DUMMY
//SYSTSPRT DD DUMMY    < Use "SYSOUT=*" instead in case of problems
//*STDPARM  DD ...     < Do not add a STDPARM DD statement to this JCL
//* ------------------------------------------------------------------
```

C.8.4 Create a large file in a file system

Example: C-47 LARGEFIL JCL (version 2, zFS file 1 sample)

```
//ZFSJOB   JOB ,'LARGEFIL',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Create large file in a file system and fill it with data
//* Property of IBM  (C) Copyright IBM Corp. 2002-2011
//* ------------------------------------------------------------------
//  SET DIRECTRY='/u/hering/r13test/zfs'          <=== Source directoy
//  SET  FNUMBER='1'                              <=== File number
//* ------------------------------------------------------------------
//* SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS13.REXX.EXEC            <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,PARM='LARGEFIL &DIRECTRY. &FNUMBER.'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=260,RECFM=VB
//* ------------------------------------------------------------------
```

C.8.5 Test with a specified amount of large random I/Os

Example: C-48 LARGEIOS JCL (version 2, zFS file 1 sample)

```
//ZFSJOBZ1 JOB ,'LARGEIOS',NOTIFY=&SYSUID.,REGION=0M,TYPRUN=HOLD
//*JOBPARM SYSAFF=SC74
//*MAIN SYSTEM=SC75
//* ------------------------------------------------------------------
//* Run test with doing a specied amount of large random I/Os
//* Property of IBM  (C) Copyright IBM Corp. 2002-2009
//* ------------------------------------------------------------------
//  SET    FDIR='/u/hering/r13test/zfs/large_file_1' <=large file name
//  SET   PROCS=20         <=== Number of parallel processes to start
//  SET  BLKIOS=200        <=== Number of 1MB block I/Os to perform
//  SET  TYPIOS=R70        <=== Type of IOs, Rxx: xx% Rs, (100-xx)% Ws
//  SET    SEED=88888      <=== Seed value for predictable random nbrs
//  SET TIMEOUT=0          <=== Timeout value in seconds, 0=no timeout
//  SET REXXLIB=HERING.ZFS13.REXX.EXEC            <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSADM   EXEC PGM=IKJEFT01,
// PARM='LARGESCD &TIMEOUT &REXXLIB &PROCS &BLKIOS &TYPIOS &SEED &FDIR'
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* ------------------------------------------------------------------
```

D

zFS performance data

This appendix contains the output from the zFS query command to obtain performance data on the running system.

D.1 Output data of a modify zfs command

Example D-1 displays the output from a **f zfs,query,all** command.

*Example: D-1 Output from the **f zfs,query,all** command*

```
IEA630I  OPERATOR HERING   NOW ACTIVE,   SYSTEM=SC43    , LU=ZFSJOB
 F ZFS,QUERY,ALL
          zFS Kernel USS PFS Calls
          ------------------------
 Operation              Count          Avg Time
 ---------          ----------        ----------
 zfs_opens                 40           506.047
 zfs_closes                40          2375.650
 zfs_reads               5600          4463.577
 zfs_writes              2400          5486.785
 zfs_ioctls                 0             0.000
 zfs_getattrs               0             0.000
 zfs_setattrs               0             0.000
 zfs_accesses              39             0.028
 zfs_lookups                1             0.164
 zfs_creates                0             0.000
 zfs_removes                0             0.000
 zfs_links                  0             0.000
 zfs_renames                0             0.000
 zfs_mkdirs                 0             0.000
 zfs_rmdirs                 0             0.000
 zfs_readdirs               0             0.000
 zfs_symlinks               0             0.000
 zfs_readlinks              0             0.000
 zfs_fsyncs                 0             0.000
 zfs_truncs                 0             0.000
 zfs_lockctls               0             0.000
 zfs_audits                 0             0.000
 zfs_inactives              1             0.013
 zfs_recoverys              0             0.000
 zfs_vgets                  0             0.000
 zfs_pfsctls                0             0.000
 zfs_statfss                0             0.000
 zfs_mounts                 0             0.000
 zfs_unmounts               0             0.000
 ---------          ----------        ----------
 *TOTALS*                8121          4713.654

