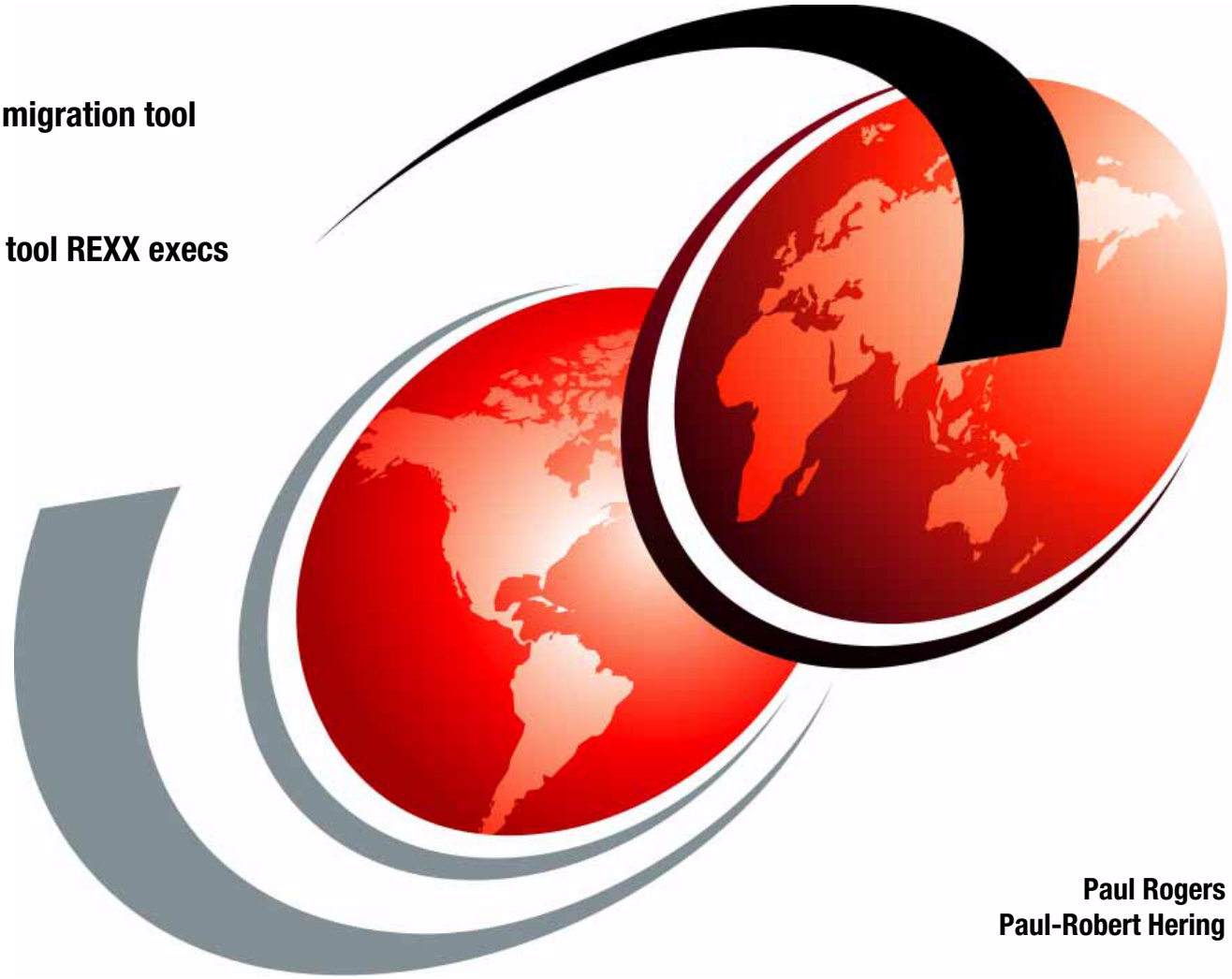
IBM

HFS to zFS Migration Tool

- Installing the migration tool
- Using the migration tool
- Migration tool REXX execs



Paul Rogers
Paul-Robert Hering

Redpaper

ibm.com/redbooks

International Technical Support Organization

HFS to zFS Migration Tool

January 2008

REDP-4328-00

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

First Edition (January 2008)

This edition applies to Versions of z/OS.

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

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

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

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

DFS™
IBM®
MVS™
OS/390®
Redbooks®
Redbooks (logo)  ®
REXX™
VM/ESA®
z/OS®
zSeries®

The following terms are trademarks of other companies:

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

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

Preface

The z/OS® Distributed File Service zSeries® File System (zFS) is the strategic z/OS UNIX® file system that is suggested to be used instead of the Hierarchical File System (HFS), beginning with z/OS V1R7. Therefore, in z/OS V1R7 an official migration tool, BPXWH2Z, is delivered with z/OS to support installations in replacing HFS with zFS.

This Redpaper describes and provides an additional HFS to zFS migration tool, named MIGRTOOL, that has been created as an alternative to BPXWH2Z. It provides more flexibility in many situations and offers options for how the migration should be done. In addition, it supports migration processing independent of the z/OS release, while BPXWH2Z is not supported in a z/OS system running at a level prior to V1R7.

The team that 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 20 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 40 years.

Paul-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, he worked with VM and all its different flavors (VM/370, VM/HPO, VM/XA, and VM/ESA®) for many years.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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® in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail 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

1

HFS to zFS migration tool

This document provides information about a tool for migration of HFS data sets to zFS compatibility mode aggregates, named MIGRTOOL. The following topics are described in this Redpaper:

- Description of migration tool MIGRTOOL
- Using the migration tool

1.1 Description of migration tool MIGRTOOL

In parallel to the official conversion tool BPXWH2Z that is available beginning with z/OS V1R7, this Redpaper introduces a new tool that can be used independently of the existing tool. BPXWH2Z requires the z/OS V1R7 level of the pax utility. The pax utility was enhanced in z/OS V1R7 to support the BPXWH2Z migration tool.

The new migration tool supports the following types of source HFS data sets and targets zFS compatibility mode aggregates.

- It supports using the standard pax utility of the system running the migration.
- It supports any version of pax picked up via the PATH environment variable set.
- It automatically detects whether the version of pax is at a z/OS V1R7 level (or higher). If it is a version previous to z/OS V1R7, it automatically adds all the missing mount points if file systems are mounted on the HFS file system that is about to be migrated.

  Note: This allows automatic replacement (copy) of an HFS file system by a new zFS file system and remounting of the file systems down the structure again without any manual interaction.

- It supports 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.
- It also provides the migration utility **`copymigr`**, which gives you the same functions for migration processing that you get with pax in z/OS V1R7, in a system running a version of z/OS prior to V1R7.
- You can define and select the file systems to be migrated and how to do it interactively. The real migration processing is done using a TSO batch job.
- You have several test or step options for running the job to see whether the migration may 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 HFS (if not already done) and the zFS (if it exists already; the zFS gets unmounted again automatically).
  - You can stop after defining and formatting the zFS aggregate.
- If the zFS aggregate exists already, you can decide to run the copy processing only if the target zFS structure is empty.
- You can decide whether to replace the HFS file system by a zFS file system after doing the migration processing, successfully or not.
- You can name specific PFS types that block automatic replacement of the HFS by the zFS even if you specified to replace the HFS.

1.1.1 Basic installation information

Preparation

Get the file zfs.migrtool.unload.bin from

```
ftp://www.redbooks.ibm.com/redbooks/REDP4328/
```

and put it into an FB80 data set on your z/OS system. Figure 1-1 on page 3 shows sample commands for doing that.

```
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 files zfs.migrtool.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/REDP4328/
lcd 'myuser'
locsite blk=3120 lrecl=80 recfm=fb
binary
get zfs.migrtool.unload.bin zfs.migrtool.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/REDP4328/
binary
get zfs.migrtool.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.migrtool.unload.bin zfs.migrtool.unload
quit
```

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

Now that the files are on the z/OS system, you need to receive them using the TSO RECEIVE command.

```
Final installation steps

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

tso receive indsn('myuser.zfs.migrtool.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.MIGRTOOL 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.migrtool')
```

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.

```
tso receive indsn(zfs.migrtool.unload)
Dataset HERING.ZFS.MIGRTOOL from HERING on WTSCPLX4
 Enter restore parameters or 'DELETE' or 'END' +
[enter]
 ... IEBCOPY MESSAGES AND CONTROL STATEMENTS ... PAGE     1
 IEB1135I IEBCOPY  FMID HDZ1180  SERVICE LEVEL UA27650  DATED 20060711 DFSMS
01.08.00 z/OS    01.08.00 HBB7730  CPU 2094
 IEB1035I HERING   IKJACCT  IKJACCNT 20:39:13 TUE 22 MAY 2007 PARM=''
  COPY INDD=((SYS00070,R)),OUTDD=SYS00069
 IEB1013I COPYING FROM PDSU  INDD=SYS00070 VOL=
DSN=SYS07142.T203912.RA000.HERING.R0100488
 IEB1014I           TO PDS  OUTDD=SYS00069 VOL=BH5ST1 DSN=HERING.ZFS.MIGRTOOL
 IEB167I FOLLOWING MEMBER(S) LOADED FROM INPUT DATA SET REFERENCED BY SYS00070
 IEB154I $INSTALL HAS BEEN SUCCESSFULLY LOADED
 IEB154I $INSTHLP HAS BEEN SUCCESSFULLY LOADED
 IEB154I $INSTRCV HAS BEEN SUCCESSFULLY LOADED
 IEB154I $INSTVFY HAS BEEN SUCCESSFULLY LOADED
 IEB154I COPYMIGR HAS BEEN SUCCESSFULLY LOADED
 IEB154I CPYMHELP HAS BEEN SUCCESSFULLY LOADED
 IEB154I DEFMHELP HAS BEEN SUCCESSFULLY LOADED
 IEB154I DEFMIGR  HAS BEEN SUCCESSFULLY LOADED
 IEB154I DMGC$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DMGH$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DMGM$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DMGP$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I DMGR$MAC HAS BEEN SUCCESSFULLY LOADED
 IEB154I MIGRDATA HAS BEEN SUCCESSFULLY LOADED
 IEB154I MIGRLOAD HAS BEEN SUCCESSFULLY LOADED
 IEB1098I 15 OF 15 MEMBERS LOADED FROM INPUT DATA SET REFERENCED BY SYS00070
 IEB144I THERE ARE 3 UNUSED TRACKS IN OUTPUT DATA SET REFERENCED BY SYS00069
 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.MIGRTOOL'
```

Figure 1-3 Restoring the MIGRTOOL PDS from the unloaded version

1.1.2 Installing the tool

Now you should be able to edit or view the PDS data set ZFS.MIGRTOOL. The first member is named $INSTALL. Near the top you find a sequence of job variables that may 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, shown in Figure 1-4 on page 6. Another option is to use your own REXX and JCL libraries and then change them with the JCL shown in Figure 1-4 on page 6.

Note: If the MIGRTOOL loadlib does not already exist, there is no need to define it because it will be created automatically.

When these requirements are met, you may submit the $INSTALL job. Figure 1-4 on page 6 lists excerpts of the sample job provided.

```
//ZFSJOB   JOB ,'Install MIGRTOOL',NOTIFY=&SYSUID.
...
//* ------------------------------------------------------------------
//  SET   MIGRTOOL=&SYSUID..ZFS.MIGRTOOL         <== This PDS Data Set
//  SET   MIGRLLIB=&SYSUID..ZFS.MIGRTOOL.LOADLIB <== MIGRTOOL Loadlib
//  SET   MIGRREXX=&SYSUID..ZFS.REXX.EXEC        <== REXX Library
//  SET   MIGRJCTL=&SYSUID..ZFS.JOB.CNTL         <== Job CNTL Library
//* ------------------------------------------------------------------
//* Verify Existence of Libraries and Directories, Check System Level
//* ------------------------------------------------------------------
//MIGRVRFY EXEC PGM=IKJEFT01,PARM='%$INSTVFY &MIGRREXX &MIGRJCTL'
//SYSEXEC  DD DSN=&MIGRTOOL.,DISP=SHR
//MIGRREXX DD DSN=&MIGRREXX.,DISP=SHR
//MIGRJCTL DD DSN=&MIGRJCTL.,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//* ------------------------------------------------------------------
//  IF MIGRVRFY.RC EQ 0 THEN
//* ------------------------------------------------------------------
//* Receive ZFS.MIGRTOOL.LOADLIB from unloaded member MIGRLOAD
...
//* Copy member COPYMIGR to the REXX Library
...
//* Copy member DEFMIGR to the REXX Library
...
//* Copy member CPYMHELP to the REXX Library
...
//* Copy member DEFMHELP to the REXX Library
...
//* Copy member MIGRDATA to the Job CNTL Library
...
//* Copy member DMGC$MAC to the REXX Library
...
//* Copy member DMGH$MAC to the REXX Library
...
//* Copy member DMGM$MAC to the REXX Library
...
//* Copy member DMGP$MAC to the REXX Library
...
//* Copy member DMGR$MAC to the REXX Library
...
//* ------------------------------------------------------------------
//  ENDIF MIGRVRFY.RC EQ 0
//* ------------------------------------------------------------------
```

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

Have a look at the step return codes and messages to be sure the job has run successfully.

