IBM

# zFS Reorganization Tool

- Installing the reorganization tool
- Using the reorganization tool
- Reorganization tool REXX execs

Paul Rogers
Robert Hering

Redpaper

ibm.com/redbooks

International Technical Support Organization

# zFS Reorganization Tool

January 2012

REDP-4769-00

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

**First Edition (January 2012)**

This edition applies to Distributed File Service (DFS/SMB/zFS) of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This document created or updated on January 24, 2012.

# 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:

| | | |
|---|---|---|
| IBM® | RACF® | Redbooks (logo) ® |
| MVS™ | Redbooks® | VM/ESA® |
| OS/390® | Redpaper™ | z/OS® |

The following terms are trademarks of other companies:

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

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

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

# Preface

ZFSREORG is a tool for reorganizing and restructuring zFS compatibility mode aggregates. It is an alternative to directly using commands like pax and copytree. It provides more flexibility in many situations and offers options for how the reorganization or copy processing should be done.

This IBM® Redpaper™ document describes the tool, explains how to install and use it, and provides the complete code for the tool.

## The team who wrote this paper

This paper was produced by a team of specialists from around the world 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®, z/OS UNIX, JES3, and Infoprint Server. Before joining the ITSO 23 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England for seven years providing OS/390® and JES support for IBM EMEA and also in the Washington Systems Center for three years. He has worked for IBM for 44 years.

**Robert Hering** is an IT Specialist at the ITS Technical Support Center, Mainz, Germany. He advises customers on 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, Robert worked with VM and all its various flavors (VM/370, VM/HPO, VM/XA, and VM/ESA®) for many years.

## 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 papers to be as helpful as possible. Send us your comments about this paper 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 reorganization tool

This chapter provides information about a tool for reorganization of zFS compatibility mode aggregates. The following topics are covered in this chapter:

- Description of reorganization tool ZFSREORG
- Using the zFS reorganization tool

# 1.1 Description of reorganization tool ZFSREORG

This chapter introduces the ZFSREORG tool, which can be used to reorganize or restructure zFS compatibility mode aggregates. Because there is not currently a defragmentation function for zFS aggregates, this tool can be helpful if you need to reorganize a zFS aggregate and want to avoid manual interactions.

Among the reasons you might want to do this are as the following situations:

- To reduce the number of extents used for a zFS aggregate.
- To internally restructure a zFS aggregate to allow growth above 4 GB because this was not initially taken into account.

> **Note:** You need to assign an SMS data class with `Data Set Name Type EXTENDED` and `Extended Addressability YES` for the new zFS aggregate.

## Reorganization tool functions

The new reorganization tool supports the following functions:

- Using the standard pax utility of the system running the reorganization to copy the data. The tool supports any version of pax picked up via the PATH environment variable set.
- Using copytree for the copy processing. This utility is picked up via the PATH environment variable set or directly from directory `/samples` if not found otherwise.
- Using IDCAMS/TSO REPRO for copy processing.
- Defining and selecting the file systems to be reorganized, and doing so interactively in TSO/ISPF foreground. The real reorganization processing is done in a second step using a TSO batch job.

> **Important:** You must have one of the following authorizations to perform the definition part in TSO/ISPF foreground:
>
> - A permanent z/OS UNIX superuser ID (UID=0)
> - Read access to profile `BPX.SUPERUSER` in the FACILITY class of RACF®
> - Read access to profile `SUPERUSER.FILESYS.PFSCTL` in class UNIXPRIV

- Several test or step options for running the job to see whether the reorganization can run okay later or to stop after doing some initial work. You can restart the job later after changing the options, as follows:
  - You can stop after doing a short syntax check.
  - You can stop after mounting the old zFS (if not already done) and the new zFS (if it exists already; the new zFS gets unmounted again automatically).
  - You can stop after defining and formatting the new zFS aggregate (if formatting is needed).
- If the new zFS aggregate exists already, the ability to specify to run the copy processing only if the target zFS structure is empty.
- Automatic replacement (after successful copy processing) of the old zFS file system by the new zFS file system and remounting of the file systems down the structure again without any manual interaction.
- Naming specific PFS types that block automatic replacement of the old zFS by the new zFS even if you specified to replace the old zFS.

### 1.1.1 Basic installation information

Get the file `zfs.zfsreorg.unload.bin` and put it into an FB80 data set on your z/OS system. Figure 1-1 shows sample commands for doing this. The file is available at:

`ftp://www.redbooks.ibm.com/redbooks/REDP4769/`

```
Installation Instructions

Note: myuser in these instructions is your User ID. So replace your User ID where you
see myuser and my.email@xx.com is your email address.

First retrieve the file zfs.zfsreorg.unload.bin in binary to your workstation or
directly into your z/OS system from this Redbook website. This can be done using your
favorite browser (not shown here) or an ftp session. Following are samples of how to do
this with ftp sessions.

Scenario 1: Getting the data directly to your z/OS system

ftp www.redbooks.ibm.com
User: anonymous
Password: my.email@xx.com
cd redbooks/REDP4769/
lcd 'myuser'
locsite blk=3120 lrecl=80 recfm=fb
binary
get zfs.zfsreorg.unload.bin zfs.zfsreorg.unload
quit

Note: You may also decide to pre-allocate the unloaded files as XMIT-ed sequential files
instead of using the LOCSITE setting.

Scenario 2: Getting the files to your workstation first

ftp www.redbooks.ibm.com
User: anonymous
Password: my.email@xx.com
cd redbooks/REDP4769/
binary
get zfs.zfsreorg.unload.bin
quit

The following example shows how to transfer the files to z/OS from your workstation
afterwards.

ftp my.zos.system
User: myuser
Password: mypasswd
cd 'myuser'
binary
quote site blk=3120 lrecl=80 recfm=fb
put zfs.zfsreorg.unload.bin zfs.zfsreorg.unload
quit
```

*Figure 1-1 Sample FTP commands to transfer the unloaded ZFSREORG PDS to your system*

After the files are placed on the z/OS system, receive them using the **TSO RECEIVE** command as shown in Figure 1-2.

```
Final installation steps

Afterwards run the following command on your z/OS system:

tso receive indsn('myuser.zfs.zfsreorg.unload')

On display of the following messages you may request to restore the original
PDS data set with a desired new data set name (You may also decide to rename
the data set after completion of the receive command.):

INMR901I Dataset HERING.ZFS.ZFSREORG from HERING on WTSCPLX4
INMR906A Enter restore parameters or 'DELETE' or 'END' +

Here is a sample to set the data set name.

dsn('myuser.zfs.zfsreorg')
```

*Figure 1-2 Final instructions to move files to the z/OS system*

Figure 1-3 on page 5 shows the results of executing the first **TSO RECEIVE** command.

```
tso receive indsn(zfs.zfsreorg.unload)
 Dataset HERING.ZFS.ZFSREORG from HERING on WTSCPLX2
 Enter restore parameters or 'DELETE' or 'END' +
[enter]
 ... IEBCOPY MESSAGES AND CONTROL STATEMENTS ... PAGE     1
 IEB1135I IEBCOPY  FMID HDZ1D10  SERVICE LEVEL NONE     DATED 20110318 DFSMS
01.13.00 z/OS    01.13.00 HBB7780  CPU 2817
 IEB1035I HERING   IKJACCT  IKJACCNT 12:10:55 WED 22 JUN 2011
PARM='WORK=4M,SIZE=1M'
  COPY INDD=((SYS00082,R)),OUTDD=SYS00081
 IEB1013I COPYING FROM PDSU  INDD=SYS00082 VOL=
DSN=SYS11173.T121055.RA000.HERING.R0100027
 IEB1014I           TO PDS  OUTDD=SYS00081 VOL=BH5ST3 DSN=HERING.ZFS.ZFSREORG
 IEB167I FOLLOWING MEMBER(S) LOADED FROM INPUT DATA SET REFERENCED BY SYS00082
 IEB154I $INSTALL HAS BEEN SUCCESSFULLY LOADED
 IEB154I $INSTHLP HAS BEEN SUCCESSFULLY LOADED
 IEB154I $INSTVFY HAS BEEN SUCCESSFULLY LOADED
 IEB154I CPYRHELP HAS BEEN SUCCESSFULLY LOADED
 IEB154I DEFREORG HAS BEEN SUCCESSFULLY LOADED
 IEB154I DEFRHELP HAS BEEN SUCCESSFULLY LOADED
 IEB154I DZRC$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DZRH$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DZRM$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DZRP$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DZRR$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I JCLREORG HAS BEEN SUCCESSFULLY LOADED
 IEB154I ZFSREORG HAS BEEN SUCCESSFULLY LOADED
 IEB1098I 13 OF 13 MEMBERS LOADED FROM INPUT DATA SET REFERENCED BY SYS00082
 IEB144I THERE ARE 1 UNUSED TRACKS IN OUTPUT DATA SET REFERENCED BY SYS00081
 IEB149I THERE ARE 5 UNUSED DIRECTORY BLOCKS IN OUTPUT DIRECTORY
 IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
 Restore successful to dataset 'HERING.ZFS.ZFSREORG'
```

*Figure 1-3 Restoring the ZFSREORG PDS from the unloaded version*

## 1.1.2 Installing the tool

This section describes how to install the tool.

As shown in Figure 1-4 on page 6, you can view and edit the PDS data set `ZFS.ZFSREORG`. The first member is named `$INSTALL`. Near the top is a sequence of job variables that can be customized according to your installation's naming conventions and file structures.

You need two partitioned libraries with record format FB80, a REXX library such as *myuserid*`.ZFS.REXX.EXEC` (which should be located in the SYSPROC or SYSEXEC library chain), and a job control library such as *myuserid*`.ZFS.JOB.CNTL`, as shown in Figure 1-4. Another option is to use your own REXX and JCL libraries and then change them with the JCL, as is also shown in the figure.

Submit the `$INSTALL` job after these requirements are met. Figure 1-4 on page 6 lists excerpts of the sample job provided.

```
//ZFSJOB   JOB ,'Install ZFSREORG',NOTIFY=&SYSUID.
...
//* ------------------------------------------------------------------
//  SET   ZFSREORG=&SYSUID..ZFS.ZFSREORG         <== This PDS Data Set
//  SET   RORGREXX=&SYSUID..ZFS.REXX.EXEC        <== REXX Library
//  SET   RORGJCTL=&SYSUID..ZFS.JOB.CNTL         <== Job CNTL Library
//* ------------------------------------------------------------------
//* Verify Existence of Libraries and Directories, Check System Level
//* ------------------------------------------------------------------
//RORGVRFY EXEC PGM=IKJEFT01,PARM='%$INSTVFY &RORGREXX &RORGJCTL'
//SYSEXEC  DD DSN=&ZFSREORG.,DISP=SHR
//RORGREXX DD DSN=&RORGREXX.,DISP=SHR
//RORGJCTL DD DSN=&RORGJCTL.,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//* ------------------------------------------------------------------
//  IF RORGVRFY.RC EQ 0 THEN
//* ------------------------------------------------------------------
//* Copy member ZFSREORG to the REXX Library
...
//* Copy member DEFREORG to the REXX Library
...
//* Copy member CPYRHELP to the REXX Library
...
//* Copy member DEFRHELP to the REXX Library
...
//* Copy member JCLREORG to the Job CNTL Library as ZFSREORG
...
//* Copy member DZRC$MAC to the REXX Library
...
//* Copy member DZRH$MAC to the REXX Library
...
//* Copy member DZRM$MAC to the REXX Library
...
//* Copy member DZRP$MAC to the REXX Library
...
//* Copy member DZRR$MAC to the REXX Library
...
//* ------------------------------------------------------------------
//  ENDIF RORGVRFY.RC EQ 0
//* ------------------------------------------------------------------
```

*Figure 1-4 JCL excerpts of member $INSTALL in PDS ZFS.ZFSREORG*

Examine the step return codes and messages to be sure the job ran successfully.

### 1.1.3 Tool customization and migration processing

The tool is composed of three commands and one job, as follows:

**DEFRHELP** This command provides information about how to set up for the reorganization tool and how to create reorganization definition files.

**DEFREORG** This command is used to create migration definition statements.

**CPYRHELP** This command provides information about how to set job and UNIX System Services environment variables and how to run reorganization processing.

**ZFSREORG** This file contains the necessary JCL to run a TSO batch job to perform the reorganization processing.

## 1.1.4 DEFREORG help information

This section describes the information provided when using the **DEFRHELP** command.

### DEFREORG control statements

When running DEFREORG, an EDIT session is opened and you need to provide the following values to create an initial migration definition list:

**ZFS_REORG_DEFINE_DSN**

The full name of a sequential MVS™ data set including HLQ or the name of a PDS. Use of a PDS is preferred because this allows you simply to use new members for new migration tasks. This data set must be pre-allocated as VB80 or FB80. The default name is *hlq*.ZFS.REORG.DEFINE, with *hlq* being your own user ID.

**ZFS_REORG_DEFINE_MBR**

Here you can specify the output member name to contain the migration control statements. If no name is provided WORK*nn* is used by default, with *nn* being a number that is not used currently. You can rename the member names using ISPF at any time.

**ZFS_REORG_DEFINE_DSP**

This value must be specified as APPEND or REPLACE. Note that only the first character is examined. With APPEND the new control statements are appended, otherwise the old content is replaced (if the member exists already).

**ZFS_DEF_DATACLASS**

Default SMS data class for a new zFS aggregate to be used. The value can be changed or removed later for specific aggregates.

**ZFS_DEF_MANAGEMENTCLASS**

Default SMS management class for a new zFS aggregate to be used. The value can be changed or removed later for specific aggregates.

**ZFS_DEF_STORAGECLASS**

Default SMS storage class for a new zFS aggregate to be used. The value can be changed or removed later for specific aggregates.

**ZFS_DATA_SETS_TO_REORG**

This statement defines the zFS data set name list to be reorganized. The value cannot be BLANK. You can use several lines using this control word. The values are the same as used on ISPF 3.4 to display a list of data sets.

**ZFS_AGGRNAME_CHANGE_CMD**

This statements allows you to specify a file system name change command, for example: /ZF1/ZF2/. If you do not specify anything the new zFS aggregate will get the same name as the old zFS LDS before. The old zFS will be renamed (with ".SAV" appended at the end by default).

### DEFREORG line commands

When running DEFREORG, the following specific line commands are available:

| | |
|---|---|
| **DEFRHELP** or **DH** | This command calls help information; DEFRHELP can easily be called from within DEFREORG. |
| **DM** | Displays the last set of messages shown (browse mode). |
| **CPYRHELP** or **CH** | Displays help information for the later reorganization job. |
| **REFRESH** or **RF** | Displays the current source statements refreshed. This is useful if you accidentally deleted some information. |

## 1.1.5 ZFSREORG help information

This section explains the information provided when you use the **CPYRHELP** command.

### Reorganization processing and job variables

The following job variables can be set:

| | |
|---|---|
| **CPYTOOL** | This is the copy utility to be used; using COPYPAX is the suggested method for a REORG. |
| Value **TSOREPRO** | Use IDCAMS/TSO REPRO function for copying. |
| Value **COPYPAX** | Use the accessible (standard) pax version for copying. |
| Value **COPYTREE** | Use the accessible (standard) copytree for copying. |
| **VERBOSE** | Specify N or Y to list all objects copied; VERBOSE is used only if COPYPAX is set. |
| **DEFREORG** | This is the member or data set containing the REORG definition statements. |

### Migration processing environment variables

The following environment variables can be set for running the job:

| | |
|---|---|
| **STOP_AFTER_SYNTAX_CHECK** | Force stopping after formal syntax check of STDIN data is done; value must be specified as N or Y. |
| **STOP_AFTER_FSS_MOUNTED** | Force stopping when old and if existing new zFS is mounted; value must be specified as N or Y. |
| **STOP_AFTER_ZFS_IS_FORMATTED** | Force stopping when the new zFS aggregate is formatted; value must be specified as N or Y. |
| **TARGET_ZFS_MUST_BE_EMPTY** | Run copy processing only if the target zFS structure is empty; value can be set to Y or N. |
| **DENIED_UMNT_FSTYPES** | Specify filesystypes blocking replacement of mounted old zFS by the new zFS. TFS is always blocking the automatic replacement because the complete contents of the TFS is lost otherwise. |
| **PATH** | Additional PATH setting; `/bin` and `/samples` are included automatically.<br>Example: `PATH=/usr/local/bin` |

## Reorganization definition statements

This section provides short descriptions of all the supported definition statements for reorganizing a zFS aggregate.

**Notes:**

- These statements are automatically generated using DEFREORG, displayed for review, saved in the data set containing the REORG definition statements, and referenced in the JCL using DD name STDIN.
- The abbreviation “FYR” as used here means a value that is provided just “For Your Reference.”

| | |
|---|---|
| **ZFS_OLD_NAME** | Name of the old zFS data set to be migrated. |
| **#ZFS_OLD_#_VOLUMES** | FYR: Number of old zFS volumes. |
| **#ZFS_OLD_DEVICE_TYPE** | FYR: Old zFS DASD device type. |
| **#ZFS_OLD_ALLOC_UNIT** | FYR: Old zFS DASD allocation unit. |
| **#ZFS_OLD_ALLOC_SPACE** | FYR: Old zFS primary and secondary allocation. |
| **#ZFS_OLD_TOTAL_UNITS_ALLOCATED** | FYR: Old zFS total number of units allocated. |
| **#ZFS_OLD_TOTAL_UNITS_FORMATTED** | FYR: Old zFS total zFS formatted units. |
| **#ZFS_OLD_TOTAL_UNITS_%USED** | FYR: Old zFS percentage used of space. |
| **#ZFS_OLD_DATACLASS** | FYR: Old zFS data class |
| **#ZFS_OLD_MGMNTCLASS** | FYR: Old zFS management class. |
| **#ZFS_OLD_STORCLASS** | FYR: Old zFS storage class. |
| **ZFS_OLD_NAME_SAV** | Suggested name for old zFS to be renamed. This is only used if old and new zFS have set the same name. |
| **ZFS_NAME_TMP** | Suggested name for zfs used initially. This is only used if old and new zFS have set the same name. |
| **ZFS_NAME** | zFS aggregate name. |
| **#ZFS_#_VOLUMES** | Number of zFS volumes you should have at least (just a comment). This value should be the sum of zFS volumes plus candidate volumes plus volumes needed for secondary allocations. |
| **ZFS_VOLUMES** | Explicit list of volumes to be used. This specification is ignored if an SMS storage class is assigned. |
| **ZFS_ALLOC_NUM_CAND_VOLUMES** | Number of zFS candidate volumes. |
| **ZFS_ALLOC_UNIT** | zFS allocation unit. You can specify CYLINDERS (CYL) or TRACKS(TRK) only. |
| **ZFS_ALLOC_SPACE** | zFS primary and secondary allocation. |
| **ZFS_ALLOC_NUM_SEC_ALLOCS** | Number of secondary allocations added before starting migration processing. |

| | |
|---|---|
| **ZFS_DATACLASS** | zFS SMS data class to be used. A blank values means that no data class is set. |
| **ZFS_MGMNTCLASS** | zFS SMS management class to be used. A blank values means that no management class is set. |
| **ZFS_STORCLASS** | zFS SMS storage class to be used. A blank value means that no storage class is set. |
| **ZFS_REPLACES_ZFS** | This value specifies whether the new zFS file system should replace the old zFS. You must specify Y or N. |

**Important:** If you specify to not replace the old zFS file system by the new zFS file system, or this cannot be done for other reasons, the old zFS file system is kept mounted read-only to assure that the contents will not differ from the new zFS file system.

## 1.2 Using the zFS reorganization tool

In this section we demonstrate how to use the tool by showing several examples. The following situations are covered:

- Reorganization of a zFS aggregate being used and having other file systems mounted down the structure.
- Reorganization of a zFS aggregate with low formatted and used space at the same time much space is allocated for the LDS.
- Reorganization of a zFS aggregate that is growing towards 4 GB and not having been assigned a data class with data set name that has a type of extended addressability.

### 1.2.1 Reorganization scenario A: Some file systems mounted below

Figure 1-5 shows an example of the status of a zFS aggregate being mounted and having other file systems mounted down the structure.

```
$> pwd
/u/hering
$> sudo /usr/sbin/mount -qv test.reorg
----A- HERING.TEST.BIG.ZFS   /u/hering/test.reorg/test.sub/subdir
R---A- HERING.TEST.ZFS       /u/hering/test.reorg/test.sub
----A- HERING.TEST.REORG.ZFS /u/hering/test.reorg
$> rxzfsmon | grep HERING.TEST | grep ZFS
HERING.TEST.ZFS                                360        309 = 85.8%  X-R-
HERING.TEST.REORG.ZFS                          360         22 =  6.1%  X--S
HERING.TEST.BIG.ZFS                         180000      67769 = 37.6%  X--S
$>
```

*Figure 1-5 Initial situation with all the file systems involved*

**Note:** As suggested previously, the migration definition data has been allocated as a PDS with record type FB 80.

## Doing the reorganization

The aggregate HERING.TEST.REORG.ZFS is the candidate to get reorganized. There are three file systems involved here.

**Attention:** Make sure the REXX library *myuser*.ZFS.REXX.EXEC is located in the SYSPROC or SYSEXEC library chain and a job control library *myuser*.ZFS.JOB.CNTL exists.

First, from ISPF Option 6, enter the **DEFREORG** command, which brings up an EDIT session as shown in Figure 1-6. This command can be issued from any ISPF command line, using the command **TSO DEFREORG**.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       SYS11170.T114445.RA000.HERING.R0300301          Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 000001 # ---------------------------------------------------------------------- #
 000002 ZFS_REORG_DEFINE_DSN=HERING.ZFS.REORG.DEFINE
 000003 ZFS_REORG_DEFINE_MBR=
 000004 ZFS_REORG_DEFINE_DSP=APPEND
 000005 # ---------------------------------------------------------------------- #
 000006
 000007 # ---------------------------------------------------------------------- #
 000008 ZFS_DEF_DATACLASS=
 000009 ZFS_DEF_MANAGEMENTCLASS=
 000010 ZFS_DEF_STORAGECLASS=
 000011 # ---------------------------------------------------------------------- #
 000012
 000013 # ---------------------------------------------------------------------- #
 000014 ZFS_DATA_SETS_TO_REORG=HERING.TEST
 000015 ZFS_AGGRNAME_CHANGE_CMD=
 000016 # ---------------------------------------------------------------------- #
 ****** **************************** Bottom of Data ****************************


 ____________________________________________________________________________
| Enter DH to show REORG definition information. Change the data as needed. To |
| continue SAVE the changes,to stop processing use CANCEL.                     |
|______________________________________________________________________________|
 Command ===>                                                 Scroll ===> CSR
  F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-6 EDIT session display after starting the DEFREORG utility*

In this file, we changed the settings as needed. Figure 1-7 on page 12 shows the display after we made the changes.