 Number of service threads: 10 (stacksize=OK)
 Requests: 0    Queued: 0 (  0.0%)

                  User File (VM) Caching System Statistics
                  ----------------------------------------

 External Requests:
 ------------------
 Reads           5599     Fsyncs             0     Opens              1
 Writes          2400     Setattrs           0     Unmaps             0
 Asy Reads          0     Getattrs           1     Schedules         40    Flushes 0

 File System Reads:
 ------------------
 Reads Faulted       42275     (Fault Ratio  755.45%)
 Writes Faulted          0     (Fault Ratio    0.00%)
```

```
Read Waits              0     (Wait Ratio    0.00%)
Total Reads         42171

File System Writes:
-------------------
Scheduled Writes         38397     Sync  Waits            0
Error Writes                 0     Error Waits            0
Scheduled deletes            0
Page Reclaim Writes          0     Reclaim Waits      10744
Write Waits                 10     (Wait Ratio    0.416%)

File Management: (File struct size=168)
---------------------------------------
Max Files:         8192     Allocated:         8192
Lookups          1     Hits            1 (Hit Ratio    100.00%)

Page Management (Segment Size = 64K) (Page Size = 4K)
-----------------------------------------------------
Total Pages       65536     Free                 32
Segments           8192
Steal Invocations      29323     Waits for Reclaim        4186

Number of dataspaces used:     4  Pages per dataspace:       16384

Dataspace   Allocated     Free
Name        Segments      Pages
--------    ----------    ----------
ZFSUCD00          1024             0
ZFSUCD01          1023            16
ZFSUCD02          1024             0
ZFSUCD03          1023            16

                              zFS Vnode Op Counts

Vnode Op                        Count   Vnode Op                        Count
------------------------  ----------   ------------------------  ----------
efs_hold                             0   efs_readdir                          0
efs_rele                             0   efs_create                           0
efs_inactive                         0   efs_remove                           0
efs_getattr                          1   efs_rename                           0
efs_setattr                          0   efs_mkdir                            0
efs_access                          79   efs_rmdir                            0
efs_lookup                           1   efs_link                             0
efs_getvolume                        0   efs_symlink                          0
efs_getlength                        0   efs_readlink                         0
efs_afsfid                           0   efs_rdwr                             0
efs_fid                              0   efs_fsync                            0
efs_vmread                       42170   efs_waitIO                       38345
efs_vmwrite                      38398   efs_cancelIO                         0
efs_clrsetid                         0   efs_audit                            0
efs_atime                            0   efs_vmblkinfo                   246960

Total zFS Vnode Ops     365954

                           LFS Vnode Cache Statistics

Vnodes      Requests     Hits    Ratio  Allocates   Deletes
---------- ---------- ---------- ----- ---------- ----------
     16384          1          1 100.0%          0          0
```

```
LFS Vnode structure size: 296 bytes
SAF Access Cache Requests:          0   hits:          0 (hit ratio   0.0%)

                       Metadata Caching Statistics

Buffers    (K bytes)  Requests     Hits    Ratio  Updates
---------- --------- ---------- ---------- ----- ----------
      4096     32768     721635     721635 100.0%      77000

                        Directory Cache Statistics

Dir Blocks (K bytes)  Requests     Hits    Ratio  Deletes
---------- --------- ---------- ---------- ----- ----------
       128      1024          0          0   0.0%          0

                       Transaction Cache Statistics

Transactions started:      38439    Lookups on tran:     269004    EC Merges:          0
Allocated Transactions:       2000  (Act=         0, Pend=         0, Comp=      1137,
Free=       863)


                I/O Summary By Type
                -------------------

Count       Waits       Cancels     Merges      Type
----------  ----------  ----------  ----------  ----------
        16          10           0           0  File System Metadata
       676          17           0         581  Log File
     80571       53680        7527           0  User File Data