1.1.3 Tool customization and migration processing

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

DEFMHELP This command provides information about how to set up for the migration tool and how to create migration definition files.

DEFMIGR This command is used to create migration definition statements.

CPYMHELP This command provides information about how to set job and USS environment variables and how to run migration processing.

MIGRDATA This file contains the necessary JCL to run a TSO batch job performing the migration processing.

1.1.4 DEFMIGR help information

The following information is provided when using the DEFMHELP command.

DEFMIGR control statements

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

ZFS_MIGRATE_DEFINE_DSN
This is the full name of a sequential MVS™ data set, including the HLQ or the name of the PDS. It is preferred to use a PDS because 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.MIGRATE.DEFINE, with hlq being your own user ID.

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

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

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.

HFS_DATA_SETS_TO_MIGRATE
This statement defines the HFS data set name list to be migrated. 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 statement allows to specify a file system name change command, for example: /HFS/ZFS/. If you do not specify anything, the new zFS aggregate will get the same name as the original HFS data set. The HFS will get renamed (with .SAV appended at the end by default).

DEFMIGR line commands

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

DEFMHELP or **DH**	The information shown with DEFMHELP is help information; so, DEFMHELP can easily be called from within DEFMIGR.
DM	Displays the last set of messages shown (browse mode).
CPYMHELP or **CH**	Displays help information for the later migration job.
REFRESH or **RF**	Displays the current source statements refreshed; this is useful if you accidentally deleted some information.

1.1.5 MIGRDATA help information

The following information is provided when using the CPYMHELP command:

Migration processing and job variables

The following job variables can be set:

MGRTOOL	This is the copy utility to be used; if you are running at level z/OS V1R7 or above, using COPYPAX is the suggested method.
COPYMIGR	Use the copy tool provided with the migration tool.
COPYPAX	Use the available (std) pax version for copying.
COPYTREE	Use the available (std) copytree for copying.
VERBOSE	Specify N or Y to list all objects copied; VERBOSE is used only if COPYMIGR or COPYPAX are set.
DEFMIGR	Member or data set containing the migration 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 HFS and, if existing, zFS is/are mounted; value must be specified as N or Y.
STOP_AFTER_ZFS_IS_FORMATTED	Force stopping when the 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 HFS by zFS; TFS is always blocking the

	automatic replacement because the complete contents of the TFS are lost otherwise.
PATH	Additional PATH setting; /bin and /samples are included automatically; example: PATH=/usr/local/bin

Migration definition statements

Following is a short description of all the supported definition statements for converting an HFS.

Note: These statements are automatically generated using DEFMIGR, displayed for review, saved in the data set containing the migration definition statements, and referenced in the JCL using DD name STDIN.

HFS_NAME	Name of the HFS data set to be migrated
#HFS_#_VOLUMES	Number of HFS volumes. This value is provided just "For Your Reference" (FYR).
#HFS_DEVICE_TYPE	HFS DASD device type (FYR)
#HFS_ALLOC_UNIT	HFS DASD allocation unit (FYR)
#HFS_ALLOC_SPACE	HFS primary and secondary allocation (FYR)
#HFS_TOTAL_UNITS_ALLOCATED	HFS total number of units allocated (FYR)
#HFS_TOTAL_UNITS_%USED	HFS percentage of space used (FYR)
#HFS_DATACLASS	HFS data class (FYR)
#HFS_MGMNTCLASS	HFS management class (FYR)
#HFS_STORCLASS	HFS storage class (FYR)
HFS_NAME_SAV	Suggested name for HFS to be renamed; this is only used if HFS and zFS have set the same name (FYR).
ZFS_NAME_TMP	Suggested name for zFS used initially; this is only used if HFS and 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 (FYR).
ZFS_VOLUMES	Explicit list of volumes to be used; this specification is ignored if an SMS storage class is assigned.
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_CAND_VOLUMES	Number of zFS candidate volumes
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 value means that no data class is set.
ZFS_MGMNTCLASS	zFS SMS management class to be used; a BLANK value 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_HFS	This value specifies whether the zFS file system should replace the HFS; you must specify Y or N.

1.2 Using the migration tool

The following demonstrates how to use the tool by showing a short example.

Migration scenario example

Figure 1-5 shows an example of the status of the HFS data sets that are going to be migrated and are mounted off of the TEST directory.

```
$> cd /u/hering
$> /usr/sbin/mount -qv test
----A- OMVS.HERING.TEST.OTHERZFS /u/hering/test/otherzfs
R---A- OMVS.HERING.TEST.OTHERHFS /u/hering/test/otherhfs
----A- OMVS.HERING.TEST          /u/hering/test
$> df -v test | grep Owner
File System Owner : SC74        Automove=Y      Client=N
$> df -v test | grep Device
HFS, Read/Write, Device:165, ACLS=Y
$> df -v test/otherhfs | grep Owner
File System Owner : SC74        Automove=Y      Client=N
$> df -v test/otherhfs | grep Device
HFS, Read Only, Device:167, ACLS=Y
$> df -v test/otherzfs | grep Owner
File System Owner : SC75        Automove=Y      Client=N
$> df -v test/otherzfs | grep Device
ZFS, Read/Write, Device:166, ACLS=Y
```

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

So, there are three file systems involved regarding the naming conventions and the mounted file structure:

- OMVS.HERING.TEST
- OMVS.HERING.TEST.OTHERHFS
- OMVS.HERING.TEST.OTHERZFS

As suggested, the migration definition data has been allocated as a PDS with record type VB80.

1.2.1 Doing the migration

This example shows how to migrate just the HFS file system named OMVS.HERING.TEST. This is done just to reduce the amount of messages in this example. The normal case would be to migrate all three HFS file systems.

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 DEFMIGR command, which brings up an EDIT session as shown in Figure 1-6. This command can be issued from any ISPF command line, using TSO DEFMIGR.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       SYS07143.T112833.RA000.HERING.R0100509          Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 000001 # ---------------------------------------------------------------------- #
 000002 ZFS_MIGRATE_DEFINE_DSN=HERING.ZFS.MIGRATE.DEFINE
 000003 ZFS_MIGRATE_DEFINE_MBR=
 000004 ZFS_MIGRATE_DEFINE_DSP=APPEND
 000005 # ---------------------------------------------------------------------- #
 000006
 000007 # ---------------------------------------------------------------------- #
 000008 ZFS_DEF_DATACLASS=
 000009 ZFS_DEF_MANAGEMENTCLASS=
 000010 ZFS_DEF_STORAGECLASS=
 000011 # ---------------------------------------------------------------------- #
 000012
 000013 # ---------------------------------------------------------------------- #
 000014 HFS_DATA_SETS_TO_MIGRATE=
 000015 ZFS_AGGRNAME_CHANGE_CMD=
 000016 # ---------------------------------------------------------------------- #
 ****** **************************** Bottom of Data ****************************



 _____________________________________________________________________
| Enter DH to show migration 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 DEFMIGR utility

In this file we changed the settings as needed. Figure 1-7 on page 12 shows the display after making 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       SYS07143.T112833.RA000.HERING.R0100509          Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 000001 # ---------------------------------------------------------------------- #
 000002 ZFS_MIGRATE_DEFINE_DSN=HERING.ZFS.MIGRATE.DEFINE
 000003 ZFS_MIGRATE_DEFINE_MBR=migr01
 000004 ZFS_MIGRATE_DEFINE_DSP=rPPEND
 000005 # ---------------------------------------------------------------------- #
 000006
 000007 # ---------------------------------------------------------------------- #
 000008 ZFS_DEF_DATACLASS=zfsbig
 000009 ZFS_DEF_MANAGEMENTCLASS=
 000010 ZFS_DEF_STORAGECLASS=
 000011 # ---------------------------------------------------------------------- #
 000012
 000013 # ---------------------------------------------------------------------- #
 000014 HFS_DATA_SETS_TO_MIGRATE=omvs.hering.test
 000015 ZFS_AGGRNAME_CHANGE_CMD=
 000016 # ---------------------------------------------------------------------- #
 ****** ***************************** Bottom of Data ****************************


 ____________________________________________________________________
| Enter DH to show migration 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 MIGR01, requested to replace the member if available, set a default zFS data class, and finally entered the HFS data sets to look for. Pressing PF03 ended the EDIT session. DEFMIGR then calculated the settings, displayed messages, 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.

```
DEFMG050I Searching for data sets OMVS.HERING.TEST ...
DEFMG048I Data set OMVS.HERING.TEST will be examined.
DEFMG048I Data set OMVS.HERING.TEST.OTHERHFS will be examined.
DEFMG048I Data set OMVS.HERING.TEST.OTHERZFS will be examined.
DEFMG048I Data set OMVS.HERING.TEST.OTHERZFS.DATA will be examined.
DEFMG042I Data set OMVS.HERING.TEST.OTHERZFS is not an HFS and will be skipped.
DEFMG042I Data set OMVS.HERING.TEST.OTHERZFS.DATA is not an HFS and will be ski
pped.
***
```

Figure 1-8 Messages displayed while the tool is creating conversion 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.MIGRATE.DEFINE(MIGR01) - 01.02       Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 # ------------------------------------------------------------------- #
000002 # MIGRATE CONTROL DEFINITIONS, CREATED 2007-05-23 18:30:36            #
000003 # ------------------------------------------------------------------- #
000004
000005 # ------------------------------------------------------------------- #
000006 HFS_NAME=                      OMVS.HERING.TEST
000007 # ------------------------------------------------------------------- #
000008 #HFS_#_VOLUMES=                1
000009 #HFS_DEVICE_TYPE=              3390
000010 #HFS_ALLOC_UNIT=               CYLINDER
000011 #HFS_ALLOC_SPACE=              1 1
000012 #HFS_TOTAL_UNITS_ALLOCATED=    1
000013 #HFS_TOTAL_UNITS_%USED=        5
000014 #HFS_DATACLASS=
000015 #HFS_MGMNTCLASS=
000016 #HFS_STORCLASS=                OPENMVS
000017 HFS_NAME_SAV=                  OMVS.HERING.TEST.SAV
000018 ZFS_NAME_TMP=                  OMVS.HERING.TEST.TMP
000019 ZFS_NAME=                      OMVS.HERING.TEST
000020 #ZFS_#_VOLUMES=                1
000021 ZFS_VOLUMES=                   BH5OE2
000022 ZFS_ALLOC_UNIT=                CYLINDERS
 _____________________________________________________________________
| Enter DM to display most recent messages collected. Change the data as |
| needed. Enter CH to show further migration processing information.     |
|_____________________________________________________________________|
Command ===>                                                Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

Figure 1-9 EDIT session displaying the conversion statements just created, part 1

Going down in the display we got what is shown in Figure 1-10 on page 14.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.MIGRATE.DEFINE(MIGR01) - 01.02       Columns 00001 00072
000022 ZFS_ALLOC_UNIT=                CYLINDERS
000023 ZFS_ALLOC_SPACE=               1 1
000024 ZFS_ALLOC_NUM_CAND_VOLUMES=    0
000025 ZFS_ALLOC_NUM_SEC_ALLOCS=      0
000026 ZFS_DATACLASS=                 ZFSBIG
000027 ZFS_MGMNTCLASS=
000028 ZFS_STORCLASS=                 OPENMVS
000029 ZFS_REPLACES_HFS=              Y
000030
000031 # ------------------------------------------------------------------- #
000032 HFS_NAME=                      OMVS.HERING.TEST.OTHERHFS
000033 # ------------------------------------------------------------------- #
000034 #HFS_#_VOLUMES=                1
000035 #HFS_DEVICE_TYPE=              3390
000036 #HFS_ALLOC_UNIT=               CYLINDER
000037 #HFS_ALLOC_SPACE=              1 1
000038 #HFS_TOTAL_UNITS_ALLOCATED=    1
000039 #HFS_TOTAL_UNITS_%USED=        2
000040 #HFS_DATACLASS=
000041 #HFS_MGMNTCLASS=
000042 #HFS_STORCLASS=                OPENMVS
000043 HFS_NAME_SAV=                  OMVS.HERING.TEST.OTHERHFS.SAV
000044 ZFS_NAME_TMP=                  OMVS.HERING.TEST.OTHERHFS.TMP
000045 ZFS_NAME=                      OMVS.HERING.TEST.OTHERHFS
000046 #ZFS_#_VOLUMES=                1
000047 ZFS_VOLUMES=                   BH50E2
000048 ZFS_ALLOC_UNIT=                CYLINDERS
Command ===>                                              Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

Figure 1-10 EDIT session displaying the conversion statements just created, part 2

Because we only wanted to migrate OMVS.HERING.TEST, we deleted all lines starting at line 31, accepted the values suggested, saved the data, and ended the EDIT session.