**Note:** If you do not specify a member name, the tool will select a non-existent name such as WORKxx.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       SYS11170.T114445.RA000.HERING.R0300301          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 ZFS_REORG_DEFINE_DSN=HERING.ZFS.REORG.DEFINE
000003 ZFS_REORG_DEFINE_MBR=reorg01
000004 ZFS_REORG_DEFINE_DSP=rPPEND
000005 # ------------------------------------------------------------------- #
000006
000007 # ------------------------------------------------------------------- #
000008 ZFS_DEF_DATACLASS=
000009 ZFS_DEF_MANAGEMENTCLASS=
000010 ZFS_DEF_STORAGECLASS=
000011 # ------------------------------------------------------------------- #
000012
000013 # ------------------------------------------------------------------- #
000014 ZFS_DATA_SETS_TO_REORG=hering.test.reorg
000015 ZFS_AGGRNAME_CHANGE_CMD=
000016 # ------------------------------------------------------------------- #
****** *************************** Bottom of Data ***************************


 _____________________________________________________________________________
| Enter DH to show REORG definition information. Change the data as needed. To |
| continue SAVE the changes,to stop processing use CANCEL.                     |
|_____________________________________________________________________________|
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-7 EDIT session display after modifying the settings*

We set the member name to REORG01, requested to replace the member if available, and finally entered the zFS aggregates to look for. Pressing PF03 ended the EDIT session. The **DEFREORG** command then calculated the settings, displayed messages as shown in Figure 1-8, and then displayed the member for reviewing and modifying the data. This is shown in Figure 1-9 on page 13, after going forward to line 5.

```
DEFR0050I Searching for data sets HERING.TEST.REORG ...
DEFR0048I Data set HERING.TEST.REORG.NO.FORMAT will be examined.
DEFR0048I Data set HERING.TEST.REORG.NO.FORMAT.DATA will be examined.
DEFR0048I Data set HERING.TEST.REORG.ZFS will be examined.
DEFR0048I Data set HERING.TEST.REORG.ZFS.DATA will be examined.
DEFR0060W Data set HERING.TEST.REORG.NO.FORMAT seems to be no valid zFS and wil
l be skipped.
DEFR0056I Data set HERING.TEST.REORG.NO.FORMAT.DATA is not a VSAM cluster and w
ill be skipped.
DEFR0056I Data set HERING.TEST.REORG.ZFS.DATA is not a VSAM cluster and will be
 skipped.
***
```

*Figure 1-8 Messages displayed while the tool is creating reorganization statements*

Note that you can redisplay the messages from the new EDIT session shown in Figure 1-9 on page 13 using the command DM.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.REORG.DEFINE(REORG01) - 01.05        Columns 00001 00072
000005 # ------------------------------------------------------------------- #
000006 ZFS_OLD_NAME=                    HERING.TEST.REORG.ZFS
000007 # ------------------------------------------------------------------- #
000008 #ZFS_OLD_#_VOLUMES=              1
000009 #ZFS_OLD_DEVICE_TYPE=            3390
000010 #ZFS_OLD_ALLOC_UNIT=             CYLINDER
000011 #ZFS_OLD_ALLOC_SPACE=            4 1
000012 #ZFS_OLD_TOTAL_UNITS_ALLOCATED=  4
000013 #ZFS_OLD_TOTAL_UNITS_FORMATTED=  4 CYLINDERS
000014 #ZFS_OLD_TOTAL_UNITS_%USED=      6
000015 #ZFS_OLD_DATACLASS=
000016 #ZFS_OLD_MGMNTCLASS=             HFS
000017 #ZFS_OLD_STORCLASS=              OPENMVS
000018 ZFS_OLD_NAME_SAV=                HERING.TEST.REORG.ZFS.SAV
000019 ZFS_NEW_NAME_TMP=                HERING.TEST.REORG.ZFS.TMP
000020 ZFS_NEW_NAME=                    HERING.TEST.REORG.ZFS
000021 #ZFS_NEW_#_VOLUMES=              1
000022 ZFS_NEW_VOLUMES=
000023 ZFS_NEW_ALLOC_NUM_CAND_VOLUMES=  1
000024 ZFS_NEW_ALLOC_UNIT=              CYLINDERS
000025 ZFS_NEW_ALLOC_SPACE=             1 1
000026 ZFS_NEW_ALLOC_NUM_SEC_ALLOCS=    0
000027 ZFS_NEW_DATACLASS=               extaddr
000028 ZFS_NEW_MGMNTCLASS=              HFS
000029 ZFS_NEW_STORCLASS=               OPENMVS
000030 ZFS_NEW_REPLACES_ZFS_OLD=        Y
000031
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-9 EDIT session displaying the reorganization statements just created*

Because HERING.TEST.REORG.ZFS was the only data set that met the search criteria, we simply accepted the values suggested, saved the data, and ended the EDIT session.

Then we edited the JCL member ZFSREORG and modified the data as shown in Figure 1-10 on page 14 and Figure 1-11 on page 15.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.JOB.CNTL(ZFSREORG) - 01.09           Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 //ZFSJOB   JOB ,'ZFSREORG',NOTIFY=&SYSUID.,REGION=0M
000002 /*JOBPARM SYSAFF=xxxx <=== JES2 system affinity
000003 //*MAIN SYSTEM=SC70   <=== JES3 system affinity
000004 //* --------------------------------------------------------------------
000005 //* Reorganize compat zFS aggregates to new zFS compat mode aggregates
000006 //* Property of IBM  (C) Copyright IBM Corp. 2011
000007 //* --------------------------------------------------------------------
000008 //  SET CPYTOOL=COPYPAX                  <=== Copy utility to be used
000009 //*             TSOREPRO: Use IDCAMS/TSO REPRO function for copying
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 COPYPAX is set.
000014 //  SET DEFREORG=&SYSUID..ZFS.REORG.DEFINE(REORG01)    <=== REORG DEFs
000015 //* --------------------------------------------------------------------
000016 //  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
000017 //* --------------------------------------------------------------------
000018 //ZFSREORG EXEC PGM=IKJEFT01,PARM='ZFSREORG &CPYTOOL. &VERBOSE.'
000019 //*STEPLIB DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
000020 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000021 //STDENV   DD DATA,DLM=##
000022 # --------------------------------------------------------------------
000023 # Force stopping after formal syntax check of STDIN data is done (N|Y)
000024   STOP_AFTER_SYNTAX_CHECK=N
000025 # Force stopping when old and existing new zFS is/are mounted (N|Y)
000026   STOP_AFTER_FSS_MOUNTED=N
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-10 JCL ZFSREORG with the modified reorganization definition member name*

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ______________________________________________________________________________
 EDIT       HERING.ZFS.JOB.CNTL(ZFSREORG) - 01.09          Columns 00001 00072
 000021 //STDENV   DD DATA,DLM=##
 000022 # ---------------------------------------------------------------------
 000023 # Force stopping after formal syntax check of STDIN data is done (N|Y)
 000024   STOP_AFTER_SYNTAX_CHECK=N
 000025 # Force stopping when old and existing new zFS is/are mounted (N|Y)
 000026   STOP_AFTER_FSS_MOUNTED=N
 000027 # Force stopping when the new zFS aggregate is formatted (N|Y)
 000028   STOP_AFTER_ZFS_IS_FORMATTED=N
 000029 # Run copy processing only if the target zFS structure is empty (Y|N)
 000030   TARGET_ZFS_MUST_BE_EMPTY=Y
 000031 # Specify filesystypes blocking replacement of mounted old by new zFS
 000032   DENIED_UMNT_FSTYPES=TFS
 000033 # Specify additional directories to be included in the PATH chain
 000034 # PATH=/usr/local/bin
 000035 # ---------------------------------------------------------------------
 000036 ##
 000037 //STDIN    DD DSNAME=&DEFREORG.,DISP=SHR
 000038 //SYSTSIN  DD DUMMY
 000039 //SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
 000040 //* ---------------------------------------------------------------------
 ****** **************************** Bottom of Data ****************************

Command ===>                                                 Scroll ===> CSR
  F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-11 JCL ZFSREORG showing unmodified environment variables*

No further changes were made in the remaining part of the JCL, shown in Figure 1-11.

Finally, we submitted the job. Figure 1-12 on page 16 shows the messages we got after the job successfully ended with RC=0.

```
ZFSRO065I Running with options Reorgtool=COPYPAX and Verbose=N ...

------------------------------------------------------------------------
The main process ID for this job is 33751184. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 33751184
Or    : kill -s SIGTERM 33751184
------------------------------------------------------------------------

------------------------------------------------------------------------
ZFSRO004I Processing zFS old data set name HERING.TEST.REORG.ZFS...
------------------------------------------------------------------------
ZFSRO075I As the new zFS aggregate name is the same as of the old zFS data set a
temporary name is used for the new zFS.
ZFSRO076I zFS temporary name: HERING.TEST.REORG.ZFS.TMP
ZFSRO086I Defining zFS aggregate HERING.TEST.REORG.ZFS.TMP ...
IOEZ00248I VSAM linear dataset HERING.TEST.REORG.ZFS.TMP successfully created.
ZFSRO087I 09:46:41 Formatting zFS aggregate HERING.TEST.REORG.ZFS.TMP ...
IOEZ00077I HFS-compatibility aggregate HERING.TEST.REORG.ZFS.TMP has been successfully
created
ZFSRO054I 09:46:42 Now starting copy processing...
ZFSRO070I 09:46:43 Copy processing has been ended...
ZFSRO105I Temporary unmounting HERING.TEST.BIG.ZFS mounted at
/u/hering/test.reorg/test.sub/subdir
ZFSRO105I Temporary unmounting HERING.TEST.ZFS mounted at /u/hering/test.reorg/test.sub
ZFSRO106I Unmounting HERING.TEST.REORG.ZFS ...
IDC0531I ENTRY HERING.TEST.REORG.ZFS ALTERED
ZFSRO103I The old zFS aggregate has been renamed to HERING.TEST.REORG.ZFS.SAV.
IDC0531I ENTRY HERING.TEST.REORG.ZFS.DATA ALTERED
ZFSRO131I The old zFS DATA part has been renamed to or is named
HERING.TEST.REORG.ZFS.SAV.DATA.
IDC0531I ENTRY HERING.TEST.REORG.ZFS.TMP ALTERED
ZFSRO104I The zFS aggregate has been renamed to HERING.TEST.REORG.ZFS.
IDC0531I ENTRY HERING.TEST.REORG.ZFS.TMP.DATA ALTERED
ZFSRO119I The zFS DATA part has been renamed to or is named HERING.TEST.REORG.ZFS.DATA.
ZFSRO107I Mounting zFS HERING.TEST.REORG.ZFS at /u/hering/test.reorg now...
ZFSRO109I Mounting file system HERING.TEST.ZFS at /u/hering/test.reorg/test.sub again
ZFSRO109I Mounting file system HERING.TEST.BIG.ZFS at
/u/hering/test.reorg/test.sub/subdir again
ZFSRO111I No errors have been recognized for the actual REORG process.

READY
END
```

*Figure 1-12 Messages shown in joblog about the reorganization processing*

## Reorganization results

The results of the reorganization process are shown in Figure 1-13 on page 17.

```
$> sudo /usr/sbin/mount -qv test.reorg
----A- HERING.TEST.BIG.ZFS   /u/hering/test.reorg/test.sub/subdir
R---A- HERING.TEST.ZFS       /u/hering/test.reorg/test.sub
----A- HERING.TEST.REORG.ZFS /u/hering/test.reorg
$> rxzfsmon | grep HERING.TEST | grep ZFS
HERING.TEST.ZFS                                360        309 = 85.8%  X-R-
HERING.TEST.REORG.ZFS                           90         22 = 24.4%  X--S
HERING.TEST.BIG.ZFS                         180000      67769 = 37.6%  X--S
$>
```

*Figure 1-13 Final situation after having done the reorganization*

This verifies that the reorganization processing was done successfully and all file systems were mounted in the same way as before.

## 1.2.2 Reorganization scenario B: Space reduction example

In this example we reorganized a zFS aggregate with low usage of the formatted space and having even more space allocated.

```
$> sudo zfsadm attach hering.test.autogrow
IOEZ00117I Aggregate HERING.TEST.AUTOGROW attached successfully
$> rxzfsmon hering.test.autogrow

Aggregate name                            8K blk tot 8K blk use Percent  XQRS
----------------------------------------- ---------- ---------- -------  ----
HERING.TEST.AUTOGROW                            3330         91 =  2.7%  X---

$> sudo zfsadm detach hering.test.autogrow
IOEZ00122I Aggregate HERING.TEST.AUTOGROW detached successfully
$> rexx 'say 3330/90 "cylinders"'
37 cylinders
$>
```

*Figure 1-14 Information about the zFS aggregate to be reorganized*

Figure 1-14 shows a situation where 37 cylinders are formatted and available but not much space is used. Furthermore, as Figure 1-17 on page 19 shows, even more space is allocated but not yet completely formatted.

We started DEFREORG and modified the values as shown in Figure 1-15. We pressed PF03 and DEFREORG searched for possible data sets as shown in Figure 1-16.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       SYS11173.T052612.RA000.HERING.R0300398          Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 000001 # ------------------------------------------------------------------- #
 000002 ZFS_REORG_DEFINE_DSN=HERING.ZFS.REORG.DEFINE
 000003 ZFS_REORG_DEFINE_MBR=REORGO2
 000004 ZFS_REORG_DEFINE_DSP=REPLACE
 000005 # ------------------------------------------------------------------- #
 000006
 000007 # ------------------------------------------------------------------- #
 000008 ZFS_DEF_DATACLASS=
 000009 ZFS_DEF_MANAGEMENTCLASS=
 000010 ZFS_DEF_STORAGECLASS=
 000011 # ------------------------------------------------------------------- #
 000012
 000013 # ------------------------------------------------------------------- #
 000014 ZFS_DATA_SETS_TO_REORG=hering.test.autogrow
 000015 ZFS_AGGRNAME_CHANGE_CMD=
 000016 # ------------------------------------------------------------------- #
 ****** *************************** Bottom of Data ****************************


 ____________________________________________________________________________
| Enter DH to show REORG definition information. Change the data as needed. To |
| continue SAVE the changes,to stop processing use CANCEL.                     |
|______________________________________________________________________________|
 Command ===>                                                 Scroll ===> CSR
  F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-15 EDIT session display after starting DEFREORG and modifying some values*

```
DEFRO050I Searching for data sets HERING.TEST.AUTOGROW ...
DEFRO048I Data set HERING.TEST.AUTOGROW will be examined.
DEFRO048I Data set HERING.TEST.AUTOGROW.DATA will be examined.
DEFRO056I Data set HERING.TEST.AUTOGROW.DATA is not a VSAM cluster and will be
skipped.
***
```

*Figure 1-16 Messages shown on searching data sets and creating reorganization statements*

DEFREORG calculated suggestions for the new zFS aggregate as shown in Figure 1-17. We simply accepted the values suggested and ended the EDIT session by pressing PF03.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.REORG.DEFINE(REORG02) - 01.01        Columns 00001 00072
000005 # ------------------------------------------------------------------- #
000006 ZFS_OLD_NAME=                    HERING.TEST.AUTOGROW
000007 # ------------------------------------------------------------------- #
000008 #ZFS_OLD_#_VOLUMES=              1
000009 #ZFS_OLD_DEVICE_TYPE=            3390
000010 #ZFS_OLD_ALLOC_UNIT=             CYLINDER
000011 #ZFS_OLD_ALLOC_SPACE=            601 600
000012 #ZFS_OLD_TOTAL_UNITS_ALLOCATED=  601
000013 #ZFS_OLD_TOTAL_UNITS_FORMATTED=  37 CYLINDERS
000014 #ZFS_OLD_TOTAL_UNITS_%USED=      3
000015 #ZFS_OLD_DATACLASS=
000016 #ZFS_OLD_MGMNTCLASS=             HFS
000017 #ZFS_OLD_STORCLASS=              OPENMVS
000018 ZFS_OLD_NAME_SAV=                HERING.TEST.AUTOGROW.SAV
000019 ZFS_NEW_NAME_TMP=                HERING.TEST.AUTOGROW.TMP
000020 ZFS_NEW_NAME=                    HERING.TEST.AUTOGROW
000021 #ZFS_NEW_#_VOLUMES=              1
000022 ZFS_NEW_VOLUMES=
000023 ZFS_NEW_ALLOC_NUM_CAND_VOLUMES=  1
000024 ZFS_NEW_ALLOC_UNIT=              CYLINDERS
000025 ZFS_NEW_ALLOC_SPACE=            2 1
000026 ZFS_NEW_ALLOC_NUM_SEC_ALLOCS=    0
000027 ZFS_NEW_DATACLASS=
000028 ZFS_NEW_MGMNTCLASS=             HFS
000029 ZFS_NEW_STORCLASS=              OPENMVS
000030 ZFS_NEW_REPLACES_ZFS_OLD=       Y
000031
Command ===>                                                Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-17 EDIT session showing the reorganization statements created for this run*

We opened the ZFSREORG JCL again, as shown in Figure 1-18, and submitted the job. Figure 1-19 on page 21 shows the messages received in the joblog during reorganization processing.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.JOB.CNTL(ZFSREORG) - 01.09           Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 //ZFSJOB   JOB ,'ZFSREORG',NOTIFY=&SYSUID.,REGION=0M
000002 /*JOBPARM SYSAFF=xxxx <=== JES2 system affinity
000003 //*MAIN SYSTEM=SC70   <=== JES3 system affinity
000004 //* --------------------------------------------------------------------
000005 //* Reorganize compat zFS aggregates to new zFS compat mode aggregates
000006 //* Property of IBM  (C) Copyright IBM Corp. 2011
000007 //* --------------------------------------------------------------------
000008 //  SET CPYTOOL=COPYPAX                  <=== Copy utility to be used
000009 //*             TSOREPRO: Use IDCAMS/TSO REPRO function for copying
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 COPYPAX is set.
000014 //  SET DEFREORG=&SYSUID..ZFS.REORG.DEFINE(REORG02)    <=== REORG DEFs
000015 //* --------------------------------------------------------------------
000016 //  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
000017 //* --------------------------------------------------------------------
000018 //ZFSREORG EXEC PGM=IKJEFT01,PARM='ZFSREORG &CPYTOOL. &VERBOSE.'
000019 //*STEPLIB DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
000020 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000021 //STDENV   DD DATA,DLM=##
000022 # --------------------------------------------------------------------
000023 # Force stopping after formal syntax check of STDIN data is done (N|Y)
000024   STOP_AFTER_SYNTAX_CHECK=N
000025 # Force stopping when old and existing new zFS is/are mounted (N|Y)
000026   STOP_AFTER_FSS_MOUNTED=N
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-18 JCL ZFSREORG with the modified reorganization member REORG02*

```
ZFSRO065I Running with options Reorgtool=COPYPAX and Verbose=N ...

------------------------------------------------------------------------
The main process ID for this job is 67305587. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 67305587
Or    : kill -s SIGTERM 67305587
------------------------------------------------------------------------

------------------------------------------------------------------------
ZFSRO004I Processing zFS old data set name HERING.TEST.AUTOGROW...
------------------------------------------------------------------------
ZFSRO075I As the new zFS aggregate name is the same as of the old zFS data set a
temporary name is used for the new zFS.
ZFSRO076I zFS temporary name: HERING.TEST.AUTOGROW.TMP
ZFSRO086I Defining zFS aggregate HERING.TEST.AUTOGROW.TMP ...
IOEZ00248I VSAM linear dataset HERING.TEST.AUTOGROW.TMP successfully created.
ZFSRO087I 09:10:31 Formatting zFS aggregate HERING.TEST.AUTOGROW.TMP ...
IOEZ00077I HFS-compatibility aggregate HERING.TEST.AUTOGROW.TMP has been successfully
created
ZFSRO054I 09:10:32 Now starting copy processing...
ZFSRO070I 09:10:33 Copy processing has been ended...
IDC0531I ENTRY HERING.TEST.AUTOGROW ALTERED
ZFSRO103I The old zFS aggregate has been renamed to HERING.TEST.AUTOGROW.SAV.
IDC0531I ENTRY HERING.TEST.AUTOGROW.DATA ALTERED
ZFSRO131I The old zFS DATA part has been renamed to or is named
HERING.TEST.AUTOGROW.SAV.DATA.
IDC0531I ENTRY HERING.TEST.AUTOGROW.TMP ALTERED
ZFSRO104I The zFS aggregate has been renamed to HERING.TEST.AUTOGROW.
IDC0531I ENTRY HERING.TEST.AUTOGROW.TMP.DATA ALTERED
ZFSRO119I The zFS DATA part has been renamed to or is named HERING.TEST.AUTOGROW.DATA.
ZFSRO111I No errors have been recognized for the actual REORG process.

READY
END
```

*Figure 1-19 Messages shown in the joblog during reorganization processing*

The results of this reorganization processing are shown in Figure 1-20 and indicate that the desired results were achieved.

```
$> sudo zfsadm attach hering.test.autogrow
IOEZ00117I Aggregate HERING.TEST.AUTOGROW attached successfully
$> rxzfsmon hering.test.autogrow

Aggregate name                              8K blk tot 8K blk use Percent  XQRS
------------------------------------------ ---------- ---------- -------  ----
HERING.TEST.AUTOGROW                              180         91 = 50.6%  X---

$> sudo zfsadm detach hering.test.autogrow
IOEZ00122I Aggregate HERING.TEST.AUTOGROW detached successfully
$>
```

*Figure 1-20 The new zFS aggregate after the reorganization*

### 1.2.3 Reorganization scenario C: Using REPRO

In this example we reorganized a zFS aggregate that had been defined with no data class assigned having the necessary attribute for extended addressability, as shown in Figure 1-21.

```
$> rxzfsmon hering.test.not.extaddr

Aggregate name                              8K blk tot 8K blk use Percent  XQRS
------------------------------------------ ---------- ---------- -------  ----
HERING.TEST.NOT.EXTADDR                         459090     459075 =100.0%  X--S
$> rexx 'say 459090/90 "cylinders"'
5101 cylinders
$> sudo /usr/sbin/mount -qv test
----A- HERING.TEST.NOT.EXTADDR /u/hering/test
$> tsocmd "listcat ent(test.not.extaddr) all" 2>/dev/null | grep DATACLASS
       DATACLASS --------(NULL)     LBACKUP ---0000.000.0000
$>
```

*Figure 1-21 Information about the original zFS aggregate with no extended addressability*

**Note:** Using REPRO can be much faster than using pax in cases of complex structures and many special attribute settings.