                I/O Summary By Circumstance
                ---------------------------

Count       Waits       Cancels     Merges      Circumstance
----------  ----------  ----------  ----------  ------------
         0           0           0           0  Metadata cache read
     42171       42170           0           0  User file cache direct read
         0           0           0           0  Log file read
         0           0           0           0  Metadata cache async delete write
         0           0           0           0  Metadata cache async write
         0           0           0           0  Metadata cache lazy write
         0           0           0           0  Metadata cache sync delete write
         0           0           0           0  Metadata cache sync write
     38400       11508        7527           0  User File cache direct write
         0           0           0           0  Metadata cache file sync write
        16          10           0           0  Metadata cache sync daemon write
         0           0           0           0  Metadata cache aggregate detach write
         0           0           0           0  Metadata cache buffer block reclaim write
         0           0           0           0  Metadata cache buffer allocation write
         0           0           0           0  Metadata cache file system quiesce write
         0           0           0           0  Metadata cache log file full write
       676          17           0         581  Log file write
         0           0           0           0  Metadata cache shutdown write

                    zFS I/O by Currently Attached Aggregate

DASD   PAV
VOLSER IOs Mode       Reads     K bytes      Writes     K bytes  Dataset Name
------ --- ----  ----------  ----------  ----------  ----------  ------------
```

```
TOTZF1   1  R/W           0           0           0           0  OMVS.MUL01.ZFS
TOTZF1   1  R/W           0           0           0           0  OMVS.MUL02.ZFS
TOTZF1   1  R/W           0           0           0           0  OMVS.MULTUSER.ZFS
TOTZF2   1  R/W       42171     2698944       30984     1981408  OMVS.LARGE.ZFS
------           ----------  ----------  ----------  ----------
     4               42171     2698944       30984     1981408  *TOTALS*


Total number of waits for I/O:      53707
Average I/O wait time:                48.448 (msecs)

                                   Locking Statistics


Untimed sleeps:          0   Timed Sleeps:          0   Wakeups:          0

Total waits for locks:                          856536
Average lock wait time:             41.075 (msecs)

Total monitored sleeps:                           4186
Average monitored sleep time:            105.836 (msecs)

                Top 15 Most Higly Contended Locks
  Thread      Async       Spin
   Wait       Disp.       Resol.     Pct.    Description
----------  ----------  ----------   -----   --------------
    850754           0          97    97.892%   Vnode lock
         0        9295           0     1.69%   Volser I/O queue lock
      3207           0           0     0.368%   User-cache segment lock
      1121           0         922     0.235%   Anode handle lock
      1588           0           0     0.182%   Vnode-cache access lock
         0        1550           0     0.178%   OSI Global queue of threads waiting for
locks
         3           0         433     0.50%   Metadata-cache buffer lock
        14           2          83     0.11%   User file cache main segment lock
         4          74           3     0.9%   Unclassified locks
         2           0           8     0.1%   Async global device lock
         5           0           0     0.0%   Log system map lock
         2           0           0     0.0%   User-cache segment slot lock
         0           2           0     0.0%   Cache Services association main lock
         1           0           0     0.0%   User-cache segment association lock
         0           0           0     0.0%   Async IO device lock

Total lock contention of all kinds:     869170

                Top 5 Most Common Thread Sleeps
Thread Wait     Pct.      Description
-----------     -----     -----------
       4186    100.0%     User file cache Page Reclaim Wait
          0      0.0%     OSI cache item cleanup wait
          0      0.0%     Directory Cache Buffer Wait
          0      0.0%     User file cache Page Wait
          0      0.0%     User file cache File Wait

                zFS Primary Address Space Storage Usage
                ---------------------------------------

Total Bytes Allocated: 96837890 (94568K) (92M)
Total Pieces Allocated: 126061
Total Allocation Requests: 578
```

```
 Total Free Requests: 0

             Storage Usage By Component
             --------------------------
 Bytes               No. of  No. of
 Allocated   Pieces  Allocs  Frees   Component
 ----------  ------  ------  ------  ---------