Then we edited the JCL member MIGRDATA and modified the data as shown in Figure 1-11 on page 15 and Figure 1-12 on page 16.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.JOB.CNTL(MIGRDATA) - 01.00          Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 //ZFSJOB   JOB ,'MIGRDATA',NOTIFY=&SYSUID.,REGION=0M
000002 /*JOBPARM SYSAFF=SC74 <=== JES2 system affinity
000003 //* -------------------------------------------------------------------
000004 //* Migrate HFS data sets to zFS compat mode aggregates
000005 //* Property of IBM  (C) Copyright IBM Corp. 2002-2007
000006 //* -------------------------------------------------------------------
000007 //  SET MGRTOOL=COPYPAX                  <=== Copy utility to be used
000008 //*             COPYMIGR: Use copy tool provided with migration tool
000009 //*             COPYPAX : Use accessible (std) pax version for copying
000010 //*             COPYTREE: Use accessible (std) copytree for copying
000011 //  SET VERBOSE=N                <=== Y or N, list all objects copied
000012 //*              VERBOSE is used only if COPYMIGR or COPYPAX are set.
000013 //  SET DEFMIGR=&SYSUID..ZFS.MIGRATE.DEFINE(MIGR01)  <=== MIGRATE DEFs
000014 //* -------------------------------------------------------------------
000015 //  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC            <=== SYSEXEC library
000016 //  SET STEPLIB=&SYSUID..ZFS.MIGRTOOL.LOADLIB     <=== STEPLIB library
000017 //* -------------------------------------------------------------------
000018 //COPYMIGR EXEC PGM=IKJEFT01,
000019 //  PARM='COPYMIGR &MGRTOOL. &VERBOSE. &OVERWRT.'
000020 //STEPLIB  DD DSNAME=&STEPLIB.,DISP=SHR
000021 //SYSEXEC  DD DSNAME=&REXXLIB.,DISP=SHR
000022 //STDENV   DD DATA,DLM=##
000023 # -------------------------------------------------------------------
000024 # Force stopping after formal syntax check of STDIN data is done (N|Y)
000025   STOP_AFTER_SYNTAX_CHECK=N
000026 # Force stopping when HFS and if existing the zFS is/are mounted (N|Y)
Command ===>                                                Scroll ===> CSR
 F3=Exit      F4=Save      F5=Rfind     F6=Rchange  F10=Retrieve F12=Cancel
```

Figure 1-11 JCL MIGRDATA with the modified migration definition member name

No further changes were made in the remaining part of the JCL, shown in Figure 1-12 on page 16.

Attention: If the DFS™ loadlib, by default named IOE.SIOELMOD, is not part of the LNKLIST concatenation, you need to add the library to the STEPLIB ddname in the MIGRDATA JCL.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________________
EDIT       HERING.ZFS.JOB.CNTL(MIGRDATA) - 01.00           Columns 00001 00072
000022 //STDENV   DD DATA,DLM=##
000023 # ---------------------------------------------------------------------
000024 # Force stopping after formal syntax check of STDIN data is done (N|Y)
000025   STOP_AFTER_SYNTAX_CHECK=N
000026 # Force stopping when HFS and if existing the zFS is/are mounted (N|Y)
000027   STOP_AFTER_FSS_MOUNTED=N
000028 # Force stopping when the zFS aggregate is formatted (N|Y)
000029   STOP_AFTER_ZFS_IS_FORMATTED=N
000030 # Run copy processing only if the target zFS structure is empty (Y|N)
000031   TARGET_ZFS_MUST_BE_EMPTY=Y
000032 # Specify filesystypes blocking replacement of mounted HFS by zFS
000033   DENIED_UMNT_FSTYPES=TFS
000034 # Specify additional directories to be included in the PATH chain
000035 # PATH=/usr/local/bin
000036 # ---------------------------------------------------------------------
000037 ##
000038 //STDIN    DD DSNAME=&DEFMIGR.,DISP=SHR
000039 //SYSTSIN  DD DUMMY
000040 //SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
000041 //* ---------------------------------------------------------------------
****** **************************** Bottom of Data ****************************









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

Figure 1-12 JCL MIGRDATA showing unmodified environment variables

Finally we submitted the job. Figure 1-13 on page 17 shows the messages we got after the job successfully ended with RC=0.

```
COPMGO65I Running with options Migrtool=COPYPAX and Verbose=N ...

------------------------------------------------------------------------------
The main process ID for this job is 83952245. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 83952245
Or    : kill -s SIGTERM 83952245
------------------------------------------------------------------------------

------------------------------------------------------------------------------
COPMGO04I Processing HFS data set name OMVS.HERING.TEST...
------------------------------------------------------------------------------
COPMGO75I As the zFS aggregate name is the same as the HFS data set a temporary name is
used for the zFS.
COPMGO76I zFS temporary name: OMVS.HERING.TEST.TMP
COPMGO86I Defining zFS aggregate OMVS.HERING.TEST.TMP ...
IOEZ00248I VSAM linear dataset OMVS.HERING.TEST.TMP successfully created.
COPMGO87I Formatting zFS aggregate OMVS.HERING.TEST.TMP ...
IOEZ00077I HFS-compatibility aggregate OMVS.HERING.TEST.TMP has been successfully
created
COPMGO54I Now starting copy processing...
COPMGO70I Copy processing has been ended...
COPMG105I Temporary unmounting OMVS.HERING.TEST.OTHERZFS mounted at
/u/hering/test/otherzfs
COPMG105I Temporary unmounting OMVS.HERING.TEST.OTHERHFS mounted at
/u/hering/test/otherhfs
COPMG106I Unmounting OMVS.HERING.TEST ...
COPMG103I The HFS data set has been renamed to OMVS.HERING.TEST.SAV.
ENTRY OMVS.HERING.TEST.TMP ALTERED
COPMG104I The zFS aggregate has been renamed to OMVS.HERING.TEST.
ENTRY OMVS.HERING.TEST.TMP.DATA ALTERED
COPMG119I The zFS DATA part has been renamed to OMVS.HERING.TEST.DATA.
COPMG107I Mounting zFS OMVS.HERING.TEST at /u/hering/test now...
COPMG109I Mounting file system OMVS.HERING.TEST.OTHERHFS at /u/hering/test/otherhfs
again
COPMG109I Mounting file system OMVS.HERING.TEST.OTHERZFS at /u/hering/test/otherzfs
again
COPMG111I No errors have been recognized for the actual migration process.

READY
END
```

Figure 1-13 Messages shown in joblog about the migration processing

1.2.2 Migration results

Finally let us have a look at the picture after having done the migration, shown in Figure 1-14 on page 18.

```
$> /usr/sbin/mount -qv test
----A- OMVS.HERING.TEST.OTHERZFS /u/hering/test/otherzfs
R---A- OMVS.HERING.TEST.OTHERHFS /u/hering/test/otherhfs
----A- OMVS.HERING.TEST          /u/hering/test
$> df -v test | grep Owner
File System Owner : SC74        Automove=Y      Client=N
$> df -v test | grep Device
ZFS, Read/Write, Device:169, ACLS=Y
$> df -v test/otherhfs | grep Owner
File System Owner : SC74        Automove=Y      Client=N
$> df -v test/otherhfs | grep Device
HFS, Read Only, Device:170, ACLS=Y
$> df -v test/otherzfs | grep Owner
File System Owner : SC75        Automove=Y      Client=N
$> df -v test/otherzfs | grep Device
ZFS, Read/Write, Device:171, ACLS=Y
```

Figure 1-14 Final situation after having done the migration

So, this verifies that the migration processing was done successfully and the sub-file systems were mounted with the same attributes as before.

1.2.3 Migration processing in backlevel z/OS releases

This tool is very useful as an alternative to the standard migration tool BPXWH2Z if you like the way it works and the flexibility it provides. There is another good reason to use it if you are running a z/OS release prior to z/OS V1R7.

Important: The standard migration tool is not supported in a system running a release of z/OS prior to V1R7.

So, we want to add another example for a z/OS V1R6 environment and show two different approaches how you can use the MIGRTOOL in this situation. Figure 1-15 shows the initial situation with two HFS file systems mounted.

```
#> echo $(uname -I) Version $(uname -Iv).$(uname -Ir)
z/OS Version 01.06.00
#> cd /tmp/heritemp/hfstest
#> /usr/sbin/mount -qv .
----A- HERI.TESTSUB.HFS /SYSTEM/tmp/heritemp/hfstest/test
----A- HERI.TEST.HFS    /SYSTEM/tmp/heritemp/hfstest
#> df -v . | grep Device
HFS, Read/Write, Device:140, ACLS=Y
#> df -v ./test | grep Device
HFS, Read/Write, Device:141, ACLS=Y
```

Figure 1-15 Initial situation before starting migration processing

We want to migrate HERI.TEST.HFS and create the migration definition statements shown in Figure 1-16 on page 19 using DEFMIGR.

```
# ------------------------------------------------------------------ #
# MIGRATE CONTROL DEFINITIONS, CREATED 2007-05-22 18:18:07           #
# ------------------------------------------------------------------ #

# ------------------------------------------------------------------ #
HFS_NAME=                       HERI.TEST.HFS
# ------------------------------------------------------------------ #
#HFS_#_VOLUMES=                 1
#HFS_DEVICE_TYPE=               3390
#HFS_ALLOC_UNIT=                CYLINDER
#HFS_ALLOC_SPACE=               9 9
#HFS_TOTAL_UNITS_ALLOCATED=     9
#HFS_TOTAL_UNITS_%USED=         1
#HFS_DATACLASS=
#HFS_MGMNTCLASS=                STANDARD
#HFS_STORCLASS=                 OMVS
HFS_NAME_SAV=                   HERI.TEST.HFS.SAV
ZFS_NAME_TMP=                   HERI.TEST.HFS.TMP
ZFS_NAME=                       HERI.TEST.HFS
#ZFS_#_VOLUMES=                 1
ZFS_VOLUMES=                    VSD6S1
ZFS_ALLOC_UNIT=                 CYLINDERS
ZFS_ALLOC_SPACE=                9 9
ZFS_ALLOC_NUM_CAND_VOLUMES=     0
ZFS_ALLOC_NUM_CAND_VOLUMES=     0
ZFS_ALLOC_NUM_SEC_ALLOCS=       0
ZFS_DATACLASS=
ZFS_MGMNTCLASS=                 STANDARD
ZFS_STORCLASS=                  OMVS
ZFS_REPLACES_HFS=               Y
```

Figure 1-16 Migration definition statements for HFS HERI.TEST.HFS

Migration using the pax version of z/OS V1R6

First, Figure 1-17 on page 20 shows the job settings used for this approach.

```
//ZFSJOB   JOB ,'MIGRDATA',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Migrate HFS data sets to zFS compat mode aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2002-2007
//* ------------------------------------------------------------------
//  SET MGRTOOL=COPYPAX                  <=== Copy utility to be used
//*             COPYMIGR: Use copy tool provided with migration tool
//*             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 COPYMIGR or COPYPAX are set.
//  SET DEFMIGR=&SYSUID..ZFS.MIGRATE.DEFINE(MIGR07)  <=== MIGRATE DEFs
//* ------------------------------------------------------------------
//  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC           <=== SYSEXEC library
//  SET STEPLIB=&SYSUID..ZFS.MIGRTOOL.LOADLIB    <=== STEPLIB library
//* ------------------------------------------------------------------
//COPYMIGR EXEC PGM=IKJEFT01,
//  PARM='COPYMIGR &MGRTOOL. &VERBOSE. &OVERWRT.'
//STEPLIB  DD DSNAME=&STEPLIB.,DISP=SHR
//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 HFS and if existing the zFS is/are mounted (N!Y)
  STOP_AFTER_FSS_MOUNTED=N
# Force stopping when the 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 HFS by zFS
  DENIED_UMNT_FSTYPES=TFS
# Specify additional directories to be included in the PATH chain
# PATH=/usr/local/bin
# ------------------------------------------------------------------
##
//STDIN    DD DSNAME=&DEFMIGR.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* ------------------------------------------------------------------
```

Figure 1-17 MIGRDATA JCL for migration using pax of z/OS V1R6

Figure 1-18 on page 21 shows the messages we got after successfully running the migration.

```
COPMGO65I Running with options Migrtool=COPYPAX and Verbose=N ...

------------------------------------------------------------------------------
The main process ID for this job is 67109528. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 67109528
Or    : kill -s SIGTERM 67109528
------------------------------------------------------------------------------

------------------------------------------------------------------------------
COPMGO04I Processing HFS data set name HERI.TEST.HFS...
------------------------------------------------------------------------------
COPMGO75I As the zFS aggregate name is the same as the HFS data set a temporary name is
used for the zFS.
COPMGO76I zFS temporary name: HERI.TEST.HFS.TMP
COPMGO86I Defining zFS aggregate HERI.TEST.HFS.TMP ...
IOEZ00248E VSAM linear dataset HERI.TEST.HFS.TMP successfully created.
COPMGO87I Formatting zFS aggregate HERI.TEST.HFS.TMP ...
IOEZ00077I HFS-compatibility aggregate HERI.TEST.HFS.TMP has been successfully created
COPMGO54I Now starting copy processing...
COPMGO66I Mount point test has been created relative to the zFS top directory.
COPMGO70I Copy processing has been ended...
COPMG105I Temporary unmounting HERI.TESTSUB.HFS mounted at
/SYSTEM/tmp/heritemp/hfstest/test
COPMG106I Unmounting HERI.TEST.HFS ...
COPMG103I The HFS data set has been renamed to HERI.TEST.HFS.SAV.
IDC0531I ENTRY HERI.TEST.HFS.TMP ALTERED
COPMG104I The zFS aggregate has been renamed to HERI.TEST.HFS.
IDC0531I ENTRY HERI.TEST.HFS.TMP.DATA ALTERED
COPMG119I The zFS DATA part has been renamed to HERI.TEST.HFS.DATA.
COPMG107I Mounting zFS HERI.TEST.HFS at /SYSTEM/tmp/heritemp/hfstest now...
COPMG109I Mounting file system HERI.TESTSUB.HFS at /SYSTEM/tmp/heritemp/hfstest/test
again
COPMG111I No errors have been recognized for the actual migration process.