For more information about using REPRO with zFS aggregates, see *z/OS Distributed File Service zSeries File System Implementation z/OS V1R13*, SG24-6580 (edition 05 or later).

Again we started DEFREORG and modified the values as shown in Figure 1-22.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       SYS11172.T105004.RA000.HERING.R0300364          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 ZFS_REORG_DEFINE_DSN=HERING.ZFS.REORG.DEFINE
000003 ZFS_REORG_DEFINE_MBR=REORG03
000004 ZFS_REORG_DEFINE_DSP=REPLACE
000005 # ------------------------------------------------------------------- #
000006
000007 # ------------------------------------------------------------------- #
000008 ZFS_DEF_DATACLASS=extaddr
000009 ZFS_DEF_MANAGEMENTCLASS=
000010 ZFS_DEF_STORAGECLASS=
000011 # ------------------------------------------------------------------- #
000012
000013 # ------------------------------------------------------------------- #
000014 ZFS_DATA_SETS_TO_REORG=hering.test.not.extaddr
000015 ZFS_AGGRNAME_CHANGE_CMD=/.not//
000016 # ------------------------------------------------------------------- #
****** ***************************** Bottom of Data ***************************


 _____________________________________________________________________________
| Enter DH to show REORG definition information. Change the data as needed. To |
| continue SAVE the changes,to stop processing use CANCEL.                     |
|_____________________________________________________________________________|
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-22 EDIT session display after starting DEFREORG and modifying the values for as needed*

We pressed PF03 to start the DEFREORG search for possible data sets as shown in Figure 1-23.

```
DEFR0050I Searching for data sets HERING.TEST.NOT.EXTADDR ...
DEFR0048I Data set HERING.TEST.NOT.EXTADDR will be examined.
DEFR0048I Data set HERING.TEST.NOT.EXTADDR.DATA will be examined.
DEFR0056I Data set HERING.TEST.NOT.EXTADDR.DATA is not a VSAM cluster and will
be skipped.
***
```

*Figure 1-23 Messages shown on DEFREORG searching for data sets in the new situation*

As before, DEFREORG calculated suggestions for the new zFS aggregate. This is shown in Figure 1-24 on page 24.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.REORG.DEFINE(REORG03) - 01.01        Columns 00001 00072
000005 # ------------------------------------------------------------------- #
000006 ZFS_OLD_NAME=                    HERING.TEST.NOT.EXTADDR
000007 # ------------------------------------------------------------------- #
000008 #ZFS_OLD_#_VOLUMES=              3
000009 #ZFS_OLD_DEVICE_TYPE=            3390
000010 #ZFS_OLD_ALLOC_UNIT=             CYLINDER
000011 #ZFS_OLD_ALLOC_SPACE=            1 1
000012 #ZFS_OLD_TOTAL_UNITS_ALLOCATED=  5101
000013 #ZFS_OLD_TOTAL_UNITS_FORMATTED=  5101 CYLINDERS
000014 #ZFS_OLD_TOTAL_UNITS_%USED=      100
000015 #ZFS_OLD_DATACLASS=
000016 #ZFS_OLD_MGMNTCLASS=             HFS
000017 #ZFS_OLD_STORCLASS=              OPENMVS
000018 ZFS_OLD_NAME_SAV=                HERING.TEST.NOT.EXTADDR.SAV
000019 ZFS_NEW_NAME_TMP=                HERING.TEST.NOT.EXTADDR.TMP
000020 ZFS_NEW_NAME=                    HERING.TEST.NOT.EXTADDR
000021 #ZFS_NEW_#_VOLUMES=              4
000022 ZFS_NEW_VOLUMES=
000023 ZFS_NEW_ALLOC_NUM_CAND_VOLUMES=  4
000024 ZFS_NEW_ALLOC_UNIT=              CYLINDERS
000025 ZFS_NEW_ALLOC_SPACE=             500 500
000026 ZFS_NEW_ALLOC_NUM_SEC_ALLOCS=    12
000027 ZFS_NEW_DATACLASS=               EXTADDR
000028 ZFS_NEW_MGMNTCLASS=              HFS
000029 ZFS_NEW_STORCLASS=               OPENMVS
000030 ZFS_NEW_REPLACES_ZFS_OLD=        Y
000031
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-24 EDIT session showing the reorganization statements created by DEFREORG*

We reduced the number of candidate volumes by 2 because the original value was too high already in our opinion. The new reorganization statements are shown in Figure 1-25. We ended the EDIT session by pressing PF03.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.REORG.DEFINE(REORG03) - 01.01        Columns 00001 00072
000005 # ------------------------------------------------------------------- #
000006 ZFS_OLD_NAME=                     HERING.TEST.NOT.EXTADDR
000007 # ------------------------------------------------------------------- #
000008 #ZFS_OLD_#_VOLUMES=               3
000009 #ZFS_OLD_DEVICE_TYPE=             3390
000010 #ZFS_OLD_ALLOC_UNIT=              CYLINDER
000011 #ZFS_OLD_ALLOC_SPACE=             1 1
000012 #ZFS_OLD_TOTAL_UNITS_ALLOCATED=   5101
000013 #ZFS_OLD_TOTAL_UNITS_FORMATTED=   5101 CYLINDERS
000014 #ZFS_OLD_TOTAL_UNITS_%USED=       100
000015 #ZFS_OLD_DATACLASS=
000016 #ZFS_OLD_MGMNTCLASS=              HFS
000017 #ZFS_OLD_STORCLASS=               OPENMVS
000018 ZFS_OLD_NAME_SAV=                 HERING.TEST.NOT.EXTADDR
000019 ZFS_NEW_NAME_TMP=                 HERING.TEST.EXTADDR
000020 ZFS_NEW_NAME=                     HERING.TEST.EXTADDR
000021 #ZFS_NEW_#_VOLUMES=               4
000022 ZFS_NEW_VOLUMES=
000023 ZFS_NEW_ALLOC_NUM_CAND_VOLUMES=  2
000024 ZFS_NEW_ALLOC_UNIT=               CYLINDERS
000025 ZFS_NEW_ALLOC_SPACE=              500 500
000026 ZFS_NEW_ALLOC_NUM_SEC_ALLOCS=    12
000027 ZFS_NEW_DATACLASS=                EXTADDR
000028 ZFS_NEW_MGMNTCLASS=               HFS
000029 ZFS_NEW_STORCLASS=                OPENMVS
000030 ZFS_NEW_REPLACES_ZFS_OLD=         Y
000031
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-25 EDIT session showing the reorganization statements after doing a small change*

For reference Figure 1-26 provides a few cylinder-to-GB calculations.

```
$> rexx
SH> Do i=5000 By 500 To 6000; -
  >   Say i "cylinders are about" Format(i*90*8/(1024*1024),,2) "GB"; -
  > End
5000 cylinders are about 3.43 GB
5500 cylinders are about 3.78 GB
6000 cylinders are about 4.12 GB
SH> Exit
$>
```

*Figure 1-26 Some cylinder to GB calculations*

We opened the ZFSREORG JCL, changed the copy tool used to TSOREPRO, and changed the member name containing the reorganization statements. The job as then submitted is shown in Figure 1-27.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.JOB.CNTL(ZFSREORG) - 01.09           Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 //ZFSJOB   JOB ,'ZFSREORG',NOTIFY=&SYSUID.,REGION=0M
000002 /*JOBPARM SYSAFF=xxxx <=== JES2 system affinity
000003 //*MAIN SYSTEM=SC70   <=== JES3 system affinity
000004 //* --------------------------------------------------------------------
000005 //* Reorganize compat zFS aggregates to new zFS compat mode aggregates
000006 //* Property of IBM  (C) Copyright IBM Corp. 2011
000007 //* --------------------------------------------------------------------
000008 //  SET CPYTOOL=TSOREPRO                  <=== Copy utility to be used
000009 //*             TSOREPRO: Use IDCAMS/TSO REPRO function for copying
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 COPYPAX is set.
000014 //  SET DEFREORG=&SYSUID..ZFS.REORG.DEFINE(REORG03)    <=== REORG DEFs
000015 //* --------------------------------------------------------------------
000016 //  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
000017 //* --------------------------------------------------------------------
000018 //ZFSREORG EXEC PGM=IKJEFT01,PARM='ZFSREORG &CPYTOOL. &VERBOSE.'
000019 //*STEPLIB DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
000020 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000021 //STDENV   DD DATA,DLM=##
000022 # --------------------------------------------------------------------
000023 # Force stopping after formal syntax check of STDIN data is done (N|Y)
000024   STOP_AFTER_SYNTAX_CHECK=N
000025 # Force stopping when old and existing new zFS is/are mounted (N|Y)
000026   STOP_AFTER_FSS_MOUNTED=N
Command ===>                                                 Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

*Figure 1-27  JCL ZFSREORG with the changed CPYTOOL and member name specification*

Figure 1-28 shows the messages received in the joblog while the job was running.

```
ZFSR0065I Running with options Reorgtool=TSOREPRO and Verbose=N ...

-------------------------------------------------------------------------
The main process ID for this job is 67305655. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 67305655
Or    : kill -s SIGTERM 67305655
-------------------------------------------------------------------------

-------------------------------------------------------------------------
ZFSR0004I Processing zFS old data set name HERING.TEST.NOT.EXTADDR...
-------------------------------------------------------------------------
ZFSR0086I Defining zFS aggregate HERING.TEST.EXTADDR ...
IOEZ00248I VSAM linear dataset HERING.TEST.EXTADDR successfully created.
ZFSR0054I 09:33:03 Now starting copy processing...
IDC0005I NUMBER OF RECORDS PROCESSED WAS 918180
ZFSR0070I 09:38:14 Copy processing has been ended...
ZFSR0106I Unmounting HERING.TEST.NOT.EXTADDR ...
ZFSR0107I Mounting zFS HERING.TEST.EXTADDR at /u/hering/test now...
ZFSR0111I No errors have been recognized for the actual REORG process.

READY
END
```

*Figure 1-28 Messages shown in the joblog for the reorganization processing using REPRO*

Figure 1-29 shows what this number of records of records processed means.

```
$> rexx
SH> Say "918180 records processed means" 918180/2 "8K blocks. These are" -
  > 918180/(2*90) "cylinders."
918180 records processed means 459090 8K blocks. These are 5101 cylinders.
SH> exit
$>
```

*Figure 1-29 Calculation for the number of records processed by REPRO*

Figure 1-30 on page 28 shows the results of the reorganization job. This verifies that extended addressability is set now, exactly 5101 cylinders are formatted, and 5500 cylinders are allocated.

**Notes:**

- If TSOREPRO is used, the specification for `ZFS_NEW_ALLOC_NUM_SEC_ALLOCS` is ignored and not used.
- If COPYPAX is used, the resulting zFS would have 6500 cylinders fully formatted (500 + 12*500 cylinders).

```
$> sudo /usr/sbin/mount -qv test
----A- HERING.TEST.EXTADDR /u/hering/test
$> rxzfsmon hering.test.extaddr

Aggregate name                              8K blk tot 8K blk use Percent  XQRS
------------------------------------------- ---------- ---------- -------  ----
HERING.TEST.EXTADDR                             459090     459075 =100.0%  X--S

$> tsocmd "listcat ent(test.extaddr) all" 2>/dev/null | grep SPACE-TYPE
       SPACE-TYPE------CYLINDER     HI-A-RBA------4055040000
$> tsocmd "listcat ent(test.extaddr) all" 2>/dev/null | grep SPACE-PRI
       SPACE-PRI------------500     HI-U-RBA------3760865280
$> tsocmd "listcat ent('hering.test.extaddr') all" 2>/dev/null | grep DATACLASS
       DATACLASS -------EXTADDR     LBACKUP ---0000.000.0000
$> tsocmd "listcat ent('hering.test.extaddr') all" 2>/dev/null | grep EXT-ADDR
       UNORDERED        NOREUSE     NONSPANNED      EXTENDED     EXT-ADDR
      ZFS
$> rexx
SH> numeric digits 10
SH> say 459090*8*1024
3760865280
SH> say (4055040000/90)/(1024*8) "cylinders"
5500 cylinders
SH> exit
$>
```

*Figure 1-30 Showing resulting information after the reorganization processing*

We hope that the description and samples provided, together with the online help that is available, is enough to understand how to install, customize, and use the ZFSREORG tool.

2

# The zFS reorganization files

This chapter contains the ZFSREORG files and JCL for running the tool. These REXX procedures are downloaded into your `myuser.ZFS.REXX.EXEC` library or they can be placed in any data set you choose.

The following files and a short description of each are included here:

- ZFSREORG
- CPYRHELP
- DEFRHELP
- DEFREORG
- DZRC$MAC
- DZRH$MAC
- DZRM$MAC
- DZRP$MAC
- DZRR$MAC
- ZFSREORG JCL

## 2.1 ZFSREORG

The ZFSREORG REXX procedure is used in the reorganization job to control the complete processing, beginning with verification steps, selecting the desired options, the reorganization copy utility, starting the copy processing, and optionally replacing the old zFS file system with the new one.

Example 2-1 displays the contents of ZFSREORG.

*Example 2-1 ZFSREORG REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: ZFSREORG (Next free MSGNO: Say "ZFSRO135x ...")      */
/*  Description: Reorganize a zFS compat mode aggregate to a new one  */
/*               Property of IBM  (C) Copyright IBM Corp. 2011        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: zfsreorg reorgtool <verbose>                         */
/*********************************************************************/

Trace O
Parse Source . calltype myname .

switched = 0
final_rc = 0
no_msgs = 1
noexit_on_error = 1
info_on_timeout = 2
fd. = -1
pp. = -1
sig_enabled = 0
tmp_dir_defined = 0
max_no_vols = 59 /* maximal number of volumes for a zFS aggregate */
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
msg_save = Msg()
zfo_temp_mounted = 0
zfs_temp_mounted = 0
zfo_switched_ro = 0
zfs_exists = 0
zfs_formatted = 0
zfs_created = 0
zfs_usable = 0
zfs_mounted = 0
zfo_processed. = 0
error_found = 0
denied_umnt_fstypes ="TFS" /* do not unmount these fstype fss */
swsu = 1 /* forces SU switching */
local_system = MVSVar("SYMDEF","SYSNAME")

Signal On Syntax Name Syntax_Error
Signal On Novalue Name Novalue_Error

Parse Upper Arg reorgtool verbose .
If Wordpos(reorgtool,"TSOREPRO COPYPAX COPYTREE")=0 Then Do
  Say "ZFSROOO1E You must provide TSOREPRO, COPYPAX or COPYTREE as the",
```

```
    "first parameter"
  Say "to specify the copytool utility to be used."
  Exit 8
End

timeout_value = "R0" /* force this as the best selection */
If verbose="" Then verbose = "N"
If verbose<>"Y" & verbose<>"N" Then Do
  Say "ZFSR0063I Option Verbose, if specified, must be Y or N."
  Exit 8
End

Say "ZFSR0065I Running with options Reorgtool=" || reorgtool "and",
  "Verbose="||verbose "..."
If verbose="Y" Then verbose = "v"
Else verbose = ""
Say
do_wait = 1
not_do_wait = 0
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 "ZFSR0002E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "ZFSR0003E The SIGNAL interface could not be established."
  Exit 8
End

If foreground Then Do
  Say "ZFSR0080E Processing is not yet supported in foreground."
  Exit 8
End

/* ---------------------------------------------------------------- */
/*  Switch effective and/or real UID to zero if needed and possible   */
/* ---------------------------------------------------------------- */

If swsu Then Do
  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 & foreground Then Do
      Say "UID setting switched to 0..."
      switched = 1
    End
  End
End

/* ---------------------------------------------------------------- */
/*  Set home and working directory, setup envvars, open /dev/null   */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "getpid"
mainprocess_pid = retval
If background & calltype="COMMAND" 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"
  Say "processing use the following UNIX command to do this smoothly."
  If do_wait Then Do
    signal_type = "SIGALRM"
    Say "Use: kill -s" signal_type mainprocess_pid
  End
  Else Do
    signal_type = "SIGTERM"
    Say "Either: kill" mainprocess_pid
    Say "Or    : kill -s" signal_type mainprocess_pid
  End
  Say dash_line
End
Call Syscall_Cmd "getcwd cur_cwd", noexit_on_error
If not_OK Then Do
  If errno="86" & errnojr="52C04DC" Then
    Say "ZFSRO126W The user's home directory or CWD is not available.",
      "Nevertheless, the processing continues."
  Else
    Say "ZFSRO127W Although ""getcwd"" failed, processing is tried to",
      "be continued."
End
If Symbol("cur_cwd")="LIT" Then cur_cwd = "/"
If cur_cwd="" Then cur_cwd = "/"
If swsu Then home = "/"
Else Do
  Call Syscall_Cmd "getpwnam" Userid() "omvs."
  home = omvs.pw_dir
End
Call Syscall_Cmd "chdir /"
Call Syscall_Cmd "realpath /tmp tmp_dir"
tmp_dir = tmp_dir||"/COPYREORG."||DelStr(Delstr(Time("L"),3,1),5,1)
Call Syscall_Cmd "mkdir (tmp_dir) 755"
tmp_dir_defined = 1
```

```
__environment.1 = "_BPX_SHAREAS=YES"
__environment.2 = "PATH="||tmp_dir||":/bin:/samples"
__environment.3 = "HOME="||home
__environment.4 = "LOGNAME="||Userid()
__environment.5 = "PWD="||home
__environment.6 = "_EDC_ADD_ERRNO2=1"
envvars = 6
Call Syscall_Cmd "chdir (tmp_dir)"
Call Syscall_Cmd "mkdir tmp_zfo 755"
tmp_zfo = tmp_dir||"/"||"tmp_zfo"
Call Syscall_Cmd "mkdir tmp_zfs 755"
tmp_zfs = tmp_dir||"/"||"tmp_zfs"
Call Syscall_Cmd "extlink IOEZADM ioezadm"
Call Syscall_Cmd "chdir /"
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 i
    If Left(envvar_line,5)="PATH=" Then Do
      __environment.2 = "PATH="||tmp_dir||":"||Substr(envvar_line,6)||,
        ":/bin:/samples"
      Iterate i
    End
    pos_equal = Pos("=",envvar_line)
    If Left(envvar_line,pos_equal-1)="DENIED_UMNT_FSTYPES" Then Do
      denied_umnt_fstypes =Translate(Substr(envvar_line,pos_equal+1))
      Iterate i
    End
    If Left(envvar_line,pos_equal-1)="STOP_AFTER_SYNTAX_CHECK" Then Do
      If Translate(Substr(envvar_line,pos_equal+1))="Y" Then
        stop_after_syntax_check = 1
      Else
        stop_after_syntax_check = 0
      Iterate i
    End
    If Left(envvar_line,pos_equal-1)="STOP_AFTER_FSS_MOUNTED" Then Do
      If Translate(Substr(envvar_line,pos_equal+1))="Y" Then
        stop_after_fss_mounted = 1
      Else
        stop_after_fss_mounted = 0
      Iterate i
    End
    If Left(envvar_line,pos_equal-1)="STOP_AFTER_ZFS_IS_FORMATTED" Then
      Do
        If Translate(Substr(envvar_line,pos_equal+1))="Y" Then
          stop_after_zfs_is_formatted = 1
        Else
          stop_after_zfs_is_formatted = 0
        Iterate i
      End
    If Left(envvar_line,pos_equal-1)="TARGET_ZFS_MUST_BE_EMPTY" Then Do
      If Translate(Substr(envvar_line,pos_equal+1))="N" Then
        target_zfs_must_be_empty = 0
```

```
      Else
        target_zfs_must_be_empty = 1
      iterate i
    end
    envvars = envvars+1
    __environment.envvars = envvar_line
  end
end
If Symbol("stop_after_syntax_check")="LIT" Then
  stop_after_syntax_check = 0
If Symbol("stop_after_fss_mounted")="LIT" Then
  stop_after_fss_mounted = 0
If Symbol("stop_after_zfs_is_formatted")="LIT" Then
  stop_after_zfs_is_formatted = 0
If Symbol("target_zfs_must_be_empty")="LIT" Then
  target_zfs_must_be_empty = 0
__environment.0 = envvars
Call Syscall_Cmd "open /dev/null (o_rdonly)"
fd.0 = retval

/* ------------------------------------------------------------------ */
/*  Handling of signal 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 Do
  Call Syscall_Cmd "sigprocmask" sig_block,
    sigaddset(sigsetempty(),sigterm) "mask"
  Call Syscall_Cmd "sigaction" sigterm sig_cat 0 "xhdl xflg"
End

/* ------------------------------------------------------------------ */
/*  Read in all the reorg statements                                  */
/* ------------------------------------------------------------------ */

If background Then Do
  Call Bpxwdyn "INFO DD(STDIN)"
  If Pos(Left(Right(D2x(result),8,"0"),4),"0438 0440")<>0 Then Do
    /* IKJ56247I FILE INFO-RETRIEVAL NOT PERFORMED, NOT ALLOCATED */
    Say "ZFSRO128E The necessary DDNAME STDIN is missing."
    Call Final_Exit 8
  End
  Else Do
    "EXECIO * DISKR STDIN (STEM INPUTD. FINIS"
    If rc<>0 Then Do
      Say "ZFSRO122E The specified member on DDNAME STDIN probably",
        "does not exist."
      Call Final_Exit 8
    End
  End
  in_lines = inputd.0
End
```

```
Else Do
  Parse Upper Pull fg_stdin
  If fg_stdin="" Then Call Final_Exit 0
  "EXECIO * DISKR" FG_STDIN "(STEM INPUTD. FINIS"
  If rc<>0 Then Call Final_Exit 8
  in_lines = inputd.0
End

/* ---------------------------------------------------------------- */
/*  Run all the reorg processing                                    */
/* ---------------------------------------------------------------- */