        472       4       0       0  USS Interface
     595320   27849     399       0  Media Manager I/O driver
   33555876       4       0       0  Trace Facility
     266156       3       0       0  Message Service
      48692      68       0       0  Miscellaneous
       1064      13       0       0  Aggregate Management
     107283      35       0       0  Filesystem Management
       9680      15       0       0  Administration Command Handling
    5312188      17       0       0  Vnode Management
    7692820   16450       0       0  Anode Management
    1074892       9       0       0  Directory Management
    7145144     739      26       0  Log File Management
   34393968    4113       0       0  Metadata Cache
     372208      13       0       0  Transaction Management
    1563080    4577     153       0  Asynchronous I/O Component
      51660     199       0       0  Lock Facility
       4576      83       0       0  Threading Services
     231687    6233       0       0  Cache Services
      10932       5       0       0  Configuration parameters processing
    4370824   65590       0       0  User File Cache
      29368      42       0       0  Storage Management

 File System Name                                Aggr #  Flg  Operations
 ----------------------------------------------  ------  ---  ----------
 OMVS.LARGE.ZFS                                  100003  AM         8120

 IOEZ00025I zFS kernel: MODIFY command - QUERY,ALL completed successfully.
READY
END
```

D.2 Output data of a zfsadm query command

Example D-2 displays the output from a **zfsadm query -usercache** command.

*Example: D-2 Output from a **zfsadm query -usercache** command*

```
$> zfsadm query -usercache
                  User File (VM) Caching System Statistics
                  ----------------------------------------

                  Direct Statistics
                  -----------------

External Requests:
------------------
Reads          10049     Fsyncs             2     Schedules     300006
Writes        300006     Setattrs      101975     Unmaps        300000
Asy Reads       5037     Getattrs     1052692     Flushes            0

File System Reads:
```

```
------------------
Reads Faulted         510     (Fault Ratio    5.75%)
Writes Faulted          0     (Fault Ratio    0.00%)
Read Waits              0     (Wait Ratio    0.00%)
Total Reads           510

File System Writes:
-------------------
Scheduled Writes        300006     Sync  Waits            0
Error Writes                 0     Error Waits            0
Scheduled deletes            0
Page Reclaim Writes          0     Reclaim Waits          0
Write Waits                  0     (Wait Ratio    0.00%)
                  Client Statistics
                  -----------------

External Requests:
------------------
Reads             0     Fsyncs             0     Schedules          0
Writes            0     Setattrs           0     Unmaps             0
Asy Reads         0     Getattrs           0     Flushes            0

File System Reads:
------------------
Reads Faulted           0     (Fault Ratio    0.00%)
Writes Faulted          0     (Fault Ratio    0.00%)
Read Waits              0     (Wait Ratio    0.00%)
Total Reads             0

File System Writes:
-------------------
Scheduled Writes             0     Sync  Waits            0
Error Writes                 0     Error Waits            0
Scheduled deletes            0
Page Reclaim Writes          0     Reclaim Waits          0
Write Waits                  0     (Wait Ratio    0.00%)

Page Management (Segment Size = (64K Local 32K Remote) ) (Page Size = 4K)
-------------------------------------------------------------------------
Total Pages           393216     Free                   393212
Segments                   0
Steal Invocations          0     Waits for Reclaim           0

Number of dataspaces used:    24  Pages per dataspace:       16384

Dataspace   Allocated     Free
Name        Segments      Pages
--------    ----------    ----------
ZFSUCD00             1         16382
ZFSUCD01             0         16384
ZFSUCD02             0         16384
ZFSUCD03             0         16384
ZFSUCD04             0         16384
ZFSUCD05             0         16384
ZFSUCD06             0         16384
ZFSUCD07             0         16384
ZFSUCD08             0         16384
ZFSUCD09             0         16384
ZFSUCD0A             0         16384
ZFSUCD0B             1         16382
```

```
ZFSUCD0C             0         16384
ZFSUCD0D             0         16384
ZFSUCD0E             0         16384
ZFSUCD0F             0         16384
ZFSUCD10             0         16384
ZFSUCD11             0         16384
ZFSUCD12             0         16384
ZFSUCD13             0         16384
ZFSUCD14             0         16384
ZFSUCD15             0         16384
ZFSUCD16             0         16384
ZFSUCD17             0         16384
```

D.3 Output data of a z/OS V1R7 zfsadm query command

Example D-3 displays the output from a **zfsadm query -knpfs** command.

*Example: D-3 Output from a **zfsadm query -knpfs** command*