READY
END
```

Figure 1-18 Messages in the joblog after successful migration using pax of z/OS V1R6

Note that the missing mount point /tmp/heritemp/test has been created after running pax because that version of pax does not support this function itself.

Migration using COPYMIGR

After resetting to the initial situation, we did the migration again and now used COPYMIGR instead of COPYPAX in the job, as shown in Figure 1-19 on page 22.

```
//ZFSJOB   JOB ,'MIGRDATA',NOTIFY=&SYSUID.,REGION=0M
//* ------------------------------------------------------------------
//* Migrate HFS data sets to zFS compat mode aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2002-2007
//* ------------------------------------------------------------------
//  SET MGRTOOL=COPYMIGR                  <=== Copy utility to be used
//*             COPYMIGR: Use copy tool provided with migration tool
//*             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 COPYMIGR or COPYPAX are set.
//  SET DEFMIGR=&SYSUID..ZFS.MIGRATE.DEFINE(MIGR07)  <=== MIGRATE DEFs
...
```

Figure 1-19 MIGRDATA JCL for migration using the COPYMIGR copy tool of MIGRTOOL

This provides the messages shown in Figure 1-20 when running the migration.

```
COPMG065I Running with options Migrtool=COPYMIGR and Verbose=N ...

------------------------------------------------------------------------------
The main process ID for this job is 33556748. If you should need to stop
processing use the following UNIX command to do this smoothly.
Either: kill 33556748
Or    : kill -s SIGTERM 33556748
------------------------------------------------------------------------------

------------------------------------------------------------------------------
COPMG004I Processing HFS data set name HERI.TEST.HFS...
------------------------------------------------------------------------------
COPMG075I As the zFS aggregate name is the same as the HFS data set a temporary name is
used for the zFS.
COPMG076I zFS temporary name: HERI.TEST.HFS.TMP
COPMG086I Defining zFS aggregate HERI.TEST.HFS.TMP ...
IOEZ00248E VSAM linear dataset HERI.TEST.HFS.TMP successfully created.
COPMG087I Formatting zFS aggregate HERI.TEST.HFS.TMP ...
IOEZ00077I HFS-compatibility aggregate HERI.TEST.HFS.TMP has been successfully created
COPMG054I Now starting copy processing...
COPMG070I Copy processing has been ended...
COPMG105I Temporary unmounting HERI.TESTSUB.HFS mounted at
/SYSTEM/tmp/heritemp/hfstest/test
COPMG106I Unmounting HERI.TEST.HFS ...
COPMG103I The HFS data set has been renamed to HERI.TEST.HFS.SAV.
IDC0531I ENTRY HERI.TEST.HFS.TMP ALTERED
COPMG104I The zFS aggregate has been renamed to HERI.TEST.HFS.
IDC0531I ENTRY HERI.TEST.HFS.TMP.DATA ALTERED
COPMG119I The zFS DATA part has been renamed to HERI.TEST.HFS.DATA.
COPMG107I Mounting zFS HERI.TEST.HFS at /SYSTEM/tmp/heritemp/hfstest now...
COPMG109I Mounting file system HERI.TESTSUB.HFS at /SYSTEM/tmp/heritemp/hfstest/test
again
COPMG111I No errors have been recognized for the actual migration process.

READY
END
```

Figure 1-20 Messages in the joblog after successful migration using the COPYMIGR copy tool

So, both ways are possible solutions offered by MIGRTOOL. Figure 1-21 finally shows a resulting view from USS.