Call Syscall_Cmd "stat /dev st."
denied_umnt_devnos = X2d(st.st_dev) /* Do not allow unmounting of */
Call Syscall_Cmd "stat /tmp st."    /* /dev and /tmp structure    */
denied_umnt_devnos = denied_umnt_devnos X2d(st.st_dev)
s_zfo_name = "ZFS_OLD_NAME"
s_zfo_name_sav = "ZFS_OLD_NAME_SAV"
s_zfs_name_tmp = "ZFS_NEW_NAME_TMP"
s_zfs_name = "ZFS_NEW_NAME"
s_zfs_volumes = "ZFS_NEW_VOLUMES"
s_zfs_alloc_unit = "ZFS_NEW_ALLOC_UNIT"
s_zfs_alloc_space = "ZFS_NEW_ALLOC_SPACE"
s_zfs_alloc_nocvols = "ZFS_NEW_ALLOC_NUM_CAND_VOLUMES"
s_zfs_alloc_nosallc = "ZFS_NEW_ALLOC_NUM_SEC_ALLOCS"
s_zfs_dataclass = "ZFS_NEW_DATACLASS"
s_zfs_mgmntclass = "ZFS_NEW_MGMNTCLASS"
s_zfs_storclass = "ZFS_NEW_STORCLASS"
s_zfs_replaces_zfo = "ZFS_NEW_REPLACES_ZFS_OLD"
not_first_zfo = 0
in_lines = in_lines+1
inputd.in_lines = s_zfo_name||"=Dummy.zFS_OLD"
Do i=1 To in_lines
  in_line = Translate(Strip(inputd.i))
  If Left(in_line,1)="#" | in_line="" Then Iterate i
  Parse Var in_line in_parm "=" in_value
  error_in_line = 0
  Select
    When in_parm=s_zfo_name Then Do
      If not_first_zfo Then Do
        Call Migrate_zFS2zFS
        Call Migrate_Data_Reset 1 /* 1= display curmig msg */
      End
      Else Do
        not_first_zfo = 1
        Call Migrate_Data_Reset 0 /* 0= no curmig msg */
      End
      Parse Var in_value zfo_name .
      If i<in_lines Then Do
        Say
        Say dash_line
        Say "ZFSR0004I Processing zFS old data set name" zfo_name||"..."
        Say dash_line
      end
      If zfo_name="" Then Do
```

```
    Say "ZFSR0005E No zFS old data set name has been provided."
    error_in_line = 1
  End
  Else If zfo_processed.zfo_name Then Do
    Say "ZFSR0006W zFS old dataset name has been processed already."
    error_in_line = 1
  End
  zfo_processed.zfo_name = 1
End
When in_parm=s_zfo_name_sav Then
  Parse Var in_value zfo_name_sav .
When in_parm=s_zfs_name_tmp Then
  Parse Var in_value zfs_name_tmp .
When in_parm=s_zfs_name Then Do
  Parse Var in_value zfs_name .
  If zfs_name="" Then Do
    Say "ZFSR0007E No zFS aggregate name has been provided."
    error_in_line = 1
  End
End
When in_parm=s_zfs_volumes Then Do
  Parse Var in_value zfs_volumes_list
  Do While zfs_volumes_list<>""
    Parse Var zfs_volumes_list zfs_volume zfs_volumes_list
    zfs_volumes = zfs_volumes zfs_volume
  End
End
When in_parm=s_zfs_alloc_unit Then Do
  Parse Var in_value zfs_alloc_unit .
  If Wordpos(zfs_alloc_unit,"CYLINDERS CYL TRACKS TRK")=0 Then Do
    Say "ZFSR0008W zfs_alloc_unit must be CYLINDERS (CYL) or",
      "TRACKS (TRK)."
    error_in_line = 1
  End
End
When in_parm=s_zfs_alloc_space Then Do
  Parse Var in_value prim_alloc sec_alloc .
  If Verify(prim_alloc||sec_alloc,"1234567890")<>0 Then Do
    Say "ZFSR0009E primary and secondary allocation values must be",
      "positive whole numbers."
    error_in_line = 1
  End
  If prim_alloc=0 Then Do
    Say "ZFSR0061E Primary allocation value cannot be zero."
    error_in_line = 1
  End
  If sec_alloc="" Then sec_alloc = "0"
  zfs_alloc_space = prim_alloc sec_alloc
End
When in_parm=s_zfs_alloc_nocvols Then Do
  Parse Var in_value zfs_alloc_nocvols .
  If Verify(zfs_alloc_nocvols,"1234567890")<>0 |,
     zfs_alloc_nocvols="" Then Do
    Say "ZFSR0010E The number of candidate volumes must be a whole",
      "number."
```

```
        error_in_line = 1
        zfs_alloc_nocvols = 0
      End
    End
    When in_parm=s_zfs_alloc_nosallc Then Do
      Parse Var in_value zfs_alloc_nosallc .
      If Verify(zfs_alloc_nosallc,"1234567890")<>0 |,
         zfs_alloc_nosallc="" Then Do
        Say "ZFSRO011E The number of repeated secondary allocations",
          "must be a whole number."
        error_in_line = 1
        zfs_alloc_nosallc = 0
      End
    End
    When in_parm=s_zfs_dataclass Then Parse Var in_value zfs_dataclass .
    When in_parm=s_zfs_mgmntclass Then
      Parse Var in_value zfs_mgmntclass .
    When in_parm=s_zfs_storclass Then Parse Var in_value zfs_storclass .
    When in_parm=s_zfs_replaces_zfo Then Do
      Parse Var in_value zfs_replaces_zfo .
      If Wordpos(zfs_replaces_zfo,"Y N")=0 Then Do
        Say "ZFSRO012E The specification for replacing the old zFS",
          "must be Y or N."
        error_in_line = 1
      End
    End
    Otherwise Do
      Say "ZFSRO013E The specified option is unknown."
      error_in_line = 1
    End
  End
  If error_in_line Then Do
    Say "ZFSRO014I Invalid specification found in line" i||":"
    Say Left("",9) inputd.i
    error_found = 1
    final_rc = 8
  End
End

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit final_rc
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Migrate_Data_Reset:
  Parse Arg curmig_msg
  If zfs_temp_mounted Then Do
    Call syscall_cmd "unmount" zfs_name "(mtm_immed)", noexit_on_error
    Call Test4_unmounted zfs_name
```

```
End
If error_found Then Do
  final_rc = 8
  If zfo_switched_ro Then Do
    If zfs_usable Then Nop
    Else Do
      Call syscall_cmd "unmount" zfo_name "(mtm_remount)",,
        noexit_on_error
      If OK Then Say "ZFSRO074W Old zFS has been remounted as",
        "read-write as the copy processing was not or not completely",
        "successful."
    End
  End
  Select
    When zfs_mounted Then Nop
    When zfs_usable Then
      Say "ZFSRO125I The migration copy processing was successful",
        "and you can replace old zFS by the new zFS yourself later."
    When zfs_exists Then
      Say "ZFSRO071I Data set" zfs_name "kept as it existed already."
    When zfs_formatted Then
      Say "ZFSRO072I Data set" zfs_name "kept as it has been",
        "formatted at least."
    When zfs_created Then Do
      Say "ZFSRO073I Data set" zfs_name "will be deleted as it",
        "has been defined only."
      "DELETE" "'"||zfs_name||"'"
    End
    Otherwise Nop
  End
End
If zfo_temp_mounted Then
  Call Syscall_Cmd "unmount (zfo_name) (mtm_immed)", noexit_on_error
If curmig_msg Then Do
  If error_found Then Say "ZFSRO110W One or more errors occurred",
    "during the actual REORG process."
  Else Say "ZFSRO111I No errors have been recognized for the actual",
    "REORG process."
End
Parse Value "" With zfo_name zfo_name_sav zfs_name_tmp zfs_name,
  zfs_volumes
zfs_alloc_unit = "CYLINDERS"
zfs_alloc_space = ""
zfs_alloc_nocvols = 0 /* number of additional candidate volumes */
zfs_alloc_nosallc = 0 /* number of initial grows using sec alloc */
Parse Value "" With zfs_dataclass zfs_mgmntclass zfs_storclass
zfs_replaces_zfo = "N"
error_found = 0
zfo_switched_ro = 0
zfo_temp_mounted = 0
zfs_temp_mounted = 0
zfs_name_same_as_zfo = 0
zfs_exists = 0
zfs_created = 0
zfs_formatted = 0
```

```
  zfs_usable = 0
  zfs_mounted = 0
Return

Migrate_zFS2zFS:
  Call Msg "OFF"
  zfo_dsn_status = Sysdsn("'"||zfo_name||"'")
  Call Msg msg_save
  Parse Value zfo_dsn_status With invalid_dsn "," .
  Select
    When zfo_dsn_status="DATASET NOT FOUND" Then Do
      Say "ZFSRO015W The old zFS data set cannot be found."
      error_found = 1
    End
    When invalid_dsn="INVALID DATASET NAME" Then Do
      Say "ZFSRO016W The old zFS data set name specified is not valid."
      error_found = 1
    End
    When zfo_dsn_status="OK" Then Nop
    When zfo_dsn_status="UNAVAILABLE DATASET" Then Nop /* expect OK */
    Otherwise Do
      Say "ZFSRO017W The old zFS data set status is not as expected:"
      Say "ZFSRO018W" zfo_dsn_status
      error_found = 1
    End
  End
  If error_found Then Nop /* Skip test for old zFS mounted already */
  Else Do
    Call Syscall_Cmd "statfs (zfo_name) st.", noexit_on_error, no_msgs
    Select
      When OK Then Do
        zfo_devno = st.stfs_fsid
        Call Syscall_Cmd "getmntent zfo." zfo_devno
        read_only = Bitand(D2c(zfo.mnte_mode.1),D2c(mnt_mode_rdonly))
        read_only = C2d(read_only)
        read_write = (read_only=0)
        zfo_mntdir = zfo.mnte_path.1
        zfo_fstype = zfo.mnte_fstype.1
        zfo_mounted = 1
        If zfo_fstype<>"ZFS" Then Do
          Say "ZFSRO090E The dataset specified as old zFS is not a zFS."
          error_found = 1
        End
      End
      When rc=0 & retval=-1 & errno=79 & errnojr="567002E" Then
        zfo_mounted = 0
      Otherwise Do
        Say "ZFSRO019W The status of the old zFS is not expected."
        Say "ZFSRO020W Rc=" rc "retval=" retval "errno=" errno,
          "errnojr=" errnojr
        error_found = 1
      End
    End /* Select */
  End
  zfs_volumes = zfs_volumes||,
```

```
  Copies(" *",Min(max_no_vols,zfs_alloc_nocvols))
If Words(zfs_volumes)>max_no_vols Then Do
  Say "ZFSRO021W The number of volumes specified cannot exceed a",
    "value of" max_no_vols||"."
  error_found = 1
End
If zfs_alloc_space="" Then Do
  Say "ZFSRO022W A primary allocation number greater zero is",
    "required."
  error_found = 1
End
Parse Var zfs_alloc_space prim_alloc sec_alloc .
If zfs_alloc_nosallc<>0 & sec_alloc=0 Then Do
  Say "ZFSRO062W To extend the zFS aggregate the secondary",
    "allocation size cannot be zero."
  error_found = 1
End
If zfs_name=zfo_name Then Do
  zfs_name_same_as_zfo = 1
  If zfo_name_sav="" Then
    zfo_name_sav = Strip(Strip(Left(zfo_name,40)),"T",".")||".SAV"
  Call Msg "OFF"
  zfos_dsn_status = Sysdsn("'"||zfo_name_sav||"'")
  Call Msg msg_save
  Parse Value zfos_dsn_status With invalid_dsn "," .
  Select
    When zfos_dsn_status="DATASET NOT FOUND" Then Nop /* not used */
    When invalid_dsn="INVALID DATASET NAME" Then Do
      Say "ZFSRO113W The zFS_OLD save name specified is not valid."
      error_found = 1
    End
    When zfo_name_sav=zfo_name Then Do
      Say "ZFSRO114W The old zFS and the new zFS save data set",
        "cannot have the same name."
      error_found = 1
    End
    When zfos_dsn_status="OK" | zfos_dsn_status="UNAVAILABLE DATASET"
      Then Do
      Say "ZFSRO115W The old zFS save data set" zfo_name "exists",
        "already."
      error_found = 1
    End
    Otherwise Do
      Say "ZFSRO116W The old zFS save data set status is not as",
        "expected:"
      Say "ZFSRO117W" zfos_dsn_status
      error_found = 1
    End
  End
  If zfs_name_tmp="" Then
    zfs_name_tmp = Strip(Strip(Left(zfs_name,40)),"T",".")||".TMP"
  zfs_name = zfs_name_tmp
  Say "ZFSRO075I As the new zFS aggregate name is the same as of the",
    "old zFS data set a temporary name is used for the new zFS."
  Say "ZFSRO076I zFS temporary name:" zfs_name
```

```
End
Call Msg "OFF"
zfs_dsn_status = Sysdsn("'"||zfs_name||"'")
Call Msg msg_save
Parse Value zfs_dsn_status With invalid_dsn "," .
Select
  When zfs_name_same_as_zfo & (zfs_name=zfo_name |,
    zfs_name=zfo_name_sav) Then Do
    Say "ZFSRO118W The new zFS aggregate temporary name cannot be",
      "the same as that of the old zFS or zFS save data set."
    error_found = 1
  End
  When zfs_dsn_status="DATASET NOT FOUND" Then Nop /* zfs_exists=0 */
  When invalid_dsn="INVALID DATASET NAME" Then Do
    Say "ZFSRO023W The zFS aggregate name specified is not valid."
    error_found = 1
  End
  When zfs_dsn_status="OK" Then zfs_exists = 1
  When zfs_dsn_status="UNAVAILABLE DATASET" Then zfs_exists = 1
  Otherwise Do
    Say "ZFSRO024W The zFS aggregate status is not as expected:"
    Say "ZFSRO025W" zfs_dsn_status
    error_found = 1
  End
End
If zfs_exists Then Do /* Test whether zFS is mounted already */
  Call Syscall_Cmd "statfs (zfs_name) st.", noexit_on_error, no_msgs
  Select
    When OK Then Do
      zfs_devno = st.stfs_fsid
      Call Syscall_Cmd "getmntent mnt." zfs_devno
      zfs_mntdir = mnt.mnte_path.1
      zfs_fstype = mnt.mnte_fstype.1
      If zfs_fstype="ZFS" Then Do
        Say "ZFSRO026W The new zFS aggregate specified is mounted",
          "already."
        Say "ZFSRO029W Mount directory :" zfs_mntdir
      End
      Else Do
        Say "ZFSRO027W The data set specified as zFS is mounted",
          "already and is not a zFS."
        Say "ZFSRO028W File system type:" zfs_fstype
      End
      error_found = 1
    End         /* next condition true means zFS is not mounted */
    When rc=0 & retval=-1 & errno=79 & errnojr="567002E" Then Nop
    Otherwise Do
      Say "ZFSRO030W The status of the zFS aggregate is not expected."
      Say "ZFSRO031W Rc=" rc "retval=" retval "errno=" errno,
        "errnojr=" errnojr
      error_found = 1
    End
  End /* Select */
End
```

```
If error_found Then Do
  Say "ZFSR0032W One or more specification errors found for current",
    "migration."
  Say "ZFSR0033E Skipping zFS" zfo_name||"..."
  Return
End

If stop_after_syntax_check Then Do
  Say "ZFSR0034I No specification errors found for current migration."
  Return
End

If zfo_mounted Then Do
  If read_write Then Do
    Call Syscall_Cmd "unmount (zfo_name)" mtm_remount,,
      noexit_on_error
    If OK Then zfo_switched_ro = 1
    Else Do
      Say "ZFSR0036E Old zFS mount mode could not be switched to R/O."
      error_found = 1
      Return
    End
  End
End
Else Do
  Call Syscall_Cmd "mount (tmp_zfo)" zfo_name "ZFS (mtm_rdonly)",,
    noexit_on_error
  If OK Then Do
    zfo_temp_mounted = 1
    zfo_mntdir = tmp_zfo
    Call Syscall_Cmd "statvfs (tmp_zfo) st."
    zfo_devno = st.stfs_fsid
    Call Syscall_Cmd "getmntent zfo." zfo_devno
    If zfo.mnte_fstype.1<>"ZFS" Then Do
      Say "ZFSR0091E The data set specified as zFS is not a zFS."
      error_found = 1
      Return
    End
  End
  Else Do
    Say "ZFSR0037E zFS file system could not be mounted read-only."
    error_found = 1
    Return
  End
End

If zfs_exists Then Do
  If reorgtool<>"TSOREPRO" Then Do
    Call Syscall_Cmd "mount (tmp_zfs)" zfs_name "ZFS (mtm_rdwr)",
      "aggrgrow", noexit_on_error
    If OK Then zfs_temp_mounted = 1
    Else Do
      Say "ZFSR0038E The new zFS aggregate could not be mounted."
      error_found = 1
      Return
```

```
    End
  End
  If stop_after_fss_mounted Then Do
    Say "ZFSR0092I Old zFS and existing new zFS are or could be",
      "mounted, if needed."
    Return
  End
End
Else Do
  If stop_after_fss_mounted Then Do
    Say "ZFSR0093I Old zFS is mounted, new zFS aggregate needs to be",
      "created."
    Return
  End
  parms = "/"tmp_dir"/"||"ioezadm define -aggregate" zfs_name
  If zfs_dataclass<>"" Then
    parms = parms "-dataclass" zfs_dataclass
  If zfs_mgmntclass<>"" Then
    parms = parms "-managementclass" zfs_mgmntclass
  If zfs_storclass<>"" Then
    parms = parms "-storageclass" zfs_storclass
  If zfs_volumes<>"" Then
    parms = parms "-volumes" zfs_volumes
    If Wordpos(zfs_alloc_unit,"CYLINDERS CYL")<>0 Then
      zfs_alloc_unit = "-cylinders"
    Else
      zfs_alloc_unit = "-tracks"
  parms = parms zfs_alloc_unit zfs_alloc_space
  parm.0 = Words(parms)
  Do j=1 To parm.0
    Parse Var parms parm.j parms
  End
  Say "ZFSR0086I Defining zFS aggregate" zfs_name "..."
  Call Shell_Cmd
  If shell_rc<>0 Then Do
    Say "ZFSR0039E zFS aggregate could not be defined correctly."
    error_found = 1
    Return
  End
  Else zfs_created = 1

  If reorgtool<>"TSOREPRO" Then Do

    parms = tmp_dir"/"||"ioezadm format -aggregate" zfs_name,
      "-compat"
    parm.0 = Words(parms)
    Do j=1 To parm.0
      Parse Var parms parm.j parms
    End
    Say "ZFSR0087I" Time() "Formatting zFS aggregate" zfs_name "..."
    Call Shell_Cmd
    If shell_rc<>0 Then Do
      Say "ZFSR0040E zFS aggregate could not be formatted correctly."
      error_found = 1
      Return
```

```
    End
    Else zfs_formatted = 1

    Call Syscall_Cmd "mount (tmp_zfs)" zfs_name "ZFS (mtm_rdwr)",
      "aggrgrow", noexit_on_error
    If OK Then zfs_temp_mounted = 1
    Else Do
      Say "ZFSRO050E zFS aggregate could not be mounted."
      error_found = 1
      Return
    End

    parms = tmp_dir"/"||"ioezadm grow -aggregate" zfs_name "-size 0"
    parm.0 = Words(parms)
    Do j=1 To parm.0
      Parse Var parms parm.j parms
    End
    If zfs_alloc_nosallc<>0 Then
      Say "ZFSRO088I Growing zFS aggregate" zfs_name,
        zfs_alloc_nosallc "times..."
    Do j=1 To zfs_alloc_nosallc
      Call Shell_Cmd
      If shell_rc<>0 Then Do
        Say "ZFSRO051E zFS aggregate could not be grown correctly."
        error_found = 1
        Return
      End
    End j
  End

End /* If reorgtool<>"TSOREPRO" */

If stop_after_zfs_is_formatted Then Do
  Say "ZFSRO094I Old zFS is mounted, new zFS aggregate has been",
    "created."
  Say "ZFSRO035I Old zFS data set name=" zfo_name
  Return
End

If target_zfs_must_be_empty & reorgtool<>"TSOREPRO" Then Do
  zfs_not_empty = 0
  Call Syscall_Cmd "opendir (tmp_zfs)"
  zfs_fd = retval
  Call Syscall_Cmd "rddir (zfs_fd) zfs_data 19"
  zfs_entries = retval
  Call Syscall_Cmd "rddir (zfs_fd) zfs_data 19",,
    noexit_on_error, no_display
  If OK Then Do
    zfs_entries = zfs_entries+retval
    If zfs_entries>2 Then zfs_not_empty = 1
  End
  Else Do
    If (errno="79" & errnojr="EF116126") Then zfs_not_empty = 1
    Else Do
      Say "ZFSRO095E Unexpected zFS rddir error, errno=" errno,
```

```
        "errnojr=" errnojr
      error_found = 1
    End
  End
  If zfs_not_empty Then Do
    Say "ZFSR0096E The target zFS aggregate is not empty, but this",
      "is requested."
    error_found = 1
  End
  Call Syscall_Cmd "closedir (zfs_fd)"
  If error_found Then Return
End

Call Syscall_Cmd "chdir (zfo_mntdir)"
Say "ZFSR0054I" Time() "Now starting copy processing..."
Select
  When reorgtool="TSOREPRO" Then Do
    repro_OK = 1
    no_vsamc = 1
    notempty = 1
    lc.0 = 0
    Call OUTTRAP "LC."
    "LISTCAT ENTRIES('"||zfs_name||"') ALLOCATION"
    retc = rc
    Call OUTTRAP "OFF"
    If retc<>0 Then Do
      Say "ZFSR0133E Unexpected error occured on LISTCAT, rc=" retc
      repro_OK = 0
      repro_rc = 8
    End
    Else Do
      Do lc=1 To lc.0-2
        lc_ln = lc.lc
        Select
          When Word(lc_ln,1)="CLUSTER" & Word(lc_ln,3)=zfs_name Then
            no_vsamc = 0
          When Word(lc_ln,1)="ALLOCATION" Then Do
            lc = lc+2
            lc_ln = lc.lc
            last_dash_pos = Lastpos("-",lc_ln)
            If Substr(lc_ln,last_dash_pos+1)=0 Then notempty = 0
            Leave lc
          End
          Otherwise Nop
        End /* Select */
      End lc
      If no_vsamc | notempty Then Do
        Say "ZFSR0134E The new zFS aggregate could not be verified",
          "to be an empty LDS, but this is needed for REPRO."
        repro_OK = 0
        repro_rc = 8
      End
    End
    /* ---------------------------------------------------------- *
    Call Syscall_Cmd "mount (tmp_zfs) (zfs_name) ZFS (mtm_rdwr)",,
```