```
$> zfsadm query -knpfs
          zFS Kernel PFS Calls
          --------------------
Operation               Count          Avg Time
---------          ----------        ----------
zfs_opens               12102             0.009
zfs_removes             12102             0.902
zfs_reads                   0             0.000
zfs_writes              12092             0.052
zfs_ioctls                  0             0.000
zfs_getattrs              103             0.009
zfs_setattrs                0             0.000
zfs_accesses            12099             0.013
zfs_lookups             24196             0.045
zfs_creates             12092             0.883
zfs_links                   0             0.000
zfs_renames                 0             0.000
zfs_mkdirs                  1             0.567
zfs_rmdirs                  2             2.438
zfs_readdirs              179             0.414
zfs_symlinks                0             0.000
zfs_readlinks               0             0.000
zfs_fsyncs                  0             0.000
zfs_truncs                  0             0.000
zfs_lockctls                0             0.000
zfs_audits                  0             0.000
zfs_inactives           24186             0.007
zfs_recoveries              0             0.000
zfs_vgets                   1             0.024
zfs_pfsctls            256402             0.182
zfs_statfss                44             0.019
zfs_mounts                 16           261.476
zfs_unmounts                1           100.341
zfs_vinacts                 0             0.000
------------       ----------        ----------
*TOTALS*               377711             0.211
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this IBM Redbooks publication.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 487. Note that some of the documents referenced here may be available in softcopy only.

- *z/OS Version 1 Release 5 Implementation*, SG24-6326
- *z/OS Version 1 Release 2 Implementation*, SG24-6235
- *z/OS Version 1 Release 6 Implementation*, SG24-6377
- *z/OS Version 1 Release 3 and 4 Implementation*, SG24-6581
- *UNIX System Services z/OS Version 1 Release 7 Implementation*, SG24-7035
- *z/OS Version 1 Release 9 Implementation*, SG24-7427

Other publications

These publications are also relevant as further information sources:

- *z/OS Planning for Multilevel Security,* GA22-7509
- *z/OS UNIX System Services Planning,* GA22-7800
- *z/OS Security Server RACF Security Administrator’s Guide,* SA22-7683
- *z/OS UNIX System Services Command Reference,* SA22-7802
- *z/OS Distributed File Service Messages and Codes,* SC24-5917
- *z/OS Distributed File Service zSeries File System Administration*, SC24-5989

Referenced websites

This website is also relevant as further information source:

UNIX System Services Tools and Toys:

`http://www-1.ibm.com/servers/eserver/zseries/zos/unix/bpxa1toy.html`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following website:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks website for information about all the CD-ROMs offered, as well as updates and formats.

Index

D

F

G

H

I

L

M

N

O

P

Q

R

S

T

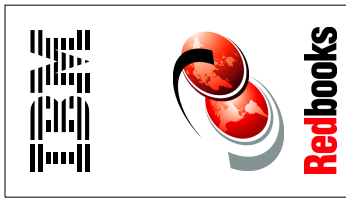
U

V

W

X

Z



z/OS Distributed File Service zSeries File System Implementation z/OS V1R13

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

z/OS Distributed File Service zSeries File System Implementation z/OS V1R13

Defining and installing a zSeries file system

Performing backup and recovery, sysplex sharing

Migrating from HFS to zFS

The z/OS Distributed File Service zSeries File System (zFS) is a z/OS UNIX file system that can be used like the Hierarchical File System (HFS). zFS file systems contain files and directories, including Access Control Lists (ACLs), that can be accessed with the z/OS HFS 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, and so on). zFS does not replace HFS, but it is the z/OS UNIX strategic file system and IBM recommends migrating HFS file systems to zFS. Beginning with z/OS V1R7, there are no restrictions for file system structures that should be kept as HFS instead of zFS.

This IBM Redbooks publication helps you to install, tailor, and configure new zFS file systems. This information can be used by system administrators who work with the zFS component of the IBM z/OS Distributed File Service base element.

The book provides a broad description of the new architecture of the zFS file system for all releases up to zFS V1R13. You can use it as a reference when converting HFS file systems to zFS file systems. It will help you to create a solution for migrating to zFS file systems, and to understand the performance differences between HFS file systems and zFS file systems.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6580-05 ISBN 0738436402