```
#> cd /tmp/heritemp/hfstest/
#> /usr/sbin/mount -qv .
----A- HERI.TESTSUB.HFS /SYSTEM/tmp/heritemp/hfstest/test
----A- HERI.TEST.HFS    /SYSTEM/tmp/heritemp/hfstest
#> df -v . | grep Device
ZFS, Read/Write, Device:143, ACLS=Y
```

Figure 1-21 Resulting view from USS

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 MIGRTOOL.

A

The MIGRTOOL files

This appendix contains the MIGRTOOL procedures and the JCL that belongs to this package. These REXX procedures are downloaded into your myuser.ZFS.REXX.EXEC library or whatever data set you choose.

This appendix includes the following files and a short description of each:

- COPYMIGR
- CPYMHELP
- DEFMHELP
- DEFMIGR
- DMGC$MAC
- DMGH$MAC
- DMGM$MAC
- DMGP$MAC
- DMGR$MAC
- MIGRDATA

A.1 COPYMIGR

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

Example A-1 displays the contents of COPYMIGR.

Example: A-1 COPYMIGR REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: COPYMIGR (Next free MSGNO: Say "COPMG126x ...")      */
/*  Description: Migrate HFS data sets to zFS compat mode aggregates  */
/*               Property of IBM  (C) Copyright IBM Corp. 2006-2007   */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: copymigr migrtool <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()
hfs_temp_mounted = 0
zfs_temp_mounted = 0
hfs_switched_ro = 0
zfs_exists = 0
zfs_formatted = 0
zfs_created = 0
zfs_usable = 0
zfs_mounted = 0
hfs_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 migrtool verbose .
If Wordpos(migrtool,"COPYMIGR COPYPAX COPYTREE")=0 Then Do
  Say "COPMG001E You must provide COPYMIGR, COPYPAX or COPYTREE as the",
    "first parameter"
```

```
  Say "to specify the migration 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 "COPMG063I Option Verbose, if specified, must be Y or N."
  Exit 8
End
Say "COPMG065I Running with options Migrtool=" || migrtool "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 "COPMG002E The SYSCALL environment could not be established."
  Exit 8
End

If Syscalls("SIGON")<>0 Then Do
  Say "COPMG003E The SIGNAL interface could not be established."
  Exit 8
End

If foreground Then Do
  Say "COPMG080E 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"
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||"/COPYMIGR."||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_hfs 755"
tmp_hfs = tmp_dir||"/"||"tmp_hfs"
Call Syscall_Cmd "mkdir tmp_zfs 755"
```

```
tmp_zfs = tmp_dir||"/"||"tmp_zfs"
Call Syscall_Cmd "extlink MIGRDATA copymigr"
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 migration statements                            */
/* ---------------------------------------------------------------- */

If background Then Do
  "EXECIO * DISKR STDIN (STEM INPUTD. FINIS"
  If rc<>0 Then Do
    Say "COPMG122E The specified member on DDNAME STDIN probably does",
      "not exist."
    Call Final_Exit 8
  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 migration 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_hfs_name = "HFS_NAME"
s_hfs_name_sav = "HFS_NAME_SAV"
```

```
s_zfs_name_tmp = "ZFS_NAME_TMP"
s_zfs_name = "ZFS_NAME"
s_zfs_volumes = "ZFS_VOLUMES"
s_zfs_alloc_unit = "ZFS_ALLOC_UNIT"
s_zfs_alloc_space = "ZFS_ALLOC_SPACE"
s_zfs_alloc_nocvols = "ZFS_ALLOC_NUM_CAND_VOLUMES"
s_zfs_alloc_nosallc = "ZFS_ALLOC_NUM_SEC_ALLOCS"
s_zfs_dataclass = "ZFS_DATACLASS"
s_zfs_mgmntclass = "ZFS_MGMNTCLASS"
s_zfs_storclass = "ZFS_STORCLASS"
s_zfs_replaces_hfs = "ZFS_REPLACES_HFS"
not_first_hfs = 0
in_lines = in_lines+1
inputd.in_lines = s_hfs_name||"=Dummy.HFS"
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_hfs_name Then Do
      If not_first_hfs Then Do
        Call Migrate_HFS2zFS
        Call Migrate_Data_Reset 1 /* 1= display curmig msg */
      End
      Else Do
        not_first_hfs = 1
        Call Migrate_Data_Reset 0 /* 0= no curmig msg */
      End
      Parse Var in_value hfs_name .
      If i<in_lines Then Do
        Say
        Say dash_line
        Say "COPMG004I Processing HFS data set name" hfs_name||"..."
        Say dash_line
      end
      If hfs_name="" Then Do
        Say "COPMG005E No HFS data set name has been provided."
        error_in_line = 1
      End
      Else If hfs_processed.hfs_name Then Do
        Say "COPMG006W HFS data set name has been processed already."
        error_in_line = 1
      End
      hfs_processed.hfs_name = 1
    End
    When in_parm=s_hfs_name_sav Then
      Parse Var in_value hfs_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 "COPMG007E 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 "COPMGOO8W 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 "COPMGOO9E primary and secondary allocation values must be",
      "positive whole numbers."
    error_in_line = 1
  End
  If prim_alloc=0 Then Do
    Say "COPMGO61E 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 "COPMGO10E 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 "COPMGO11E 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_hfs Then Do
      Parse Var in_value zfs_replaces_hfs .
      If Wordpos(zfs_replaces_hfs,"Y N")=0 Then Do
        Say "COPMG012E The specification for replacing the HFS must be",
          "Y or N."
        error_in_line = 1
      End
    End
    Otherwise Do
      Say "COPMG013E The specified option is unknown."
      error_in_line = 1
    End
  End
  If error_in_line Then Do
    Say "COPMG014I 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
    Call syscall_cmd "unmount" zfs_name "(mtm_immed)", noexit_on_error
  If error_found Then Do
    final_rc = 8
    If hfs_switched_ro Then Do
      If zfs_usable Then Nop
      Else Do
        Call syscall_cmd "unmount" hfs_name "(mtm_remount)",,
          noexit_on_error
        If OK Then Say "COPMG074W HFS 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 "COPMG125I The migration copy processing was successful",
          "and you can replace the HFS by the zFS yourself later."
      When zfs_exists Then
        Say "COPMG071I Data set" zfs_name "kept as it existed already."
      When zfs_formatted Then
```

```
          Say "COPMG072I Data set" zfs_name "kept as it has been",
            "formatted at least."
        When zfs_created Then Do
          Say "COPMG073I Data set" zfs_name "will be deleted as it",
            "has been defined only."
          "DELETE" "'"||zfs_name||"'"
        End
        Otherwise Nop
      End
    End
    If hfs_temp_mounted Then
      Call Syscall_Cmd "unmount (hfs_name) (mtm_immed)", noexit_on_error
    If curmig_msg Then Do
      If error_found Then Say "COPMG110W One or more errors occurred",
        "during the actual migration process."
      Else Say "COPMG111I No errors have been recognized for the actual",
        "migration process."
    End
    Parse Value "" With hfs_name hfs_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_hfs = "N"
    error_found = 0
    hfs_switched_ro = 0
    hfs_temp_mounted = 0
    zfs_temp_mounted = 0
    zfs_name_same_as_hfs = 0
    zfs_exists = 0
    zfs_created = 0
    zfs_formatted = 0
    zfs_usable = 0
    zfs_mounted = 0
  Return

  Migrate_HFS2zFS:
    Call Msg "OFF"
    hfs_dsn_status = Sysdsn("'"||hfs_name||"'")
    Call Msg msg_save
    Parse Value hfs_dsn_status With invalid_dsn "," .
    Select
      When hfs_dsn_status="DATASET NOT FOUND" Then Do
        Say "COPMG015W The HFS data set cannot be found."
        error_found = 1
      End
      When invalid_dsn="INVALID DATASET NAME" Then Do
        Say "COPMG016W The HFS data set name specified is not valid."
        error_found = 1
      End
      When hfs_dsn_status="OK" Then Nop
      When hfs_dsn_status="UNAVAILABLE DATASET" Then Nop /* expect OK */
      Otherwise Do
```

```
    Say "COPMG017W The HFS data set status is not as expected:"
    Say "COPMG018W" hfs_dsn_status
    error_found = 1
  End
End
If error_found Then Nop /* Skip test whether HFS is mounted already */
Else Do
  Call Syscall_Cmd "statfs (hfs_name) st.", noexit_on_error, no_msgs
  Select
    When OK Then Do
      hfs_devno = st.stfs_fsid
      Call Syscall_Cmd "getmntent hfs." hfs_devno
      read_only = Bitand(D2c(hfs.mnte_mode.1),D2c(mnt_mode_rdonly))
      read_only = C2d(read_only)
      read_write = (read_only=0)
      hfs_mntdir = hfs.mnte_path.1
      hfs_fstype = hfs.mnte_fstype.1
      hfs_mounted = 1
      If hfs_fstype<>"HFS" Then Do
        Say "COPMG090E The data set specified as HFS is not an HFS."
        error_found = 1
      End
    End
    When rc=0 & retval=-1 & errno=79 & errnojr="567002E" Then
      hfs_mounted = 0
    Otherwise Do
      Say "COPMG019W The status of the HFS data set is not expected."
      Say "COPMG020W 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 "COPMG021W The number of volumes specified cannot exceed a",
    "value of" max_no_vols||"."
  error_found = 1
End
If zfs_alloc_space="" Then Do
  Say "COPMG022W 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 "COPMG062W To extend the zFS aggregate the secondary",
    "allocation size cannot be zero."
  error_found = 1
End
If zfs_name=hfs_name Then Do
  zfs_name_same_as_hfs = 1
  If hfs_name_sav="" Then
    hfs_name_sav = Strip(Strip(Left(hfs_name,40)),"T",".")||".SAV"
```

```
  Call Msg "OFF"
  hfss_dsn_status = Sysdsn("'"||hfs_name_sav||"'")
  Call Msg msg_save
  Parse Value hfss_dsn_status With invalid_dsn "," .
  Select
    When hfss_dsn_status="DATASET NOT FOUND" Then Nop /* not used */
    When invalid_dsn="INVALID DATASET NAME" Then Do
      Say "COPMG113W The HFS save name specified is not valid."
      error_found = 1
    End
    When hfs_name_sav=hfs_name Then Do
      Say "COPMG114W The HFS and the HFS save data set cannot have",
        "the same name."
      error_found = 1
    End
    When hfss_dsn_status="OK" | hfss_dsn_status="UNAVAILABLE DATASET"
      Then Do
      Say "COPMG115W The HFS save data set" hfs_name "exists already."
      error_found = 1
    End
    Otherwise Do
      Say "COPMG116W The HFS save data set status is not as expected:"
      Say "COPMG117W" hfss_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 "COPMGO75I As the zFS aggregate name is the same as the HFS",
    "data set a temporary name is used for the zFS."
  Say "COPMGO76I 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_hfs & (zfs_name=hfs_name |,
    zfs_name=hfs_name_sav) Then Do
    Say "COPMG118W The zFS aggregate temporary name cannot be the",
      "same as that of the HFS or the HFS 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 "COPMGO23W 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 "COPMGO24W The zFS aggregate status is not as expected:"
    Say "COPMGO25W" 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
        Say "COPMGO26W The zFS aggregate specified cannot be mounted",
          "already."
      Else Do
        Say "COPMGO27W The data set specified as zFS is mounted",
          "already and is not a zFS."
        Say "COPMGO28W File system type:" zfs_fstype
      End
      Say "COPMGO29W Mount directory :" zfs_mntdir
      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 "COPMGO30W The status of the zFS aggregate is not expected."
      Say "COPMGO31W Rc=" rc "retval=" retval "errno=" errno,
        "errnojr=" errnojr
      error_found = 1
    End
  End /* Select */
End

If error_found Then Do
  Say "COPMGO32W One or more specification errors found for current",
    "migration."
  Say "COPMGO33E Skipping HFS" hfs_name||"..."
  Return
End

If stop_after_syntax_check Then Do
  Say "COPMGO34I No specification errors found for current migration."
  Return
End

If hfs_mounted Then Do
  If read_write Then Do
    Call Syscall_Cmd "unmount (hfs_name)" mtm_remount,,
      noexit_on_error
    If OK Then hfs_switched_ro = 1
    Else Do
      Say "COPMGO36E HFS mount mode could not be switched to R/O."
      error_found = 1
      Return
    End
  End
End
```

```
Else Do
  Call Syscall_Cmd "mount (tmp_hfs)" hfs_name "HFS (mtm_rdonly)",,
    noexit_on_error
  If OK Then Do
    hfs_temp_mounted = 1
    hfs_mntdir = tmp_hfs
    Call Syscall_Cmd "statvfs (tmp_hfs) st."
    hfs_devno = st.stfs_fsid
    Call Syscall_Cmd "getmntent hfs." hfs_devno
    If hfs.mnte_fstype.1<>"HFS" Then Do
      Say "COPMGO91E The data set specified as HFS is not an HFS."
      error_found = 1
      Return
    End
  End
  Else Do
    Say "COPMGO37E HFS file system could not be mounted read-only."
    error_found = 1
    Return
  End
End

If zfs_exists 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 "COPMGO38E zFS aggregate could not be mounted."
    error_found = 1
    Return
  End
  If stop_after_fss_mounted Then Do
    Say "COPMGO92I HFS and existing zFS are or could be mounted."
    Return
  End
End
Else Do
  If stop_after_fss_mounted Then Do
    Say "COPMGO93I HFS is mounted, 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 "COPMG086I Defining zFS aggregate" zfs_name "..."
  Call Shell_Cmd
  If shell_rc<>0 Then Do
    Say "COPMG039E zFS aggregate could not be defined correctly."
    error_found = 1
    Return
  End
  Else zfs_created = 1

  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 "COPMG087I Formatting zFS aggregate" zfs_name "..."
  Call Shell_Cmd
  If shell_rc<>0 Then Do
    Say "COPMG040E 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 "COPMG050E 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 "COPMG088I 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 "COPMG051E zFS aggregate could not be grown correctly."
      error_found = 1
      Return
    End
  End j
End
```

```
If stop_after_zfs_is_formatted Then Do
  Say "COPMGO94I HFS is mounted, zFS aggregate has been created."
  Say "COPMGO35I HFS data set name=" hfs_name
  Return
End

If target_zfs_must_be_empty 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 "COPMGO95E Unexpected zFS rddir error, errno=" errno,
        "errnojr=" errnojr
      error_found = 1
    End
  End
  If zfs_not_empty Then Do
    Say "COPMGO96E 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 (hfs_mntdir)"
Say "COPMGO54I Now starting copy processing..."
Select
  When migrtool="COPYMIGR" Then Do
    parms = tmp_dir"/"||"copymigr -rw"||verbose "-peW -XCM ." tmp_zfs
    parm.0 = Words(parms)
    Do j=1 To parm.0
      Parse Var parms parm.j parms
    End
    Call Shell_Cmd
    If shell_rc<>0 Then Do
      Say "COPMGO52E copymigr processing ended with rc=" shell_rc||"."
      error_found = 1
      Return
    End
  End
  When migrtool="COPYPAX" Then Do
    If Symbol("pax_is_r7")="LIT" Then Do
      pax_is_r7 = 0
      not_pax_rw_found = 1
```

```
    Call Bpxwunix "pax -Z", , , "stderr.", "__environment."
    Do ps=1 To stderr.0 While not_pax_rw_found
      pax_syntax_line = stderr.ps
      If Pos("pax -r -w",pax_syntax_line)<>0 Then Do
        not_pax_rw_found = 0
        Parse Var pax_syntax_line "AD"x pax_rw_opts "BD"x /* [] */
        If Pos("C",pax_rw_opts)<>0 Then pax_is_r7 = 1
      End
    End
    pax_is_pre_r7 = (pax_is_r7=0)
  End
  parm.1 = "sh"
  parm.2 = "-c"
  If pax_is_r7 Then
    parm.3 = "pax -rw"||verbose "-peW -XCM ." tmp_zfs
  Else
    parm.3 = "pax -rw"||verbose "-pe -X ." tmp_zfs
  parm.0 = 3
  Call Shell_Cmd
  If shell_rc<>0 Then Do
    Say "COPMG053E copypax processing ended with rc=" shell_rc||"."
    error_found = 1
    Return
  End
  Else If pax_is_pre_r7 Then Do
    Call Syscall_Cmd "statfs (hfs_name) st."
    hfs_devno = st.stfs_fsid
    Call Syscall_Cmd "getmntent mnt."
    Do kk=1 To mnt.0
      If mnt.mnte_pardev.kk=hfs_devno Then Do
        msg_zfs_mp = Substr(mnt.mnte_path.kk,Length(hfs_mntdir)+2)
        new_zfs_mp = tmp_zfs || "/" || msg_zfs_mp
        Call Syscall_Cmd "stat (new_zfs_mp) st.", noexit_on_error,,
          no_msgs
        Select
          When errno=81 Then Do
            Call Syscall_Cmd "mkdir (new_zfs_mp) 700"
            Say "COPMG066I Mount point" msg_zfs_mp "has been",
              "created relative to the zFS top directory."
          End
          When OK Then
            Say "COPMG067I Mount point" msg_zfs_mp "exists",
              "already relative to the zFS top directory."
          Otherwise Do
            Say "COPMG068E Unexpected error on ""stat"" command..."
            Say "COPMG069I Rc=" rc "Retval=" retval "Errno=" errno,
              "Errnojr=" errnojr
          End
        End
      End
    End
    Drop mnt.
  End
End
When migrtool="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
      error_found = 1
      Return
    End
  End
  Otherwise Nop /* this will be never selected */
End
Say "COPMG070I Copy processing has been ended..."
zfs_usable = 1
Call Syscall_Cmd "chdir /"

If hfs_temp_mounted Then Do
  Call Syscall_Cmd "unmount" hfs_name "(mtm_normal)", noexit_on_error
  If OK Then hfs_temp_mounted = 0
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
End
hfs_still_mounted = hfs_mounted
If zfs_replaces_hfs="Y" & hfs_mounted Then Do
  Call Syscall_Cmd "statfs" hfs_name "st."
  hfs_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
  hfs_fss = 0
  Call Build_List_Of_Subfss hfs_devno
  do_unmounting = 1
  Do j=hfs_fss By -1 To 1
    mnt_index = hfs_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 "COPMG123E 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 "COPMG124E 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 = hfs_fss+1
    Do j=hfs_fss By -1 To 1 While do_unmounting
      mnt_index = hfs_fs.j
      fsname = Strip(mnt.mnte_fsname.mnt_index)
      fspath = mnt.mnte_path.mnt_index
      Say "COPMG105I 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 "COPMG082E Error ocurred, stopping unmount processing..."
        error_found = 1
        do_unmounting = 0
      End
    End
    If do_unmounting Then Do
      Say "COPMG106I Unmounting" hfs_name "..."
      Call Syscall_Cmd "unmount" hfs_name "(mtm_immed)",,
        noexit_on_error
      If OK Then Do
        rmnt_index = 0
        hfs_switched_ro = 0
        Do kk=1 By 1
          Call Bpxwdyn "ALLOC DSN("||hfs_name||") OLD"
          If result=0 Then Do
            Call Bpxwdyn "FREE Dsn("||hfs_name||")"
            hfs_still_mounted = 0
            Leave kk
          End
          Else Do
            If kk=30 Then Do
              Say "COPMG089E HFS 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 "COPMG083E Error with HFS, stopping unmount processing..."
        error_found = 1
      End
    End
  End

  If zfs_name_same_as_hfs Then Do
    If hfs_still_mounted Then Do
      Say "COPMG081E HFS and zFS data sets cannot be renamed as the",
        "HFS is still mounted."
      error_found = 1
      Return
```

```
  End
  new_zfs_name = hfs_name
  new_hfs_name = hfs_name_sav
  "RENAME '"||hfs_name||"' '"||new_hfs_name||"'"
  If rc=0 Then Do
    hfs_name = new_hfs_name
    Say "COPMG103I The HFS data set has been renamed to" hfs_name||"."
    "ALTER '"||zfs_name||"' NEWNAME('"||new_zfs_name||"')"
    If rc=0 Then Do
      zfs_name = new_zfs_name
      Say "COPMG104I 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 "COPMG119I The zFS DATA part has been",
        "renamed to or is named" new_lds_data||"."
      Else Say "COPMG077W The zFS DATA part could not be renamed,",
        "but the zFS aggregate is usable correctly."
    End
    Else Do
      Say "COPMG078E The zFS aggregate name could not be renamed,",
        "but it is usable correctly."
      Say "COPMG120I Therefore, mounting the zFS aggregate with its",
        "temporary name" zfs_name
      error_found = 1
    End
  End
  Else Do
    Say "COPMG079E The HFS file system could not be renamed, but the",
      "zFS aggregate is usable correctly."
    Say "COPMG121I Therefore, mounting the zFS aggregate with its",
      "temporary name" zfs_name
    error_found = 1
  End
End
```

```
If zfs_replaces_hfs="Y" & hfs_mounted Then Do
  If rmnt_index=0 Then Do
    m. = ""
    m.mnte_filetag = hfs.mnte_filetag.1
    m.mnte_path = hfs.mnte_path.1
    If zfs_usable Then Do
      Say "COPMG107I Mounting zFS" zfs_name "at" m.mnte_path "now..."
      m.mnte_fsname = zfs_name
      m.mnte_parm = "aggrgrow"
      m.mnte_fstype = "ZFS"
    End
    Else Do
      Say "COPMG108I Mounting HFS" hfs_name "at" m.mnte_path,
        "again..."
      m.mnte_fsname = hfs_name
      m.mnte_parm = hfs.mnte_parm.1
      m.mnte_fstype = hfs.mnte_fstype.1
    End
    m.mnte_mode = hfs.mnte_mode.1
    m.mnte_syslist = hfs.mnte_syslist.1
    If hfs.mnte_sysname.1<>local_system Then
      m.mnte_sysname = hfs.mnte_sysname.1
    Call Syscall_Cmd "mount m.", noexit_on_error
    If not_OK Then Do
      Say "COPMG085E 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 hfs_fss
    mnt_index = hfs_fs.j
    fsname = Strip(mnt.mnte_fsname.mnt_index)
    fspath = mnt.mnte_path.mnt_index
    Say "COPMG109I 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 "COPMG084E 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 hfs_temp_mounted Then Do
        Call Syscall_Cmd "unmount (hfs_name)" mtm_normal,,
          noexit_on_error
        If OK Then hfs_temp_mounted = O
      End
      If zfs_temp_mounted Then Do
        Call Syscall_Cmd "unmount (zfs_name)" mtm_normal,,
          noexit_on_error
        If OK Then zfs_temp_mounted = O
      End
      Address SYSCALL  "rmdir tmp_hfs"
      Address SYSCALL  "rmdir tmp_zfs"
      Address SYSCALL  "unlink copymigr"
      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=O 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 "COPMGO97W REXX error in sourceline" sigl "of" myname
  Say "COPMGO98I Line" sigl||":" Strip(Sourceline(sigl))
  Say "COPMGO99E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
```

```
  retc = rc
  Say "COPMG100W REXX error in sourceline" sigl "of" myname
  Say "COPMG101I Line" sigl||":" Strip(Sourceline(sigl))
  Say "COPMG102E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Build_List_Of_Subfss: Procedure Expose mnt. hfs_fss hfs_fs. subfss.,
    mnte_dev
  Parse Arg devno
  subfss = Strip(subfss.devno)
  Do While subfss<>""
    Parse Var subfss mnt_index subfss
    hfs_fss = hfs_fss+1
    hfs_fs.hfs_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 "COPMG055I Signal SIGTERM received, terminating..."
        Say "COPMG056I 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 "COPMGO57I Signal SIGALRM received or timeout occurred,",
      "terminating..."
    Say "COPMGO58I 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 "COPMGO59I Too many lines have been created, aborting..."
    Say "COPMGO60I 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 "COPMG112E Severe error occurred, stopping..."
        Call Final_Exit 8
      End
    End
  End
Return

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

A.2 CPYMHELP

The REXX procedure CPYMHELP displays help information for migration and copy processing.

Example A-2 displays the contents of COPYMIGR.

Example: A-2 CPYMHELP REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: MIGRHELP (Next free MSGNO: Say "MIGHP001x ...")      */
/*  Description: Help information for migration processing            */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: migrhelp                                             */
/**********************************************************************/

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

Trace O
Parse Source . . myname .

Parse Value "" With mighpdsn mighpddn 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(MIGHPDSN) RTDDN(MIGHPDDN) NEW",
    "MSG(MSG.) LRECL(80) RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
  If result<>0 | mighpddn="" Then Do
    If msg.0=0 Then
      Say "Disp_Msg MIGHP001E 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" mighpddn "(STEM HLINE. FINIS"
  If rc<>0 Then Do
    Call Disp_Msg "MIGHP002E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

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

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

  Call Final_Exit 0
Exit 9999

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

Syntax_Error:
  Say "MIGHP003W REXX error in sourceline" sigl "of" myname
  Say "MIGHP004I Line" sigl||":" Strip(Sourceline(sigl))
  Say "MIGHP005E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999
```

```
Novalue_Error:
  retc = rc
  Say "MIGHPO06W REXX error in sourceline" sigl "of" myname
  Say "MIGHPO07I Line" sigl||":" Strip(Sourceline(sigl))
  Say "MIGHPO08E 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 */

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

# ================================================================== #
#   MIGRDATA Help Information                                        #
# ================================================================== #


1. Migration Processing JCL Information
---------------------------------------

a) Job Variables

SET MGRTOOL=          - This is the copy utility to be used; if you are
                        running at level z/OS V1R7 or above using
                        COPYPAX is the suggested method.
COPYMIGR:             > Use copy tool provided with migration tool
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 COPYMIGR or COPYPAX are set.

SET DEFMIGR=          - Member or data set containing the migration
                        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 HFS and if existing the zFS is/are mounted; value
  must be specified as N or Y.


STOP_AFTER_ZFS_IS_FORMATTED:

  Force stopping when the 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 HFS by 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. Migration Definition Statements
----------------------------------

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

HFS_NAME                     - Name of the HFS data set to be migrated

#HFS_#_VOLUMES=              - FYR: Number of HFS volumes

#HFS_DEVICE_TYPE=            - FYR: HFS DASD device type

#HFS_ALLOC_UNIT=             - FYR: HFS DASD allocation unit

#HFS_ALLOC_SPACE=            - FYR: HFS primary and secondary allocation

#HFS_TOTAL_UNITS_ALLOCATED=  - FYR: HFS total number of units allocated

#HFS_TOTAL_UNITS_%USED=      - FYR: HFS percentage used of space

#HFS_DATACLASS=              - FYR: HFS data class

#HFS_MGMNTCLASS=             - FYR: HFS management class

#HFS_STORCLASS=              - FYR: HFS storage class

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

ZFS_NAME_TMP=                - Suggested name for zfs used initially;
                               this is only used if HFS and 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_UNIT               - zFS allocation unit; you may specify
                               CYLINDERS (CYL) or TRACKS(TRK) only

ZFS_ALLOC_SPACE              - zFS primary and secondary allocation

ZFS_ALLOC_NUM_CAND_VOLUMES   - Number of zFS candidate volumes

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_HFS             - This value specifies whether the zFS
                               file system should replace the HFS; you
                               must specify Y or N.

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

A.3 DEFMHELP

The REXX procedure DEFMHELP displays help information for creating migration definition statements.

Example A-3 displays the contents of DEFMHELP.

Example: A-3 DEFMHELP REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DEFMHELP (Next free MSGNO: Say "MIGHP001x ...")      */
/*  Description: Help information for migration processing            */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: migrhelp                                             */
/**********************************************************************/
```