```
      noexit_on_error, no_msgs
    If OK Then Do
      zfs_temp_mounted = 1
      Call Syscall_Cmd "unmount (zfs_name) (mtm_immed)",,
        noexit_on_error
      If OK Then zfs_temp_mounted = 0
      Say "ZFSRO129E The new zFS aggregate is formatted already, but",
        "this is not allowed for REPRO."
      repro_OK = 0
      repro_rc = 8
    End
     * ---------------------------------------------------------- */
    If repro_OK Then Do
      Call Bpxwdyn "ALLOC DA("||zfo_name||") DD(ZFSOLD) SHR",
        "MSG(MSG99.)"
      If result<>0 Then Do
        repro_OK = 0
        repro_rc = result
        Do kk=1 To msg99.0; Say msg99.i; End
      End
    End
    If repro_OK Then Do
      Call Bpxwdyn "ALLOC DA("||zfs_name||") DD(ZFSNEW) OLD",
        "MSG(MSG99.)"
      If result<>0 Then Do
        repro_OK = 0
        repro_rc = result
        Do kk=1 To msg99.0; Say msg99.i; End
      End
    End
    If repro_OK Then Do
      "REPRO INFILE(ZFSOLD) OUTFILE(ZFSNEW)"
      If rc<>0 Then Do
        repro_OK = 0
        repro_rc = rc
      End
    End
    Call Bpxwdyn "FREE DD(ZFSOLD) MSG(MSG99.)"
    Call Bpxwdyn "FREE DD(ZFSNEW) MSG(MSG99.)"
    If repro_OK=0 Then Do
      Say "ZFSRO052E TSOREPRO processing ended with rc=" repro_rc||"."
      error_found = 1
      Return
    End
  End
  When reorgtool="COPYPAX" Then Do
    parm.1 = "sh"
    parm.2 = "-c"
    parm.3 = "pax -rw"||verbose "-peW -XCM ." tmp_zfs
    parm.0 = 3
    Call Shell_Cmd
    If shell_rc<>0 Then Do
      Say "ZFSRO053E COPYPAX processing ended with rc=" shell_rc||"."
      error_found = 1
      Return
```

```
    End
  End
  When reorgtool="COPYTREE" Then Do
    parm.0 = 3
    parm.1 = "sh"
    parm.2 = "-c"
    parm.3 = "copytree -a ." tmp_zfs
    parm.0 = 3
    Call Shell_Cmd
    If shell_rc<>0 Then Do
      Say "ZFSRO130E COPYTREE processing ended with rc=" shell_rc||"."
      error_found = 1
      Return
    End
  End
  Otherwise Nop /* this will be never selected */
End
Say "ZFSRO070I" Time() "Copy processing has been ended..."
zfs_usable = 1
Call Syscall_Cmd "chdir /"

If zfo_temp_mounted Then Do
  Call Syscall_Cmd "unmount" zfo_name "(mtm_normal)", noexit_on_error
  If OK Then zfo_temp_mounted = 0
  Call Test4_unmounted zfo_name
End
If zfs_temp_mounted Then Do
  Call Syscall_Cmd "unmount" zfs_name "(mtm_normal)", noexit_on_error
  If OK Then zfs_temp_mounted = 0
  Call Test4_unmounted zfs_name
End
zfo_still_mounted = zfo_mounted
If zfs_replaces_zfo="Y" & zfo_mounted Then Do
  Call Syscall_Cmd "statfs" zfo_name "st."
  zfo_devno = st.stfs_fsid
  Call Syscall_Cmd "getmntent mnt."
  subfss. = ""
  Do j=1 To mnt.0
    par_devno = mnt.mnte_pardev.j
    subfss.par_devno = subfss.par_devno j
  End
  zfo_fss = 0
  Call Build_List_Of_Subfss zfo_devno
  do_unmounting = 1
  Do j=zfo_fss By -1 To 1
    mnt_index = zfo_fs.j
    fsdevno = mnt.mnte_dev.mnt_index
    fstype = mnt.mnte_fstype.mnt_index
    If Wordpos(mnt.mnte_fstype.mnt_index,denied_umnt_fstypes)<>0 Then
      Do
        Say "ZFSRO123E One PFS down the USS structure is marked to",
          "prevent unmounting of the sub structures; no replacement",
          "of the file systems is done."
        error_found = 1
        do_unmounting = 0
```

```
        End
      If Wordpos(mnt.mnte_dev.mnt_index,denied_umnt_devnos)<>0 Then
        Do
          Say "ZFSRO124E The /dev and the /tmp structure cannot be",
            "unmounted; no replacement of the file systems is done."
          error_found = 1
          do_unmounting = 0
        End
    End
    rmnt_index = zfo_fss+1
    Do j=zfo_fss By -1 To 1 While do_unmounting
      mnt_index = zfo_fs.j
      fsname = Strip(mnt.mnte_fsname.mnt_index)
      fspath = mnt.mnte_path.mnt_index
      Say "ZFSRO105I Temporary unmounting" fsname "mounted at" fspath
      Call Syscall_Cmd "unmount" fsname "(mtm_immed)", noexit_on_error
      If OK Then rmnt_index = j
      If not_OK Then Do
        Say "ZFSRO082E Error ocurred, stopping unmount processing..."
        error_found = 1
        do_unmounting = 0
      End
    End
    If do_unmounting Then Do
      Say "ZFSRO106I Unmounting" zfo_name "..."
      Call Syscall_Cmd "unmount" zfo_name "(mtm_immed)",,
        noexit_on_error
      If OK Then Do
        rmnt_index = 0
        zfo_switched_ro = 0
        Do kk=1 By 1
          Call Bpxwdyn "ALLOC DSN("||zfo_name||") OLD"
          If result=0 Then Do
            Call Bpxwdyn "FREE Dsn("||zfo_name||")"
            zfo_still_mounted = 0
            Leave kk
          End
          Else Do
            If kk=30 Then Do
              Say "ZFSRO089E Old zFS is still blocked; rename/replace",
                "processing will not be performed."
              error_found = 1
              Leave kk
            End
            Call Syscall_Cmd "sleep 1"
          End
        End
      End
      Else Do
        Say "ZFSRO083E Error with old zFS, stopping unmount",
          "processing..."
        error_found = 1
      End
    End
  End
```

```
If zfs_name_same_as_zfo Then Do
  If zfo_still_mounted Then Do
    Say "ZFSRO081E Old and new zFS data sets cannot be renamed as",
      "the old zFS is still mounted."
    error_found = 1
    Return
  End
  new_zfs_name = zfo_name
  new_zfo_name = zfo_name_sav
  zfo_rnd = 0
  "ALTER '"||zfo_name||"' NEWNAME('"||new_zfo_name||"')"
  If rc=0 Then Do
    zfo_name = new_zfo_name
    zfo_rnd = 1
    Say "ZFSRO103I The old zFS aggregate has been renamed to",
      zfo_name||"."
    zfo_lds_fnd = 0 /* lds data part name found   */
    zfo_lds_rnd = 0 /* lds data part name renamed */
    ldsdata. = ""
    Call Outtrap "LDSDATA."
    "LISTCAT ENTRIES('"||zfo_name||"')"; retc = rc
    Call Outtrap "OFF"
    If retc=0 Then Do ldsi=1 To ldsdata.0
      Parse Var ldsdata.ldsi "DATA" . zfo_lds_data
      If zfo_lds_data<>"" Then Do
        zfo_lds_fnd = 1
        Leave ldsi
      End
    End
    If zfo_lds_fnd Then Do
      neo_lds_data = Strip(Strip(Left(zfo_name,39)),"T",".")||,
        ".DATA"
      If zfo_lds_data<>neo_lds_data Then Do
        "ALTER '"||zfo_lds_data||"' NEWNAME('"||neo_lds_data||"')"
        If rc=0 Then zfo_lds_rnd = 1
      End
      Else zfo_lds_rnd = 1
    End
    If zfo_lds_rnd Then Say "ZFSRO131I The old zFS DATA part has",
      "been renamed to or is named" neo_lds_data||"."
    Else Say "ZFSRO132W The old zFS DATA part could not be renamed,",
      "but the old zFS aggregate is still consistent."
  End
  If zfo_rnd Then Do
    "ALTER '"||zfs_name||"' NEWNAME('"||new_zfs_name||"')"
    If rc=0 Then Do
      zfs_name = new_zfs_name
      Say "ZFSRO104I The zFS aggregate has been renamed to",
        zfs_name||"."
      zfs_lds_fnd = 0 /* lds data part name found   */
      zfs_lds_rnd = 0 /* lds data part name renamed */
      ldsdata. = ""
      Call Outtrap "LDSDATA."
      "LISTCAT ENTRIES('"||zfs_name||"')"; retc = rc
```

```
      Call Outtrap "OFF"
      If retc=0 Then Do ldsi=1 To ldsdata.0
        Parse Var ldsdata.ldsi "DATA" . zfs_lds_data
        If zfs_lds_data<>"" Then Do
          zfs_lds_fnd = 1
          Leave ldsi
        End
      End
      If zfs_lds_fnd Then Do
        new_lds_data = Strip(Strip(Left(zfs_name,39)),"T",".")||,
          ".DATA"
        If zfs_lds_data<>new_lds_data Then Do
          "ALTER '"||zfs_lds_data||"' NEWNAME('"||new_lds_data||"')"
          If rc=0 Then zfs_lds_rnd = 1
        End
        Else zfs_lds_rnd = 1
      End
      If zfs_lds_rnd Then Say "ZFSRO119I The zFS DATA part has been",
        "renamed to or is named" new_lds_data||"."
      Else Say "ZFSRO077W The zFS DATA part could not be renamed,",
        "but the zFS aggregate is usable correctly."
    End
    Else Do
      Say "ZFSRO078E The zFS aggregate name could not be renamed,",
        "but it is usable correctly."
      Say "ZFSRO120I Therefore, mounting the zFS aggregate with its",
        "temporary name" zfs_name
      error_found = 1
    End
  End
  Else Do
    Say "ZFSRO079E The old zFS aggregate could not be renamed, but",
      "the new zFS aggregate is usable correctly."
    Say "ZFSRO121I Therefore, mounting the zFS aggregate with its",
      "temporary name" zfs_name
    error_found = 1
  End
End

If zfs_replaces_zfo="Y" & zfo_mounted Then Do
  If rmnt_index=0 Then Do
    m. = ""
    m.mnte_filetag = zfo.mnte_filetag.1
    m.mnte_path = zfo.mnte_path.1
    m.mnte_parm = zfo.mnte_parm.1
    m.mnte_fstype = zfo.mnte_fstype.1
    If zfs_usable Then Do
      Say "ZFSRO107I Mounting zFS" zfs_name "at" m.mnte_path "now..."
      m.mnte_fsname = zfs_name
    End
    Else Do
      Say "ZFSRO108I Mounting old zFS" zfo_name "at" m.mnte_path,
        "again..."
      m.mnte_fsname = zfo_name
    End
```

```
      m.mnte_mode = zfo.mnte_mode.1
      m.mnte_syslist = zfo.mnte_syslist.1
      If zfo.mnte_sysname.1<>local_system Then
        m.mnte_sysname = zfo.mnte_sysname.1
      Call Syscall_Cmd "mount m.", noexit_on_error
      If not_OK Then Do
        Say "ZFSRO085E File System" m.mnte_fsname "could not be",
          "mounted at" m.mnte_path
        error_found = 1
      End
      Else Do
        If zfs_usable Then zfs_mounted = 1
      End
      rmnt_index=1
    End
    Do j=rmnt_index To zfo_fss
      mnt_index = zfo_fs.j
      fsname = Strip(mnt.mnte_fsname.mnt_index)
      fspath = mnt.mnte_path.mnt_index
      Say "ZFSRO109I Mounting file system" fsname "at" fspath "again"
      m. = ""
      m.mnte_filetag = mnt.mnte_filetag.mnt_index
      m.mnte_fsname = Strip(mnt.mnte_fsname.mnt_index)
      m.mnte_fstype = mnt.mnte_fstype.mnt_index
      m.mnte_mode = mnt.mnte_mode.mnt_index
      m.mnte_parm = mnt.mnte_parm.mnt_index
      m.mnte_path = mnt.mnte_path.mnt_index
      m.mnte_syslist = mnt.mnte_syslist.mnt_index
      If mnt.mnte_sysname.mnt_index<>local_system Then
        m.mnte_sysname = mnt.mnte_sysname.mnt_index
      Call Syscall_Cmd "mount m.", noexit_on_error
      If not_OK Then Do
        Say "ZFSRO084E File System" m.mnte_fsname "could not be",
          "remounted at" m.mnte_path
        error_found = 1
      End
    End
  End
Return

Final_Exit: Trace O
  Parse Arg final_rc
  Call Migrate_Data_Reset O /* O= no curmig_msg */
  If tmp_dir_defined Then Do
    Call Syscall_Cmd "chdir" tmp_dir, noexit_on_error
    If OK Then Do
      If zfo_temp_mounted Then Do
        Call Syscall_Cmd "unmount (zfo_name)" mtm_normal,,
          noexit_on_error
        If OK Then zfo_temp_mounted = O
        Call Test4_unmounted zfo_name
      End
      If zfs_temp_mounted Then Do
        Call Syscall_Cmd "unmount (zfs_name)" mtm_normal,,
          noexit_on_error
```

```
        If OK Then zfs_temp_mounted = 0
        Call Test4_unmounted zfs_name
      End
      Address SYSCALL  "rmdir tmp_zfo"
      Address SYSCALL  "rmdir tmp_zfs"
      Address SYSCALL  "unlink ioezadm"
    End
    Call Syscall_Cmd "chdir /", noexit_on_error
    Call Syscall_Cmd "rmdir" tmp_dir, noexit_on_error
  End
  If foreground Then Do
    If Symbol("cur_cwd")="VAR" Then
      Call Syscall_Cmd "chdir" cur_cwd, noexit_on_error
  End
  If sig_enabled Then
    Call Syscall_Cmd "sigaction" sigalrm ohdl oflg "x y",,
      noexit_on_error
  Call Syscalls "SIGOFF"
  Do fei=0 To 2
    If fd.i<>-1 Then Call Syscall_Cmd "close (fd.fei)", 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
  Say
Exit final_rc

Syntax_Error:
  Say "ZFSR0097W REXX error in sourceline" sigl "of" myname
  Say "ZFSR0098I Line" sigl||":" Strip(Sourceline(sigl))
  Say "ZFSR0099E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  retc = rc
  Say "ZFSR0100W REXX error in sourceline" sigl "of" myname
  Say "ZFSR0101I Line" sigl||":" Strip(Sourceline(sigl))
  Say "ZFSR0102E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Build_List_Of_Subfss: Procedure Expose mnt. zfo_fss zfo_fs. subfss.,
    mnte_dev
  Parse Arg devno
  subfss = Strip(subfss.devno)
  Do While subfss<>""
    Parse Var subfss mnt_index subfss
    zfo_fss = zfo_fss+1
    zfo_fs.zfo_fss = mnt_index
    Call Build_List_Of_Subfss mnt.mnte_dev.mnt_index
  End
Return
```

```
Shell_Cmd:
  shell_cmd = ""
  Do sci=1 To parm.0
    shell_cmd = shell_cmd parm.sci
  End
  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 "ZFSR0055I Signal SIGTERM received, terminating..."
        Say "ZFSR0056I 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 sci=1 To 2
    If fd.sci<>-1 Then Call Syscall_Cmd "close (fd.sci)"
```

```
    fd.sci = -1
  End
  Call Get_Output_Data
  Call Syscall_Cmd "close (pp.1)"
  pp.1 = -1
  If time_out Then Do
    Say "ZFSRO057I Signal SIGALRM received or timeout occurred,",
      "terminating..."
    Say "ZFSRO058I 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
  End
Return

Get_Output_Data: Trace O
  Call Syscall_Cmd "fstat (pp.1) st."
  output_size = st.st_size
  If Verify(output_size,"1234567890")<>0 Then Do
    Say "ZFSRO059I Too many lines have been created, aborting..."
    Say "ZFSRO060I Command:" shell_cmd
    Call Final_Exit 8
  End

  If output_size<>0 Then Do
    Call Syscall_Cmd "read (pp.1) output_data (output_size)"
    all_output = all_output||output_data
    Do While Length(all_output)>0
      Parse Var all_output output_line (esc_n) all_output
      Say output_line
    End
  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
  Select
    When rc=0 & retval=-1 & errno=79 & errnojr="55B005C" &,
      syscall_cmd="mount m." Then Do /* Problem with mount point */
      not_OK = 1
      Address SYSCALL "statfs (m.mnte_fsname) st."
      If retval>=0 Then Do
        ffsid = st.stfs_fsid
        Address SYSCALL "statvfs (m.mnte_path) st."
        If retval>=0 Then Do
          If ffsid=st.stfs_fsid Then Do
            not_OK = 0
            retval = 0
          End
        End
```

```
      End
      If not_OK Then Do
        rc = 0; retval = -1; errno = 79; errnojr = "55B005C"
      End
    End
    When rc=0 & retval=-1 & errno=70 & errnojr="59D0135" Then Do
      not_OK = 0 /* Pipe is currently empty */
      retval = 0
    End
    Otherwise
      not_OK = (rc<>0 | retval<0 | retval=0 & (errno<>0 | errnojr<>0))
  End /* Select */
  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)
      is_zFS = Left(Right(errnojr,8,"0"),2)="EF"
      If Symbol("nfnd_text")="LIT" Then nfnd_text =,
        "Notice: unknown modid, reason text may be incorrect"
      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<>"" & err.se_reason<>nfnd_text
          Then Say "OMVS Reason Code Explanation -" err.se_reason
      End
      errno = errno_save
      errnojr = errnojr_save
      If Symbol("shell_cmd")="VAR" & Word(syscall_cmd,1)="spawnp" Then
      Say "Shell Command:" Strip(shell_cmd)
      If exit_on_error Then Do
        Say "ZFSR0112E Severe error occurred, stopping..."
        Call Final_Exit 8
      End
    End
  End
Return

Test4_unmounted:
  Parse Arg uss_fsn
```

```
  Do kk=1 By 1
    Call Bpxwdyn "ALLOC DSN("||uss_fsn||") OLD"
    If result=0 Then Do
      Call Bpxwdyn "FREE Dsn("||uss_fsn||")"
      Leave kk
    End
    Else Do
      If kk=30 Then Do
        /* No way to get it unmounted; Leave handling to further code */
        Leave kk
      End
      Call Syscall_Cmd "sleep 1"
    End
  End
Return