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

Trace O
Parse Source . . myname .

Parse Value "" With mighpdsn mighpddn 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.O = O
  Call Bpxwdyn "ALLOC RTDSN(MIGHPDSN) RTDDN(MIGHPDDN) NEW",
    "MSG(MSG.) LRECL(8O) RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
  If result<>O | mighpddn="" Then Do
    If msg.O=O Then
      Say "Disp_Msg MIGHPOO1E Error allocating temporary file"
    Else Do i=1 to msg.O
      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" mighpddn "(STEM HLINE. FINIS"
  If rc<>O Then Do
    Call Disp_Msg "MIGHPOO2E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

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

  Address ISPEXEC "BROWSE DATAID("||mighpdid||")"
  If rc<>O Then Do
    Call Disp_Msg Strip(zerrmsg) Strip(zerrlm) "ISPF gave rc" rc "on",
      "browsing data ID for MIGHPDDN."
```

```
    Call Final_Exit 8
  End

  Call Final_Exit 0
Exit 9999

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

Syntax_Error:
  Say "MIGHP003W REXX error in sourceline" sigl "of" myname
  Say "MIGHP004I Line" sigl||":" Strip(Sourceline(sigl))
  Say "MIGHP005E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  retc = rc
  Say "MIGHP006W REXX error in sourceline" sigl "of" myname
  Say "MIGHP007I Line" sigl||":" Strip(Sourceline(sigl))
  Say "MIGHP008E 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 */

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

# ================================================================ #
#   DEFMIGR Help Information                                       #
# ================================================================ #


1. DEFMIGR Control Statements
-----------------------------

ZFS_MIGRATE_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.MIGRATE.DEFINE with "hlq"
being your own userid.


ZFS_MIGRATE_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_MIGRATE_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.


HFS_DATA_SETS_TO_MIGRATE=

This statement defines the HFS data set name list to be migrated. 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: /HFS/ZFS/ . If you do not specify anything the new zFS
aggregate will get the same name as the HFS data set before. The HFS
will get renamed (with ".SAV" appended at the end by default).


2. DEFMIGR Line Commands
------------------------

DEFMHELP or DH  - displays this help information
DM              - displays the last set of messages shown again
CPYMHELP or CH  - displays help information for the later migration job
REFRESH or RF   - displays the current source statements refreshed

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

A.4 DEFMIGR

The REXX procedure DEFMIGR is used in the TSO/ISPF foreground to define migration 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 migration processing job.

Example A-4 displays the contents of DEFMIGR.

Example: A-4 DEFMIGR REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DEFMIGR  (Next free MSGNO: DEFMG053x)                 */
/*  Description: Migrate HFS data sets to zFS compat mode aggregates  */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: defmigr                                               */
/*********************************************************************/

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

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

msg_save = Msg()
msg_say = 0
defmcopn = 0
listaccd = 0
no_add2imsg = 0
foreground = (Sysvar("SYSENV")="FORE")
background = (foreground=0)
Parse Value "" With defmgddn defmmddn defmcddn zerrmsg zerrlm
no_sms_class = "**None**"

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,
    "DEFMG001E The procedure needs to be run in foreground."
  Call Final_Exit 8
End

msg.0 = 0
Call Bpxwdyn "ALLOC RTDDN(DEFMGDDN) NEW MSG(MSG.) LRECL(80)",
  "RECFM(F,B) SPACE(1,1) TRACKS"||tmp_unit
If result<>0 | defmgddn="" 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 "DEFMG002E 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(DEFMMDDN) NEW MSG(MSG.) LRECL(84)",
  "RECFM(V,B) SPACE(1,1) TRACKS"||tmp_unit
If result<>0 | defmmddn="" 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 "DEFMG003E 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 "DEFMG004W 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.MIGRATE.DEFINE"
If zfsm$dsp="" Then zfsm$dsp = "APPEND"
cmntline = Left("#" Copies("----",17) "#",80)
blnkline = Left("",80)
cline.1  = cmntline
cline.2  = Left("ZFS_MIGRATE_DEFINE_DSN="||zfsm$dsn,80)
cline.3  = Left("ZFS_MIGRATE_DEFINE_MBR="||zfsm$mbr,80)
cline.4  = Left("ZFS_MIGRATE_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("HFS_DATA_SETS_TO_MIGRATE="||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" defmgddn "(STEM CLINE. FINIS"
If rc<>0 Then Do
  Call Disp_Msg "DEFMGOO5E 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(DEFMGDID) DDNAME("||defmgddn||")"
If rc<>0 Then Do
  If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
  Else message = ""
  Call Disp_Msg "DEFMGOO6E ISPF service LMINIT gave RC" rc "on",
    "generating a data ID DEFMGDID."||message
  Call Final_Exit 8
End

Call Disp_Msg "Enter DH to show migration 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("||defmgdid||") MACRO(DMGP$MAC)"
  Select
    When rc=0 Then Nop
    When rc=4 Then Do
      Call Disp_Msg "DEFMGOO7W Define processing canceled..."
      Call Final_Exit 4
    End
    Otherwise Do
      Call Disp_Msg "DEFMGOO8E Error on editing occurred,",
        "Rc("||retc||")"
      Call Final_exit 8
    End
  End

  "EXECIO * DISKR" defmgddn "(STEM CLINE. FINIS"
  If rc<>0 Then Do
    Call Disp_Msg "DEFMGOO9E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  Parse Value "" With defmigr_dsn defmigr_mbr defmigr_dsp,
    zfs_def_dataclass zfs_def_mgmntclass zfs_def_storclass,
    hfs_dsns_to_migr hfs_chg_str zfs_chg_str hfs_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_MIGRATE_DEFINE_DSN" Then
```

```
      Parse Var cntl_value defmigr_dsn .
    When cntl_parm="ZFS_MIGRATE_DEFINE_MBR" Then
      Parse Var cntl_value defmigr_mbr .
    When cntl_parm="ZFS_MIGRATE_DEFINE_DSP" Then
      Parse Var cntl_value defmigr_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="HFS_DATA_SETS_TO_MIGRATE" Then
      hfs_dsns_to_migr = Strip(hfs_dsns_to_migr cntl_value)
    When cntl_parm="ZFS_AGGRNAME_CHANGE_CMD" Then
      Parse Var cntl_value hfs_dsns_chcmd .
    Otherwise Do
      Call WMsg "DEFMG010E 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
defmigr_dsn = Strip(defmigr_dsn,"B","'")
If defmigr_dsn="" Then rc=1
Else
  Address ISPEXEC "ISPEXEC DSINFO DATASET('"||defmigr_dsn||"')"
Select
  When rc=0 Then Do
    Select
      When Wordpos(zdsorg,"PS PO")=0 Then Do
        Call WMsg "DEFMG011E Migration control data set must be a PS",
          "or PO data set."
        no_error_found = 0
      End
      When zdsdsnt="HFS" Then Do
        Call WMsg "DEFMG012E Migration 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 "DEFMG013E Migration control data set must be FB80",
          "or VB80."
        no_error_found = 0
      End
      When zdsorg="PS" & defmigr_mbr<>"" Then Do
        Call WMsg "DEFMG014E 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 defmigr_mbr="" Then Do
          mbr.0 = 0
```