Sigsetempty: Return Copies(0,64)
Sigaddset: Return Overlay(1,Arg(1),Arg(2))
```

## 2.2 CPYRHELP

The CPYRHELP REXX procedure displays help information for reorganization and copy processing.

Example 2-2 displays the contents of COPYRHELP.

*Example 2-2 CPYRHELP REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: CPYRHELP (Next free MSGNO: Say "RGRHPOO9x ...")      */
/*  Description: Help information for migration processing            */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: cpyrhelp                                             */
/*********************************************************************/

tmp_unit = ""             /* UNIT for temporary files: VIO, 3390 ...  */

Trace O
Parse Source . . myname .

Parse Value "" With rgrhpdsn rgrhpddn zerrmsg zerrlm

Signal On Syntax Name Browse_Migr_Help
Signal On Novalue Name Novalue_Error

If tmp_unit="" Then tmp_unit = ""
Else tmp_unit = "" "UNIT("||Strip(tmp_unit)||")"

Call Compute_Help_Lines
Exit 9999

Browse_Migr_Help:
  Signal On Syntax Name Syntax_Error
  Parse Value Sourceline(sigl) With . . skip_top skip_end .
  help_strt = sigl + skip_top
  help_stop = sourceline() - skip_end
  msg.0 = 0
  Call Bpxwdyn "ALLOC RTDSN(RGRHPDSN) RTDDN(RGRHPDDN) NEW",
    "MSG(MSG.) LRECL(80) RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
  If result<>0 | rgrhpddn="" Then Do
    If msg.0=0 Then
      Say "Disp_Msg RGRHP001E Error allocating temporary file"
    Else Do i=1 to msg.0
      Say Strip(msg.i,"T")
    End
    Say "Rc("||result||")"
    Exit 8
  End
  Do i=help_strt To help_stop
    j=i+1-help_strt
    hline.j = Sourceline(i)
  End
  "EXECIO" (help_stop+1-help_strt) "DISKW" rgrhpddn "(STEM HLINE. FINIS"
```

```
  If rc<>0 Then Do
    Call Disp_Msg "RGRHP002E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  Address ISPEXEC "LMINIT DATAID(RGRHPDID) DDNAME("||rgrhpddn||")"
  If rc<>0 Then Do
    Call Disp_Msg Strip(zerrmsg) Strip(zerrlm) "ISPF gave rc" rc "on",
      "generating a data ID for RGRHPDDN."
    Call Final_Exit 8
  End

  Address ISPEXEC "BROWSE DATAID("||rgrhpdid||")"
  If rc<>0 Then Do
    Call Disp_Msg Strip(zerrmsg) Strip(zerrlm) "ISPF gave rc" rc "on",
      "browsing data ID for RGRHPDDN."
    Call Final_Exit 8
  End

  Call Final_Exit 0
Exit 9999

Final_Exit:
  Parse Arg final_rc
  If Symbol("RGRHPDID")="VAR" Then
    Address ISPEXEC "LMDFREE LISTID("||rgrhpdid||")"
  If rgrhpddn<>"" Then Call Bpxwdyn "FREE DD("||rgrhpddn||")"
Exit final_rc

Syntax_Error:
  Say "RGRHP003W REXX error in sourceline" sigl "of" myname
  Say "RGRHP004I Line" sigl||":" Strip(Sourceline(sigl))
  Say "RGRHP005E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  retc = rc
  Say "RGRHP006W REXX error in sourceline" sigl "of" myname
  Say "RGRHP007I Line" sigl||":" Strip(Sourceline(sigl))
  Say "RGRHP008E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Compute_Help_Lines:
  Call Browse_MigrHelp 3 1 /* skip next 2 lines and 1 at the end */

/* ---------------------------------------------------------------- *

# ================================================================ #
#   REORGDATA Help Information                                     #
# ================================================================ #

1. REORG Processing JCL Information
-----------------------------------
```

```
a) Job Variables

SET CPYTOOL=          - This is the copy utility to be used; using
                        COPYPAX is the suggested method for a REORG.
TSOREPRO:             > Use IDCAMS/TSO REPRO function for copying
COPYPAX :             > Use accessable (std) pax version for copying
COPYTREE:             > Use accessable (std) copytree for copying

SET VERBOSE=          - N or Y, list all objects copied; VERBOSE is used
                        only if COPYPAX is set.

SET DEFREORG=         - Member or data set containing the REORG
                        definition statements


b) Environment Variables

STOP_AFTER_SYNTAX_CHECK:

  Force stopping after formal syntax check of STDIN data is done; value
  must be specified as N or Y.


STOP_AFTER_FSS_MOUNTED:

  Force stopping when old and if existing new zFS is/are mounted; value
  must be specified as N or Y.


STOP_AFTER_ZFS_IS_FORMATTED:

  Force stopping when the new zFS aggregate is formatted; value must be
  specified as N or Y.


TARGET_ZFS_MUST_BE_EMPTY:

  Run copy processing only if the target zFS structure is empty; value
  may be set to Y or N.


DENIED_UMNT_FSTYPES:

  Specify filesystypes blocking replacement of mounted old zFS by the
  new zFS; TFS is always blocking the automatic replacement as the
  complete contents of the TFS is lost otherwise.


PATH:

  Additional PATH setting; "/bin" and "/samples" are included
  automatically; example: PATH=/usr/local/bin
```

```
2. REORG Definition Statements
------------------------------

Note: The abbreviation "FYR:" used below means a value that is provided
      just "For Your Reference".

ZFS_OLD_NAME                    - Name of the old zFS data set to be migrated

#ZFS_OLD_#_VOLUMES=             - FYR: Number of old zFS volumes

#ZFS_OLD_DEVICE_TYPE=           - FYR: old zFS DASD device type

#ZFS_OLD_ALLOC_UNIT=            - FYR: old zFS DASD allocation unit

#ZFS_OLD_ALLOC_SPACE=           - FYR: old zFS primary and secondary allocation

#ZFS_OLD_TOTAL_UNITS_ALLOCATED= - FYR: old zFS total number of units allocated

#ZFS_OLD_TOTAL_UNITS_FORMATTED= - FYR: old zFS total zFS formatted units

#ZFS_OLD_TOTAL_UNITS_%USED=     - FYR: old zFS percentage used of space

#ZFS_OLD_DATACLASS=             - FYR: old zFS data class

#ZFS_OLD_MGMNTCLASS=            - FYR: old zFS management class

#ZFS_OLD_STORCLASS=             - FYR: old zFS storage class

ZFS_OLD_NAME_SAV=               - Suggested name for old zFS to be renamed;
                                  this is only used if old and new zFS have
                                  set the same name

ZFS_NAME_TMP=                   - Suggested name for zfs used initially;
                                  this is only used if old and new zFS have
                                  set the same name

ZFS_NAME=                       - zFS aggregate name

#ZFS_#_VOLUMES=                 - Number of zFS volumes you should have at
                                  least (just a comment); sum of zFS
                                  volumes plus candidate volumes plus
                                  volumes needed for secondary allocations
                                  should result in this value

ZFS_VOLUMES=                    - Explicit list of volumes to be used;
                                  this specification is ignored if a SMS
                                  storage class is assigned

ZFS_ALLOC_NUM_CAND_VOLUMES=     - Number of zFS candidate volumes

ZFS_ALLOC_UNIT=                 - zFS allocation unit; you may specify
                                  CYLINDERS (CYL) or TRACKS(TRK) only

ZFS_ALLOC_SPACE=                - zFS primary and secondary allocation
```

```
ZFS_ALLOC_NUM_SEC_ALLOCS=       - Number of secondary allocations added
                                  before starting migration processing

ZFS_DATACLASS=                  - zFS SMS data class to be used; a BLANK
                                  values means that no data class is set

ZFS_MGMNTCLASS=                 - zFS SMS mgmnt class to be used; a BLANK
                                  values means that no mgmnt class is set

ZFS_STORCLASS=                  - zFS SMS storage class to be used; a BLANK
                                  values means that no storage class is set

ZFS_REPLACES_ZFS=               - This value specifies whether the new zFS
                                  file system should replace the old zFS; you
                                  must specify Y or N.

Important: If you specify not to replace the old by the new zFS or this
           cannot be done for other reasons the old zFS file system is kept
           mounted read-only to assure that the contents will not differ
           again from the new zFS.

 * ------------------------------------------------------------------ */
```

## 2.3 DEFRHELP

The DEFRHELP REXX procedure displays help information for creating reorganization definition statements.

Example 2-3 displays the contents of DEFRHELP.

*Example 2-3 DEFRHELP REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DEFRHELP (Next free MSGNO: Say "RGRHP009x ...")      */
/*  Description: Help information for migration processing            */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: defrhelp                                             */
/*********************************************************************/

tmp_unit = ""             /* UNIT for temporary files: VIO, 3390 ...  */

Trace O
Parse Source . . myname .

Parse Value "" With rgrhpdsn rgrhpddn zerrmsg zerrlm

Signal On Syntax Name Browse_Migr_Help
Signal On Novalue Name Novalue_Error

If tmp_unit="" Then tmp_unit = ""
Else tmp_unit = "" "UNIT("||Strip(tmp_unit)||")"

Call Compute_Help_Lines
Exit 9999

Browse_Migr_Help:
  Signal On Syntax Name Syntax_Error
  Parse Value Sourceline(sigl) With . . skip_top skip_end .
  help_strt = sigl + skip_top
  help_stop = sourceline() - skip_end
  msg.0 = 0
  Call Bpxwdyn "ALLOC RTDSN(RGRHPDSN) RTDDN(RGRHPDDN) NEW",
    "MSG(MSG.) LRECL(80) RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
  If result<>0 | rgrhpddn="" Then Do
    If msg.0=0 Then
      Say "Disp_Msg RGRHP001E Error allocating temporary file"
    Else Do i=1 to msg.0
      Say Strip(msg.i,"T")
    End
    Say "Rc("||result||")"
    Exit 8
  End
  Do i=help_strt To help_stop
    j=i+1-help_strt
    hline.j = Sourceline(i)
  End
  "EXECIO" (help_stop+1-help_strt) "DISKW" rgrhpddn "(STEM HLINE. FINIS"
```

```
  If rc<>0 Then Do
    Call Disp_Msg "RGRHP002E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  Address ISPEXEC "LMINIT DATAID(RGRHPDID) DDNAME("||rgrhpddn||")"
  If rc<>0 Then Do
    Call Disp_Msg Strip(zerrmsg) Strip(zerrlm) "ISPF gave rc" rc "on",
      "generating a data ID for RGRHPDDN."
    Call Final_Exit 8
  End

  Address ISPEXEC "BROWSE DATAID("||rgrhpdid||")"
  If rc<>0 Then Do
    Call Disp_Msg Strip(zerrmsg) Strip(zerrlm) "ISPF gave rc" rc "on",
      "browsing data ID for RGRHPDDN."
    Call Final_Exit 8
  End

  Call Final_Exit 0
Exit 9999

Final_Exit:
  Parse Arg final_rc
  If Symbol("RGRHPDID")="VAR" Then
    Address ISPEXEC "LMDFREE LISTID("||rgrhpdid||")"
  If rgrhpddn<>"" Then Call Bpxwdyn "FREE DD("||rgrhpddn||")"
Exit final_rc

Syntax_Error:
  Say "RGRHP003W REXX error in sourceline" sigl "of" myname
  Say "RGRHP004I Line" sigl||":" Strip(Sourceline(sigl))
  Say "RGRHP005E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  retc = rc
  Say "RGRHP006W REXX error in sourceline" sigl "of" myname
  Say "RGRHP007I Line" sigl||":" Strip(Sourceline(sigl))
  Say "RGRHP008E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Compute_Help_Lines:
  Call Browse_MigrHelp 3 1 /* skip next 2 lines and 1 at the end */

/* ---------------------------------------------------------------- *

# ================================================================ #
#   DEFREORG Help Information                                      #
# ================================================================ #


1. DEFREORG Control Statements
```

```
------------------------------

ZFS_REORG_DEFINE_DSN=

This is the full name of a sequential MVS data set including HLQ or the
name of PDS. Preferred is to use a PDS as this allows simply to use new
members for new migration tasks. This data set must be pre-allocated as
VB80 or FB80. The default name is hlq.ZFS.REORG.DEFINE with "hlq" being
your own userid.


ZFS_REORG_DEFINE_MBR=

Here you can specify the output member name to contain the migration
the migration control statements. If no name is provided "WORKnn" is
used by default with "nn" being a number that is not used currently.
You can rename the member names using ISPF at any time as you like it.


ZFS_REORG_DEFINE_DSP=

This value must be specified as APPEND or REPLACE. Note, only the first
character is examined. With "APPEND" the new control statements are
appended, otherwise the old contents is replaced (if the member exists
already).


ZFS_DEF_DATACLASS=
ZFS_DEF_MANAGEMENTCLASS=
ZFS_DEF_STORAGECLASS=

These three statements allow to specify default SMS classes to be used
for the new zFS aggregates. If you do not want to use a specific class
for a specific zFS aggregate specify "ZFS_DATACLASS=", "ZFS_MGMNTCLASS="
or "ZFS_STORCLASS=" with BLANK as value later on.


ZFS_DATA_SETS_TO_REORG=

This statement defines the zFS data set name list to be reorganized. The
value cannot be BLANK. You may use several lines using this control
word. The values are the same as used on ISPF 3.4 to display a list of
data sets.


ZFS_AGGRNAME_CHANGE_CMD=

This statements allows to specify a file system name change command,
for example: /ZF1/ZF2/ . If you do not specify anything the new zFS
aggregate will get the same name as the old zFS LDS before. The old zFS
will get renamed (with ".SAV" appended at the end by default).


2. DEFREORG Line Commands
-------------------------
```

```
DEFRHELP or DH  - displays this help information
DM              - displays the last set of messages shown again
CPYRHELP or CH  - displays help information for the later reorganize job
REFRESH or RF   - displays the current source statements refreshed