```
        Call OUTTRAP "MBR."
        Address TSO "LISTDS ('"||defmigr_dsn||"') MEMBERS"
        retc = rc
        Call OUTTRAP "OFF"
        If retc<>0 Then Do
          message = "DEFMG015E 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
          defmigr_mbr = "WORK"||Right(i,2,"0")
          If mname_free.defmigr_mbr Then Leave i
          If i=99 Then Do
            Call WMsg "DEFMG016E No free number available for",
              "WORKxx member name between 01 and 99."
            defmigr_mbr = ""
            member_ok = 0
            no_error_found = 0
          End
        End
      End
      If member_ok Then Do
        Call Msg "OFF"
        dm_status = Sysdsn("'"||defmigr_dsn||"("||defmigr_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 "DEFMG017E 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 "DEFMGO18E Data set name for ZFS_MIGRATE_DEFINE_DSN",
      "cannot be BLANK."
    no_error_found = 0
  End
  When rc=8 Then Do
    Call WMsg "DEFMGO19E Data set" defmigr_dsn "could not be found."
    no_error_found = 0
  End
  Otherwise Do
    Call WMsg "DEFMGO20I ISPF service DSINFO gave rc" rc||"."
    no_error_found = 0
  End
End
Select
  When Left(defmigr_dsp,1)="A" Then defmigr_dsp = "APPEND"
  When Left(defmigr_dsp,1)="R" Then defmigr_dsp = "REPLACE"
  Otherwise Do
    Call WMsg "DEFMGO21E The value for ZFS_MIGRATE_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 "DEFMGO22E The value for ZFS_DEF_DATACLASS is invalid."
  no_error_found = 0
End
If IsNotValid(zfs_def_mgmntclass) Then Do
  Call WMsg "DEFMGO23E The value for ZFS_DEF_MANAGEMENTCLASS is",
    "invalid."
  no_error_found = 0
End
If IsNotValid(zfs_def_storclass) Then Do
  Call WMsg "DEFMGO24E The value for ZFS_DEF_STORAGECLASS is invalid."
  no_error_found = 0
End
If hfs_dsns_to_migr="" Then Do
  Call WMsg "DEFMGO25E The value for HFS_DATA_SETS_TO_MIGRATE cannot",
    "be BLANK."
  no_error_found = 0
End
Do ww = 1 To Words(hfs_dsns_to_migr)
  If Left(Word(hfs_dsns_to_migr,ww),1)="*" Then Do
    Call WMsg "DEFMGO51E A word within HFS_DATA_SETS_TO_MIGRATE",
      "cannot start with '*'."
    no_error_found = 0
  End
  If Left(Word(hfs_dsns_to_migr,ww),1)="%" Then Do
    Call WMsg "DEFMGO49E A word within HFS_DATA_SETS_TO_MIGRATE",
      "cannot start with '%'."
    no_error_found = 0
  End
End ww
Parse Var hfs_dsns_chcmd "/" hfs_chg_str "/" zfs_chg_str "/",
```

```
  hfs_dsns_junk
Select
  When hfs_dsns_chcmd="" Then Nop
  When Left(hfs_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(hfs_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(hfs_dsns_chcmd,2))=Length(hfs_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 hfs_dsns_junk<>"" Then Do
    Call WMsg "The value for ZFS_AGGRNAME_CHANGE_CMD must contain",
      "exactly three '/' characters."
    no_error_found = 0
  End
  When hfs_chg_str="" Then Do
    Call WMsg "DEFMGO26E The value for the HFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD cannot be BLANK."
    no_error_found = 0
  End
  When IsInvalidStr(hfs_chg_str) Then Do
    Call WMsg "DEFMGO27E The value for the HFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD contains invalid characters."
    no_error_found = 0
  End
  When IsInvalidStr(zfs_chg_str) Then Do
    Call WMsg "DEFMGO28E The value for the zFS source string in",
      "ZFS_AGGRNAME_CHANGE_CMD contains invalid characters."
    no_error_found = 0
  End
  Otherwise Nop
End
zfsm$dsn = defmigr_dsn
zfsm$mbr = defmigr_mbr
zfsm$dsp = defmigr_dsp
zfsm$dcl = zfs_def_dataclass
zfsm$mcl = zfs_def_mgmntclass
zfsm$scl = zfs_def_storclass
zfsm$hdm = hfs_dsns_to_migr
If hfs_dsns_chcmd="" Then zfsm$ccm = ""
Else zfsm$ccm = "/"||hfs_chg_str||"/"||zfs_chg_str||"/"
Address ISPEXEC "VPUT (ZFSM$DSN ZFSM$MBR ZFSM$DSP ZFSM$DCL ZFSM$MCL",
  "ZFSM$SCL ZFSM$HDM ZFSM$CCM)"
If rc<>0 Then
  Call WMsg "DEFMGO29W ISPF Service VPUT gave rc" rc||"."
Drop zfsm$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 defmigr_mbr="" Then Do
  defmigr_ctl = defmigr_dsn
  If defmigr_dsp="APPEND" Then acc_mode = "MOD"
  Else acc_mode = "OLD"
End
Else Do
  defmigr_ctl = defmigr_dsn||"("||defmigr_mbr||")"
  acc_mode = "SHR"
End
msg.0 = 0
Call Bpxwdyn "ALLOC DSN("||defmigr_ctl||") RTDDN(DEFMCDDN)" acc_mode,
  "MSG(MSG.)"
If result<>0 | defmcddn="" 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 "DEFMG030E Error allocating DEFMIGR 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 defmigr_dsp="APPEND" & defmigr_mbr<>"" Then Do
  If Sysdsn("'"||defmigr_ctl||"'")="MEMBER NOT FOUND" Then
    migrcntl.0 = 0
  Else Do
    "EXECIO * DISKR" defmcddn "(STEM MIGRCNTL. FINIS"
    If rc<>0 Then Do
      Call Disp_Msg "DEFMG031E EXECIO error occurred, Rc("||rc||")"
      Call Final_Exit 8
    End
  End
End
Else migrcntl.0 = 0
clines = migrcntl.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 = "MIGRATE CONTROL DEFINITIONS, CREATED" yyyy||"-"||mm||"-"||dd,
  Time()
Call WMctl Overlay(cline,"#" Left("",68) "#",3)
Call WMctl cmnt_line
Do i=1 To Words(hfs_dsns_to_migr)
  dsnp = Strip(Word(hfs_dsns_to_migr,i),"B","'")
  Call WMsg "DEFMGO50I Searching for data sets" dsnp "..."
  Address ISPEXEC "LMDINIT LISTID(DEFMDDID) LEVEL("||dsnp||")"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFMGO32E ISPF service LMDINIT gave RC" rc "on",
      "generating a data ID DEFMDDID."||message
    Call Final_Exit 8
  End

  dsn = ""
  Do Forever
    Address ISPEXEC "LMDLIST LISTID("||defmddid||") DATASET(DSN)",
      "STATS(YES) OPTION(LIST)"
    Select
      When rc=0 Then Do
        listaccd = 1
        If zdlmigr="YES" Then Call WMsg,
          "DEFMGO33W 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 "DEFMGO48I Data set" dsn "will be examined."
          End
          Else Call WMsg,
            "DEFMGO34I Data set" dsn "has been processed already"
        End
      End
      When rc=4 Then Do
        Call WMsg "DEFMGO35E No data sets match """||dsnp||"""."
        Leave
      End
      When rc=8 Then Leave /* All data set names processed */
      Otherwise Do
        Call Disp_Msg,
          "DEFMGO36E 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("||defmddid||") OPTION(FREE)"
    If rc<>0 Then Do
      If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
      Else message = ""
      Call Disp_Msg "DEFMG037E ISPF service LMDLIST gave RC" rc "on",
        "freeing storage for data ID DEFMDDID."||message
      Call Final_Exit 8
    End
    listaccd = 0
  End
  Address ISPEXEC "LMDFREE LISTID("||defmddid||")"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFMG038E ISPF service LMDFREE gave RC" rc "on",
      "freeing data ID DEFMDDID."||message
    Call Final_Exit 8
  End
  Drop defmddid
End i

nn = 30
Call WMctl " "
hfss = 0
Do i=1 To dsns
  dsname = "'"||dsn.i||"'"
  Address ISPEXEC "ISPEXEC DSINFO DATASET("||dsname||")"
  Select
    When rc=8 Then Do
      Call WMsg "DEFMG039I Data set" dsn.i "could not be found or is",
        "not an HFS."
      Iterate i
    End
    When rc<>0 Then Do
      Call WMsg "DEFMG040E ISPF Service DSINFO gave rc" rc||"."
      Call WMsg "DEFMG041W Data set" dsn.i "has been skipped therefore."
      Iterate i
    End
    When zdsdsnt<>"HFS" Then Do
      Call WMsg "DEFMG042I Data set" dsn.i "is not an HFS and will be",
        "skipped."
      Iterate i
    End
    Otherwise Nop /* OK */
  End

  hfss = hfss+1
  hfs_name = Strip(dsname,"B","'") /* HFS data set name     */
  hfs_firstvol = zdsvol            /* First volume serial   */
  hfs_#volumes = Strip(zds#vols)   /* Number of volumes     */
  hfs_devt = Strip(zdsdevt)        /* Device type           */
  hfs_primu = zdsspc               /* Primary space units   */
  hfs_prima = C2n(zds1ex)          /* Primary space alloc   */
  hfs_seca = C2n(zds2ex)           /* Secondary space alloc */
```

```
hfs_spaca = C2n(zdstota)         /* Allocated space units */
hfs_percu = Strip(zdsperu)       /* Percent used          */
hfs_mc = Strip(zdsmc)            /* Management class      */
If hfs_mc=no_sms_class Then hfs_mc = ""
hfs_sc = Strip(zdssc)            /* Storage class         */
If hfs_sc=no_sms_class Then hfs_sc = ""
hfs_dc = Strip(zdsdc)            /* Data class            */
If hfs_dc=no_sms_class Then hfs_dc = ""
hfs_name_sav = Strip(Strip(Left(hfs_name,40)),"T",".")||".SAV"
                                 /* Name for the bkup HFS */
zfs_name = hfs_name              /* zFS LDS name          */
If hfs_dsns_chcmd<>"" Then Do
  hfs_chg_len = Length(hfs_chg_str)
  zfs_chg_len = Length(zfs_chg_str)
  pos_zfs = 1
  Do Forever
    nxt_pos = Pos(hfs_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+hfs_chg_len)
    pos_zfs = nxt_pos+zfs_chg_len
  End
End
zfs_name_tmp = Strip(Strip(Left(zfs_name,40)),"T",".")||".TMP"
                                 /* zFS initial temp name */
zfs_#volumes = hfs_#volumes      /* Number of volumes     */
zfs_primu = hfs_primu            /* Primary space units   */
zfs_prima = hfs_prima            /* Primary space alloc   */
zfs_seca = hfs_seca              /* Secondary space alloc */
If zfs_def_mgmntclass<>"" Then   /* Management class      */
  zfs_mc = zfs_def_mgmntclass
Else zfs_mc = hfs_mc
If zfs_def_storclass<>"" Then    /* Storage class         */
  zfs_sc = zfs_def_storclass
Else zfs_sc = hfs_sc
If zfs_def_dataclass<>"" Then    /* Data class            */
  zfs_dc = zfs_def_dataclass
Else zfs_dc = hfs_dc
zfs_volumes = ""                 /* List of volumes       */

Select
  When hfs_#volumes>1 & hfs_percu>90 Then
    zfs_#volumes = hfs_#volumes+1
  When hfs_percu>75 Then zfs_prima = Format(hfs_prima*1.1,,0)
  When hfs_percu>90 Then zfs_prima = Format(hfs_prima*1.2,,0)
  Otherwise Nop
End
Select
  When zfs_primu="CYLINDER" Then
    zfs_primu = "CYLINDERS"
  When zfs_primu="TRACK" Then Do
    If zfs_prima<6 Then zfs_prima = 6
    zfs_primu = "TRACKS"
  End
  When zfs_primu="BYTE" Then Do
```

```
    zfs_prima = zfs_prima%(720*1024)+1
    zfs_seca = zfs_seca%(720*1024)+1
    zfs_primu = "CYLINDERS"
  End
  When zfs_primu="BLOCK" Then Do
    zfs_prima = zfs_prima%180+1
    zfs_seca = zfs_seca%180+1
    zfs_primu = "CYLINDERS"
  End
  When zfs_primu="KILOBYTE" Then Do
    zfs_prima = zfs_prima%720+1
    zfs_seca = zfs_seca%720+1
    zfs_primu = "CYLINDERS"
  End
  When zfs_primu="MEGABYTE" Then Do
    zfs_prima = zfs_prima*64%45+1
    zfs_seca = zfs_seca*64%45+1
    zfs_primu = "CYLINDERS"
  End
  Otherwise Nop
End
If zfs_primu<>"CYLINDERS" & zfs_primu<>"TRACKS" Then Do
  Call WMsg "DEFMG052W You must choose CYLINDERS or TRACKS as the",
    "unit type."
End

zfs_sec_allocs = (hfs_spaca-hfs_prima)%Max(hfs_seca,1)
If hfs_#volumes=1 Then Do
  zfs_prima_red = (zfs_prima+zfs_sec_allocs*zfs_seca)*hfs_percu%100
  zfs_prima = Max(zfs_prima,zfs_prima_red)
  zfs_sec_allocs = 0
  zfs_volumes = hfs_firstvol
End