 * ------------------------------------------------------------------ */
```

## 2.4 DEFREORG

The DEFREORG REXX procedure is used in the TSO/ISPF foreground to define reorganization definition statements, and put them to a data set or member of a partioned data set or PDSE. These definition files are used in the second step as input for a reorganization processing job.

Example 2-4 displays the contents of DEFREORG.

*Example 2-4 DEFREORG REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DEFREORG (Next free MSGNO: DEFRO070x)                */
/*  Description: REORG zFS compat mode aggregates                     */
/*               Property of IBM  (C) Copyright IBM Corp. 2011        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: defreorg                                             */
/*********************************************************************/

tmp_unit  = ""         /* UNIT for temp files, e.g. SYSDA, 3390, VIO  */

Trace O
Parse Source . . myname . . . . omvs .
myname = Substr(myname,Lastpos("/",myname)+1)
If omvs="OMVS" Then Do
  Say "DEFRO053E The procedure needs to be run in TSO/ISPF foreground."
  Exit 2
End

If Sysvar("SYSISPF")<>"ACTIVE" Then Do
  Address TSO "ISPSTART CMD(%"||myname||")"
  Exit rc
End

msg_save = Msg()
msg_say = 0
defrcopn = 0
listaccd = 0
no_add2imsg = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
Parse Value "" With defroddn defrmddn defrcddn zerrmsg zerrlm
no_sms_class = "**None**"
get_size_needed = 0
Numeric Digits 10
final_rc = 0
no_msgs = 1
noexit_on_error = 1
switched = 0
zero_parm = D2c(0,4)
zero_01 = "00"x
zero_33 = Copies(zero_01,33)
aid_reserved = zero_33            /* for pfsctl() calls - char(33)   */
zero_st = Copies(zero_01,164)     /* aggr_status initialization part */
zfs_cmd = X2d("40000005")         /* zFS pfsctl() cmd ZFSCALL_AGGR   */
```

```
parm_len = 32          /* for pfsctl() calls - 1 x op_code + 7 x parm */
parm_123456 = Copies(zero_parm,6)  /* pfsctl() parms 1 to 6 not used  */
parm_23456 = Copies(zero_parm,5)   /* pfsctl() parms 2 to 6 not used  */
parm_3456 = Copies(zero_parm,4)    /* pfsctl() parms 3 to 6 not used  */
not_aggr_attached. = 1         /* not_aggr_attached stem initialized  */

Signal On Syntax Name Syntax_Error
Signal On Novalue Name Novalue_Error
Signal On Halt Name Halt_Request

If tmp_unit="" Then tmp_unit = ""
Else tmp_unit = "" "UNIT("||Strip(tmp_unit)||")"

If background Then Do
  Call Disp_Msg,
    "DEFROOO1E The procedure needs to be run in foreground."
  Call Final_Exit 8
End

If Syscalls("ON")>4 Then Do
  Call Disp_Msg,
    "DEFROO54E The SYSCALL environment could not be established."
  Call Final_Exit 8
End

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

msg.0 = 0
Call Bpxwdyn "ALLOC RTDDN(DEFRODDN) NEW MSG(MSG.) LRECL(80)",
  "RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
If result<>0 | defroddn="" Then Do
  If msg.0=0 | msg.0>1 Then Do
    Do i=1 to msg.0
      Say Strip(msg.i)
    End
    Call Disp_Msg "DEFROOO2E Error allocating temporary file,",
      "Rc("||result||")"
  End
  Else Call Disp_Msg Strip(msg.1)||", Rc("||result||")"
  Call Final_Exit 8
End

msg.0 = 0
Call Bpxwdyn "ALLOC RTDDN(DEFRMDDN) NEW MSG(MSG.) LRECL(84)",
  "RECFM(V,B) SPACE(1,1) TRACKS"||tmp_unit
If result<>0 | defrmddn="" Then Do
  If msg.0=0 | msg.0>1 Then Do
    Do i=1 to msg.0
```

```
      Say Strip(msg.i)
    End
    Call Disp_Msg "DEFRO003E Error allocating temporary message file,",
      "Rc("||result||")"
  End
  Else Call Disp_Msg Strip(msg.1)||", Rc("||result||")"
  Call Final_Exit 8
End

mline00 = 4
mline.2 = "Messages created during previous processing"
mline.1 = Copies("=",Length(mline.2))
mline.3 = mline.1
mline.4 = ""
Address ISPEXEC "VGET (ZFSM$DSN ZFSM$MBR ZFSM$DSP ZFSM$DCL ZFSM$MCL",
  "ZFSM$SCL ZFSM$HDM ZFSM$CCM)"
If rc>8 Then Do
  Call WMsg "DEFRO004W ISPF Service VGET gave rc" rc||".", no_add2imsg
  Parse Value "" With zfsm$dsn zfsm$mbr zfsm$dsp zfsm$dcl zfsm$mcl,
    zfsm$scl zfsm$hdm zfsm$ccm
End
If zfsm$dsn="" Then zfsm$dsn = Userid()||".ZFS.REORG.DEFINE"
If zfsm$dsp="" Then zfsm$dsp = "APPEND"
cmntline = Left("#" Copies("----",17) "#",80)
blnkline = Left("",80)
cline.1  = cmntline
cline.2  = Left("ZFS_REORG_DEFINE_DSN="||zfsm$dsn,80)
cline.3  = Left("ZFS_REORG_DEFINE_MBR="||zfsm$mbr,80)
cline.4  = Left("ZFS_REORG_DEFINE_DSP="||zfsm$dsp,80)
cline.5  = cmntline
cline.6  = blnkline
cline.7  = cmntline
cline.8  = Left("ZFS_DEF_DATACLASS="||zfsm$dcl,80)
cline.9  = Left("ZFS_DEF_MANAGEMENTCLASS="||zfsm$mcl,80)
cline.10 = Left("ZFS_DEF_STORAGECLASS="||zfsm$scl,80)
cline.11 = cmntline
cline.12 = blnkline
cline.13 = cmntline
hdm_words = Max(Words(zfsm$hdm),1)
Do i=1 To Max(Words(zfsm$hdm),1)
  clines = 13+i
  cline.clines = Left("ZFS_DATA_SETS_TO_REORG="||Word(zfsm$hdm,i),80)
End
clines = 13+hdm_words+1
cline.clines = Left("ZFS_AGGRNAME_CHANGE_CMD="||zfsm$ccm,80)
clines = clines+1
cline.clines = cmntline
"EXECIO" clines "DISKW" defroddn "(STEM CLINE. FINIS"
If rc<>0 Then Do
  Call Disp_Msg "DEFRO005E EXECIO error occurred, Rc("||rc||")"
  Call Final_Exit 8
End

Drop zfsm$dsn zfsm$mbr zfsm$dsp zfsm$dcl zfsm$mcl zfsm$scl,
  zfsm$hdm zfsm$ccm
```

```
Address ISPEXEC "LMINIT DATAID(DEFRODID) DDNAME("||defroddn||")"
If rc<>0 Then Do
  If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
  Else message = ""
  Call Disp_Msg "DEFROOO6E ISPF service LMINIT gave RC" rc "on",
    "generating a data ID DEFRODID."||message
  Call Final_Exit 8
End

Call Disp_Msg "Enter DH to show REORG definition information.",
  "Change the data as needed. To continue SAVE the changes,to stop",
  "processing use CANCEL."

Do Forever

  no_error_found = 1
  Address ISPEXEC "EDIT DATAID("||defrodid||") MACRO(DZRP$MAC)"
  Select
    When rc=0 Then Nop
    When rc=4 Then Do
      Call Disp_Msg "DEFROOO7W Define processing canceled..."
      Call Final_Exit 4
    End
    Otherwise Do
      Call Disp_Msg "DEFROOO8E Error on editing occurred,",
        "Rc("||rc||")"
      Call Final_exit 8
    End
  End

  "EXECIO * DISKR" defroddn "(STEM CLINE. FINIS"
  If rc<>0 Then Do
    Call Disp_Msg "DEFROOO9E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  Parse Value "" With defreorg_dsn defreorg_mbr defreorg_dsp,
    zfs_def_dataclass zfs_def_mgmntclass zfs_def_storclass,
    zfs_dsns_to_reorg zfs_chg_str zfs_chg_str zfs_dsns_chcmd ispfmsg
    mlines = mline00
  Do i=1 To cline.0
    cntl_line = Translate(Strip(Left(cline.i,72)))
    If Left(cntl_line,1)="#" | cntl_line="" Then Iterate i
    Parse Var cntl_line cntl_parm "=" cntl_value
    Select
      When cntl_parm="ZFS_REORG_DEFINE_DSN" Then
        Parse Var cntl_value defreorg_dsn .
      When cntl_parm="ZFS_REORG_DEFINE_MBR" Then
        Parse Var cntl_value defreorg_mbr .
      When cntl_parm="ZFS_REORG_DEFINE_DSP" Then
        Parse Var cntl_value defreorg_dsp .
      When cntl_parm="ZFS_DEF_DATACLASS" Then
        Parse Var cntl_value zfs_def_dataclass .
      When cntl_parm="ZFS_DEF_MANAGEMENTCLASS" Then
        Parse Var cntl_value zfs_def_mgmntclass .
```

```
    When cntl_parm="ZFS_DEF_STORAGECLASS" Then
      Parse Var cntl_value zfs_def_storclass .
    When cntl_parm="ZFS_DATA_SETS_TO_REORG" Then
      zfs_dsns_to_reorg = Strip(zfs_dsns_to_reorg cntl_value)
    When cntl_parm="ZFS_AGGRNAME_CHANGE_CMD" Then
      Parse Var cntl_value zfs_dsns_chcmd .
    Otherwise Do
      Call WMsg "DEFRO010E Invalid specification found in line" i||"."
      If Length(cntl_line)<70 Then message = Left(">",9) cntl_line
      Else message = ">" Right(cntl_line,77)
      Call WMsg message
      no_error_found = 0
    End
  End
End
defreorg_dsn = Strip(defreorg_dsn,"B","'")
If defreorg_dsn="" Then rc=1
Else
  Address ISPEXEC "ISPEXEC DSINFO DATASET('"||defreorg_dsn||"')"
Select
  When rc=0 Then Do
    Select
      When Wordpos(zdsorg,"PS PO")=0 Then Do
        Call WMsg "DEFRO011E REORG control data set must be a PS",
          "or PO data set."
        no_error_found = 0
      End
      When zdsdsnt="HFS" Then Do
        Call WMsg "DEFRO012E REORG control data set cannot be an",
          "HFS data set."
        no_error_found = 0
      End
      When Wordpos(zdsrf,"FB VB")=0 | zdslrec<>"80" Then Do
        Call WMsg "DEFRO013E REORG control data set must be FB80",
          "or VB80."
        no_error_found = 0
      End
      When zdsorg="PS" & defreorg_mbr<>"" Then Do
        Call WMsg "DEFRO014E You cannot specify a member name for a",
          "PS data set."
        no_error_found = 0
      End
      When zdsorg="PO" Then Do
        member_ok = 1
        If defreorg_mbr="" Then Do
          mbr.0 = 0
          Call OUTTRAP "MBR."
          Address TSO "LISTDS ('"||defreorg_dsn||"') MEMBERS"
          retc = rc
          Call OUTTRAP "OFF"
          If retc<>0 Then Do
            message = "DEFRO015E TSO LISTDS ... MEMBERS gave",
              "Rc("||retc||"):"
            Do i=1 To mbr.0
              message = message mbr.i
```

```
            End
            Call Disp_Msg message
            member_ok = 0
            no_error_found = 0
            Call Final_Exit 8
          End
          a_mbr_name = 0
          mname_free. = 1
          Do i=1 To mbr.0
            If a_mbr_name Then Do
              mbr_name = Strip(mbr.i)
              If Left(mbr_name,4)="WORK" Then
                mname_free.mbr_name = 0
            End
            Else If mbr.i="--MEMBERS--" Then a_mbr_name = 1
          End
          Do i=1 To 99
            defreorg_mbr = "WORK"||Right(i,2,"0")
            If mname_free.defreorg_mbr Then Leave i
            If i=99 Then Do
              Call WMsg "DEFRO016E No free number available for",
                "WORKxx member name between 01 and 99."
              defreorg_mbr = ""
              member_ok = 0
              no_error_found = 0
            End
          End
        End
        If member_ok Then Do
          Call Msg "OFF"
          dm_status = Sysdsn("'"||defreorg_dsn||"("||defreorg_mbr||,
            ")'")
          Call Msg msg_save
          Select
            When dm_status="OK" Then mbr_exists = 1
            When dm_status="MEMBER NOT FOUND" Then mbr_exists = 0
            Otherwise Do
              mbr_exists = 0
              Call WMsg "DEFRO017E The member name specified cannot",
                "be used, status is:" dm_status||"."
              no_error_found = 0
            End
          End
        End
      End
      Otherwise Nop
    End /* Select */
  End
  When rc=1 Then Do
    Call WMsg "DEFRO018E Data set name for ZFS_REORG_DEFINE_DSN",
      "cannot be BLANK."
    no_error_found = 0
  End
  When rc=8 Then Do
    Call WMsg "DEFRO019E Data set" defreorg_dsn "could not be found."
```

```
    no_error_found = 0
  End
  Otherwise Do
    Call WMsg "DEFRO020I ISPF service DSINFO gave rc" rc||"."
    no_error_found = 0
  End
End
Select
  When Left(defreorg_dsp,1)="A" Then defreorg_dsp = "APPEND"
  When Left(defreorg_dsp,1)="R" Then defreorg_dsp = "REPLACE"
  Otherwise Do
    Call WMsg "DEFRO021E The value for ZFS_REORG_DEFINE_DSP is",
      "invalid (must be APPEND or REPLACE)."
    no_error_found = 0
  End
End /* Select */
If IsNotValid(zfs_def_dataclass) Then Do
  Call WMsg "DEFRO022E The value for ZFS_DEF_DATACLASS is invalid."
  no_error_found = 0
End
If IsNotValid(zfs_def_mgmntclass) Then Do
  Call WMsg "DEFRO023E The value for ZFS_DEF_MANAGEMENTCLASS is",
    "invalid."
  no_error_found = 0
End
If IsNotValid(zfs_def_storclass) Then Do
  Call WMsg "DEFRO024E The value for ZFS_DEF_STORAGECLASS is invalid."
  no_error_found = 0
End
If zfs_dsns_to_reorg="" Then Do
  Call WMsg "DEFRO025E The value for ZFS_DATA_SETS_TO_REORG cannot",
    "be BLANK."
  no_error_found = 0
End
Do ww = 1 To Words(zfs_dsns_to_reorg)
  If Left(Word(zfs_dsns_to_reorg,ww),1)="*" Then Do
    Call WMsg "DEFRO051E A word within ZFS_DATA_SETS_TO_REORG",
      "cannot start with '*'."
    no_error_found = 0
  End
  If Left(Word(zfs_dsns_to_reorg,ww),1)="%" Then Do
    Call WMsg "DEFRO049E A word within ZFS_DATA_SETS_TO_REORG",
      "cannot start with '%'."
    no_error_found = 0
  End
End ww
Parse Var zfs_dsns_chcmd "/" zfo_chg_str "/" zfs_chg_str "/",
  zfs_dsns_junk
Select
  When zfs_dsns_chcmd="" Then Nop
  When Left(zfs_dsns_chcmd,1)<>"/" Then Do
    Call WMsg "The first character in ZFS_AGGRNAME_CHANGE_CMD",
      "must be a '/'."
    no_error_found = 0
  End
```

```
  When Right(zfs_dsns_chcmd,1)<>"/" Then Do
    Call WMsg "The last character in ZFS_AGGRNAME_CHANGE_CMD",
      "must be a '/'."
    no_error_found = 0
  End
  When Pos("/",Substr(zfs_dsns_chcmd,2))=Length(zfs_dsns_chcmd)-1
    Then Do
    Call WMsg "The value for ZFS_AGGRNAME_CHANGE_CMD must contain",
      "another '/' character somewhere in middle."
    no_error_found = 0
  End
  When zfs_dsns_junk<>"" Then Do
    Call WMsg "The value for ZFS_AGGRNAME_CHANGE_CMD must contain",
      "exactly three '/' characters."
    no_error_found = 0
  End
  When zfo_chg_str="" Then Do
    Call WMsg "DEFRO026E The value for the ZFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD cannot be BLANK."
    no_error_found = 0
  End
  When IsInvalidStr(zfs_chg_str) Then Do
    Call WMsg "DEFRO027E The value for the zFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD contains invalid characters."
    no_error_found = 0
  End
  When IsInvalidStr(zfs_chg_str) Then Do
    Call WMsg "DEFRO028E The value for the zFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD contains invalid characters."
    no_error_found = 0
  End
  Otherwise Nop
End
zfsm$dsn = defreorg_dsn
zfsm$mbr = defreorg_mbr
zfsm$dsp = defreorg_dsp
zfsm$dcl = zfs_def_dataclass
zfsm$mcl = zfs_def_mgmntclass
zfsm$scl = zfs_def_storclass
zfsm$hdm = zfs_dsns_to_reorg
If zfs_dsns_chcmd="" Then zfsm$ccm = ""
Else zfsm$ccm = "/"||zfs_chg_str||"/"||zfs_chg_str||"/"
Address ISPEXEC "VPUT (ZFSM$DSN ZFSM$MBR ZFSM$DSP ZFSM$DCL ZFSM$MCL",
  "ZFSM$SCL ZFSM$HDM ZFSM$CCM)"
If rc<>0 Then
  Call WMsg "DEFRO029W ISPF Service VPUT gave rc" rc||"."
Drop zfsm$dsn zfsm$mbr zfsm$dsp zfsm$dcl zfsm$mcl zfsm$scl,
  zfsm$hdm zfsm$ccm
If mlines>mline00 Then Do
  Call Write_MFile
  message = "Enter DM to display most recent messages collected. "
End
Else message = ""
If no_error_found Then Leave
message = message ||,
```

```
    "Make changes and SAVE or use CANCEL to stop processing."
  Call Disp_Msg Strip(ispfmsg message)

End /* Do Forever */

mlines = mline00
If defreorg_mbr="" Then Do
  defreorg_ctl = defreorg_dsn
  If defreorg_dsp="APPEND" Then acc_mode = "MOD"
  Else acc_mode = "OLD"
End
Else Do
  defreorg_ctl = defreorg_dsn||"("||defreorg_mbr||")"
  acc_mode = "SHR"
End
msg.0 = 0
Call Bpxwdyn "ALLOC DSN("||defreorg_ctl||") RTDDN(DEFRCDDN)" acc_mode,
  "MSG(MSG.)"
If result<>0 | defrcddn="" Then Do
  dyn_rc = result
  If msg.0=0 | msg.0>1 Then Do
    Do i=1 to msg.0
      Call WMsg Strip(msg.i)
    End
    Call Disp_Msg "DEFRO030E Error allocating DEFREORG control file,",
      "Rc("||dyn_rc||")"
  End
  Else Call Disp_Msg Strip(msg.1)||", Rc("||dyn_rc||")"
  If mlines>mline00 Then Do
    Call Write_MFile
  End
  Call Final_Exit 8
End

If defreorg_dsp="APPEND" & defreorg_mbr<>"" Then Do
  If Sysdsn("'"||defreorg_ctl||"'")="MEMBER NOT FOUND" Then
    reorgcntl.0 = 0
  Else Do
    "EXECIO * DISKR" defrcddn "(STEM REORGCNTL. FINIS"
    If rc<>0 Then Do
      Call Disp_Msg "DEFRO031E EXECIO error occurred, Rc("||rc||")"
      Call Final_Exit 8
    End
  End
End
Else reorgcntl.0 = 0
clines = reorgcntl.0
not_dsnref. = 1
dsns = 0
msg_say = 1

cmnt_line = "#" Copies("----",17) "#"
Call WMctl cmnt_line
Parse Value Date("S") With 1 yyyy 5 mm 7 dd
cline = "REORG CONTROL DEFINITIONS, CREATED" yyyy||"-"||mm||"-"||dd,
```

```
  Time()
Call WMctl Overlay(cline,"#" Left("",68) "#",3)
Call WMctl cmnt_line
Do i=1 To Words(zfs_dsns_to_reorg)
  dsnp = Strip(Word(zfs_dsns_to_reorg,i),"B","'")
  Call WMsg "DEFRO050I Searching for data sets" dsnp "..."
  Address ISPEXEC "LMDINIT LISTID(DEFRDDID) LEVEL("||dsnp||")"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFRO032E ISPF service LMDINIT gave RC" rc "on",
      "generating a data ID DEFRDDID."||message
    Call Final_Exit 8
  End

  dsn = ""
  Do Forever
    Address ISPEXEC "LMDLIST LISTID("||defrddid||") DATASET(DSN)",
      "STATS(YES) OPTION(LIST)"
    Select
      When rc=0 Then Do
        listaccd = 1
        If zdlmigr="YES" Then Call WMsg,
          "DEFRO033W Migrated data set" dsn "has been skipped."
        Else Do
          If not_dsnref.dsn Then Do
            not_dsnref.dsn = 0
            dsns = dsns+1
            dsn.dsns = dsn
            Call WMsg "DEFRO048I Data set" dsn "will be examined."
          End
          Else Call WMsg,
            "DEFRO034I Data set" dsn "has been processed already"
        End
      End
      When rc=4 Then Do
        Call WMsg "DEFRO035E No data sets match """||dsnp||"""."
        Leave
      End
      When rc=8 Then Leave /* All data set names processed */
      Otherwise Do
        Call Disp_Msg,
          "DEFRO036E ISPF Service LMDLIST/LIST gave rc" rc||"."
        Call Final_Exit 8
      End
    End /* Select */
  End /* Do Forever */
  If listaccd Then Do
    Address ISPEXEC "LMDLIST LISTID("||defrddid||") OPTION(FREE)"
    If rc<>0 Then Do
      If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
      Else message = ""
      Call Disp_Msg "DEFRO037E ISPF service LMDLIST gave RC" rc "on",
        "freeing storage for data ID DEFRDDID."||message
      Call Final_Exit 8
```

```
    End
    listaccd = 0
  End
  Address ISPEXEC "LMDFREE LISTID("||defrddid||")"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFRO038E ISPF service LMDFREE gave RC" rc "on",
      "freeing data ID DEFRDDID."||message
    Call Final_Exit 8
  End
  Drop defrddid
End i

nn = 32
Call WMctl " "
Call Rxlsaggr /* Retrieve information about all currently active zFSs */
zfss = 0
Do i=1 To dsns
  aggr_name = dsn.i
  /* ------------------------------------------------------------ */
  /* Step 1: Verifying aggr_name is a VSAM cluster                */
  /* ------------------------------------------------------------ */
  lc. = ""
  Call OUTTRAP "LC."
  "LISTCAT ENTRIES('"||aggr_name||"') CLUSTER NAME"
  retc = rc
  Call OUTTRAP "OFF"
  Select
    When retc=0 Then Nop
    When retc=4 & Word(lc.1,1)="IDC1565I" Then Do
      /* IDC1565I aggr_name NOT A REQUESTED TYPE */
      Call WMsg "DEFRO056I Data set" aggr_name "is not a VSAM cluster",
        "and will be skipped."
      Iterate i
    End
    Otherwise Do
      Call WMsg "DEFRO057E Unexpected LISTCAT error occurred, rc=" retc
      Call WMsg "DEFRO058I" lc.1
      Call WMsg "DEFRO059I Data set" aggr_name "will be skipped."
      Iterate i
    End
  End /* Select */

  /* ------------------------------------------------------------ */
  /* Step 2: Retrieving total number 8K blks and free percentage */
  /* ------------------------------------------------------------ */
  If not_aggr_attached.aggr_name Then Do
    Call Rxattach /* aggr_name */
    If not_OK Then Do
      Call WMsg "DEFRO060W Data set" aggr_name "seems to be no valid",
        "zFS and will be skipped."
      Iterate i
    End
  End
```

```
Call Rxaggrinfo /* aggr_name */
If not_aggr_attached.aggr_name Then Call Rxdetach /* aggr_name */
If not_aggrinfo_OK Then Do
  Call WMsg "DEFRO042E Unexpected error occurred on aggrinfo call.",
    "Data set" aggr_name "will be skipped."
  Iterate i
End
zfss = zfss+1
zfo_name = aggr_name             /* zFS data set name          */
zfo_tot8k = blks_8K              /* Formatted 8K blocks        */
zfo_use8k = blks_8K-free_8K      /* Used 8K blocks             */
zfo_percu = Format((zfo_use8k/blks_8k)*100,,0) /* Percentage   */
zfo_trks = zfo_tot8k/6           /* Used tracks                */
If zfo_trks//15=0 Then Do
  zfo_cyls = zfo_trks/15         /* Used cylinders             */
  zfo_fmt_info = zfo_cyls "CYLINDERS"
End
Else Do
  zfo_cyls = (zfo_trks/15)%1+1
  zfo_fmt_info = zfo_trks "TRACKS"
End

/* ---------------------------------------------------------- */
/* Step 3: Retrieving DSINFO data for the cluster part        */
/* ---------------------------------------------------------- */
Address ISPEXEC "ISPEXEC DSINFO DATASET('"||aggr_name||"')"
Select
  When rc=8 Then Do
    Call WMsg "DEFRO039I Data set" aggr_name "could not be found."
    zfss = zfss-1
    Iterate i
  End
  When rc<>0 Then Do
    Call WMsg "DEFRO040E ISPF Service DSINFO gave rc" rc||"."
    Call WMsg "DEFRO041W Data set" aggr_info "will be skipped."
    zfss = zfss-1
    Iterate i
  End
  Otherwise Nop /* OK */
End
zfo_mc = Strip(zdsmc)            /* Management class      */
If zfo_mc=no_sms_class Then zfo_mc = ""
zfo_sc = Strip(zdssc)            /* Storage class         */
If zfo_sc=no_sms_class Then zfo_sc = ""
zfo_dc = Strip(zdsdc)            /* Data class            */
If zfo_dc=no_sms_class Then zfo_dc = ""

/* ---------------------------------------------------------- */
/* Step 4: Retrieving the cluster data part name              */
/* ---------------------------------------------------------- */
ldsdata. = ""
Call Outtrap "LDSDATA."
"LISTCAT ENTRIES('"||zfo_name||"') NAMES"
retc = rc
Call Outtrap "OFF"
```

```
not_zfo_lds_fnd = 1
If retc=0 Then Do ldsi=1 To ldsdata.0
  Parse Var ldsdata.ldsi "DATA" . zfo_lds_data
  If zfo_lds_data<>"" Then Do
    not_zfo_lds_fnd = 0
    Leave ldsi
  End
End
Else Do
  Call WMsg "DEFRO062E Unexpected LISTCAT error occurred, rc=" retc
  Call WMsg "DEFRO063I" lc.1
  Call WMsg "DEFRO064I Data set" aggr_name "will be skipped."
  zfss = zfss-1
  Iterate i
End
If not_zfo_lds_fnd Then Do
  Call WMsg "DEFRO065I No data part for cluster" aggr_name "could be",
    "found; the data set will be skipped."
  zfss = zfss-1
  Iterate i
End

/* ---------------------------------------------------------- */
/* Step 5: Retrieving DSINFO data for the data part           */
/* ---------------------------------------------------------- */
Address ISPEXEC "ISPEXEC DSINFO DATASET('"||zfo_lds_data||"')"
Select
  When rc=8 Then Do
    Call WMsg "DEFRO066I Data part" zfo_lds_data "could not be found."
    Call WMsg "DEFRO067W Data set" aggr_info "will be skipped."
    zfss = zfss-1
    Iterate i
  End
  When rc<>0 Then Do
    Call WMsg "DEFRO068E ISPF Service DSINFO gave rc" rc||"."
    Call WMsg "DEFRO069W Data set" aggr_info "will be skipped."
    zfss = zfss-1
    Iterate i
  End
  Otherwise Nop /* OK */
End
zfo_firstvol = zdsvol            /* First volume serial   */
zfo_#volumes = Strip(zds#vols)   /* Number of volumes     */
zfo_devt = Strip(zdsdevt)        /* Device type           */
zfo_primu = zdsspc               /* Primary space units   */
zfo_prima = C2n(zds1ex)          /* Primary space alloc   */
zfo_seca = C2n(zds2ex)           /* Secondary space alloc */
zfo_spaca = C2n(zdstota)         /* Allocated space units */

/* ---------------------------------------------------------- */
/* Step 6: Putting all the information together               */
/* ---------------------------------------------------------- */
zfo_name_sav = Strip(Strip(Left(zfo_name,40)),"T",".")||".SAV"
                                 /* Name for the bkup zFS */
zfs_name = zfo_name              /* zFS LDS name          */
```

```
If zfs_dsns_chcmd<>"" Then Do
  zfo_chg_len = Length(zfo_chg_str)
  zfs_chg_len = Length(zfs_chg_str)
  pos_zfs = 1
  Do Forever
    nxt_pos = Pos(zfo_chg_str,zfs_name,pos_zfs)
    If nxt_pos=0 Then Leave
    zfs_name = Substr(zfs_name,1,nxt_pos-1)||,
      zfs_chg_str||Substr(zfs_name,nxt_pos+zfo_chg_len)
    pos_zfs = nxt_pos+zfs_chg_len
  End
End
zfs_name_tmp = Strip(Strip(Left(zfs_name,40)),"T",".")||".TMP"
zfs_primu = zfo_primu
zfs_prima = zfo_prima
zfs_seca = zfo_seca
If zfs_def_mgmntclass<>"" Then   /* Management class      */
  zfs_mc = zfs_def_mgmntclass
Else zfs_mc = zfo_mc
If zfs_def_storclass<>"" Then    /* Storage class         */
  zfs_sc = zfs_def_storclass
Else zfs_sc = zfo_sc
If zfs_def_dataclass<>"" Then    /* Data class            */
  zfs_dc = zfs_def_dataclass
Else zfs_dc = zfo_dc
zfs_volumes = ""                 /* List of volumes       */
zfs_primu = "CYLINDERS"
If zfo_#volumes>1 & zfo_percu>90 Then zfs_#volumes = zfo_#volumes+1
Else zfs_#volumes = zfo_#volumes
Select
  When zfo_percu>90 Then Do
    zfs_prima = Min((zfo_cyls*1.2+1)%1,500)
    If zfs_prima>60 & zfs_prima//5<>0 Then
      zfs_prima = zfs_prima+5-zfs_prima//5
    zfs_seca = Min(Max((zfo_cyls/10)%1,1),500)
    If zfs_seca>60 & zfs_seca//5<>0 Then
      zfs_seca = zfs_seca+5-zfs_seca//5
    zfs_sec_allocs = ((zfo_cyls*1.2+1-zfs_prima)/zfs_seca+.99)%1
  End
  When zfo_percu>75 Then Do
    zfs_prima = Min(zfo_cyls,500)
    If zfs_prima>60 & zfs_prima//5<>0 Then
      zfs_prima = zfs_prima+5-zfs_prima//5
    zfs_seca = Min(Max((zfo_cyls/10)%1,1),500)
    If zfs_seca>60 & zfs_seca//5<>0 Then
      zfs_seca = zfs_seca+5-zfs_seca//5
    zfs_sec_allocs = ((zfo_cyls+1-zfs_prima)/zfs_seca+.99)%1
  End
  Otherwise Do
    zfs_prima = Min((zfo_cyls*Min(zfo_percu*1.6/100,1)+1)%1,500)
    If zfs_prima>60 & zfs_prima//5<>0 Then
      zfs_prima = zfs_prima+5-zfs_prima//5
    zfs_seca = Max((zfo_cyls/10*Min(zfo_percu*1.6/100,1))%1,1)
    zfs_seca = Min(zfs_seca,500)
    If zfs_seca>60 & zfs_seca//5<>0 Then
```

```
        zfs_seca = zfs_seca+5-zfs_seca//5
      zfs_sec_allocs = zfo_cyls*Min(zfo_percu*1.6/100,1)-zfs_prima
      zfs_sec_allocs = (zfs_sec_allocs%Max(zfs_seca,1)+.99)%1
    End
  End

  Call WMctl cmnt_line
  Call WMctl Left("ZFS_OLD_NAME=",nn) zfo_name
  Call WMctl cmnt_line
  Call WMctl Left("#ZFS_OLD_#_VOLUMES=",nn) zfo_#volumes
  Call WMctl Left("#ZFS_OLD_DEVICE_TYPE=",nn) zfo_devt
  Call WMctl Left("#ZFS_OLD_ALLOC_UNIT=",nn) zfo_primu
  Call WMctl Left("#ZFS_OLD_ALLOC_SPACE=",nn) zfo_prima zfo_seca
  Call WMctl Left("#ZFS_OLD_TOTAL_UNITS_ALLOCATED=",nn) zfo_spaca
  Call WMctl Left("#ZFS_OLD_TOTAL_UNITS_FORMATTED=",nn) zfo_fmt_info
  Call WMctl Left("#ZFS_OLD_TOTAL_UNITS_%USED=",nn) zfo_percu
  Call WMctl Left("#ZFS_OLD_DATACLASS=",nn) zfo_dc
  Call WMctl Left("#ZFS_OLD_MGMNTCLASS=",nn) zfo_mc
  Call WMctl Left("#ZFS_OLD_STORCLASS=",nn) zfo_sc
  Call WMctl Left("ZFS_OLD_NAME_SAV=",nn) zfo_name_sav
  Call WMctl Left("ZFS_NEW_NAME_TMP=",nn) zfs_name_tmp
  Call WMctl Left("ZFS_NEW_NAME=",nn) zfs_name
  Call WMctl Left("#ZFS_NEW_#_VOLUMES=",nn) zfs_#volumes
  Call WMctl Left("ZFS_NEW_VOLUMES=",nn) zfs_volumes
  Call WMctl Left("ZFS_NEW_ALLOC_NUM_CAND_VOLUMES=",nn) zfs_#volumes
  Call WMctl Left("ZFS_NEW_ALLOC_UNIT=",nn) zfs_primu
  Call WMctl Left("ZFS_NEW_ALLOC_SPACE=",nn) zfs_prima zfs_seca
  Call WMctl Left("ZFS_NEW_ALLOC_NUM_SEC_ALLOCS=",nn) zfs_sec_allocs
  Call WMctl Left("ZFS_NEW_DATACLASS=",nn) zfs_dc
  Call WMctl Left("ZFS_NEW_MGMNTCLASS=",nn) zfs_mc
  Call WMctl Left("ZFS_NEW_STORCLASS=",nn) zfs_sc
  Call WMctl Left("ZFS_NEW_REPLACES_ZFS_OLD=",nn) "Y" /* Yes or No */
  Call WMctl " "
End i

If defreorg_mbr<>"" Then Do
  Address ISPEXEC "LMINIT DATAID(DEFRCDID) DATASET('"||defreorg_dsn||,
  "')"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFRO043E ISPF service LMINIT gave RC" rc "on",
      "generating a data ID DEFRCDID."||message
    Call Final_Exit 8
  End

  If mbr_exists Then Do
    Address ISPEXEC "LMOPEN DATAID("||defrcdid||")"
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave RC" rc "on opening DEFRCDDN."
      Call Final_Exit 8
    End
    defrcopn = 1
    Address ISPEXEC "LMMFIND DATAID("||defrcdid||") MEMBER("||,
      defreorg_mbr||") STATS(YES)"
```

```
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave RC" rc "on retrieving statistics for",
        "DEFRCDDN."
      Call Final_Exit 8
    End
    Address ISPEXEC "LMCLOSE DATAID("||defrcdid||")"
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave rc" rc "on closing DEFRCDDN."
      Call Final_Exit 8
    End
    defrcopn = 0
  End
  Else Parse Value "" With zlvers zlmod zlcdate zlmdate zlmtime,
    zlmsec zlcnorc zlinorc zluser zlc4date zlm4date
End

reorgcntl.0 = clines
"EXECIO * DISKW" defrcddn "(STEM REORGCNTL. FINIS"
If rc<>0 Then Do
  Call Disp_Msg "DEFRO044E EXECIO error occurred, Rc("||rc||")"
  Call Final_Exit 8
End

If defreorg_mbr<>"" Then Do
  If zlmod="" Then zlmod = 0
  Else If Verify(zlmod,"1234567890")=0 & zlmod<99 Then zlmod = zlmod+1
  lmmstats_parm = "MODLEVEL("||zlmod||")"
  If zlcdate<>"" Then
    lmmstats_parm = lmmstats_parm "CREATED("||zlcdate||")"
  If zlc4date<>"" Then
    lmmstats_parm = lmmstats_parm "CREATED4("||zlc4date||")"
  If zlinorc<>"" Then
    lmmstats_parm = lmmstats_parm "INITSIZE("||zlinorc||")"
  Address ISPEXEC "LMMSTATS DATAID("||defrcdid||") MEMBER("||,
    defreorg_mbr||")" lmmstats_parm
  If rc<>0 Then Do
    Call Disp_Msg "ISPF gave rc" rc "on generating statistics for",
      "DEFRCDID."
    Call Final_Exit 8
  End
End

If mlines>mline00 Then Do
  Call Write_MFile
  message = "Enter DM to display most recent messages collected. "
End
Else message = ""
message = message || "Change the data as needed. Enter CH to show",
  "further REORG processing information."
Call Disp_Msg Strip(ispfmsg message)
Address ISPEXEC "EDIT DATASET('"||defreorg_ctl||"') MACRO(DZRP$MAC)"
Select
  When rc=0 Then Nop
  When rc=4 Then Nop
  Otherwise Do
```

```
    Call Disp_Msg "DEFRO045E Error on editing occurred, Rc("||rc||")"
    Call Final_exit 8
  End
End

/* ---------------------------------------------------------------- */
/*  End of processing                                               */
/* ---------------------------------------------------------------- */

Call Final_Exit 0
Exit 9999

/* ---------------------------------------------------------------- */
/*  Subroutines                                                     */
/* ---------------------------------------------------------------- */

Final_Exit: Trace O
  Parse Arg final_rc
  If Symbol("DEFRODID")="VAR" Then
    Address ISPEXEC "LMFREE DATAID("||defrodid||")"
  If defroddn<>"" Then Call Bpxwdyn "FREE DD("||defroddn||")"
  If Symbol("DEFRMDID")="VAR" Then Do
    Address ISPEXEC "LMFREE DATAID("||defrmdid||")"
    Address ISPEXEC "VERASE (ZFSM$DID)"
  End
  If defrmddn<>"" Then Call Bpxwdyn "FREE DD("||defrmddn||")"
  If Symbol("DEFRCDID")="VAR" Then Do
    If defrcopn Then Address ISPEXEC "LMCLOSE DATAID("||defrcdid||")"
    Address ISPEXEC "LMFREE DATAID("||defrcdid||")"
  End
  If defrcddn<>"" Then Call Bpxwdyn "FREE DD("||defrcddn||")"
  If Symbol("DEFRDDID")="VAR" Then Do
    If listaccd Then
      Address ISPEXEC "LMDLIST LISTID("||defrddid||") OPTION(FREE)"
    Address ISPEXEC "ISPEXEC LMDFREE LISTID("||defrddid||")"
  End
  If switched Then Do
    Call Syscall_Cmd "setreuid" cur_uid cur_euid, noexit_on_error
    If not_OK Then
      Say "DEFRO055E UID settings could not be switched back..."
  End
Exit final_rc

Syntax_Error:
  Say "DEFRO096W REXX error in sourceline" sigl "of" myname
  Say "DEFRO097I Line" sigl||":" Strip(Sourceline(sigl))
  Say "DEFRO098I This may be caused by invalid data that was received."
  Say "DEFRO099E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  Say "DEFRO100W REXX error in sourceline" sigl "of" myname
  Say "DEFRO101I Line" sigl||":" Strip(Sourceline(sigl))
  Say "DEFRO102E Variable not initialized..."
```

```
  Call Final_Exit 12
  /* Return code "rc" not used yet */
Exit 9999

Halt_Request:
  Say "DEFRO103W REXX processing halted at line" sigl "of" myname
  Say "DEFRO104I Line" sigl||":" Strip(Sourceline(sigl))
  Call Final_Exit 8
Exit 9999

C2n: Procedure
  Parse Arg cstring
  cstring = Strip(cstring)
  Do Forever
    cpos = Pos(",",cstring)
    If cpos>0 Then cstring = Delstr(cstring,cpos,1)
    Else Leave
  End
Return cstring

IsNotValid: Procedure
  Parse Arg clname
  Select
    When Length(clname)>8 Then is_not_valid = 1
    When Verify(Left(clname,1),"1234567890")=0 Then is_not_valid = 1
    When Verify(clname,"ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890@#$")=0 Then
      is_not_valid = 0
    Otherwise is_not_valid = 1
  End
Return is_not_valid

IsInvalidStr: Procedure
  Parse Arg string
  If Verify(string,"ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890@#$.")=0 Then
    is_not_valid = 0
  Else is_not_valid = 1
Return is_not_valid

WMsg: Trace O
  Parse Arg msgline, add2ispfmsg
  add2ispfmsg = (add2ispfmsg<>0)
  If msg_say Then add2ispfmsg = 0
  If add2ispfmsg & ispfmsg = "" Then ispfmsg = msgline
  If msg_say Then Say msgline
  Do While Length(msgline)<>0
    mlines = mlines+1
    Parse Var msgline mline.mlines 81 msgline
  End
Return

WMctl: Trace O
  Parse Arg mdcline
  clines = clines+1
  reorgcntl.clines = mdcline
Return
```

```
Write_MFile:
  "EXECIO" mlines "DISKW" defrmddn "(STEM MLINE. FINIS"
  If rc<>0 Then Do
    Call Disp_Msg "DEFRO046E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  If Symbol("defrmdid")="LIT" Then Do
    Address ISPEXEC "LMINIT DATAID(DEFRMDID) DDNAME("||defrmddn||")"
    If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFRO047E ISPF service LMINIT gave RC" rc "on",
      "generating a data ID DEFRMDID."||message
      Call Final_Exit 8
    End
    zfsm$did = defrmdid
    Address ISPEXEC "VPUT (ZFSM$DID)"
  End

Return

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
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 errno="8A" & errnojr="EF18626F" Then display_msgs = O
  If rc=O & retval=-1 & errno=7O & errnojr="59DO135" Then not_OK = O
  Else not_OK = (rc<>O | retval<O | retval=O & (errno<>O | errnojr<>O))
  If get_size_needed & errno="91" & errnojr="EF176274" Then Do
    not_OK = O
    get_size_needed = O
  End
  OK = (not_OK = O)
  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,"O"),4))<=X2d(2OFF)
    is_zfs = (Left(Right(errnojr,8,"O"),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 8
  End
Return

/* ------------------------------------------------------------------ */
/* */ Rxattach:                                                      /* */
/*    Attaching an aggregate locally with NBS + aggr_grow             */
/*    Attach aggregate/opcode 105 for pfsctl API ZFSCALL_AGGR command */
/* ------------------------------------------------------------------ */

op_code = D2c(105,4) /* Attach aggregate - AGOP_ATTACH_PARMDATA */

aid_eye      = "AGID"                    /* char(4)                  */
aid_len      = "00"x                     /* char(1), sizeof(aggr_id) */
aid_ver      = "01"x                     /* char(1)                  */
aid_name     = Left(aggr_name,45,zero_01) /* char(45)                */
aggr_id      = aid_eye || aid_len || aid_ver || aid_name || aid_reserved
aggr_id      = Overlay(D2c(Length(aggr_id),1),aggr_id,5)

at_eye        = "AGAT"           /* char(4)                          */
at_len        = "0000"x          /* short, sizeof(aggr_attach)       */
at_ver        = "01"x            /* char(1)                          */
at_res1       = "00"x            /* char(1)                          */
at_threshold  = D2c(90,1)        /* char(1), 90                      */
at_increment  = D2c(5,1)         /* char(1), 5                       */
at_flags      = "24"x            /* char(1), NBS + aggrgrow          */
at_res2       = "00"x            /* char(1)                          */
at_reserved = Copies(zero_parm,64)/* int(64), reserved for future use */
aggr_attach   = at_eye || at_len || at_ver || at_res1 || at_threshold,
  || at_increment || at_flags || at_res2 || at_reserved
aggr_attach   = Overlay(D2c(Length(aggr_attach),2),aggr_attach,5)

parm_0 = D2c(parm_len,4)         /* offset to aggr_id                */
parm_1 = D2c(parm_len+Length(aggr_id),4) /* offset to aggr_attach     */
zfs_buf = op_code || parm_0 || parm_1 || parm_23456 || aggr_id ||,
  aggr_attach
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf),,
  noexit_on_error
```

```
/* ---------------------------------------------------------------- */
/* */  Return  /* from Rxattach                                     */
/* ---------------------------------------------------------------- */

/* ---------------------------------------------------------------- */
/* */ Rxdetach:                                                  /* */
/*    Detaching an aggregate                                      */
/*    Detach aggregate/opcode 104 for pfsctl API ZFSCALL_AGGR command */
/* ---------------------------------------------------------------- */

op_code = D2c(104,4) /* Detach aggregate - AGOP_DETACH_PARMDATA */

aid_eye      = "AGID"                    /* char(4)                 */
aid_len      = "00"x                     /* char(1), sizeof(aggr_id) */
aid_ver      = "01"x                     /* char(1)                 */
aid_name     = Left(aggr_name,45,zero_01) /* char(45)               */
aggr_id      = aid_eye || aid_len || aid_ver || aid_name || aid_reserved
aggr_id      = Overlay(D2c(Length(aggr_id),1),aggr_id,5)

parm_0 = D2c(parm_len,4)                 /* offset to aggr_id       */
zfs_buf = op_code || parm_0 || parm_123456 || aggr_id
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf" Length(zfs_buf),,
  noexit_on_error

/* ---------------------------------------------------------------- */
/* */  Return  /* from Rxdetach                                     */
/* ---------------------------------------------------------------- */

/* ---------------------------------------------------------------- */
/* */ Rxlsaggr:                                                  /* */
/*    Retrieving information for all attached aggregates          */
/*    List attached aggregate names (version 2)/opcode 140        */
/* ---------------------------------------------------------------- */

op_code = D2c(140,4) /* Listing attached aggregate names (Version 2) */
aggr_struct_size = 84 /* length of aggregate structure */
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 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 8
  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_name.i = aggr_name_c
  aggr_sysn.i = aggr_system
  not_aggr_attached.aggr_name_c = 0
  aggr_attindex.aggr_name_c = i
  aggr_off = aggr_off+aggr_struct_size
End

/* ---------------------------------------------------------------- */
/* */  Return  /* from Rxlsaggr                                     */
/* ---------------------------------------------------------------- */

/* ---------------------------------------------------------------- */
/* */ Rxaggrinfo:                                                /* */
/*    Retrieving attributes for a specific (attached) aggregate     */
/*    List aggregate status (Version 2)/opcode 146 - ZFSCALL_AGGR cmd */
/* ---------------------------------------------------------------- */

op_code = D2c(146,4) /* List aggregate status (Version 2) */
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
not_aggrinfo_OK = 1
Call Syscall_Cmd "pfsctl ZFS (zfs_cmd) zfs_buf2" Length(zfs_buf2),,
  noexit_on_error
If OK Then Do
  aggr_status = Substr(zfs_buf2,parm_len+aggr_len+1)
  If Left(aggr_status,4)<>"AGST" Then Do
    Say "DEFRO061E Unexpected data for aggregate" aggr_name||"..."
    Return /* Unexpected data error, very unlikely */
  End
  not_aggrinfo_OK = 0
  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          */
  blks_1K = C2d(Substr(aggr_status,21,4))   /* 1K blocks total        */
  blks_8K = C2d(Substr(aggr_status,33,4))/8 /* 8K blocks total        */
  free_sp = C2d(Substr(aggr_status,37,4))   /* aggregate free space K */
  free_8K = C2d(Substr(aggr_status,57,4))/8 /* 8K blocks free         */
  free_1K = C2d(Substr(aggr_status,61,4))   /* 1K fragments free      */
  logf_sz = C2d(Substr(aggr_status,65,4))   /* Log file size in KB    */
  fstb_sz = C2d(Substr(aggr_status,73,4))   /* File system table in K */
  bitm_sz = C2d(Substr(aggr_status,77,4))   /* Bitmap file size in KB */
End

/* ---------------------------------------------------------------- */
/* */  Return  /* from Rxaggrinfo                                   */
/* ---------------------------------------------------------------- */
```