/* If zds#vols>1: SMS classes should be set: zfs_mc zfs_sc zfs_dc */
Call WMctl cmnt_line
Call WMctl Left("HFS_NAME=",nn) hfs_name
Call WMctl cmnt_line
Call WMctl Left("#HFS_#_VOLUMES=",nn) hfs_#volumes
Call WMctl Left("#HFS_DEVICE_TYPE=",nn) hfs_devt
Call WMctl Left("#HFS_ALLOC_UNIT=",nn) hfs_primu
Call WMctl Left("#HFS_ALLOC_SPACE=",nn) hfs_prima hfs_seca
Call WMctl Left("#HFS_TOTAL_UNITS_ALLOCATED=",nn) hfs_spaca
Call WMctl Left("#HFS_TOTAL_UNITS_%USED=",nn) hfs_percu
Call WMctl Left("#HFS_DATACLASS=",nn) hfs_dc
Call WMctl Left("#HFS_MGMNTCLASS=",nn) hfs_mc
Call WMctl Left("#HFS_STORCLASS=",nn) hfs_sc
Call WMctl Left("HFS_NAME_SAV=",nn) hfs_name_sav
Call WMctl Left("ZFS_NAME_TMP=",nn) zfs_name_tmp
Call WMctl Left("ZFS_NAME=",nn) zfs_name
Call WMctl Left("#ZFS_#_VOLUMES=",nn) zfs_#volumes
Call WMctl Left("ZFS_VOLUMES=",nn) zfs_volumes
Call WMctl Left("ZFS_ALLOC_UNIT=",nn) zfs_primu
Call WMctl Left("ZFS_ALLOC_SPACE=",nn) zfs_prima zfs_seca
Call WMctl Left("ZFS_ALLOC_NUM_CAND_VOLUMES=",nn) "0"
```

```
  Call WMctl Left("ZFS_ALLOC_NUM_SEC_ALLOCS=",nn) zfs_sec_allocs
  Call WMctl Left("ZFS_DATACLASS=",nn) zfs_dc
  Call WMctl Left("ZFS_MGMNTCLASS=",nn) zfs_mc
  Call WMctl Left("ZFS_STORCLASS=",nn) zfs_sc
  Call WMctl Left("ZFS_REPLACES_HFS=",nn) "Y" /* Yes or No */
  Call WMctl " "
End i

If defmigr_mbr<>"" Then Do
  Address ISPEXEC "LMINIT DATAID(DEFMCDID) DATASET('"||defmigr_dsn||"')"
  If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFMGO43E ISPF service LMINIT gave RC" rc "on",
      "generating a data ID DEFMCDID."||message
    Call Final_Exit 8
  End

  If mbr_exists Then Do
    Address ISPEXEC "LMOPEN DATAID("||defmcdid||")"
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave RC" rc "on opening DEFMCDDN."
      Call Final_Exit 8
    End
    defmcopn = 1
    Address ISPEXEC "LMMFIND DATAID("||defmcdid||") MEMBER("||,
      defmigr_mbr||") STATS(YES)"
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave RC" rc "on retrieving statistics for",
        "DEFMCDDN."
      Call Final_Exit 8
    End
    Address ISPEXEC "LMCLOSE DATAID("||defmcdid||")"
    If rc<>0 Then Do
      Call Disp_Msg "ISPF gave rc" rc "on closing DEFMCDDN."
      Call Final_Exit 8
    End
    defmcopn = 0
  End
  Else Parse Value "" With zlvers zlmod zlcdate zlmdate zlmtime,
    zlmsec zlcnorc zlinorc zluser zlc4date zlm4date
End

migrcntl.0 = clines
"EXECIO * DISKW" defmcddn "(STEM MIGRCNTL. FINIS"
If rc<>0 Then Do
  Call Disp_Msg "DEFMGO44E EXECIO error occurred, Rc("||rc||")"
  Call Final_Exit 8
End

If defmigr_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("||defmcdid||") MEMBER("||,
    defmigr_mbr||")" lmmstats_parm
  If rc<>0 Then Do
    Call Disp_Msg "ISPF gave rc" rc "on generating statistics for",
      "DEFMCDID."
    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 migration processing information."
Call Disp_Msg Strip(ispfmsg message)
Address ISPEXEC "EDIT DATASET('"||defmigr_ctl||"') MACRO(DMGP$MAC)"
Select
  When rc=0 Then Nop
  When rc=4 Then Nop
  Otherwise Do
    Call Disp_Msg "DEFMG045E 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("DEFMGDID")="VAR" Then
    Address ISPEXEC "LMFREE DATAID("||defmgdid||")"
  If defmgddn<>"" Then Call Bpxwdyn "FREE DD("||defmgddn||")"
  If Symbol("DEFMMDID")="VAR" Then Do
    Address ISPEXEC "LMFREE DATAID("||defmmdid||")"
    Address ISPEXEC "VERASE (ZFSM$DID)"
  End
  If defmmddn<>"" Then Call Bpxwdyn "FREE DD("||defmmddn||")"
  If Symbol("DEFMCDID")="VAR" Then Do
    If defmcopn Then Address ISPEXEC "LMCLOSE DATAID("||defmcdid||")"
```

```
    Address ISPEXEC "LMFREE DATAID("||defmcdid||")"
  End
  If defmcddn<>"" Then Call Bpxwdyn "FREE DD("||defmcddn||")"
  If Symbol("DEFMDDID")="VAR" Then Do
    If listaccd Then
      Address ISPEXEC "LMDLIST LISTID("||defmddid||") OPTION(FREE)"
    Address ISPEXEC "ISPEXEC LMDFREE LISTID("||defmddid||")"
  End
Exit final_rc

Syntax_Error:
  Say "DEFMGO97W REXX error in sourceline" sigl "of" myname
  Say "DEFMGO98I Line" sigl||":" Strip(Sourceline(sigl))
  Say "DEFMGO99E" Errortext(rc)||", Rc("||rc||")"
  Call Final_Exit 12
Exit 9999

Novalue_Error:
  retc = rc
  Say "DEFMG1OOW REXX error in sourceline" sigl "of" myname
  Say "DEFMG1O1I Line" sigl||":" Strip(Sourceline(sigl))
  Say "DEFMG1O2E Variable not initialized..."
  Call Final_Exit 12
Exit 9999

Halt_Request:
  Say "DEFMG1O3W REXX processing halted at line" sigl "of" myname
  Say "DEFMG1O4I 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
  migrcntl.clines = mdcline
Return

Write_MFile:
  "EXECIO" mlines "DISKW" defmmddn "(STEM MLINE. FINIS"
  If rc<>0 Then Do
    Call Disp_Msg "DEFMGO46E EXECIO error occurred, Rc("||rc||")"
    Call Final_Exit 8
  End

  If Symbol("defmmdid")="LIT" Then Do
    Address ISPEXEC "LMINIT DATAID(DEFMMDID) DDNAME("||defmmddn||")"
    If rc<>0 Then Do
    If rc=8 Then message = "" Strip(zerrmsg) Strip(zerrlm)
    Else message = ""
    Call Disp_Msg "DEFMGO47E ISPF service LMINIT gave RC" rc "on",
      "generating a data ID DEFMMDID."||message
      Call Final_Exit 8
    End
    zfsm$did = defmmdid
    Address ISPEXEC "VPUT (ZFSM$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
```

A.5 DMGC$MAC

The REXX procedure DMGC$MAC is an ISPF macro used from within procedure DEFMIGR to easily call CPYMHELP.

Example A-5 displays the contents of DMGC$MAC.

Example: A-5 DMGC$MAC REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DMGC$MAC                                             */
/*  Description: Call CPYMHELP from within DEFMIGR EDIT session       */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dmgc$mac | cpymhelp | 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

"CPYMHELP"
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
```

A.6 DMGH$MAC

The REXX procedure DMGH$MAC is an ISPF macro used from within procedure DEFMIGR to easily call DEFMHELP.

Example A-6 displays the contents of DMGH$MAC.

Example: A-6 DMGH$MAC REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DMGH$MAC                                             */
```

```
/*  Description: Call DEFMHELP from within DEFMIGR EDIT session       */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dmgh$mac | defmhelp | 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

"DEFMHELP"
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
```

A.7 DMGM$MAC

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

Example A-7 displays the contents of DMGM$MAC.

Example: A-7 DMGM$MAC REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DMGM$MAC                                             */
/*  Description: Display Messages from the previous processing step   */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dmgm$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 DEFMMDDN."
    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
```

A.8 DMGP$MAC

The REXX procedure DMGP$MAC is the ISPF profile macro used for the edit sessions in procedure DEFMIGR.

Example A-8 displays the contents of DMGP$MAC.

Example: A-8 DMGP$MAC REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DMGP$MAC                                             */
/*  Description: Profile MACRO for all DEFMIGR EDIT sessions          */
/*               Property of IBM  (C) Copyright IBM Corp. 2006        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: called as initial macro on editing DEFMGDID 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 DMGM$MAC MACRO"
"ISREDIT DEFINE DM ALIAS DMGM$MAC"
"ISREDIT DEFINE DMGH$MAC MACRO"
"ISREDIT DEFINE DEFMHELP ALIAS DMGH$MAC"
"ISREDIT DEFINE DH ALIAS DMGH$MAC"
"ISREDIT DEFINE DMGC$MAC MACRO"
"ISREDIT DEFINE CPYMHELP ALIAS DMGC$MAC"
"ISREDIT DEFINE CH ALIAS DMGC$MAC"
"ISREDIT DEFINE REFRESH  ALIAS DMGR$MAC"
"ISREDIT DEFINE RF ALIAS DMGR$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(ISRZ00"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

A.9 DMGR$MAC

The REXX procedure DMGR$MAC is an ISPF macro that refreshes or resets the current migration control data definitions that are displayed with procedure DEFMIGR.

Example A-9 displays the contents of DMGR$MAC.

Example: A-9 DMGR$MAC REXX procedure

```
/* REXX ***************************************************************/
/*    Procedure: DMGR$MAC                                             */
/*  Description: Refresh DEFMIGR control data definitions             */
/*               Property of IBM  (C) Copyright IBM Corp. 2007        */
/*               Robert Hering (robert.hering@de.ibm.com)             */
/*    Format is: dmgr$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 "DEFMIGR"
Exit 0

Disp_Msg: Procedure
  Trace O
  Parse Arg zedlmsg, type
  zedsmsg = ""
  type = (type<>"I")
  Address ISPEXEC
  "VPUT (ZEDSMSG ZEDLMSG)"
  "SETMSG MSG(ISRZ00"||type||")"
  "CONTROL DISPLAY REFRESH"
Return
```

A.10 MIGRDATA

The file MIGRDATA is the JCL used for the job running the migration copy processing and optionally replaces the old HFS file system with the new one of type zFS.

Example A-10 displays the contents of JCL MIGRDATA.

Example: A-10 MIGRDATA JCL

```
//ZFSJOB   JOB ,'MIGRDATA',NOTIFY=&SYSUID.,REGION=0M
/*JOBPARM SYSAFF=SCxx <=== JES2 system affinity
//*MAIN SYSTEM=SCxx   <=== JES3 system affinity
//* ------------------------------------------------------------------
//* Migrate HFS data sets to zFS compat mode aggregates
//* Property of IBM  (C) Copyright IBM Corp. 2002-2007
//* ------------------------------------------------------------------
//  SET MGRTOOL=COPYPAX                  <=== Copy utility to be used
//*             COPYMIGR: Use copy tool provided with migration tool
//*             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 COPYMIGR or COPYPAX are set.
//  SET DEFMIGR=&SYSUID..ZFS.MIGRATE.DEFINE(xxxxxx)  <=== MIGRATE DEFs
//* ------------------------------------------------------------------
//  SET REXXLIB=&SYSUID..ZFS.REXX.EXEC           <=== SYSEXEC library
//  SET STEPLIB=&SYSUID..ZFS.MIGRTOOL.LOADLIB    <=== STEPLIB library
//* ------------------------------------------------------------------
```

```
//COPYMIGR EXEC PGM=IKJEFT01,
//  PARM='COPYMIGR &MGRTOOL. &VERBOSE. &OVERWRT.'
//STEPLIB  DD DSNAME=&STEPLIB.,DISP=SHR
//*        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 HFS and if existing the zFS is/are mounted (N|Y)
  STOP_AFTER_FSS_MOUNTED=N
# Force stopping when the 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 HFS by zFS
  DENIED_UMNT_FSTYPES=TFS
# Specify additional directories to be included in the PATH chain
# PATH=/usr/local/bin
# ---------------------------------------------------------------------
##
//STDIN    DD DSNAME=&DEFMIGR.,DISP=SHR
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
//* ---------------------------------------------------------------------
```

HFS to zFS Migration Tool

Redpaper™

Installing the migration tool

Using the migration tool

Migration tool REXX execs

The z/OS® Distributed File Service zSeries® File System (zFS) is the strategic z/OS UNIX® file system that is suggested to be used instead of the Hierarchical File System (HFS), beginning with z/OS V1R7. Therefore, in z/OS V1R7 an official migration tool, BPXWH2Z, is delivered with z/OS to support installations in replacing HFS with zFS.

This Redpaper describes and provides an additional HFS to zFS migration tool, named MIGRTOOL, that has been created as an alternative to BPXWH2Z. It provides more flexibility in many situations and offers options for how the migration should be done. In addition, it supports migration processing independent of the z/OS release, while BPXWH2Z is not supported in a z/OS system running at a level prior to V1R7.

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-4328-00