## 2.5 DZRC$MAC

The DZRC$MAC REXX procedure is an ISPF macro used from within procedure DEFREORG to easily call CPYRHELP.

Example 2-5 displays the contents of DZRC$MAC.

*Example 2-5 DZRC$MAC REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DZRC$MAC                                             */
/*  Description: Call CPYRHELP from within DEFREORG EDIT session      */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dzrc$mac | cpyrhelp | ch                             */
/**********************************************************************/

Trace O
Parse Source . . myname .

"ISREDIT MACRO"
If rc<>0 Then Do
  message = "MACRO" myname "gave Rc=" rc
  If rc=28 Then message = message||", probably pending prefix",
    "commands..."
  Call Disp_Msg message
  Exit rc
End

"CPYRHELP"
Exit rc

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

## 2.6 DZRH$MAC

The DZRH$MAC REXX procedure is an ISPF macro used from within procedure DEFREORG to easily call DEFRHELP.

Example 2-6 displays the contents of DZRH$MAC.

*Example 2-6 DZRH$MAC REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DZRH$MAC                                             */
/*  Description: Call DEFRHELP from within DEFREORG EDIT session      */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dzrh$mac | defrhelp | dh                             */
/**********************************************************************/

Trace O
Parse Source . . myname .

"ISREDIT MACRO"
If rc<>0 Then Do
  message = "MACRO" myname "gave Rc=" rc
  If rc=28 Then message = message||", probably pending prefix",
    "commands..."
  Call Disp_Msg message
  Exit rc
End

"DEFRHELP"
Exit rc

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

## 2.7 DZRM$MAC

The DZRM$MAC REXX procedure is an ISPF macro used from within procedure DEFREORG to display messages from the previous processing step.

Example 2-7 displays the contents of DZRM$MAC.

*Example 2-7 DZRM$MAC REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DZRM$MAC                                             */
/*  Description: Display Messages from the previous processing step   */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dzrm$mac | dm                                        */
/*********************************************************************/

Trace O
Parse Source . . myname .

"ISREDIT MACRO"
If rc<>0 Then Do
  message = "MACRO" myname "gave Rc=" rc
  If rc=28 Then message = message||", probably pending prefix",
    "commands..."
  Call Disp_Msg message
  Exit rc
End

Parse Value "" With zerrmsg zerrlm
Address ISPEXEC "VGET (ZFSM$DID)"
If zfsm$did<>"" Then Do
  Address ISPEXEC "BROWSE DATAID("||zfsm$did||")"
  If rc<>0 Then Do
    message = zerrmsg zerrlm "ISPF gave Rc" rc "on browsing data ID",
      "for DEFRMDDN."
    Call Disp_Msg Strip(message)
  End
End
Else
  Call Disp_Msg "No message file has been created yet or can be found."
Exit 0

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

## 2.8 DZRP$MAC

The DZRP$MAC REXX procedure is the ISPF profile macro used for the edit sessions in procedure DEFREORG. Example 2-8 displays the contents of DZRP$MAC.

*Example 2-8 DZRP$MAC REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DZRP$MAC                                             */
/*  Description: Profile MACRO for all DEFREORG EDIT sessions         */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: called as initial macro on editing DEFRGDID data set */
/*********************************************************************/

Trace O
Parse Source . . myname .

"ISREDIT MACRO"
If rc<>0 Then Do
  message = "MACRO" myname "gave Rc=" rc
  If rc=28 Then message = message||", probably pending prefix",
    "commands..."
  Call Disp_Msg message
  Exit rc
End

"ISREDIT DEFINE DZRM$MAC MACRO"
"ISREDIT DEFINE DM ALIAS DZRM$MAC"
"ISREDIT DEFINE DZRH$MAC MACRO"
"ISREDIT DEFINE DEFRHELP ALIAS DZRH$MAC"
"ISREDIT DEFINE DH ALIAS DZRH$MAC"
"ISREDIT DEFINE DZRC$MAC MACRO"
"ISREDIT DEFINE CPYRHELP ALIAS DZRC$MAC"
"ISREDIT DEFINE CH ALIAS DZRC$MAC"
"ISREDIT DEFINE REFRESH  ALIAS DZRR$MAC"
"ISREDIT DEFINE RF ALIAS DZRR$MAC"
"ISREDIT CAPS ON"
"ISREDIT LINE_AFTER 0 = ""TEMPLINE"""
"ISREDIT DELETE 1"
"ISREDIT RESET"

Exit 0

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

## 2.9 DZRR$MAC

The DZRR$MAC REXX procedure is an ISPF macro that refreshes or resets the current reorganization control data definitions that are displayed with procedure DEFREORG.

Example 2-9 displays the contents of DZRR$MAC.

*Example 2-9 DZRR$MAC REXX procedure*

```
/* REXX ***************************************************************/
/*    Procedure: DZRR$MAC                                             */
/*  Description: Refresh DEFREORG control data definitions            */
/*               Property of IBM  (C) Copyright IBM Corp. 2007-2011   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dzrr$mac | refresh | rf                              */
/**********************************************************************/

Trace O
Parse Source . . myname .

"ISREDIT MACRO"
If rc<>0 Then Do
  message = "MACRO" myname "gave Rc=" rc
  If rc=28 Then message = message||", probably pending prefix",
    "commands..."
  Call Disp_Msg message
  Exit rc
End

Parse Value "" With zerrmsg zerrlm
"ISREDIT BUILTIN CANCEL"
If rc<>0 Then Call Disp_Msg "Builtin CANCEL command gave Rc=" rc
Queue "DEFREORG"
Exit 0

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZOO"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

## 2.10 ZFSREORG JCL

The ZFSREORG file is the JCL used for the job running the reorganization copy processing and optionally replaces the old zFS file system with the new one.

Example 2-10 displays the contents of JCL ZFSREORG.

*Example 2-10 ZFSREORG JCL*

```
//ZFSJOB   JOB ,'ZFSREORG',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=xxxx <=== JES2 system affinity
//*MAIN SYSTEM=xxxx   <=== JES3 system affinity
//* ------------------------------------------------------------------
//* Reorganize compat zFS aggregates to new zFS compat mode aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2011
//* ------------------------------------------------------------------
//  SET CPYTOOL=COPYPAX                  <=== Copy utility to be used
//*             TSOREPRO: Use IDCAMS/TSO REPRO function for copying
//*             COPYPAX : Use accessible (std) pax version for copying
//*             COPYTREE: Use accessible (std) copytree for copying
//  SET VERBOSE=N                <=== Y or N, list all objects copied
//*              VERBOSE is used only if COPYPAX is set.
//  SET DEFREORG=&SYSUID..ZFS.REORG.DEFINE(xxxxxxxx)   <=== REORG DEFs
//* ------------------------------------------------------------------
//  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
//* ------------------------------------------------------------------
//ZFSREORG EXEC PGM=IKJEFT01,PARM='ZFSREORG &CPYTOOL. &VERBOSE.'
//*STEPLIB DD DSNAME=IOE.SIOELMOD,DISP=SHR <Uncom'nt if not in LNKLIST
//SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
//STDENV   DD DATA,DLM=##
# ------------------------------------------------------------------
# Force stopping after formal syntax check of STDIN data is done (N|Y)
  STOP_AFTER_SYNTAX_CHECK=N
# Force stopping when old and existing new zFS is/are mounted (N|Y)
  STOP_AFTER_FSS_MOUNTED=N
# Force stopping when the new zFS aggregate is formatted (N|Y)
  STOP_AFTER_ZFS_IS_FORMATTED=N
# Run copy processing only if the target zFS structure is empty (Y|N)
  TARGET_ZFS_MUST_BE_EMPTY=Y
# Specify filesystypes blocking replacement of mounted old by new zFS
  DENIED_UMNT_FSTYPES=TFS
# Specify additional directories to be included in the PATH chain
# PATH=/usr/local/bin
# ------------------------------------------------------------------
##
//STDIN    DD DSNAME=&DEFREORG.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* ------------------------------------------------------------------
```

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *z/OS Distributed File Service zSeries File System Implementation* , SG24-6580

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

**ibm.com**/redbooks

## Other publications

These publications are also relevant as further information sources:

- *z/OS Distributed File Service zSeries File System Administration*, SC24-5989

## Online resources

These websites are also relevant as further information sources:

- Get the file zfs.zfsreorg.unload.bin from:

  ftp://www.redbooks.ibm.com/redbooks/REDP4769/Description2

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# zFS Reorganization Tool

Redpaper™

**Installing the reorganization tool**

**Using the reorganization tool**

**Reorganization tool REXX execs**

ZFSREORG is a tool for reorganizing and restructuring zFS compatibility mode aggregates. It is an alternative to directly using commands like pax and copytree. It provides more flexibility in many situations and offers options for how the reorganization or copy processing should be done.

This IBM Redpaper document describes the tool, explains how to install and use it, and provides the complete code for the tool.

**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

REDP-4769-00